IBM

IBM Fault Analyzer for z/OS
User's Guide and Reference
V17.1.0

Note

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

Edition notice

This edition (published October 2026) applies to Version 17 Release 1 Modification Level 0 of IBM Fault Analyzer for z/OS® (program number 5755-AD2) and to any subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this document ... xii
Who should use this document ... xii
Organization of this document ... xii
How to read the syntax diagrams ... xii
Summary of changes ... xvi
Part I. Using Fault Analyzer ... 18
Chapter 1. Introduction ... 19
The analysis engine ... 19
The analysis process ... 19
Real-time abend analysis ... 19
Real-time SNAP analysis ... 21
Fault reanalysis ... 22
Batch reanalysis ... 23
Interactive reanalysis ... 24
Fault history files ... 24
Associated dump data sets ... 25
Special members in the history file data set ... 25
Fault Analyzer supported application environments ... 26
Binder-related dependencies ... 29
Setting up existing programs for fault analysis ... 29
Additional region size required ... 30
Compiler listing or side file selection criteria ... 30
Special processing of Language Environment CEEWUCHA abends ... 30
WTO routing and descriptor codes used by Fault Analyzer ... 31
CICS Storage Accounting Area (SAA) overlay assistance ... 31
Chapter 2. Real-time analysis ... 32
Dump suppression ... 32
Fault Analyzer and CICS global user exits ... 33
Fault history file selection ... 33
Controlling the real-time analysis with options ... 34
JCL DD statements ... 35
Pointing to listings with JCL DD statements ... 35
The real-time analysis report ... 35
Combining Fault Analyzer real-time reports ... 36
Controlling the SYSOUT class of real-time reports ... 36
Suppressing real-time reports ... 37
The SYSLOG summary ... 37
Using the program SNAP interface (IDISNAP) ... 37
IDISNAP invocation ... 38
Dump registration processing ... 47
Real-time exclusion processing ... 47
Duplicate fault processing ... 50
Recovery fault recording ... 51
RFR dump titles ... 52
Using SLIP,COMP=0C4 with Fault Analyzer ... 53
Extended minidump data set (XDUMP) ... 53
Abend traps can prevent fault analysis ... 54
Capturing and displaying the abending job JCL ... 54
Chapter 3. The Fault Analyzer ISPF interface ... 55
Invoking the interface ... 56
ISPF split screen support ... 56
The Fault Entry List display ... 56
Using views ... 59
Changing the history file or view displayed ... 59
Fault entry list column configuration ... 65
Sorting and matching fault entries ... 71
Additional ways to match and select faults ... 77
Applying an action to a particular fault ... 81
History file properties ... 84
New history file allocation ... 87
Change fault history file settings ... 88
Resetting history file access information ... 91
Refreshing fault entry information ... 92
Fault entry expiration control ... 94
Action-bar pull-down menus ... 94
Commands ... 98
CE ... 98
CICSD ... 98
CICSLINK ... 99
CICSSTG ... 99
CMDS ... 99
COLS ... 99
COPY ... 100
CUROPTS ... 100
DISASM ... 100
DSECT ... 101
DUPS ... 101
EDIT ... 101
EXEC ... 101
FIND ... 102
IDISINFO ... 105
HTTP ... 105
INFO ... 106
JCL ... 106
LOOKUP ... 106
MATCH ... 108
NEXT ... 108

NOTE ... 108
NOTELIST ... 109
PREV ... 109
QUIT ... 109
REFRESH ... 110
RESET ... 110
RPTFIND ... 110
RUNCHAIN ... 110
SHOW ... 111
SHOWFREE ... 111
SIT ... 111
STCK ... 112
STGMAP ... 112
VIEWS ... 112
XML ... 112
Fault Analyzer preferences ... 113
BatchOpts ... 116
Viewing a saved report ... 120
Adding or removing blank lines ... 122
Adding or removing help text ... 122
Setting preferred formatting width ... 122
Displaying user-selected message or abend code explanations ... 123
Copying interactive displays to a file ... 125
Displaying product copyright, license, and version information ... 126
Deleting history file entries ... 126
Locking fault entries ... 127
Viewing fault entry information ... 129
Viewing the fault entry duplicate history ... 134
Expanding and collapsing duplicate fault entries ... 137
Copying history file entries ... 138
Moving history file entries ... 140
Transmitting history file entries ... 140
Packaging fault entries ... 141
Security considerations ... 141
Specifying 64-bit addresses ... 142
Using the interactive IDIS subsystem interface ... 142
Managed history files ... 143
Excluded history files ... 144
Selected history files ... 145
Displaying options currently in effect ... 147
Recovering user notes ... 147
Chapter 4. Performing batch reanalysis ... 149
Batch reanalysis options ... 149
Initiating batch reanalysis ... 154
Data sets used for batch reanalysis ... 154
Creating your own batch reanalysis job ... 155
Chapter 5. Performing interactive reanalysis ... 157
Interactive reanalysis options ... 157
Initiating interactive reanalysis ... 164
Status pop-up display ... 164
General information about the interactive report ... 165
Exit from the interactive report ... 167
Primary option: Synopsis ... 168
Primary option: Event Summary ... 168
Detailed Event Information ... 170
Primary option: Open Files ... 173
Primary option: CICS Information ... 176
CICS Control Blocks ... 177
CICS Transaction Storage Summary ... 177
CICS Transaction Storage ... 177
Last CICS 3270 Screen Buffer ... 178
Summarized CICS Trace ... 179
CICS Trace Formatting ... 180
CICS Task Trace Table ... 186
CICS Recovery Manager ... 186
CICS Levels, Commareas, and Channels ... 186
CICS Monitoring Data ... 187
CICS Event Program Information ... 187
CICS System Initialization Parameters ... 188
Primary option: Messages ... 188
Primary option: DB2 Information ... 188
Primary option: IMS Information ... 191
Primary option: Storage Areas ... 197
Primary option: Java Information ... 198
Primary option: Language Environment Heap Analysis ... 198
Primary option: MTRACE Records ... 199
Primary option: Abend Job Information ... 200
Primary option: User Notes ... 201
Primary option: Fault Analyzer Options ... 201
Displaying associated storage areas ... 202
Hiding the hex-value column ... 203
Collapsing level 88 items ... 204
Showing all COBOL base locators ... 205
Expanding messages and abend codes ... 206
Displaying source code ... 206
Deferred Breakpoints Feature ... 209
Displaying storage locations ... 209
Displaying data areas ... 211
Creating and managing user notes ... 212
Displaying CICS transaction storage (CICSSTG) ... 217
Displaying the address space storage map (STGMAP) ... 218
Displaying formatted HTTP data (HTTP) ... 219

Displaying formatted XML (XML)..... 220
Displaying PSW information..... 222
Mapping storage areas using DSECT information.....223
IDIDSECT concatenation.....226
Indexing your DSECT data sets ($DINDEX member).....226
Displaying chained data areas.....227
Disassembling object code.....229
Converting STORE CLOCK values.....231
User-specific report formatting.....233
Prompting for compiler listing or side file.....235
Controlling prompting..... 240
Data sets used for interactive reanalysis..... 240
Refresh processing.....240
COBOL Explorer..... 241
COBOL Explorer example.....242
Deferred Breakpoints Feature.....246
Chapter 6. Performing CICS system abend dump analysis.....247
Setting options for CICS system abend analysis..... 247
User exits..... 247
Selecting a CICS dump data set..... 247
Selecting an address space to analyze..... 249
Displaying the CICS system abend interactive report..... 250
Fastpath navigation.....251
Option 1: Synopsis..... 251
Option 2: Abend Job Information.....252
Option 3: CICS System Information..... 252
Option 4: Options in Effect..... 254
Creating a fault entry.....255
Chapter 7. Formatting a CICS auxiliary trace data set.... 257
Selecting a CICS auxiliary trace data set..... 257
Specifying CICS Trace Selection Parameters..... 258
Chapter 8. Performing Java analysis.....259
Selecting a Java dump data set..... 259
Java fault entry reanalysis..... 260
Displaying the Java information in the interactive report..... 262
Java event replacement in Fault Analyzer event list.....266
Java information reporting limitations.....267
Java application samples..... 268
Sample 1 (IDISJAV1): Enterprise PL/I invoking Java in batch 268
Sample 2 (IDISJAV2): Enterprise COBOL invoking Java in batch 269
Sample 3 (IDISJAV3): 64-bit Enterprise PL/I invoking Java in batch 269
Sample 4 (IDISJAV4): IMS Java batch processing sample.....269
Sample 5 (IDISJAV5): C++ program invoking Java in batch..... 270
Sample 6 (IDISJAV6): Call a Java batch application with the Fault Analyzer wrapper utility.....270
Chapter 9. The Fault Analyzer report.....272
General report information.....272
Most significant abend code.....272
Open file record information..... 272
COBOL suppressed copybooks.....274
Main report sections..... 275
The prolog section..... 275
The synopsis section.....275
The summary section..... 275
The event details section.....279
The system-wide information section..... 281
The user section.....281
The abend job information section.....281
The options section..... 281
The epilog section.....282
Sample reports.....282
Chapter 10. Using non-ISPF interfaces to access Fault Analyzer history files..... 283
Fault Analyzer plug-in for Eclipse..... 283
Fault Analyzer on VS Code.....283
Performing interactive reanalysis under CICS®..... 284
Asynchronous AI analysis using the Model Context Protocol (MCP) server.....284
Part II. Fault Analyzer installation and administration.... 286
Chapter 11. Migrating from an earlier version of Fault Analyzer.....287
Migrating from V16.1 to V17.1..... 287
Migrating from V15.1 to V16.1..... 287
Migrating from V14.1 to V15.1..... 287
Migrating from V13.1 to V14.1..... 288
Migrating from V12.1 to V13.1..... 288
Migrating from V11.1 to V12.1..... 289
Migrating from V10.1 to V11.1..... 290
Migrating from V9.1 to V10.1..... 290
Chapter 12. Preparing to customize Fault Analyzer..... 291
Checklist for installing and customizing Fault Analyzer..... 291
Library names after you finish installing..... 295
Storage recommendations.....296
Exits for invoking Fault Analyzer..... 298
Invocation for non-CICS transaction abends.....298
Invocation for CICS transaction abends..... 302
SVC dump registration.....302

Language Environment options required for invocation of Fault Analyzer....303
LE options required for non-CICS abends....303
LE options required for CICS abends....304
Running Fault Analyzer with similar third-party products....304
MVS dump data set size....304
Application-handled error conditions....304
Setting Fault Analyzer SLIP traps....305
Chapter 13. Customizing the operating environment for Fault Analyzer....306
Making Fault Analyzer modules available....306
Defining program control access to Fault Analyzer programs....307
Restricting change of history file settings....307
Setting up the message and abend code explanation repository....308
Managing recovery fault recording data set access....309
SDUMP recovery fault recording data sets....310
TDUMP recovery fault recording data sets....311
Managing XDUMP data set access....313
Managing copied SDUMP data set access....314
Registering Fault Analyzer in the IFAPRDxx parmlib member....316
Chapter 14. Using the Fault Analyzer IDIS subsystem....318
Sysplex-wide subsystem inter-communication....319
Caching of history file $$INDEX data....319
Starting the IDIS subsystem....320
IDIS subsystem storage requirements....322
IDIS subsystem requirements for DB2....323
IDIS subsystem requirements for Java....324
IDIS subsystem requirements for deletion of migrated tightly-coupled associated dump data sets....325
Stopping the IDIS subsystem....325
Chapter 15. Modifying your ISPF environment....327
Allocating ISPF data sets....327
Making the Fault Analyzer IDISCMDS command table available....328
Updating the ISPF selection panel....328
Invoking Fault Analyzer using an ISPF 3.4 line command....329
Invoking Fault Analyzer from SDSF (SFA command)....330
Invoking the LOOKUP command using cursor selection (LOOKC command)....330
Providing ISPF interface defaults for new users....330
Providing installation-specific batch reanalysis JCL control statements....331
Increasing the display area for Fault Analyzer reports....331
Chapter 16. Customize Fault Analyzer by using USERMODs....333
Enabling Fault Analyzer to be invoked....333
Installing the MVS change options/suppress dump exit IDIXDCAP....333
Installing the Language Environment abnormal termination exits IDIXCEE and IDIXCE64....333
Working with applications that use a non-Language Environment run time....335
Obtaining load modules from CA-Panexec....335
Chapter 17. Customize Fault Analyzer by using an IDIOPTLM configuration-options module....337
Specifying an alternative parmlib data set for IDICNFxx (CNFDSN)....339
Changing the default recovery fault recording IEATDUMP data set name (RFRDSN)....339
Changing the action of Fault Analyzer when invoked via the CICS XDUREQ global user exit (CICSNOA)....340
Specifying an alternative security server test data set name (SSCHKDSN)....340
Changing the default extended minidump (XDUMP) data set name pattern (XDUMPDSN)....340
Specifying the copied SDUMP data set name pattern (SDUMPDSN)....341
Ignoring IBM Application Delivery Foundation for z/OS Common Components options (NOIPVOPT)....341
Suppressing IDIXDCAP real-time analysis if prior exit RC=8 (PREVEXIT)....342
IDIOPTLM data set name symbol substitution....342
Chapter 18. Setting up history files....343
Determining what size history files to allocate....343
Allocating a PDS or PDSE for a history file....343
Allocating secondary space for history files....344
Converting history files between PDS and PDSE....345
AUTO-managed PDSE history files....345
Changing the size of a PDSE history file....347
Providing the name of a history file to Fault Analyzer....347
Setting up views....347
Specifying a default column layout....348
Specifying initial fault entry selection criteria....349
View considerations if not using XFACILIT resource class....350
Managing history files across MVS systems without shared DASD....350
Automated implementation....350
On-demand implementation....359
Sharing of history files across a sysplex....362
Managing history file fault entry access....363
Using the XFACILIT resource class for history file fault entries....363
Chapter 19. Setting and changing default options for the site....369

Parmlib member IDICNFxx....369
Controlling which jobs are analyzed with Exclude processing....371
Fast Exclude options processing....372
Controlling report detail....372
Limiting the size of minidumps....373
Pointing to listings and REXX exec libraries....373
Suppressing noncritical SYSLOG messages....373
Customizing the Fault Entry List display....373
Specifying the cultural environment....373
Chapter 20. Providing compiler listings or Fault Analyzer side files....374
COBOL Report Writer Precompiler....375
Required compiler options for IDILANGX....375
TEST option considerations....377
DEBUG option considerations....378
Limitations when using COBOL NOTEST(DWARF,SOURCE)....378
Limitations when using PL/I TEST(NOSEPARATE NOSOURCE)....378
Naming compiler listings or side files....378
Naming CSECTs for Fault Analyzer....379
Locating compiler listings or side files....380
IDITRACE information....383
Compiler listings and side file access requirements....384
Using the IDIXSFOR compiler listing/side file search and override exit....384
IDIXSFOR invocation....384
Example (Assembler)....387
Compiler listings and side file attributes....387
Using the IDIRLOAD DDname for CSECT mapping....388
IDIDATST side file availability test utility....389
ISPF-packed compiler listings....390
Providing Java source information to Fault Analyzer....391
Examples: Using IDIJAVA DD and IDIJAVA IDIOPTS....392
Compiling Java for optimal debugging....393
Generating debugging information with common Java build tools....394
IDISBLKO....395
Chapter 21. Customizing the CICS environment....396
Configuring Language Environment for CICS to invoke Fault Analyzer....397
Defining required programs to CICS....397
Adding the required programs to the startup PLT....398
Adding the required programs to the shutdown PLT....398
Enabling dynamic control of analysis of CICS transaction abends....399
Using CFA to FORCEPURGE the currently analyzed task....399
SVC dump screening....399
Sample definition job....399
Controlling CICS transaction abend analysis....400
Using CFA from a CICS® terminal....401
Using CFA from an MVS console....403
CICS web services transaction....405
Ensuring transaction abend analysis is not suppressed by DUMP(NO)....408
CICS NoDup(CICSFAST) override assembler exit (IDINDFUE)....408
Invocation....408
CICS trace considerations....410
Preventing LE from causing the CICS trace to wrap....410
Specifying CICS options through the IDIOPTS DDname....411
Language Environment abend considerations....411
Capture of abends running on CICS user key open TCBs (L9 TCBs)....411
Installing the MVS post-dump exit IDIXTSEL....412
Maximizing CICS transaction abend analysis performance....412
Implementing an XEIIN global user exit....412
Disabling 3270 screen buffer capture....414
Chapter 22. Customizing the DB2 environment....415
Binding DB2....415
DB2 and Language Environment....415
Improving Fault Analyzer DB2 performance....415
Chapter 23. Customizing the IMS environment....417
IMS and Language Environment....417
Chapter 24. Customizing the Java environment....418
Invoking Fault Analyzer from a Java try-catch block....418
Invoking Fault Analyzer from Java dump events....421
Using the Fault Analyzer Java wrapper utility....422
Using the Fault Analyzer JVMTI Agent....424
JVM version 9 issues and limitations....425
Chapter 25. Customizing the Fault Analyzer Japanese feature....426
Allocating ISPF data sets....426
Setting the national language....426
Chapter 26. Verifying Fault Analyzer customization....428
Verifying the use of XDUMPs....428
Verifying the use of Fault Analyzer with assembler....428
Verifying the use of Fault Analyzer with COBOL....429
Verifying the use of Fault Analyzer with PL/I....430
Verifying the use of Fault Analyzer with C....431

Verifying the IDIXCEE Language Environment exit enablement....432
Verifying Fault Analyzer customization under CICS....432
CICS IVP: 0C1 in program IDIXFA....433
CICS IVP: EXEC CICS® DUMP DUMPCODE(FAD1)....433
CICS IVP: EXEC CICS® ABEND ABCODE(FLT1)....434
CICS IVP: EXEC CICS ABEND ABCODE(FLT2)....434
Verifying the use of Fault Analyzer with DB2....434
Using a C program....435
Using a COBOL program....436
Verifying the use of Fault Analyzer through ISPF....438
Verifying the recovery fault recording setup....439
Verifying dump registration (IEAVTSEL) 439
Chapter 27. Managing history files (IDIUTIL utility)....441
IDIUTIL control statements....441
FILES control statement....442
LISTHF control statement....442
LISTHFDUP control statement....443
DELETE control statement....444
SETFAULTPREFIX control statement....446
SETMAXFAULTENTRIES control statement....446
SETMINFAULTENTRIES control statement....447
IMPORT control statement....448
EXPORT control statement....449
EXITS control statement....450
Examples....450
Example 1. Listing history file entries....451
Example 2. Listing history file abend instances....451
Example 3. Deleting history file entries by date....451
Example 4. Deleting history file entries by utilization....451
Example 5. Changing history file fault prefix characters....452
Example 6. Creating a self-maintaining history file....452
Example 7. Importing history file entries....452
IDIUTIL batch utility user exit samples....453
Chapter 28. Providing application-specific explanations and descriptions....455
User-defined message explanations....455
User-defined abend code explanations....456
User-defined program descriptions....457
Chapter 29. Maintaining Fault Analyzer....459
Chapter 30. Disabling Fault Analyzer....461
Temporarily uninstalling Fault Analyzer....461
Turning off Fault Analyzer using the IFAPRDxx parmlib member....462
Turning off Fault Analyzer with a JCL switch (IDIOFF)....462
Turning off Fault Analyzer using an environment variable (_IDI_OFF)....463
Chapter 31. Customizing Fault Analyzer by using user exits....464
Invocation parameters....467
Global environment data area (ENV)....467
User exit type specific data area....467
Supported exit programming languages....467
Determining which exit type is invoking a common user exit....469
Data area version checking....469
Diagnostic tracing....470
Tracing user exit parameter list values....470
Tracing REXX EXECs....474
Descriptions of fault analysis user exit types....474
Analysis Control user exit....475
Analysis Control user exit (MVS SVC Dump registration)....479
Compiler Listing Read user exit....480
Message and Abend Code Explanation user exit....485
Formatting user exit....490
End Processing user exit....494
End Processing user exit (Fault entry refresh)...497
Notification user exit....498
Notification user exit (MVS SVC Dump registration)....506
Descriptions of IDIUTIL batch utility user exit types....507
IDIUTIL Import user exit....507
IDIUTIL Delete user exit....510
IDIUTIL ListHF user exit....511
IDIUTIL ListHFDUP user exit....515
User exit REXX commands....517
Evaluate command....517
IDIALLOC command....521
IDIDDTEST command....525
IDIDSECTdsn command....526
IDIDSNTEST command....527
IDIEventInfo command....528
IDIFREE command....529
IDIGET command....530
IDIModQry command....530
IDIPUT command....531
IDIRegisterFaultEntry command....532
IDIWRITE command....533

IDIWTO command.....535
List command.....536
Note command.....538
Formatting tags.....539
ADDR (address).....542
AREA (area).....542
DD (definition description).....543
DATA (data).....543
DL (definition list).....543
DT (definition term).....544
DUMP (EBCDIC dump).....545
DUMPA (ASCII dump).....546
HP (highlighted phrase).....547
L (line).....547
LI (list item).....547
NOTEL (note list).....548
P (paragraph).....548
TH (table heading).....549
U (underline).....549
UL (unordered list).....550
The IDIXUFMT load module Formatting user exit.....550
IDIXUFMT functions.....551
Interfacing with the ADFzCC Event Processing user exit.....559
Samples provided by FA.....559
Chapter 32. Installing non-ISPF interfaces to access Fault Analyzer history files.....561
Installing the IBM Fault Analyzer plug-in for Eclipse.....561
Customizing the IBM Application Delivery Foundation for z/OS® Common Components server.....561
Installing the IBM Fault Analyzer plug-in for Eclipse.....562
Installing IBM Fault Analyzer for z/OS® on VS code.....562
Enabling interactive reanalysis under CICS.....563
Making the required CICS resource definitions.....563
Making the required CICS JCL changes.....564
Configuring the MCP Server.....564
Configuring local authentication using Windows Credential Manager.....565
Chapter 33. Installing and configuring Abend Intelligence.....567
The STDENV DD statement environment variables for Abend Intelligence.....569
Part III. Fault Analyzer reference information.....570
Chapter 34. Options.....571
Where to specify options.....573
Parmlib member IDICNFxx.....573
User-options module IDICNFUM.....574
The _IDI_OPTSFILE environment variable.....575
User options file IDIOPTS.....575
The JCL EXEC statement PARM field.....576
The _IDI_OPTS environment variable.....576
When changes to options take effect.....576
Option descriptions.....577
AdditionalIDIOffDD.....577
BindExclude.....577
CallEqaueDat.....578
CICSDumpTableExclude.....578
CICSTraceMax.....579
CICSTranAnalysisUser.....580
CompileDateCheck.....580
DataSets.....581
DeferredReport.....589
Detail.....591
DumpDSN.....592
DumpRegistrationExits.....593
ErrorHandler.....594
Exclude/Include.....595
Exits.....600
FAISPFopts.....603
FaultID.....610
GenerateSavedReport.....611
Include.....612
JclCapture.....612
Language.....612
LangxCapture.....613
Locale.....614
LoopProtection.....614
MaxMinidumpPages.....615
NoDup.....616
PermitLangx.....626
PreferredFormattingWidth.....628
PrintInactiveCOBOL.....628
Quiet.....629
RefreshExits.....630
RetainCICSDump.....631
RetainDump.....632
Snapdata.....633
Source.....633
SpinIDIREPRT.....634
StoragePrintLimit.....634
StorageRange.....635
SystemWidePreferred.....636
UnreferencedPagesValid.....638

Chapter 35. Data areas..... 639
Non-REXX user exit buffered data format.....639
Data area descriptions..... 640
CTL - Analysis Control user exit parameter list.....640
ENV - Common exit environment information....653
EPC - End Processing user exit parameter list... 668
LST - Compiler Listing Read user exit parameter list.....670
NFY - Notification user exit parameter list.....673
UFM - Formatting user exit parameter list.....674
UTL - IDIUTIL Batch Utility user exit parameter list.....688
XPL - Message and Abend Code Explanation user exit parameter list..... 690
Chapter 36. Return codes..... 693
Chapter 37. Messages..... 695
Fault Analyzer messages..... 695
Appendix A. Sample customized ISPF interface front-end.....741
Fault History File or VIEW name.....742
MATCH on program name.....743
Installing the sample application.....743
How the sample works.....743
Appendix B. Java API to download Fault Analyzer report.....744
Appendix C. Technical details for screen size adjustments..... 753
Appendix D. Support resources.....754
Appendix E. Accessibility features for Fault Analyzer.... 755
Notices.....dcclix
Glossary..... dcclxiii
Index..... 767

About this document

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

The following names are used in this document:

IBM® Application Delivery Foundation for z/OS (ADFz) family of products

Previously known as IBM® Problem Determination Tools (PD Tools) products.

IBM® z/OS® Debugger (z/OS® Debugger)

Includes the IBM® Debug Tool engine.

IBM® Developer for z/OS

Previously known as IBM® Rational® Developer for z Systems.

IBM® Explorer for z/OS® and plug-ins for ADFz family of products

Previously known as IBM® Problem Determination Tools Studio.

Who should use this document

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

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

Organization of this document

Using Fault Analyzer on page 18 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.

Fault Analyzer installation and administration on page 286 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.

Fault Analyzer reference information on page 570 provides information about options, return codes, data areas, and messages that are issued by Fault Analyzer.

Support resources and problem solving information contains information about IBM® web sites that can help you answer questions and solve problems.

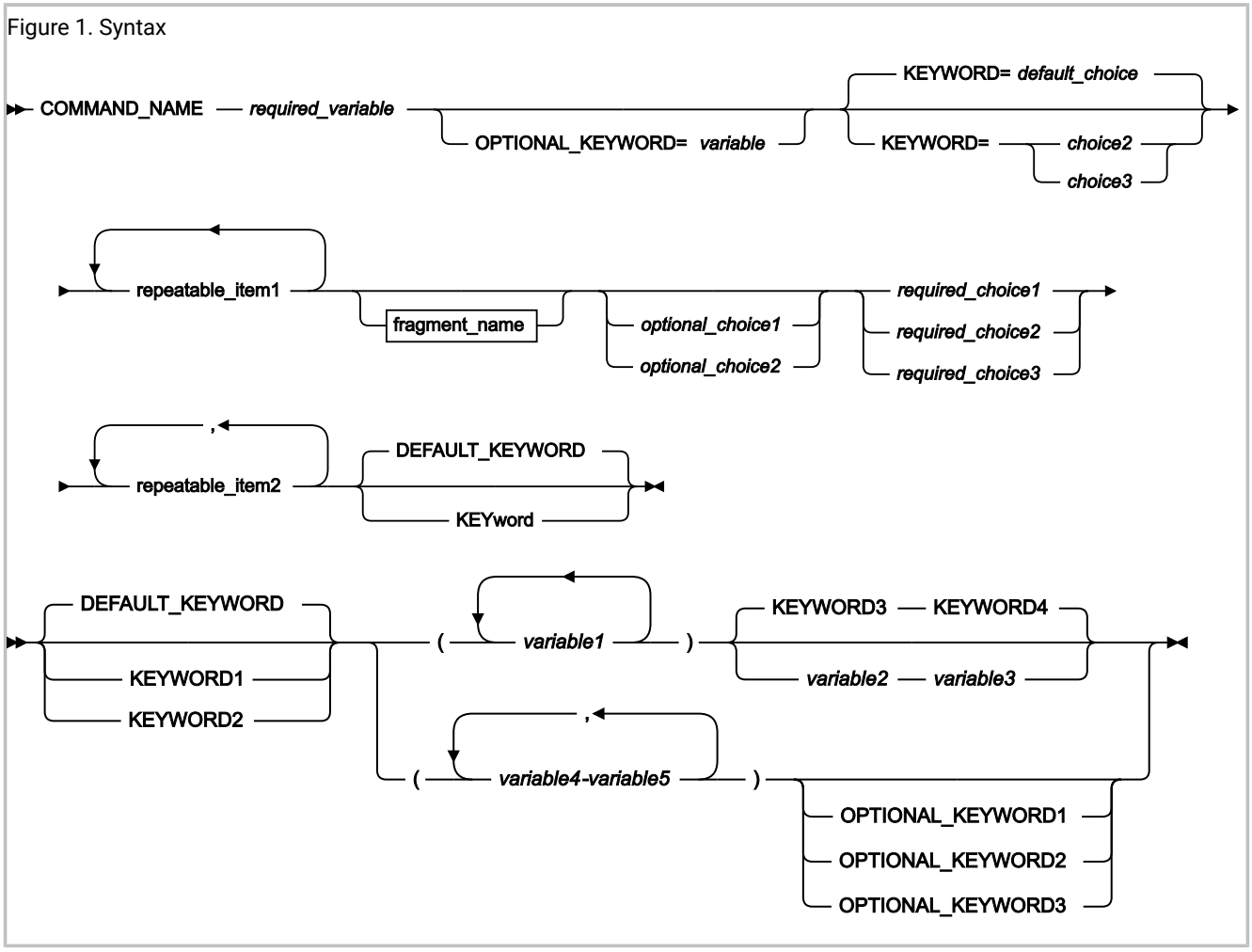
How to read the syntax diagrams

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

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

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

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 the beginning of a statement.

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

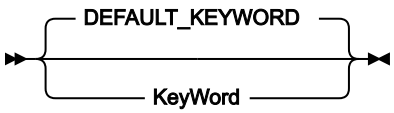
The ►—— symbol indicates that a statement is continued from the previous line.

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

Keywords

Keywords appear in uppercase letters.

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

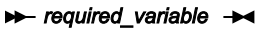


In this example, you could type "KW" or "KEYWORD".

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

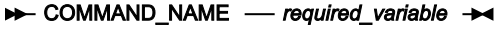
Variables

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



Required items

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



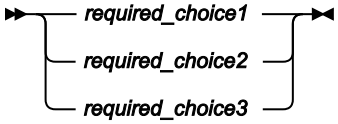
Optional items

Optional items appear below the main path.

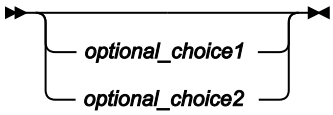
Choice of items

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

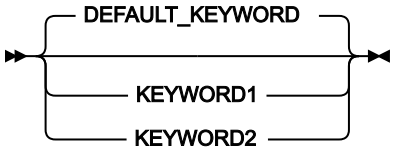
If you *must* choose one of the items, one item of the stack appears on the main path.



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

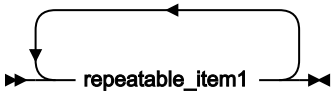


If a default value applies when you do not choose any of the items, the default value appears above the main path.

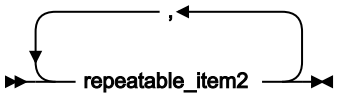


Repeatable items

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

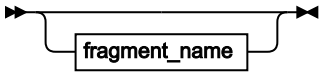


If you need to specify a separator character (such as a comma) between repeatable items, the line with the arrow returning to the left shows the separator character you must specify.

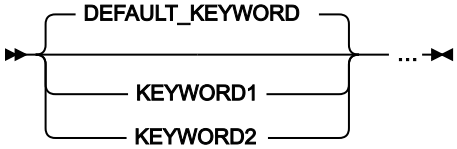


Fragments

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



fragment_name



Summary of changes

This edition of the documentation provides information applicable to Fault Analyzer Version 17 Release 1. Changes to the latest release are indicated by a "|" change bar in the left margin of the page in the PDF format of the documentation.

Latest service: For information about the problems that have been addressed in each APAR, see Latest PTFs for IBM Fault Analyzer for z/OS.

October 2026 (V17R1M0, first edition)

This edition of the documentation contains minor clarifications and corrections, as well as the following significant changes relative to the previous version:

- **Modernizing the mainframe with Artificial Intelligence enhancements**: IBM Fault Analyzer for z/OS® 17.1.0 introduces the following AI capabilities:
  - **IBM Fault Analyzer for z/OS Abend Intelligence** provides real-time host-based AI analysis. It automatically analyzes abends on z/OS using a large language model (LLM) and displays formatted Markdown reports directly in interactive reanalysis. For details, see:
    - The AI analysis report on page 281
    - Installing and configuring IBM Fault Analyzer for z/OS Abend Intelligence on page 567
  - **Model Context Protocol (MCP) server**: Provides a workstation-hosted Java server on Windows that connects to external AI assistants to z/OSMF REST APIs, enabling assistants to search history files, read SDSF job spools, retrieve fault reports, and extract application source code. For details, see:
    - Asynchronous AI analysis using the Model Context Protocol (MCP) server on page 284
    - Configuring the Model Context Protocol (MCP) Server on page 564
- The new CompileDateCheck option specifies whether Fault Analyzer scans user programs in the event list to check if any were compiled within a specified number of days before the abend . If Fault Analyzer detects recently compiled user programs, the synopsis section of the fault analysis report includes a note. For details, see CompileDateCheck on page 580.
- Two new commands have been added:
  - HTTP command: displays formatted HTTP data located at a specified storage address. See HTTP on page 105 for the complete command syntax and see Displaying formatted HTTP data (HTTP) on page 219 for information about how to use this command and interact with the results.
  - XML command: displays formatted XML data located at a specified storage address. See XML on page 112 for the complete command syntax and see Displaying formatted XML (XML) on page 220 for information about how to use this command and interact with the results.
- The LISTHF, LISTHFDUP, and DELETE control statements of the IDIUTIL utility are updated to support the **IMS_PGM** parameter. You can use this qualifier to filter fault entries based on the IMS program name.

For more details, see LISTHF control statement on page 442, LISTHFDUP control statement on page 443, and DELETE control statement on page 444.

- The ability to provide a user name or title when creating a fault entry from interactive MVS dump analysis has been provided. For details, see Creating a fault entry on page 255.

Part I. Using Fault Analyzer

Chapter 1. Introduction

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

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 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 topics that follow.

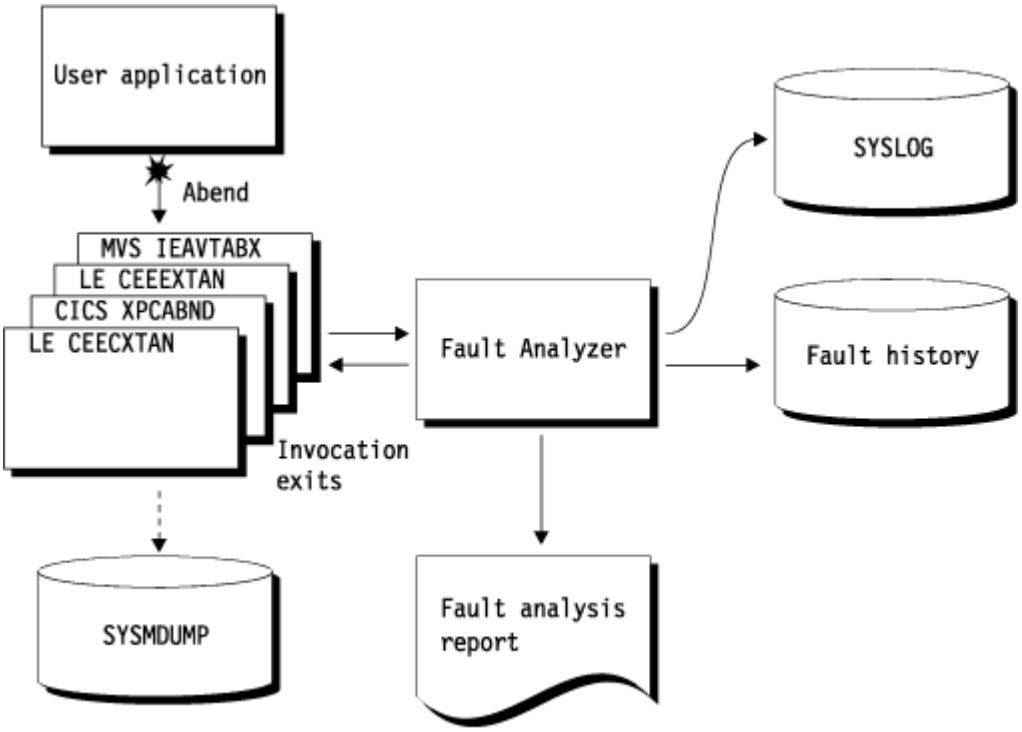
Real-time abend analysis

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 298 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 3: Real-time abend analysis on page 20 illustrates this process.

Figure 3. Real-time abend analysis



For an example of a real-time fault analysis report, see Sample reports on page 282.

Fault Analyzer is designed to only be invoked for the first abend under a given TCB. For this to happen, the required Fault Analyzer invocation exits must have been installed and the criteria for their invocation satisfied. For details, see Exits for invoking Fault Analyzer on page 298.

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 615)
- The fault is a duplicate of another fault (see NoDup on page 616)
- An End Processing user exit requested the minidump to be suppressed (see End Processing user exit on page 494).

The minidump permits reanalysis of faults that occurred in any processing environment, regardless of whether an MVS dump was also written.

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

Instead of forcing application developers and maintainers to interpret a low-level system dump or system-level error messages, the fault analysis report describes the fault in terms of the application code. Where possible, the report quotes the source statement where the abend occurred and, for COBOL and PL/I, the names and values of data items used in the statement.

Generally, when the associated compiler listing (or side file) of the abending program is available online, 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 30 for information about the criteria used when selecting compiler listings or side files). When the listing is not available, the problem is diagnosed down to program name and offset, with disassembly of the machine instructions.

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

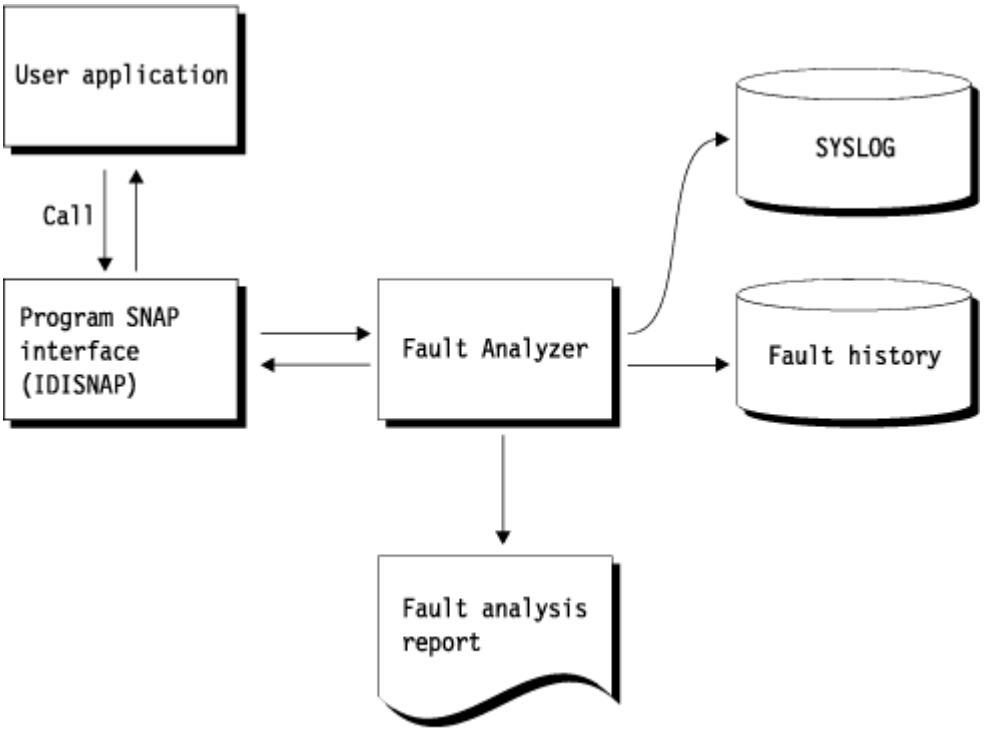
Real-time SNAP analysis

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

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

An illustration of the real-time SNAP analysis process is provided in Figure 4: Real-time SNAP analysis on page 22.

Figure 4. Real-time SNAP analysis



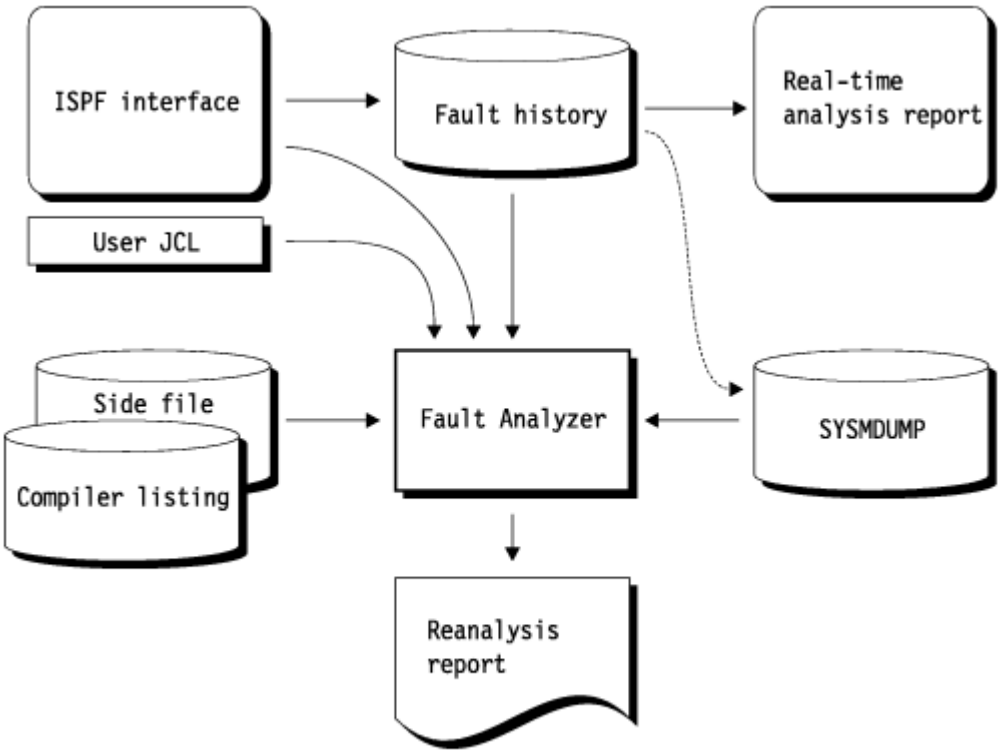
Fault reanalysis

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 (or associated MVS dump data set) instead of the abending program's virtual storage.
- The history file is not updated.
- No summary is written to the SYSLOG.

Figure 5: Reanalysis on page 23 illustrates the reanalysis process.

Figure 5. Reanalysis



If an MVS dump data set is associated with the fault entry, 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 fault entry. If neither a minidump nor an associated MVS dump 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 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 The Fault Analyzer ISPF interface on page 55 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.

Further, it lets you view the contents of storage areas included in the minidump and associated MVS™ dump 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 or PDSE data sets that contain information about faults that have been analyzed by Fault Analyzer. Fault entries, which are stored as separate members in the history file, contain the following type of information:

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

Each new fault entry in a history file is given an identifier, which is unique to that history file. The ID consists of a 1-3 character prefix, followed by a 5-digit sequence number. The default prefix is "F", but this default can be changed in one of the following ways:

- Via the Fault Analyzer ISPF interface. Click **File** > **Change Fault History File Settings** from the action-bar pull-down menu option. For more information, see Change fault history file settings on page 88.
- Via the IDIUTIL batch utility. For more information, see Managing history files (IDIUTIL utility) on page 441.

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.

You can set the minimum number of fault entries that are maintained in a PDSE history file, or the maximum number of fault entries that can be contained in a PDS history file:

- By using the Fault Analyzer ISPF interface. Click **File** > **Change Fault History File Settings** from the action-bar pull-down menu option. For more information, see Change fault history file settings on page 88.
- By using the IDIUTIL batch utility. For more information, see Managing history files (IDIUTIL utility) on page 441.

The minimum number of fault entries 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 129.) Setting this limit has no impact on the assignment of sequence numbers to new fault entries created. For PDSE history files, the default is to maintain a minimum of 25 fault entries. For PDS history files, the default is not to limit the number of fault entries.

Although a history file can be either a PDS or a PDSE data set, PDSE data sets are recommended because they provide concurrent member write capability which is not possible with PDS data sets. Fault Analyzer provides better sharing and performance when using PDSE history files, and allows the history file to be managed by the IDIS subsystem (see "Caching of history file $$INDEX data on page 319").

You can view fault entry details, browse the real-time analysis report, start batch and interactive fault reanalysis, or delete fault entries from an interactive display of a history file. These actions are described in The Fault Analyzer ISPF interface on page 55.

A view member can be created in a data set identified by the IDIVIEWS DDname that contain the names of any number of history files that you want to display simultaneously using the Fault Analyzer ISPF interface. For details, see Using views on page 59.

Associated dump data sets

There are two types of dump data sets that might be associated with a fault entry in a history file.

Tightly coupled

These are dump data sets that were created by Fault Analyzer during real-time processing and are uniquely linked with the fault entry. When the fault entry is deleted, the associated dump data set is also automatically deleted.

Examples of tightly coupled dump data sets are:

- Extended minidump data sets (XDUMP)
- IEATDUMP RFR data sets (TDUMP)
- SVC dump RFR or Java data sets (SDUMP)

Loosely coupled

These are dump data sets that are not uniquely linked with a fault entry. The fault entry was created as the result of analysis of an already existing dump data set. For example: an SVC dump created by setting a SLIP trap was selected for analysis using **File > Analyze MVS Dump Data Set** from the **Fault Entry List** display.

The same dump data set might be associated with more than one fault entry. When the fault entry is deleted, the dump data set is not deleted as it is assumed to be managed outside of Fault Analyzer.

Special members in the history file data set

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

$$INDEX

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

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

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

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

$$BACKUP

This member contains a copy of the history file specific settings that have been set in the $$INDEX member by using the IDIUTIL batch utility SetFaultPrefix, SetMaxFaultEntries or SetMinFaultEntries control statements (for details, see Managing history files (IDIUTIL utility) on page 441), or by using the Fault Analyzer ISPF interface (for details, see Change fault history file settings on page 88). If, for any reason, the $$INDEX member information is lost, then these settings are recovered from the $$BACKUP member.

Fault Analyzer supported application environments

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

- COBOL
- PL/I
- Assembler
- C/C++
- Language Environment
- UNIX® System Services
- CICS®
- IMS™
- Db2®
- MQSeries®
- Java™

Only execution in home-space mode, or primary-space mode with primary address space equal to home address space, is supported.

Fault Analyzer is formally supported only on current versions of z/OS, and the application environments listed in the preceding paragraphs. In the z/OS environment, Fault Analyzer:

- Executes in 31-bit addressing mode
- Performs analysis on 24-bit, 31-bit, or 64-bit addressing mode applications
- Supports multi-threaded, DLL, and XPLink applications
- Can run in any hardware environment that supports the current versions of z/OS.

Assembler, COBOL, and Enterprise PL/I are the only application programming languages for which formatting of associated storage areas and source line data field values is provided when the data resides in 64-bit storage.

C++ support does not provide any class information.

Real-time analysis is limited to tasks running in the following TCB protection keys:

- For CICS: key 8 or 9
- For non-CICS: key 8

Fault Analyzer issues the message IDI0123S on page 722 for tasks that run in any other protection key. To analyze abends in tasks that run in another protection key, set a SLIP trap on the IDI0123S message to capture an SVC dump. You can analyze the SVC dump by selecting **File > Analyze MVS Dump Data Set** from the **Fault Entry List** display.

Install the latest Fault Analyzer maintenance to resolve issues that might not be directly associated with the basic support for an application environment. For additional information, see Maintaining Fault Analyzer on page 459.

Fault Analyzer supports application environments with either:

- A general availability (GA) release with no maintenance applied.
- A GA release with the specified PTF (UI*nnnnn*) or APAR (PH*nnnnn*) applied.

The following table shows the minimum maintenance level to apply to Fault Analyzer Version 17.1 for each supported environment version.

Note: To get the most up-to-date information on supported operating systems, detailed system requirements, and hardware requirements, you can generate a dynamic software product compatibility report.

1. Go to the Software Product Compatibility Reports page.
2. Type the product name in the Search by product name field and click the product name in the list.
3. In the search results, select a release.

Table 1. Supported application environments

Supported environment	Supported environment version	Fault Analyzer 17.1 minimum maintenance level
CICS Transaction Server	V6R3 (760)	16.1.2 (UO05275)
	V6R2 (750) and earlier	GA
Enterprise COBOL	V6R5	16.1.2 (UO05275)
	V6R4 and earlier	GA
Enterprise PL/I	V6R2	16.1.4 (UO07796)
	V6R1 and earlier	GA
DB2	V13 and earlier	GA
IMS	V15 and earlier	GA
IBM Java for z/OS	Version 17 and earlier	GA
z/OS	V3R2	16.1.2 (UO05275)
	V3R1 and earlier	GA

Compiler support

With the minimum maintenance level applied, Fault Analyzer V17.1.0 produces the same results for a program compiled with a given environment version as it produced for the same program compiled with an earlier version of that environment.

The Fault Analyzer minimum maintenance level might not support all new features of a compiler version. Keep maintenance levels current to get the latest Fault Analyzer enhancements.

MQSeries support

MQSeries support consists of one of the following:

- Abends occurring in an MQSeries call

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

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

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

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

Java support

Java support has the following restrictions:

- Fault Analyzer works with currently supported versions of Java. Fault Analyzer might work successfully with earlier versions of Java, but they are not formally supported.
- Java support under z/OS 2.3 requires z/OS PTF UA96120. If this PTF is not applied, ABEND EC6 with reason code 0B26 C04A occurs during Java analysis.
- Fault Analyzer support for 31-bit C or C++ to 64-bit Java applications is limited due to lack of Language Environment (LE) condition handling support across the addressing mode boundary.

Binder-related dependencies

In order to map CSECTs in load modules, Fault Analyzer calls the IBM® binder program by means of its application programming interface (API). This call is generally done during real-time analysis, but might also be done during interactive analysis of CICS® system dumps. Because the binder supports load modules residing in PDS or PDSE data sets only, Fault Analyzer cannot 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. Nor do you need to recompile programs. However, if you store compiler listings or side files in the appropriate repository, then Fault Analyzer is able to identify the source statement of the abending program. (If you choose to not store listings or side files, you can still provide one after an abend has occurred. This approach makes it possible for Fault Analyzer to extract more information when you perform reanalysis.)

To provide a side file, you might need to recompile your programs, since appropriate side files are only produced when certain compiler options are requested. If you already have compiler listings that were produced with the correct compiler options, you can create side files without needing to compile again. The advantage of the side file is that it is more compact than a listing. For more information, see Providing compiler listings or Fault Analyzer side files on page 374.

Additional region size required

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

For detailed information about storage requirements, see Storage recommendations on page 296.

In situations where Fault Analyzer is unable to obtain sufficient storage for the real-time analysis of a fault, a Recovery Fault Recording fault entry is normally created (see Recovery fault recording on page 51). However, if Recovery Fault Recording has not been enabled, then a SYSMDUMP can be taken for subsequent batch or interactive reanalysis.

Compiler listing or side file selection criteria

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

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

To obtain detailed information about why a particular compiler listing or side file was selected or rejected, the IDITRACE facility can be used. See IDITRACE information on page 383 for details on how to use this facility.

Special processing of Language Environment CEEWUCHA abends

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

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

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

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

WTO routing and descriptor codes used by Fault Analyzer

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

CICS Storage Accounting Area (SAA) overlay assistance

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

- Concurrent CICS® task storage information

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

  For details, see CICS Transaction Storage Summary on page 177.

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

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

  For details, see Implementing an XEIIN global user exit on page 412.

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 298), 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 35).
- A fault entry in a history file, which provides the ability to perform reanalysis of the fault.

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

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

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

All virtual storage pages that were referenced during the analysis in the abending task's address space are written to the history file as a minidump, or to an associated XDUMP, depending on the storage content. If the number of storage pages destined for the minidump exceeds the MaxMinidumpPages option in effect, then neither a minidump nor an XDUMP is written.

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

Dump suppression

The types of dumps that can be suppressed are:

- SYSABEND, SYSUDUMP, or SYSMDUMP (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 Customizing the CICS environment on page 396).

Suppression of dumps upon *successful completion* of analysis is the default. (Successful completion is defined as Fault Analyzer having reached the normal end of analysis, without having abended or issued any S-level messages.) However, when using the following statement under CICS®, the transaction dump is only suppressed if this was requested before Fault Analyzer was invoked, for example by an earlier exit:

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

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 on page 33 for more information on transaction dump suppression and CICS® global user exits.

To override the default dump suppression, use either:

- The RetainCICSDump(ALL) option for CICS® transaction faults (see RetainCICSDump on page 631).
- The RetainDump(ALL) option for non-CICS transaction faults (see RetainDump on page 632).
- An End Processing user exit (see End Processing user exit on page 494).

Dump suppression should not be confused with suppression of analysis or suppression of fault entry creation (see Real-time exclusion processing on page 47).

Fault Analyzer and CICS global user exits

Fault Analyzer can be installed in either or both of the CICS® global user exits (GLUE) - XPCABND and XDUREQ. Installation in both exits is recommended. If other products are installed in the XDUREQ exit, then Fault Analyzer must also be installed there. This is because, by default, a successful abend analysis by Fault Analyzer through the XPCABND exit suppresses the CICS® transaction dump if the Fault Analyzer is not installed in XDUREQ exit. If the dump is suppressed from XPCABND, CICS® does not invoke any of the XDUREQ exits. When Fault Analyzer is installed in both XPCABND and XDUREQ, it defers dump suppression to XDUREQ, allowing CICS® will invoke XDUREQ exits.

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 631 for more details.

Fault history file selection

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

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

1. The product default name, IDI.HIST.
2. The IDIHIST suboption of a DataSets option that is specified in the parmlib config member, IDICNFxx. This information includes either the logical parmlib concatenation or the installation-wide alternate parmlib data set name provided via the IDIOPTLM configuration-options module CNFDSN setting. For details, see Customize Fault Analyzer by using an IDIOPTLM configuration-options module on page 337.
3. The IDIHIST suboption of a DataSets option that is specified in a config member identified via the IDICNFUM user options module.

Note: If a user options module is used, it replaces the default IDICNFxx 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 IDICNFxx parmlib config member would not be recognized.

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

Controlling the real-time analysis with options

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

See Options on page 571 for information about all available options and the different ways in which they can be specified.

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

RetainDump(ALL)

Specify this option if you want to want to retain the SYSABEND, SYSUDUMP, or SYSMDUMP unconditionally. Without this option, many dumps are suppressed when Fault Analyzer deems that the analysis it has performed is adequate. This option does not affect the writing of the minidump to the history file. See RetainDump on page 632 for more details.

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

Detail

Specify this option if you want to adjust the level of detail given in the real-time analysis report. (If a dump is produced, you can change this option when you perform a reanalysis.) See Detail on page 591 for more detail.

Exclude

Specify this option if you want to exclude this job from analysis. See Exclude/Include on page 595 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 616 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 578 for more detail.

You can also use the DataSets option to point to listings and side files. See DataSets on page 581 for more detail.

JCL DD statements

No DD statements are required to run Fault Analyzer in either batch or real time.

Pointing to listings with JCL DD statements

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

IDILC

PDS or PDSE data set containing C compiler listings

IDILCOB

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

IDILCOBO

PDS or PDSE data set containing OS/VS COBOL compiler listings

IDISYSDB

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

IDILPLI

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

IDILPLIE

PDS or PDSE data set containing Enterprise PL/I compiler listings

IDIADATA

PDS or PDSE data set containing SYSADATA from Assembler compilations

IDILANGX

PDS or PDSE data set containing LANGX side files for all languages

Do not specify a member name on any of the above DD statements. See Providing compiler listings or Fault Analyzer side files on page 374 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.faultanalyzer.Snap.dump Java™ class, unless the DeferredReport option is used (see DeferredReport on page 589) or the report is suppressed (see Suppressing real-time reports on page 37). The report is written to the IDIREPRT DDname, which is dynamically allocated to SYSOUT=*class*, if no prior allocation exists, and thus is included as part of the normal job output on the JES spool.

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

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

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

Alternatively, a user exit can be used to allocate IDIREPRT to a different output class. For details, see Controlling the SYSOUT class of real-time reports on page 36.

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 The Fault Analyzer report on page 272 for general information about the contents of the Fault Analyzer report, and for report examples.

Combining Fault Analyzer real-time reports

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

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

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

Controlling the SYSOUT class of real-time reports

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

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

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

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

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

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

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

In a CICS® environment, it is preferable to use the DeferredReport option instead, which is also the default. For details, see DeferredReport on page 589.

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 298. This interface permits a call to Fault Analyzer from anywhere within an application program to request an analysis of the current environment. For 24-bit and 31-bit programs, the program SNAP interface module name is IDISNAP. For 64-bit programs, the module name is IDISNAP6.

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.

This exit can extract WTO console messages related to the current job from the main trace table and include these messages in the analysis report, except when running under CICS.

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

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

IDISNAP invocation

The following describes the entry and return specifications for IDISNAP.

Entry specifications

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

Register

Contents

1

Zero, or address of input parameter list (see below).

13

Address of 72-byte register save area.

14

Return address.

15

Entry point address of IDISNAP.

Input parameter list

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

Table 2. IDISNAP input parameters

Parameter	Number of bytes	Description
Parameter 1	4	Specifies the number and type of optional parameters that follow, in the format: `tnnn` where: ***t*** 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. ***nnn*** Specifies the number of parameters that follow parameter 1 as either: **000** Only parameter 1 is specified. **001** Parameters 1 and 2 are specified. **002** Parameters 1, 2, and 3 are specified. **00V** Parameters 1, 2, and 3 are specified, as well as one or more storage range address pairs in subsequent parameters. **Note:** Specifying 0000 in this parameter is equivalent to invoking IDISNAP with register 1 set to zero.
Parameter 2	140	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.

Table 2. IDISNAP input parameters

(continued)

Parameter	Number of bytes	Description
		The specified user title is added to the heading of the real-time report, and can be viewed and updated from the Fault Analyzer ISPF interface Fault Entry List display. Parameter 1 must be 0001 or 0002 in order for this parameter to be processed.
Parameter 3	140 if parameter 1 is 0002, or varying if parameter 1 is N002.	Fault Analyzer options. For example: • To prevent the creation of a history file fault entry, specify the DATASETS(IDIHIST(NULLFILE)) option. • To request that specific areas of storage are to be shown in the analysis report, specify the StorageRange option (see StorageRange on page 635). • To request the return of ENV data area information, specify the SNAPDATA option (see Using the SNAPDATA option on page 41). Parameter 1 must be either 0002, 000V, N002, or N00V in order for this parameter to be processed: • If parameter 1 is 0002, then the area pointed to by this parameter must be 140 bytes, with any unused portion set to blanks. • 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').
The following parameters are for pairs of storage range begin and end addresses. The maximum number of address ranges that can be specified is 160. Using the address range parameters is an alternative to, and overrides, specification of the StorageRange option in parameter 3. To use these extra parameters, parameter 1 must be either 000V or N00V: • If parameter 1 is 000V, then the area pointed to by parameter 2 must be 140 bytes, with any unused portion set to blanks. • If parameter 1 is N00V, then the area pointed to by parameter 2 must be 1 - 1024 bytes, with a null-character (X'00') immediately following the last character in the options string.		
Parameter *n*	4	Storage range begin address.

Table 2. IDISNAP input parameters

(continued)

Parameter	Number of bytes	Description
Parameter *n*+1	4	Storage range end address.

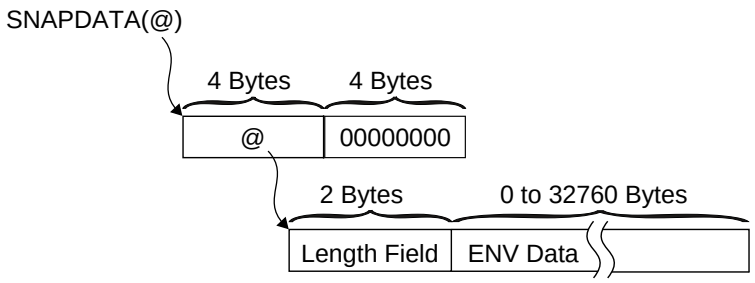
Options that are specified in parameter 3, or implicitly via the storage range address pairs starting with parameter 4, are passed as PARM field options to the main Fault Analyzer analysis module, IDIDA.

Using the SNAPDATA option

IDISNAP also processes the SNAPDATA option (see Snapdata on page 633). SNAPDATA specifies a storage address, in an 8-byte character format, that points to a list of pointers. The list is terminated by a zero address field.

Figure 6: SNAPDATA parameter list on page 41 shows an example of the SNAPDATA parameter list.

Figure 6. SNAPDATA parameter list



Only the first pointer in the parameter list is currently used. It must be the address of a two-byte input length field followed by a buffer where IDISNAP can store the final version of the common exit environment information in the ENV data area after all exits have run. A zero length field causes Fault Analyzer to ignore the parameter. A length field shorter than the length of the ENV data area is permitted.

Review the sample members IDISNAPB (COBOL) and IDISNAPP (PL/1) for examples of how to use this feature.

Return specifications

On return from IDISNAP, the contents of registers are:

Register

Contents

0-1

Undefined.

2-14

Unchanged.

15

Zero.

Example 1 (COBOL)

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

```
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 (COBOL: 64-bit DLL)

The following is an example of a 64-bit COBOL program calling IDISNAP as a DLL.

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. 'IDISNAPC'
       ENVIRONMENT DIVISION.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
       DATA DIVISION.
       FILE SECTION.
       WORKING-STORAGE SECTION.
       77  STORAGE-AMODE31-PTR USAGE IS POINTER-32.
       LINKAGE SECTION.
       01 STORAGE-AMODE31.
         03 PTRPARM USAGE IS POINTER-32.
         03 PTRS.
           04 PTR-PARM-1 USAGE IS POINTER-32.
           04 PTR-LAST USAGE IS POINTER-32.
         03 WORK PIC X(4) .
         03 WORK_TITLE PIC X(140).
         03 WORK_PARM.
           04 WORKP_1 PIC X(9).
           04 WORKP_2 PIC X(8).
           04 WORKP_3 PIC X(2).
           04 WORKP_4 PIC X(122).
         03 WORK_RANGE.
           04 R1S     PIC X(4).
           04 R1E     PIC X(4).
         03 PARM-1.
          05 PARM-1-L PIC 9(4) COMP.
          05 ENV.
      * ******************************************
      * THIS EXAMPLE RETRIEVES THE FAULT-ID AND    *
      * IDIHIST VALUES BUT IS NOT INTERESTED IN    *
      * ANYTHING ELSE.                             *
      * THEREFORE IT PROVIDES A "SHORT" DATA AREA. *
      * ******************************************

             07 VERSION                      PIC X(4).
             07 EXIT-CALL-TYPE               PIC X(1).
             07 FAULT-ID                     PIC X(8).
             07 ABEND-DATE                   PIC X(10).
             07 ABEND-TIME                   PIC X(8).
             07 REALTIME                     PIC X(1).
             07 SYSTEM-NAME                  PIC X(8).
             07 JOB-NAME                     PIC X(8).
             07 EXEC-PGM-NAME                PIC X(8).
             07 USER-ID                      PIC X(8).
             07 RESERVED-1                   PIC X(4).
             07 ABEND-MODULE-NAME            PIC X(8).
             07 CICS-TRANSACTION-ID          PIC X(4).
             07 CICS-TASK-NUMBER             PIC X(5).
             07 JOB-TYPE                     PIC X(1).
             07 JOB-CLASS                    PIC X(1).
             07 ACCOUNTING-FIELDS            PIC X(3).
             07 ACCOUNTING-INFO              PIC X(144).
             07 USER-1                       PIC X(4).
```

```
            07 USER-2                          PIC X(4).
            07 RESERVED-2                      PIC X(1).
            07 LOOPPROTECTION-OPT              PIC X(1).
            07 RESERVED-3                      PIC X(4).
            07 WRITE-ROUTINE-EP              USAGE IS PROCEDURE-POINTER.
            07 INVOCATION-EXIT                 PIC X(1).
            07 STEP-NAME                       PIC X(8).
            07 JOB-ID                          PIC X(8).
            07 IMS-PROGRAM-NAME                PIC X(8).
            07 USER-NAME                       PIC X(8).
            07 USER-TITLE                      PIC X(40).
            07 APPLID                          PIC X(8).
            07 TERMID                          PIC X(4).
            07 NETNAME                         PIC X(8).
            07 TCB-ADDRESS                     PIC X(8).
            07 CSA-ADDRESS                     PIC X(8).
            07 TCA-ADDRESS                     PIC X(8).
            07 IDIHIST                         PIC X(44).

      PROCEDURE DIVISION.
      MAIN SECTION.
      START000.
           ALLOCATE STORAGE-AMODE31 LOC 31
             RETURNING STORAGE-AMODE31-PTR.
           SET ADDRESS OF STORAGE-AMODE31 TO STORAGE-AMODE31-PTR.
           MOVE LENGTH OF ENV TO PARM-1-L.
           SET PTR-PARM-1 TO ADDRESS OF PARM-1.
           SET PTR-LAST TO NULL.
           SET PTRPARM TO ADDRESS OF PTRS.
           MOVE "0002" TO WORK.
           MOVE "TEST TITLE" TO WORK_TITLE.
           MOVE FUNCTION HEX-OF(PTRPARM) TO WORKP_2.
           MOVE 'SNAPDATA(' TO WORKP_1.
           MOVE ') ' TO WORKP_3.
           MOVE 'DATASETS(IDIHIST(MY.TEST.HIST))' TO WORKP_4.
           DISPLAY "STARTING" UPON SYSOUT.
           CALL "IDISNPD"   USING                WORK
                                                 WORK_TITLE
                                                 WORK_PARM.
           DISPLAY "FAULTID: " FAULT-ID OF ENV UPON SYSOUT.
           DISPLAY "HISTDSN: " IDIHIST OF ENV UPON SYSOUT.
           DISPLAY "STOPPING" UPON SYSOUT.
           STOP RUN.
          EXIT.
      END PROGRAM 'IDISNAPC'.
```

Note: This 64-bit DLL supports only the first three parameters; the storage range parameter is ignored.

This example is provided as sample job IDIVPCB6 in data set IDI.SIDISAM1.

Example 3 (PL/I: non-DLL)

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

```
*PROCESS COMPILE,ATTRIBUTES,AGGREGATE,MAP,LIST,ESD,NEST;
 @960IDI:PROC OPTIONS(MAIN) REORDER;
```

```
   DCL  WKPTR              PTR ;
   DCL  WORK               CHAR(4) INIT('0001') ;
   DCL  WORK140            CHAR(140) INIT(' ');
   DCL  WORK1402           CHAR(140) INIT(' ');
   DCL  NUMWK              FIXED DEC(9) INIT(0)  ;
   DCL  NUMWK2             FIXED DEC(9) INIT(0)  ;
   DCL  PICWK              PIC'999' INIT(0);
   DCL  IDISNAP EXTERNAL ENTRY OPTIONS(ASM);
/* 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);
   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 4 (PL/I: 31-bit DLL)

The following is an example of a 31-bit PL/I program calling IDISNAP as a DLL.

```
*PROCESS AGGREGATE,ATTRIBUTES(FULL),LIST,MAP,NEST,SOURCE,STMT,
         NONUMBER,OFFSET,XREF(FULL),OPTIONS,NOBLKOFF,RENT;
*PROCESS LIMITS(EXTNAME(8));
 IDISPLI1: PROC                OPTIONS(MAIN) ;

  DECLARE
    IDISNPD     External Entry Options(asm),
    Work        Char(4),
    Work140     Char(140),
    Work1402    Char(140);

  Work ='0002';
  Work140 = 'User Title Data';
  Work1402 = 'Datasets(IDIHIST(MY.HIST))';
  Call IDISNPD(Work,Work140,Work1402);

 End IDISPLI1;
```

This example is provided as sample job IDIVPLS1 in data set IDI.SIDISAM1.

Example 5 (PL/I: 64-bit DLL)

The following is an example of a 64-bit PL/I program calling IDISNAP as a DLL.

```
*PROCESS AGGREGATE,ATTRIBUTES(FULL),LIST,MAP,NEST,SOURCE,STMT,
         NONUMBER,OFFSET,XREF(FULL),OPTIONS,NOBLKOFF,RENT;
*PROCESS LIMITS(EXTNAME(8)),LP(64);
 IDISPLI2: PROC                OPTIONS(MAIN) ;

  DECLARE
    IDISNPD     External Entry Options(asm),
    plist_pointer  pointer,
    1 snap_list based(plist_pointer),
    3 args_list(3) pointer(32),
      3 Work        Char(4),
      3 Work140     Char(140),
      3 Work1402    Char(140),
    lastargflag bit(1) based;

  plist_pointer = alloc31(stg(snap_list));
  Work ='0002';
  Work140 = 'User Title Data';
  Work1402 = 'Datasets(IDIHIST(MY.HIST))';
  args_list(1) = addr(work);
  args_list(2) = addr(work140);
  args_list(3) = addr(work1402);
  addr( args_list(3 ))->lastargflag = '1'b;

  Call IDISNPD(args_list);

  Call plifree(plist_pointer);
 End IDISPLI2;
```

This example is provided as sample job IDIVPLS2 in data set IDI.SIDISAM1.

Example 6 (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
```

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 MVS™ post-dump exit IDIXTSEL. SVC dumps occur for system abends, and are also used by CICS® for its system dumps.

If you install the MVS™ post-dump exit IDIXTSEL, a "skeleton" fault entry is created whenever an SVC dump is written. For more information about installing the MVS™ post-dump exit IDIXTSEL, see Installing the MVS post-dump exit IDIXTSEL on page 412. This processing differs from normal real-time processing in that no analysis is performed, and therefore no report or minidump is produced. This Fault Analyzer process is known as "dump registration".

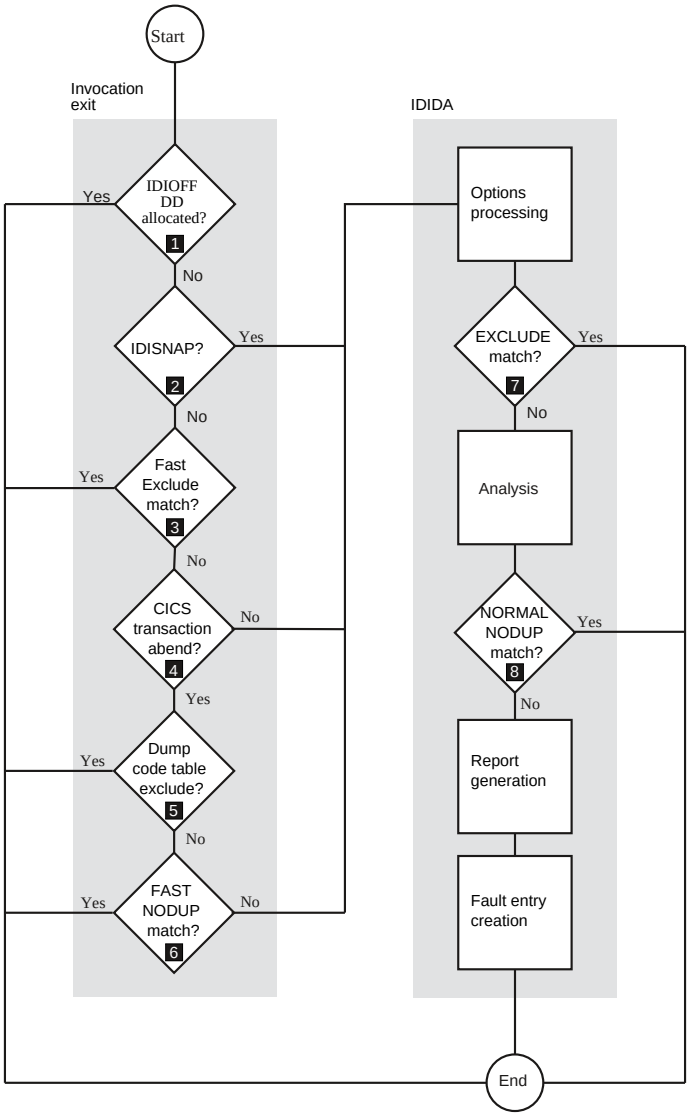
The dump registration processing permits the use of two user exits which effectively are the equivalent of the normal Analysis Control and Notification user exits. These are specified using the DumpRegistrationExits option (see DumpRegistrationExits on page 593).

The dump registration fault entry contains only limited information, such as the time of its creation, the system name, and the name of the job that caused the SVC dump to be written. If available, the abend code and abending program name is also provided. However, the first reanalysis of the dump registration fault entry refreshes the fault entry and save a report and minidump with it. For details, see Refresh processing on page 240.

Real-time exclusion processing

While real-time analysis is never performed if the primary subsystem (JES) is unavailable, there are a number of ways to selectively exclude various elements of the Fault Analyzer processing, as illustrated in Figure 7: Real-time exclusion processing overview on page 48.

Figure 7. Real-time exclusion processing overview



Note:

❶

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

```
//IDIOFF   DD      DUMMY
```

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

Notice: In the z/OS Unix System Services environment, setting the environment variable _IDI_OFF to "Y" is equivalent to using the IDIOFF DDname switch. For more information, see Turning off Fault Analyzer using an environment variable (_IDI_OFF) on page 463.

❷

If calling IDISNAP from within your application (see Using the program SNAP interface (IDISNAP) on page 37), then no further exclusions are available prior to options processing being performed.

❸

If fast Exclude options processing is enabled, and the job is eligible (see Fast Exclude options processing on page 372), then a matching Exclude option (see Exclude/Include on page 595) can be used to terminate Fault Analyzer processing early.

❹

Additional invocation exit exclusions are provided for CICS® transaction abends only.

❺

If the CICSDumpTableExclude option is in effect (see CICSDumpTableExclude on page 578), and the CICS® transaction abend that is associated with the fault is specified in the CICS® transaction dump code table to not require a CICS® dump, no further processing is performed. That is, no analysis report is produced and no fault entry is written.

❻

If either of the following is true, 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 616.

❼

The EXCLUDE option (see Exclude/Include on page 595) can be used to terminate Fault Analyzer processing once the options have been read in the mainline code.

❽

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 616), then processing terminates for the current fault. The real-time report is written to IDIREPRT, but no history file fault entry is created.

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

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

Duplicate fault processing

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

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

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

The different types of duplicate processing are controlled using the NoDup option (for details, see NoDup on page 616). 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 56).

Recovery fault recording

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

- Insufficient storage. (See IDI0005S on page 696.)
- Fault Analyzer abended. (See IDI0047S on page 705.)
- Fault Analyzer timed out. (See IDI0092S on page 715.)
- Invalid negative storage length request. (See IDI0105S on page 718.)

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

First a check is made to see if security access is granted to use SDUMP as the recovery fault recording dump type, since this type is the preferred dump type for performance reasons. (For details, see Using the XFACILIT resource class for SDUMP RFR data sets on page 310.)

Note: The MVS™ post-dump exit IDIXTSEL is required to support the recovery fault recording feature when using SDUMPs. For details of this exit, see SVC dump registration on page 302.

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

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

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

Note: To enable recovery fault recording processing, the IDIS subsystem must be started and the `UPDINDEX` parameter must be in effect (this is the default).

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

Message IDI0126I on page 723 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 498.

If the RFR dump is an IEATDUMP, then the default data set name pattern used is:

```
IDIRFRHQ.IDIRFR.&SYSNAME..D&YYMMDD..T&HHMMSS..S&SEQ..
```

To change this name (or to disable IEATDUMP RFR dumps), see Changing the default recovery fault recording IEATDUMP data set name (RFRDSN) on page 339. 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 309 for more information.

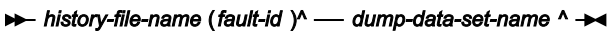
Depending on where in the real-time analysis process the problem occurred, reanalysis of the recovery fault recording fault entry is capable of producing a reanalysis report, which is effectively identical to the one that would have been produced if the real-time analysis had completed normally. The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

When a recovery fault recording fault entry is deleted, then the associated RFR dump data set is also automatically deleted to ensure that these data sets are not taking up disk space unnecessarily. Failure to delete the RFR dump data set results in message IDI0187I on page 737.

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

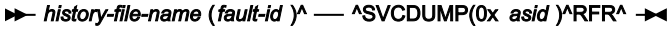


Note: In the above, ^ represents the non-printable character X'00'.

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

```
MY.HIST(F32752).                                IDIRFRHQ.TDUMP.FAE1.D110216.T003630.S00405.
```

SVCDUMP dump title



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

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

```
MY.HIST(F32753).                                .SVCDUMP(0x0080).RFR.
```

Using SLIP,COMP=0C4 with Fault Analyzer

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

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

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

Extended minidump data set (XDUMP)

XDUMPs are enabled by default using the data set name pattern:

```
IDIXDPHQ.XDUMP.&SYSNAME..D&YYMMDD..T&HHMMSS..S&SEQ.
```

To change this data set name pattern, see "Specifying the extended minidump data set name pattern (XDUMPDSN) on page 340". Fault Analyzer allocates the data set with RECFM=FBS LRECL=4096 and uses the access authorization mechanism described in Managing XDUMP data set access on page 313. If XDUMP data set allocation fails, Fault Analyzer issues IDI0184W on page 737.

An extended minidump data set (XDUMP) is a Fault Analyzer specific data set that is sometimes created to hold certain types of storage pages and which is tightly coupled with a fault entry. The usage of the XDUMP data set, and its final disposal, is normally transparent to the user.

An XDUMP data set is designed to contain:

- 64-bit storage pages
- Application storage that is not directly involved with essential aspects of the analysis, such as program linkage or the point of failure
- "Spilled" minidump pages

"Spilled" minidump pages require an XDUMP data set to save 64-bit storage pages and associate them with a fault entry. 64-bit storage pages are not included in a minidump.

For 24-bit and 31-bit applications, an XDUMP data set can provide relief from re-analysis storage constraints.

If the MaxMinidumpPages option value in effect is exceeded during real time processing, the number of pages in excess of the MaxMinidumpPages value are written ("spilled") to the XDUMP. This prevents the issue otherwise resulting in message IDI0133W on page 725. However, it also creates a dependency on the XDUMP data set, which must continue to exist for correct reanalysis to occur. This is generally not an issue since the XDUMP should only be deleted automatically by Fault Analyzer when the fault entry is deleted.

If pages are spilled to the XDUMP data set during real time analysis, IDI0194W on page 739 is issued.

If an XDUMP data set containing spilled pages is not available during reanalysis, warnings will be issued. For batch reanalysis, message IDI0195W on page 739 is also issued.

XDUMPs are enabled by default. To test that XDUMPs have been configured correctly, see Verifying the use of XDUMPs on page 428.

Abend traps can prevent fault analysis

Using abend trapping methods in a program (such as ESTAE/ESPIE routines or EXEC CICS HANDLE routines in CICS) can prevent Fault Analyzer from gaining control after an abend occurs. As a result, Fault Analyzer cannot analyze the abend.

Capturing and displaying the abending job JCL

When the JCLcapture option is in effect during real-time processing, Fault Analyzer attempts to obtain the JCL and save it in the fault entry. This feature is available with JES2 only.

See JclCapture on page 612.

To retrieve the saved JCL:

- Use the JCL command from within the interactive reanalysis report. See JCL on page 106.
- Select the "Abend Job JCL" point-and-shoot field from the "Abend Job Information" section of the interactive reanalysis report.
- Use the IDIGET command from within a REXX Formatting user exit to retrieve the saved _JCL stem. For example:

```
/* REXX */
"IDIGET _JCL"
if rc = 0 then do
  "IDIWRITE '<L>Abend job JCL:'"
  "IDIWRITE '<L>'"
  do i = 1 to _jcl.0
    "IDIWRITE '<L>"_jcl.i"'"
  end
end
else do
  "IDIWRITE '<L>JCL not available'"
end
exit 0
```

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 Performing batch reanalysis on page 149).
- Start an interactive reanalysis (for details, see Performing interactive reanalysis on page 157).
- 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 Performing CICS system abend dump analysis on page 247).
- Analyze Java™ dumps (for details, see Performing Java analysis on page 259).

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 284.

Reanalysis:

You can only perform reanalysis of a fault entry if it contains a minidump or is associated with an MVS dump data set.

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

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

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

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

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

If you want to supply a listing or side file so that Fault Analyzer can provide source line information when it performs the fault reanalysis, you must compile the program and then store the compiler listing or side file. For more information on this process, see Providing compiler listings or Fault Analyzer side files on page 374.

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 35 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 Modifying your ISPF environment on page 327.

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 session.
  - Only user notes from the analysis that is ended last are saved.
- When user notes recovery is enabled, the same TSO/ISPF user cannot perform interactive reanalysis of the same fault entry or the same MVS dump data set in two separate ISPF sessions simultaneously.
- The last used history file and fault entry lists available from the "File" pull-down menu of the Fault Entry List display are maintained for each ISPF session separately. The last Fault Analyzer ISPF interface session, during which a change was made to this information, that is ended, updates the information in the user's ISPF profile.

The Fault Entry List display

The Fault Entry List display is shown when the Fault Analyzer ISPF interface is started. Figure 8: Sample Fault Entry List display on page 57 shows an example of a Fault Entry List display:

Figure 8. Sample 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  : '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), P (Package), L (Lock), U (Unlock), A (Display source), J (Edit JCL).}

   Fault ID Job/Tran User ID  Sys/Job  Abend Date       Time
__   F00323 IDIVPCOB IBMUSER  MVS2     S0C7  2019/12/21 13:02:25
__   F00445 ALLANT01 JACKIED  MVS8     S0C7  2019/12/19 03:29:57
__   F00444 ALLANT01 JACKIED  MVS8     S0C7  2019/11/28 20:25:30
__   F00442 ALLANT01 ALLANT   MVS8     S0C7  2019/09/10 22:20:10
__   F00349 CS05     CICSUSER CSCB0050 ASRA  2019/08/23 07:47:23
__   F00348 CS04     CICSUSER CSCB0040 ASRA  2019/08/23 07:46:36
__   F00345 CS01     CICSUSER CSCB0010 AEIL  2019/08/23 07:43:35
__   F00050 PSTRANDR PSTRAND  STPLEX4B S0C4  2019/08/02 17:03:18
__   F00035 CICS53   n/a      MVS2     n/a    2019/04/05 14:49:11
 F1=Help      F3=Exit      F4=MatchCSR  F5=RptFind   F6=Actions   F7=Up
 F8=Down     F10=Left     F11=Right    F12=MatchALL
```

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

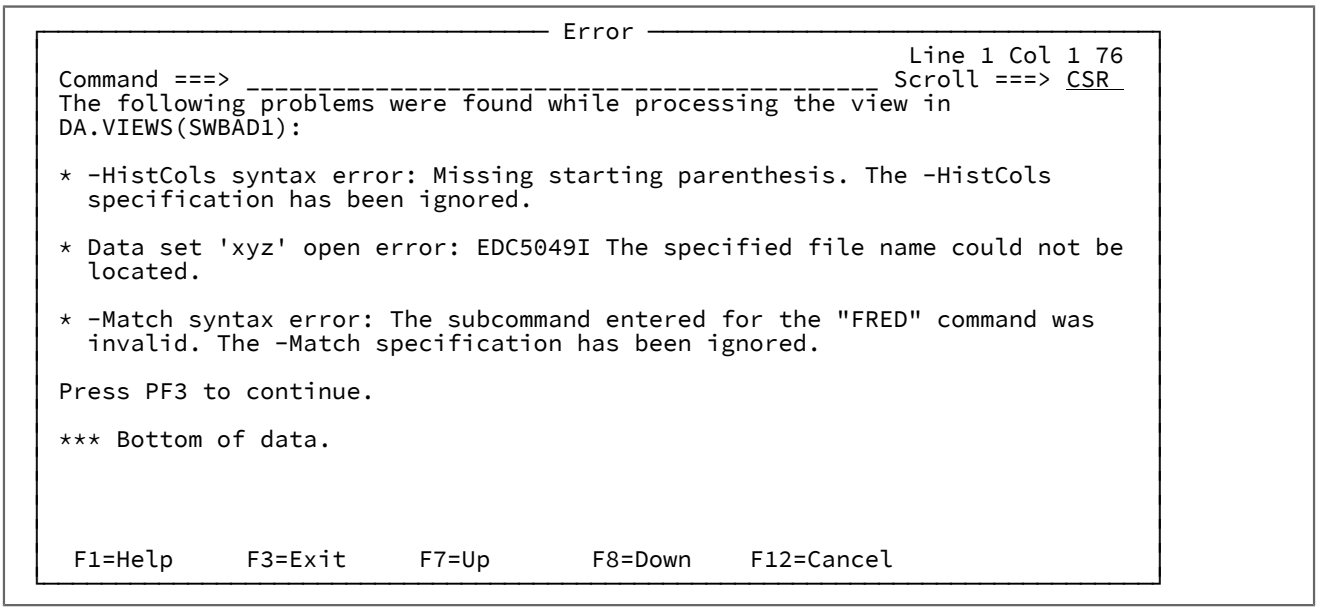
```
FKA ON
```

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

The history file or view (see Using views on page 59) 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 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 59.

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 9: Sample Error display on page 58.

Figure 9. Sample Error display



To exit from the error display, press PF3.

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

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

Each fault entry in the list occupies a single line and is identified by a fault ID on the left side of the display. Other information that might be displayed for each fault entry can be determined by the user. For details on this, refer to Fault entry list column configuration on page 65. The default information displayed, when no HistCols option is specified and no customization is made by the user, is as shown in Figure 8: Sample Fault Entry List display on page 57.

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 71.

As shown at the top of the display if help text is enabled (as illustrated in Figure 8: Sample Fault Entry List display on page 57), 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 81. For information about how to show or hide help text, see Adding or removing help text on page 122.

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

Optional help text that lists the available line commands is displayed only when the top-most line of the display is shown. If the display is scrolled down any number of lines, this help text disappears, but reappears again if the display is scrolled to the top. For general information about help text, see Adding or removing help text on page 122.

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

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

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

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

```
*** Bottom of data.
```

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

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

Using views

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

- A view name can be specified instead of a history file name on the Fault Entry List display.
- A view name can be selected via the File menu List Views option. For information about selecting this option in general, refer to Action-bar pull-down menus on page 94.

The definition of these views must be set up in the PDS or PDSE data set identified by the IDIVIEWS suboption of the DataSets option in the IDICNFxx parmlib config member.

With views you can:

- View fault entries from multiple history files simultaneously.
- Provide a specific column layout for the Fault Entry List display (see Specifying a default column layout on page 348).
- Provide a selection criteria for the initially displayed list of fault entries (see Specifying initial fault entry selection criteria on page 349).

For information about how to set up views, see Setting up views on page 347.

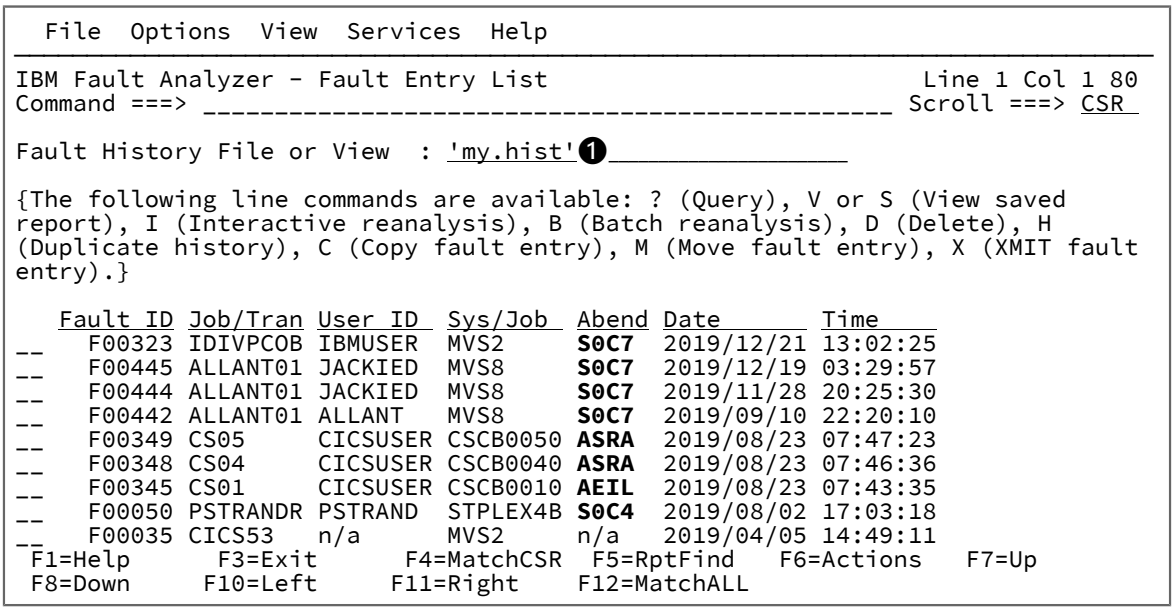
Changing the history file or view displayed

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

Type a different history file or view name

To specify a history file or view name to be displayed, type its name on the *"Fault History File or View"* line as the example shown at ❶ in Figure 10: Typing a different history file or view name on page 60 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.

Figure 10. Typing a different history file or view name



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

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

  If missing, the ending quote is automatically added.

- View names are member names in one of the data sets that is associated with the IDIVIEWS DDname. These are specified by enclosing them in parenthesis. For example, to specify that the view member ABC is to be displayed, type `(ABC)` on the *"Fault history file or view"* line.

  If missing, the closing parenthesis is automatically added.

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

*

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

******

A double asterisk indicates that zero or more qualifiers can occupy that position. A double asterisk within a qualifier is invalid.

%

A single percent sign indicates that any one single alphanumeric or national character can occupy that position.

%%...

One to eight percent signs can be specified in each qualifier.

The following examples are valid history file patterns:

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

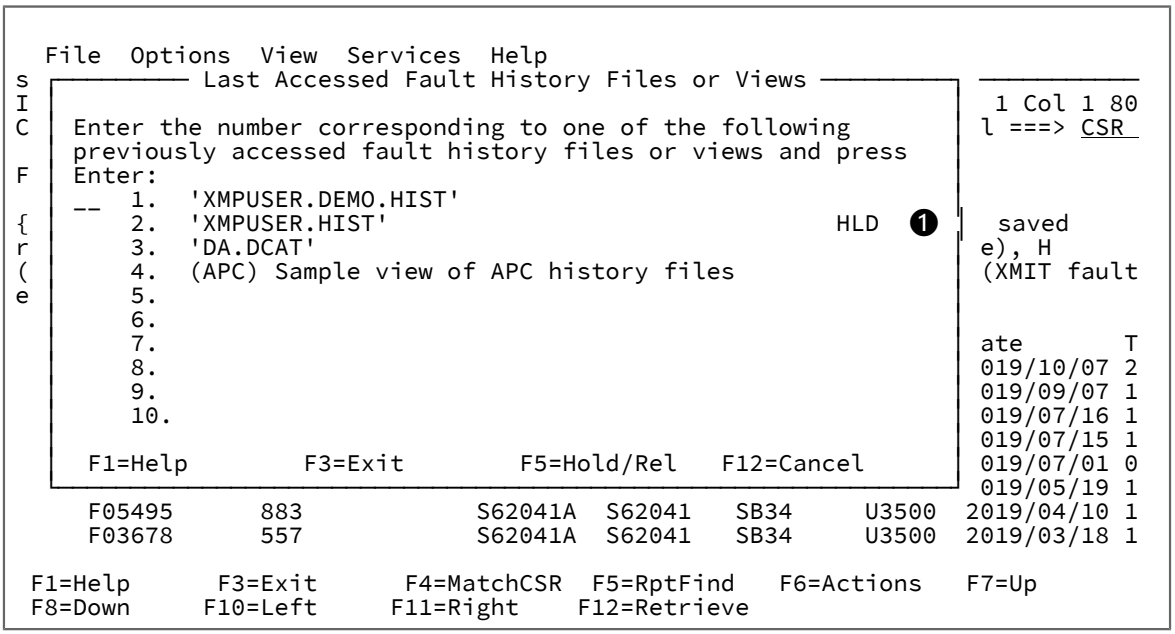
The rules for using quotes around history file names also apply to history file patterns.

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

Select a previously used history file or view

A record is maintained of the last 10 history files or views displayed. To select a previously displayed history file or view, first select the File menu Last Accessed Fault History Files or Views option (for information about selecting this option in general, refer to Action-bar pull-down menus on page 94). This opens the Last Accessed Fault History Files or Views display as the example shown in Figure 11: Sample Last Accessed Fault History Files or Views display on page 62.

Figure 11. Sample Last Accessed Fault History Files or Views display



From the Last Accessed Fault History Files or Views display shown in Figure 11: Sample Last Accessed Fault History Files or Views display on page 62, type the number corresponding to the desired history file or view name at the initial cursor position and press Enter to display the entries for the selected history file or view.

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

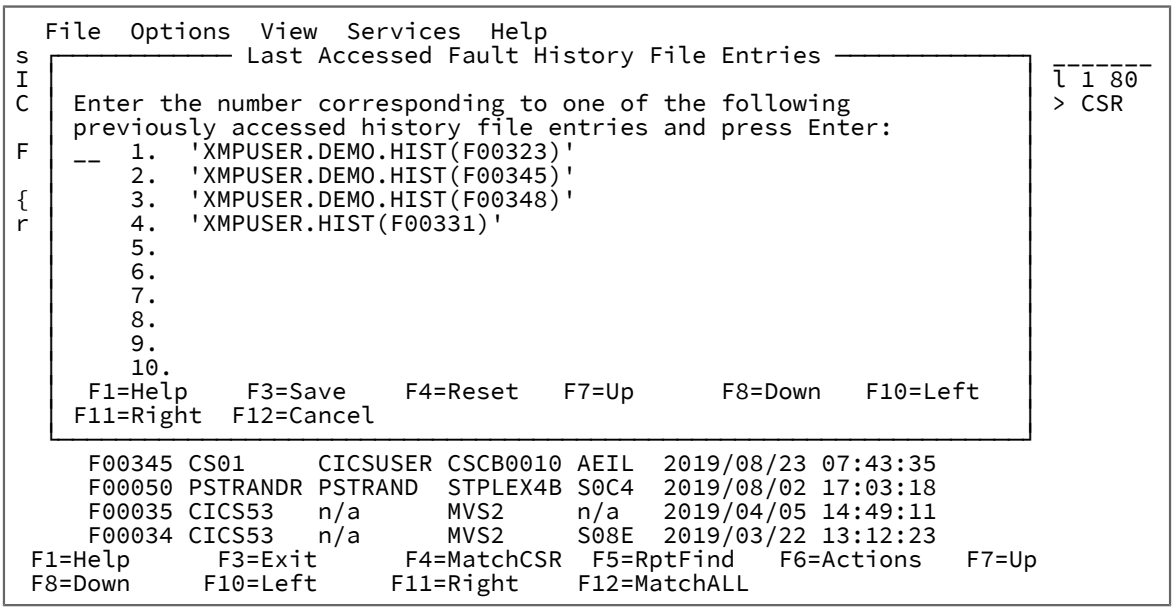
To prevent a history file name from dropping off the list, enter the number of the list item in the input field and press PF5. The "HLD" indicator will be shown on the right hand side of the entry, see ❶ in the above example.

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

Select a previously used fault entry

A record is maintained of the last 10 history file entries used. To select a previously displayed fault 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 94). This opens the Last Accessed Fault History File Entries display as the example shown in Figure 12: Sample Last Accessed Fault History File Entries display on page 63.

Figure 12. Sample Last Accessed Fault History File Entries display



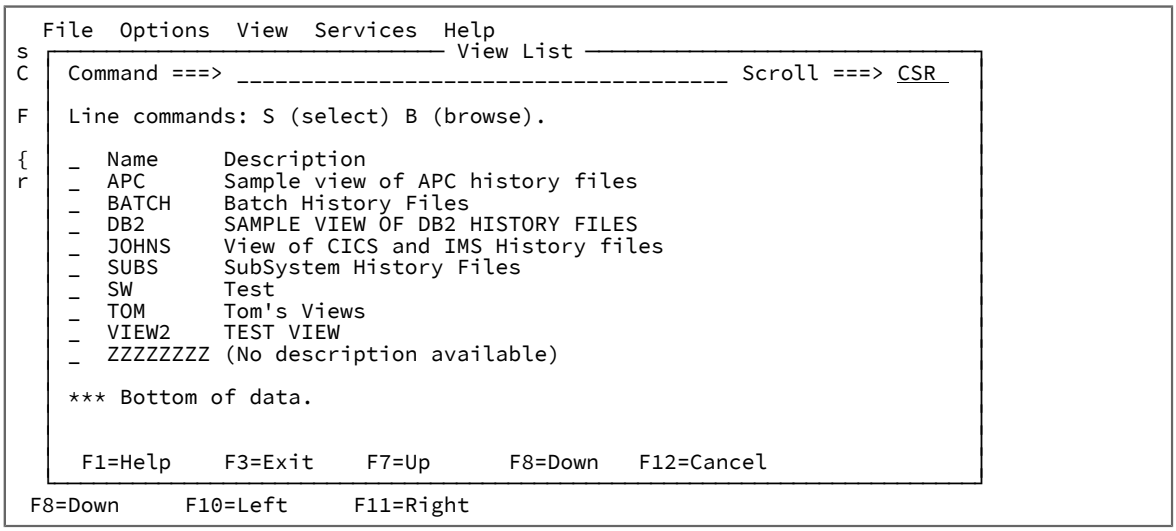
From the Last Accessed Fault History File Entries display shown in Figure 12: Sample Last Accessed Fault History File Entries display on page 63, type the number corresponding to the desired fault entry at the initial cursor position and press Enter to display it.

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 94). This opens a display as shown in Figure 13: Sample View List display on page 64.

Figure 13. Sample View List display



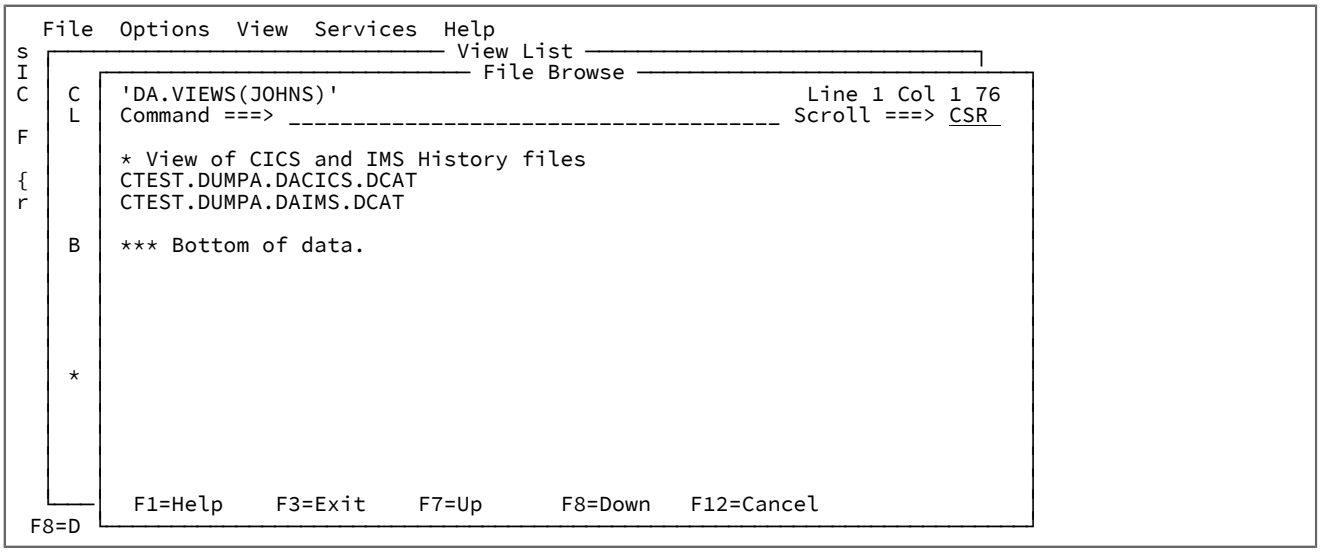
From here, you can enter one of the following line commands against each view:

B

This command permits you to browse the view member. For example, if the view member JOHNS was selected for browse, the contents of this member would be displayed as shown in Figure 14: Sample File Browse display on page 64.

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

Figure 14. Sample File Browse display



S

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

To display the chosen view, press PF3.

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

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

Fault entry list column configuration

The fault information that is shown on the Fault Entry List display is determined by the FAISPFopts(HistCols(...)) option in effect. If no FAISPFopts(HistCols(...)) option is used, the default is as illustrated in Figure 8: Sample Fault Entry List display on page 57.

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 94). This opens the Fault Entry List Column Configuration display as the example shown in Figure 15: Sample Fault Entry List Column Configuration display on page 65.

Figure 15. Sample Fault Entry List Column Configuration display

```
  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  2019/11/22 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 is shown above.}

Order Column
1     Fault_ID
2     Job/Tran
3     User_ID
4     Sys/Job        Default column configuration used   ❶
 F1=Help      F3=Sa                                          F8=Down
F10=Left     F11=Right    F12=Cancel
```

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

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

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

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

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

When the Fault Entry List Column Configuration display is first presented, a message is issued which identifies from where the current column configuration was read (see ❶ in Figure 15: Sample Fault Entry List Column Configuration display on page 65). There are four different possibilities:

- **Default column configuration used**

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

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

  Entering the RESET command (PF4) has no effect.

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

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

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

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

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

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

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

  Entering the RESET command (PF4) has no effect.

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

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

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

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

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

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

Available columns

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

Table 4. Fault entry list columns

Column label	Data type	Description
Fault_ID	Character	The ID assigned to the fault.
Abend	Character	The initial (if more than one) abend code. For a fault entry created using IDISNAP, the abend code is shown as "SNAP".
Appl_ID	Character	The CICS application ID.
CICS_Trn	Character	The failing CICS transaction ID. This column is only applicable to CICS.
Class	Character	The job class in which the job was executing.
CPU_sec		CPU seconds used at end of report processing during fault entry creation.
Date	Date	The date when the fault occurred in LOCALE-option dependent format.
Date_Time	Date & Time	This column is a combined Date and Time column with date and time values separated by a blank.
DTFJ_Status	Character	If applicable, the Java analysis DTFJ processing status: Dump error Pending Started

Table 4. Fault entry list columns (continued)

Column label	Data type	Description
		Finished Failed
Dups	Integer	The number of duplicate faults detected. See NoDup on page 616 for details about duplicate determination.
Dup_Count	Integer	Deprecated. Use Dups instead.
Dup_Date	Date	The date when the most recent duplicate fault occurred in LOCALE-option dependent format. If no duplicates have occurred, then this date is set to the initial abend date (see Date).
Dup_Date_Time	Date & Time	This column is a combined Dup_Date and Dup_Time column with date and time values separated by a blank.
Dup_Time	Character	The time when the most recent duplicate fault occurred in LOCALE-option dependent format. If no duplicates have occurred, then this time is set to the initial abend time (see Time).
EXEC_Pgm	Character	The abending job step program name from the JCL EXEC PGM= parameter.
Group_ID	Character	The security server default group ID.
History_File_DSN	Character	The history file data set name containing the displayed fault entry.
I_Abend	Character	The abend code for which Fault Analyzer was invoked. For a fault entry created using IDISNAP, the abend code is shown as "SNAP".
IMS_Pgm	Character	The IMS™ application program name for faults involving IMS™.
Job/Tran	Character	For a CICS® transaction abend, this value is the CICS® transaction ID. Otherwise,it is the name of the job that abended. This column combines information from the Jobname and CICS_Trn columns.
Job_ID	Character	The JES job ID of the abending job.
Job_Type	Character	The abending job type as one of the following: **Batch** Batch job **CICS®** CICS® transaction

Table 4. Fault entry list columns (continued)

Column label	Data type	Description
		CICSsys® CICS® system **DumpReg** Dump registration **STC** Started task **TSO** TSO user
Jobname	Character	The name of the abending job.
Lock	Character	The fault entry lock flag. For information about the purpose of this flag. See Viewing fault entry information on page 129.
Locked	Character	Indicates the lock status of the current fault entry: **Yes** The fault entry is locked. **No** The fault entry is not locked.
Lock_Usr	Character	The user ID who last changed the Lock flag.
Minidump	Character	Indication of minidump availability as follows: **Yes** Minidump is available **No** Minidump is not available
Module	Character	The point-of-failure module name. For dump registration fault entries, this value is the name of the module identified in the SVC dump header SDWA as the load module involved in the error. Following initial reanalysis and fault entry refresh, this value is updated to become the point-of-failure module name, which might be different.
MD_Pages	Integer	The number of minidump pages saved for the fault.

Table 4. Fault entry list columns (continued)

Column label	Data type	Description
MVS_Dump	Character	Indication of MVS™ dump availability as follows: **Yes** MVS™ dump is available **No** MVS™ dump is not available **Note:** This column indicates if an MVS™ dump data set was available at the time of abend. The associated MVS™ dump data might now be deleted or not exist on the system on which the fault entry is being displayed.
MVS_Dump_DSN	Character	The name of any associated MVS™ dump data set written at the same time as when the fault occurred. **Note:** This column indicates if an MVS™ dump data set was available at the time of abend. The associated MVS™ dump data might now be deleted or not exist on the system on which the fault entry is being displayed.
Netname	Character	CICS® transaction netname.
Offset	Hex	The point-of-failure offset.
Program	Character	The point-of-failure program name. This column allows up to 8 character program names to be displayed. Program names longer than 8 characters will be truncated.
Program_(long)	Character	The point-of-failure program name. This column allows up to 32 character program names to be displayed.
Spilled	Character	Indicates whether the associated XDUMP data set includes spilled minidump pages: **Yes** The XDUMP includes spilled minidump pages. **No** The XDUMP data set does not include spilled minidump pages.

Table 4. Fault entry list columns (continued)

Column label	Data type	Description
Stepname	Character	The job step name of the abending job.
Sys/Job	Character	For a CICS® transaction abend, this name is the CICS® job name. Otherwise, it is the ID of the system on which the job abended. This column combines information from the System and Jobname columns.
System	Character	The system ID on which the abend occurred.
Task	Character	The failing CICS® transaction task number. This column is only applicable to CICS® faults.
Term_ID	Character	CICS® transaction terminal ID.
Time	Time	The time when the fault occurred in LOCALE-option dependent format.
Tran_ID	Character	Deprecated. Use CICS_Trn instead.
User_ID	Character	For CICS® and for IMS™ message processing regions (MPP), the user ID that is associated with the abending transaction. Otherwise, the user ID that is associated with the abending job.
User_Title	Character	User-maintained title information. See Viewing fault entry information on page 129 for details.
Username	Character	User-maintained name information. See Viewing fault entry information on page 129 for details.

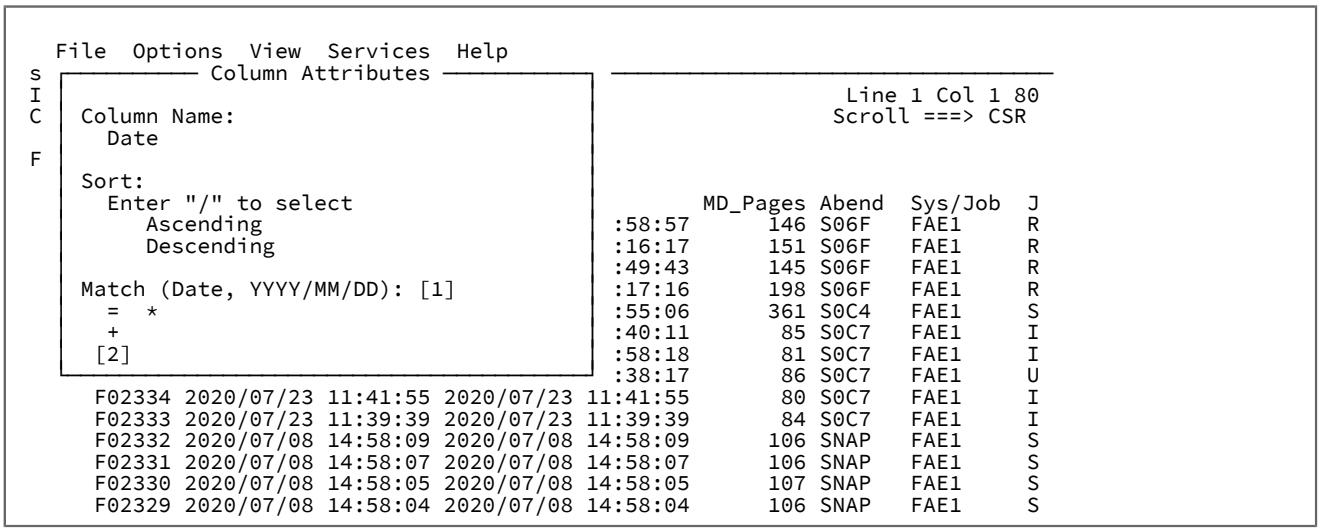
Sorting and matching fault entries

You can use column attributes to display a subset of fault entries that satisfy a given match criteria, such as a similar job name or the same abend code. You can sort the complete list of fault entries or the subset of entries that matches your criteria in ascending or descending order. Matching and sorting column attributes can help you to find faults with a similar pattern, for example, or to collect entries into a contiguous group so that you can delete a range of entries.

To display the Column Attributes, tab to a column heading in the Fault Entry List display, place the cursor on the heading, and press Enter. Use the Column Attributes display to specify the column sort order, the column matching criteria, or both. Initially, the Column Attributes display shows a single match criterion that specifies the default match operator (=) and default match value(*). The default matching criterion matches all entries in the display.

In Figure 16: Sample Column Attributes display with Date column selected on page 72, the Date column is selected.

Figure 16. Sample Column Attributes display with Date column selected



- To sort all fault entries in the column, type a forward slash (/) in the ascending or descending attribute input field and press Enter.
- To remove fault entries that do not match your criteria from the display, specify your matching criteria and press Enter.
- To apply a sort order to the subset of fault entries that match your criteria, specify both your matching criteria and the order in which you want the matching entries sorted, and press Enter.

Match operators

Overtype the default match operator (=) to change it. Valid match operators are:

=

Equal

!=

Not equal

>

Greater than

>=

Greater than or equal to

<

Less than

<=

Less than or equal to

The match operator is restricted to = or != when either of the following is true:

- The data type of the column is character.
- The match value includes one or more wildcard characters, unless the entire match value is a single * character. See Match values on page 73 for details about wildcard characters.

Match values

The match value is case-insensitive and can include wildcards:

*****

An asterisk represents zero, one, or more characters.

%

A percent sign represents a single character.

The default match value * matches any data value.

When the match value includes wildcard characters, all data types are evaluated as character data, unless the entire match value is a single * character.

The Column Attributes display shows the data type of the selected column in parentheses in the Match header (see [1] in Figure 16: Sample Column Attributes display with Date column selected on page 72). The Match values you supply must be valid for the data type of the column.

Table 5. Data types and values of column attributes

Data type	Valid values
Character	Any character.
Date, *date_format*	When the match value does not include wildcard characters, specify the date as one of the following: • Day, month, and year, in locale-specific format. • The current date, or a date relative to the current date, by using the TODAY keyword. Unless you use the TODAY keyword, the Date match-value must conform to the date format displayed in the Match header, including delimiters (such as the / character). ***YY*** 2-digit year. Values in the range 69-99 refer to years in the twentieth century (1969 to 1999); values in the range 00-68 refer to years in the twenty-first century (2000 to 2068).

Table 5. Data types and values of column attributes (continued)

Data type	Valid values
	YYYY 4-digit year. ***MM*** Month (01-12). ***DD*** Day of the month (01-31). **TODAY | TODAY-*days*** The TODAY keyword matches the current date (`TODAY`) or an earlier date, relative to the current date (`TODAY-days`). For example, if the Match header displays the date format as MM/DD/YYYY: • `09/27/2020` is an example of a valid match value. • `TODAY-2` and `TODAY` are examples of valid match values. • `9/27/20` and `09.27.2020` are examples of invalid match values.
Time, *time_format*	When the match value does not include wildcard characters, specify the Time match-value in the format displayed in the Match header, including delimiters (such as the : character). ***HH*** Hours, 12-hour clock (1-12). ***hh*** Hours, 24-hour clock (0-24). ***mm*** Minutes (0-59). ***ss*** Seconds (0-59). ***ampm*** The locale-specific equivalent of AM or PM (12-hour clock). For example, if the Match header displays the time format as HH:mm:ss ampm

Table 5. Data types and values of column attributes (continued)

Data type	Valid values
	• `01:25:59 PM` is an example of a valid match value. • `13:25:59` is an example of an invalid match value.
Date & Time, *date_format time_format*	A combined date and time value. When the match value does not include wildcard characters, specify the date value in the format indicated by the date format in the Match header. The time is optional. If included, specify the time value in the format indicated by the time format in the Match header.
Decimal	Decimal digits 0-9, decimal point period (.) or decimal comma (,) or negative sign (-).
Integer	Decimal digits 0-9 or negative sign (-).
Hex	Hexadecimal digits 0-9 or A-F.
Address	Hexadecimal digits 0-9 or A-F. Optionally, the address can include a single underscore (_) character. If included: • Digits to the left of the underscore are right-aligned as bits 0-31 of a 64-bit address. • Digits to the right of the underscore are right aligned as bits 32-63 of a 64-bit address. For example: `1_1` is equivalent to address `100000001`.

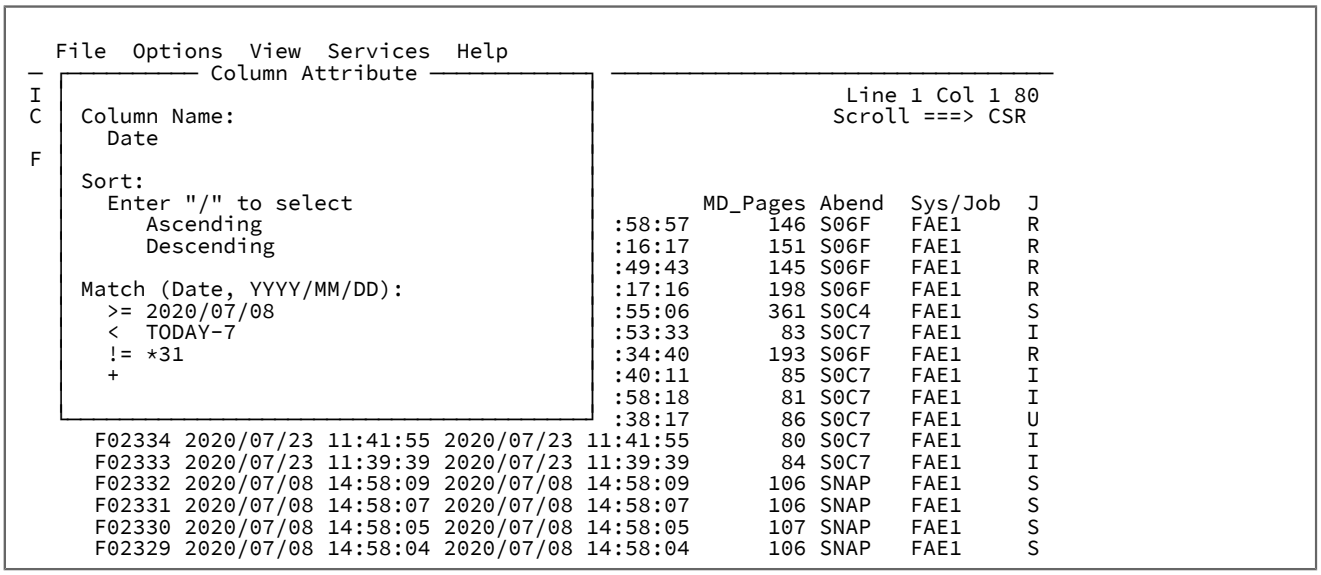
Using multiple matching criteria

You can specify up to 10 matching criteria per column. A data value must meet all criteria to match (logical AND).

- To add matching criteria, position the cursor on the **+** point-and-shoot field and press Enter. (See [2] in Figure 16: Sample Column Attributes display with Date column selected on page 72.)
- To remove match criteria, clear the operator field and press Enter.

Figure 17: Sample Column Attributes display specifying multiple matching criteria on page 76 shows multiple matching criteria for the Date column.

Figure 17. Sample Column Attributes display specifying multiple matching criteria



Applying the column attributes

Sorting and matching is applied to the rows of data that are currently displayed. If you change the sort or match values for the same column twice, or for first one column, then another, the second sort or match is applied to the rows of data that are displayed after the first sort or match.

After fault entries that do not match the criteria are removed from the display, you must perform a reset to restore them. You can reset the display by either:

- Placing the cursor on the reset point-and-shoot field and pressing Enter.
- Entering the RESET command on the command line.

Figure 18: Fault entries that match criteria in previous figure on page 77 shows a sample result of applying the matching criteria in Figure 17: Sample Column Attributes display specifying multiple matching criteria on page 76.

Figure 18. Fault entries that match criteria in previous figure

```
  File  Options  View  Services  Help
_______________________________________________________________________________
IBM Fault Analyzer - Fault Entry List                  13 of 322 rows matched
Command ===> _____________________________________________ Scroll ===> CSR

Fault History File or View  : 'NWILKES.HIST'

   Fault_ID Date       Time     Date_Time           MD_Pages Abend  Sys/Job  J
     F60415 2020/08/07 09:55:06 2020/08/07 09:55:06      361 S0C4   FAE1     S
     F02343 2020/08/04 09:53:33 2020/08/04 09:53:33       83 S0C7   FAE1     I
     F02342 2020/08/03 15:34:40 2020/08/03 15:34:40      193 S06F   FAE1     R
     F02340 2020/08/03 09:40:11 2020/08/03 09:40:11       85 S0C7   FAE1     I
     F02335 2020/07/29 10:38:17 2020/07/29 10:38:17       86 S0C7   FAE1     U
     F02334 2020/07/23 11:41:55 2020/07/23 11:41:55       80 S0C7   FAE1     I
     F02333 2020/07/23 11:39:39 2020/07/23 11:39:39       84 S0C7   FAE1     I
     F02332 2020/07/08 14:58:09 2020/07/08 14:58:09      106 SNAP   FAE1     S
     F02331 2020/07/08 14:58:07 2020/07/08 14:58:07      106 SNAP   FAE1     S
     F02330 2020/07/08 14:58:05 2020/07/08 14:58:05      107 SNAP   FAE1     S
     F02329 2020/07/08 14:58:04 2020/07/08 14:58:04      106 SNAP   FAE1     S
     F02328 2020/07/08 14:58:02 2020/07/08 14:58:02      106 SNAP   FAE1     S
     F02327 2020/07/08 12:15:44 2020/07/08 12:15:44      211 S06F   FAE1     R

**  Bottom of data.
```

Additional ways to match and select faults

In addition to the Column Attributes display match capability described in Sorting and matching fault entries on page 71, you can match fault entries by:

- Cursor-selecting a matching value
- Over-typing existing values
- Using the MATCH command

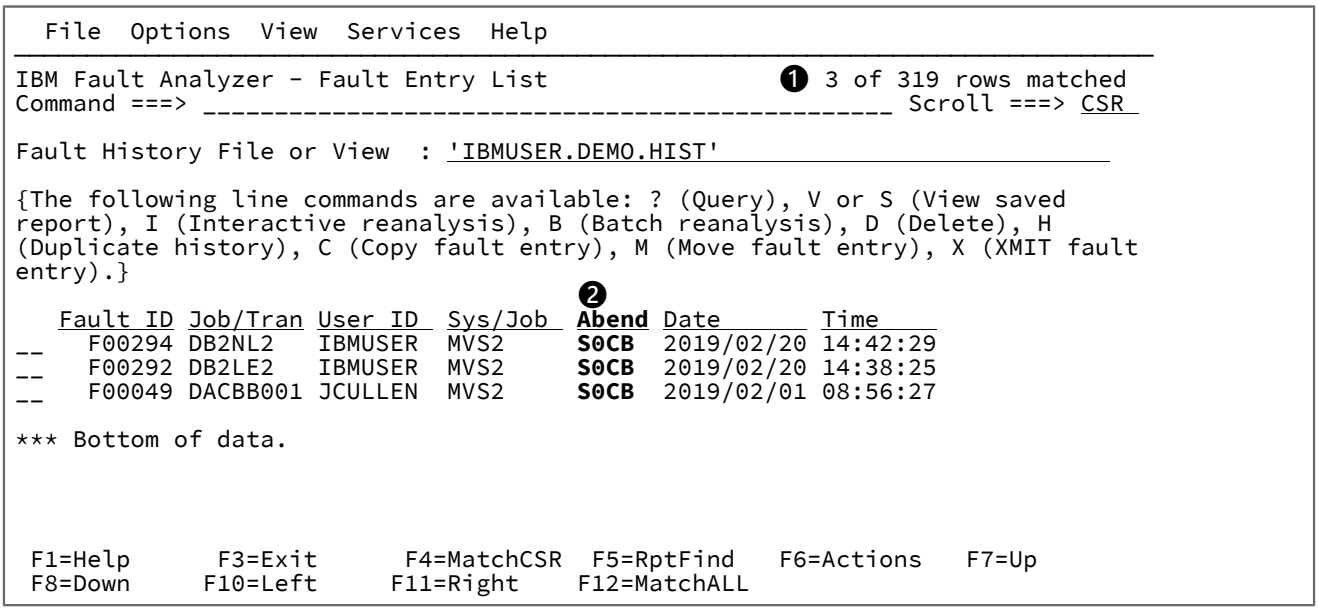
Each of these methods is integrated with the Column Attributes display, and updates the MATCH criteria of the column as if you entered the match criterion explicitly.

Cursor-selecting a matching value

If you move the cursor under a value that you want matched, then press PF4 (MatchCSR), Fault Analyzer refills the fault history window with faults that share this value for this field.

For example, if on the sample screen shown at Figure 8: Sample Fault Entry List display on page 57 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:

Figure 19. The Fault Entry List after one match



Notes:

❶

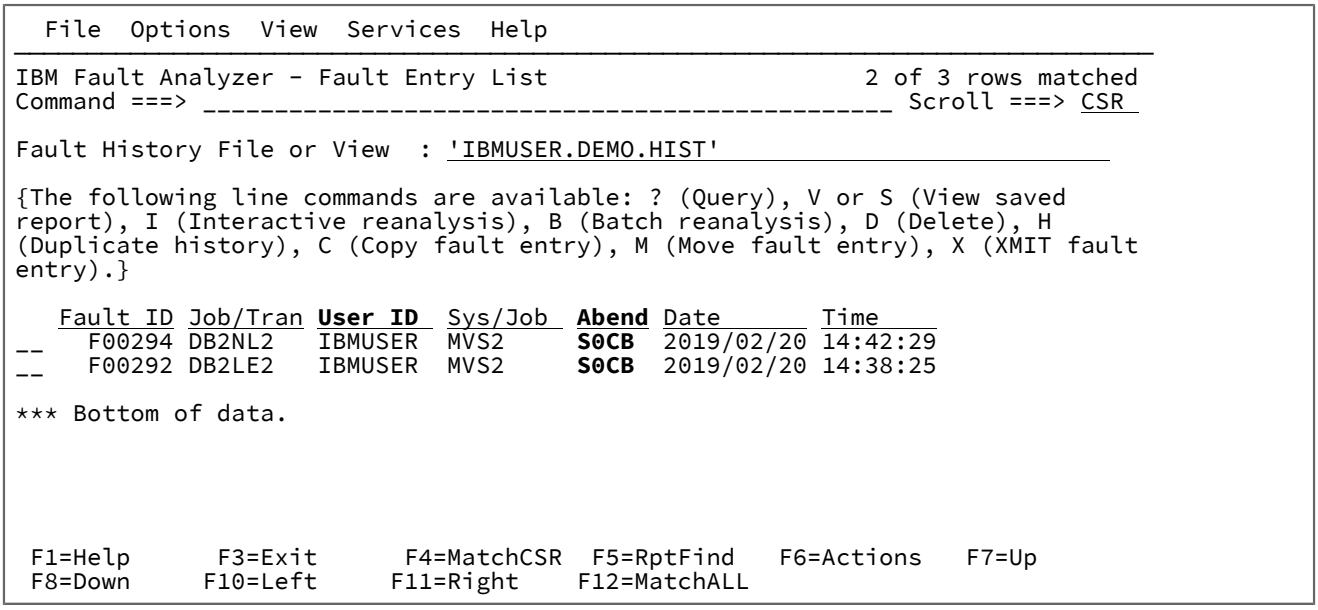
A message is issued that shows how many rows, of the ones previously displayed, that matched the selected value. This message only remains until a function key or the Enter key is pressed.

❷

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

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

Figure 20. 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 user ID, 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

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

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

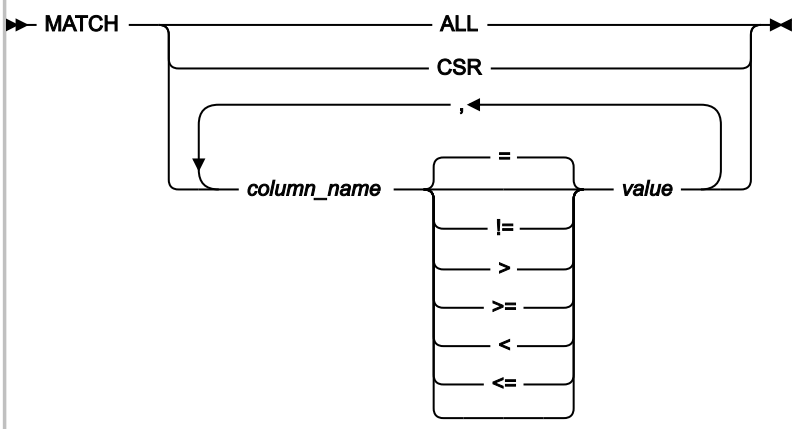
Any number of values can be over-typed before pressing the Enter key. However, if values for the same column are over-typed on multiple rows, then only the last over-typed value for that column is used.

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

Using the MATCH command

The MATCH command has the following syntax:

Figure 21. Syntax



Note: Either commas or blank characters are permitted as delimiters.

In the above syntax diagram, *column_name* can be any one of the column names shown in Available columns on page 67, for example, Abend or IMS_Pgm. The column name and the specified value are not case sensitive.

MATCH ALL (which is the same as PF12) removes all match conditions including the limit, so that the display shows the same entries as it had when you first displayed the history file. This option is not the same as doing a REFRESH, which

looks at the history file, and so can display new entries that have been written to the history file since you first started Fault Analyzer. When you refresh, you also remove all match conditions.

MATCH CSR is the cursor match. To make this work you have to move the cursor under a value and press Enter, which is essentially the same as placing the cursor and pressing PF4 (see Cursor-selecting a matching value on page 77).

The other keywords correspond to a field, and you follow the field name with a value. Optionally, you can specify an operator between the field name and the value. The default operator is =. When matching, values are not case-sensitive.

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

```
A "B",C
```

then specify the match command as:

```
MATCH USER_TITLE "A ""B"",C"
```

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

```
MATCH DATE 2019/07*
```

which would display all entries for the month of July, for 2019.

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

Specify the match values for any column in the same format as the data displayed in that column. For example, if a date column contains dates in the format year/month/day, then the match value must likewise be in the format year/month/day.

See Sorting and matching fault entries on page 71 for more information about data types and valid values. The rules that apply to the match criteria on the Column Attributes display also apply to the MATCH command.

Example:

If a Date & Time column contains dates in the format `month.day.year` and time in the format `hours:minutes:seconds`, the following examples are valid match values:

```
1.13.2020
06.5.2020 0:0:0
TODAY
TODAY-365
```

Use the TODAY keyword in the **Date** column or the **Date & Time** column to specify either the current date (`TODAY`) or a range of dates relative to the current date (`TODAY-days`). For example:

```
TODAY
TODAY-10
```

When you specify `TODAY-days`, no blanks or other delimiters can precede or follow the minus sign.

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. A match on a column that is not selected for display is ignored.

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

Applying an action to a particular fault

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

A - display source

Display the source-level debugging information captured during real-time analysis when the LangxCapture option is enabled. See LangxCapture on page 613.

B - batch reanalysis

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

See Performing batch reanalysis on page 149 for details.

C - copy

Copy the fault entry to a different history file.

See Copying history file entries on page 138 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 126 for details.

H - duplicate history

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

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

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 Performing interactive reanalysis on page 157 for details.

J - edit JCL

Display and edit the JCL captured during real-time analysis when the JclCapture option is enabled. This command performs the same function as the JCL command used in interactive reanalysis. See JCL on page 106 and JclCapture on page 612.

L - lock

Lock fault entry using default lock flag setting.

To change the default lock flag setting, see Fault Analyzer preferences on page 113.

M - move

Move the fault entry to a different history file.

See Moving history file entries on page 140 for details.

P - package fault entry

Generate JCL to package related fault entry data to a tersed data set suitable for transmission to the Fault Analyzer product support team. The JCL is displayed in an edit session which can then be amended and submitted.

See Packaging fault entries on page 141 for details.

U - unlock

Unlock fault entry.

V (or S) - view report

View the saved fault analysis report:

See Viewing a saved report on page 120 for details.

X - XMIT

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

See Transmitting history file entries on page 140 for details.

? - view fault entry information

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

See Viewing fault entry information on page 129 for details.

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

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

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

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

History file properties

To display attributes and statistics for the currently selected fault history file, first select **File > Fault History File Properties** from the action bar.

This opens the Fault History File Properties display.

Figure 22. Sample Fault History File Properties display

```
──────────────────────── Fault History File Properties ─────────────────────────
Fault History File Properties                                 Line 1 Col 1 76
Command ===> ____________________________________________ Scroll ===> CSR

Enter the Exit command (PF3) to return to the fault history file display.

History File Name . . . . . : DA.DCAT
History File Type . . . . . : PDSE (Library type 1)
Primary Space . . . . . : 1,000 Cylinders (180,000 Pages)
Secondary Space . . . . . :   200 Cylinders ( 36,000 Pages)
Allocated Space . . . . . : 4,347 Cylinders (782,460 Pages)
Used Space. . . . . : 599,907 Pages
Allocated Extents . . . . . : 27
Data Set Utilization. . . . : 76.67%
History File Access . . . . : Update
Fault ID Prefix . . . . . : F
Logical History File Size . : 600,000 Pages (76.68% of Allocated Space)
History File Utilization. . : 99.98%
Minimum Fault Entries . . . : 25,ThenAUTO
Current Fault Entries . . . : 4,192
Number of Entries With
  Minidump. . . . . : 4,092 (97.61% of Total)
Maximum Minidump Size . . . : 2,840 Pages
Enter the Exit command (PF3) to return to the fault history file display.
```

The following information is available from this display:

Allocated Extents

The total number of extents allocated for the history file. Shown only if the user has data set READ access to the history file.

Allocated Space

The total allocated data set space in original allocation units. For PDSE history files the equivalent number of pages is also shown. Shown only if the user has data set READ access to the history file.

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.

Data Set Utilization

Pages used as percentage of pages allocated. Only available for PDSE history files. Shown only if the user has data set READ access to 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 Managing history files (IDIUTIL utility) on page 441). Alternatively, this setting can be changed using the Fault Analyzer ISPF interface (see Change fault history file settings on page 88).

History File Access

Read or Update.

History File Name

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

History File Type

This name is indicated as one of the following:

- PDSE (Library)
- PDSE (Library type 1)
- PDSE (Library type 2)
- PDS (Partitioned Data Set)

History File Utilization

Used space as a percentage of history file size. (Only shown for PDSE history files.) Shown only if the user has data set READ access to the history file.

Logical History File Size

If available, the current logical history file size. Shown only if the user has data set READ access to the history file.

Minimum Fault Entries

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

The minimum number of fault entries can be changed with the IDIUTIL batch utility SetMinFaultEntries control statement (see Managing history files (IDIUTIL utility) on page 441). Alternatively, this setting can be changed using the Fault Analyzer ISPF interface (see Change fault history file settings on page 88).

Maximum Fault Entries

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

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

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

For PDS history files, the maximum number of fault entries can be changed with the IDIUTIL batch utility SetMaxFaultEntries control statement, and for PDSE history files, the IDIUTIL batch utility SetMinFaultEntries control statement (see Managing history files (IDIUTIL utility) on page 441). Alternatively, this setting can be changed using the Fault Analyzer ISPF interface (see Change fault history file settings on page 88).

Maximum Minidump Size

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

Minimum Minidump Size

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

Number of Entries With Associated MVS™ Dump

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

Number of Entries With Minidump

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

Number of History Files

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

Primary Space

The primary data set space allocation in original allocation units. For PDSE history files the equivalent number of pages is also shown. Shown only if the user has data set READ access to the history file.

Secondary Space

The secondary data set space allocation in original allocation units. For PDSE history files the equivalent number of pages is also shown. Shown only if the user has data set READ access to the history file.

Total Number of Entries

The total number of entries in the view.

Used Space

For a PDSE history file, this is the total number of pages used. For a PDS history file, this is the total number of original allocation units used. Shown only if the user has data set READ access to the history file.

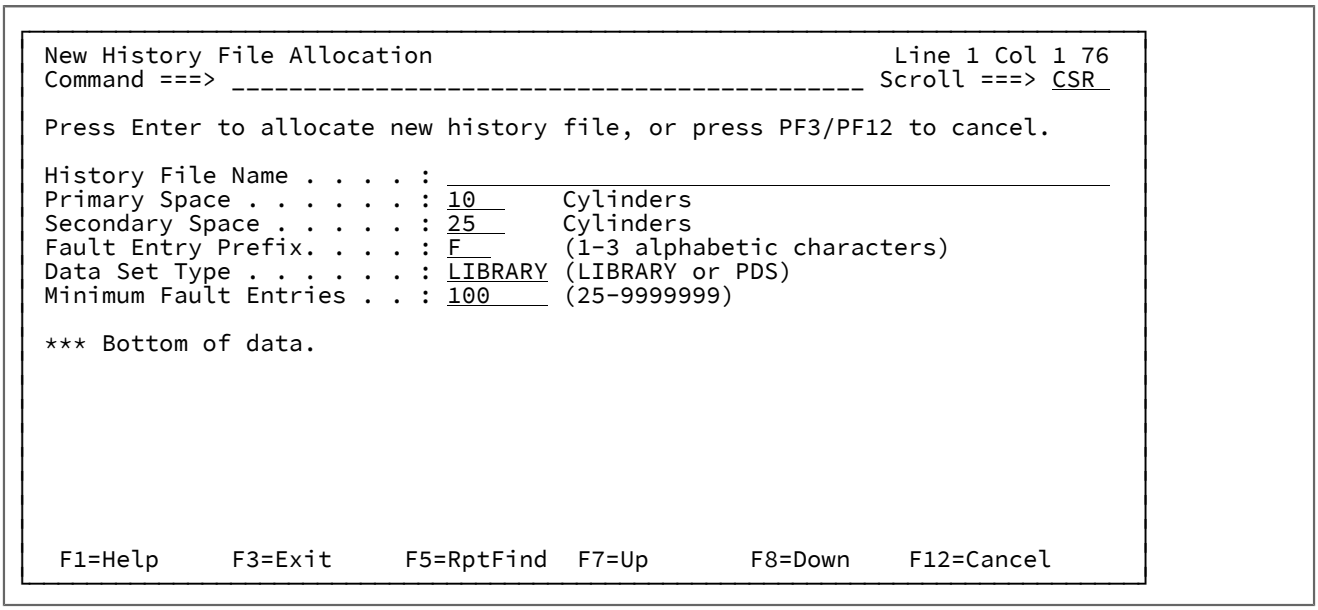
View Name

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 94). This opens the New History File Allocation display as the example shown in Figure 23: Sample New History File Allocation display on page 87.

Figure 23. Sample New History File Allocation display



The following fields are provided on this display:

History File Name

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

Note: If this display is shown due to the user having specified a history file that does not exist in one of the following situations, then the **History File Name** field is not available for input, but shows the target history file name that was specified:

- Creating an initial fault entry during Java dump analysis.
- Creating a fault entry after analysis of a MVS dump data set.
- Moving or copying one or more fault entries between history files.

Primary Space

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

Secondary Space

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

Fault Entry Prefix

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

Data Set Type

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

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

The default is LIBRARY.

Minimum Fault Entries

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

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

Maximum Fault Entries

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

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

Change fault history file settings

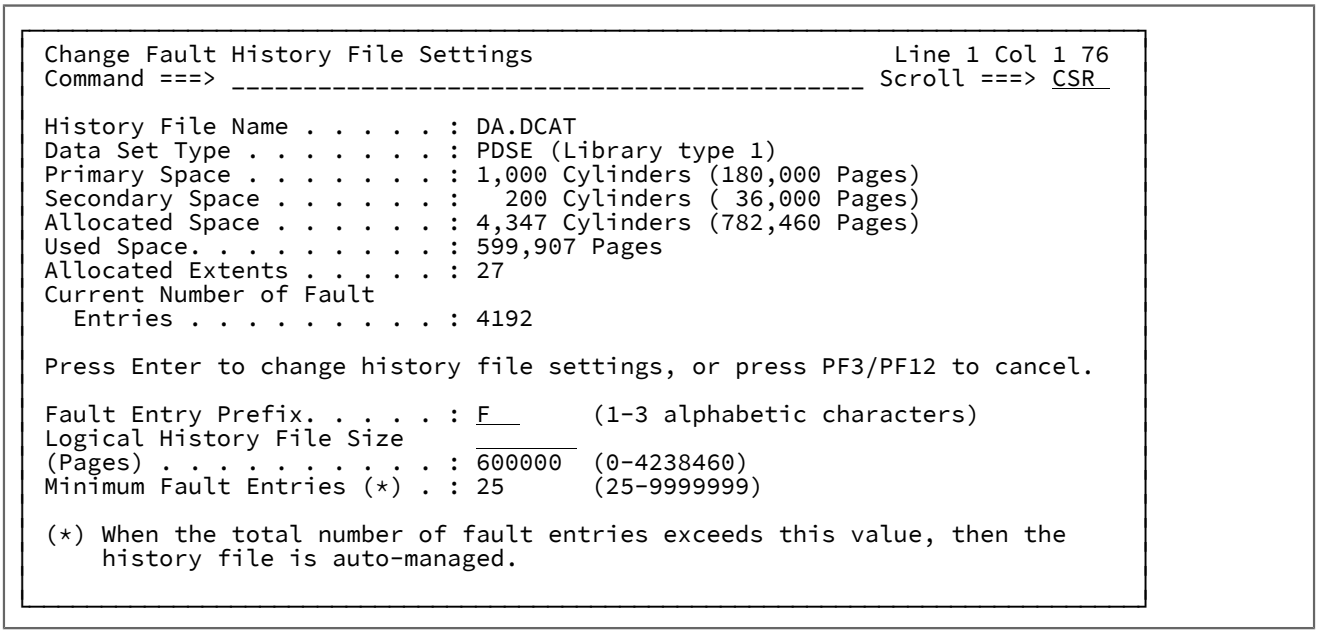
To change a history file's settings from the ISPF interface, ensure that the history file is selected on the Fault Entry List display, or is included in the currently selected View.

You can change the fault ID prefix or the minimum or maximum number of fault entries. First, select **File > Change Fault History File Settings** from the action bar.

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

Next, the Change Fault History File Settings display, as the example shown in Figure 24: Sample Change Fault History File Settings display on page 89, is presented.

Figure 24. 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)
- PDSE (Library type 1)
- PDSE (Library type 2)
- PDS

Primary Space

The primary data set space allocation in original allocation units. For PDSE history files the equivalent number of pages is also shown. Shown only if the user has data set READ access to the history file.

Secondary Space

The secondary data set space allocation in original allocation units. For PDSE history files the equivalent number of pages is also shown. Shown only if the user has data set READ access to the history file.

Allocated Space

The total allocated data set space in original allocation units. For PDSE history files the equivalent number of pages is also shown. Shown only if the user has data set READ access to the history file.

Used Space

For a PDSE history file, this is the total number of pages used. For a PDS history file, this is the total number of original allocation units used. Shown only if the user has data set READ access to the history file.

Allocated Extents

The total number of extents allocated for the history file. Shown only if the user has data set READ access to the history file.

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 are given this prefix, and by using different prefixes for different history files, this can make the fault IDs easier to recognize.

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

Logical History File Size (Pages)

This field is only displayed for a PDSE history file and only if the user has data set READ access to the history file.

The value specifies the logical history file size as a number of 4K pages, and is used by Fault Analyzer to aid in space management of the history file. Until the minimum number of fault entries has been created in the history file, this setting will be shown as 0.

If you change a non-zero value using the procedure described in Changing the size of a PDSE history file on page 347, set the value to 0. This will automatically set the logical history file size to the current allocated number of pages at the time of creating the next fault entry which exceeds the minimum number of fault entries specified for the history file.

The maximum allowed value represents the theoretical maximum number of pages that could possibly be allocated, assuming the maximum number of data set extents is reached, with all remaining extents being of the secondary space size. Do not specify a value that is close to this size, because it will likely result in out-of-space conditions on the history file.

A warning will be displayed if the value specified is less than the Used Space value or greater than the Allocated Space value.

Minimum Fault Entries

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

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

Maximum Fault Entries

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

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

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

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

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

After changing any settings, press Enter to save.

To exit without saving, press PF3 or PF12.

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

Note: The action-bar option **File > Change Fault History File Settings** is not selectable if the user's administrator authorization to the history file currently being displayed, or to all history files in the current View, has been restricted. For details about restricting authorization, see Restricting change of history file settings on page 307.

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 94).

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

Refreshing fault entry information

While displaying a history file or view, it is possible that new entries are being added, for example due to real-time analysis of abending jobs. To reread the history file or view to include any such entries, you can either issue the REFRESH command or select the View menu Refresh option (for information about selecting this option in general, refer to Action-bar pull-down menus on page 94).

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

Implicit refresh

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

Updates pending

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

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

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

Figure 25. Sample Fault Entry List display with updates pending

```
  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. ❶
   Fault_ID MD_Pages Dups  Dup_Date  Dup_Time Abend  I_Abend Module   System
__ IMS14434       96      2019/08/03 13:51:52 S0CB   U4039   IDCB0060 FAE1
__ IMS14433       99      2019/08/03 13:51:40 SNAP   SNAP    IDCB0040 FAE1
__ IMS14432       98      2019/08/03 13:51:26 S0CB   U4039   IDCB0020 FAE1
__ IMS14431       51      2019/08/03 13:51:16 U0428  U0428   DFSPCC20 FAE1
__ IMS14430       98      2019/08/03 13:51:01 S0CB   U4039   IDCB0020 FAE1
__ IMS14429       99      2019/08/03 13:50:46 S0CB   U4039   IDCB0010 FAE1
__ IMS14428      106      2019/08/03 13:50:33 S0CB   U4039   IDCB0090 FAE1
__ IMS14427      735      2019/08/03 13:50:11 S0CB   U4039   IDCB0080 FAE1
__ IMS14426      105      2019/08/03 13:49:54 S0CB   U4039   IDCB0070 FAE1
__ IMS14424       99      2019/08/03 13:49:35 S0CB   U4039   IDCB0060 FAE1
__ IMS14423       51      2019/08/03 13:49:26 U0456  U0456   DFSPCC20 FAE1
__ IMS14422      100      2019/08/03 13:49:15 SNAP   SNAP    IDCB0040 FAE1
__ IMS14421       98      2019/08/03 13:49:02 S0CB   U4039   IDCB0020 FAE1
__ IMS14420       96      2019/08/03 13:48:48 S0CB   U4039   IDCB0020 FAE1
__ IMS14419       51      2019/08/03 13:48:39 U0428  U0428   DFSPCC20 FAE1
__ IMS14418       97      2019/08/03 13:48:26 S0CB   U4039   IDCB0010 FAE1
```

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

Figure 26. Sample History File Updates Pending 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      2019/08/03 13:50:11 S0CB   U4039   IDCB0080 FAE1
 IMS14426      105      2019/08/03 13:49:54 S0CB   U4039   IDCB0070 FAE1
 IMS14424       99      2019/08/03 13:49:35 S0CB   U4039   IDCB0060 FAE1
 IMS14423       51      2019/08/03 13:49:26 U0456  U0456   DFSPCC20 FAE1
 IMS14422      100      2019/08/03 13:49:15 SNAP   SNAP    IDCB0040 FAE1
 IMS14421       98      2019/08/03 13:49:02 S0CB   U4039   IDCB0020 FAE1
 IMS14420       96      2019/08/03 13:48:48 S0CB   U4039   IDCB0020 FAE1
 IMS14419       51      2019/08/03 13:48:39 U0428  U0428   DFSPCC20 FAE1
 IMS14418       97      2019/08/03 13:48:26 S0CB   U4039   IDCB0010 FAE1
```

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

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

Fault entry expiration control

To prevent premature deletion of fault entries, these can be locked either indefinitely, or for a specified number of days following the initial creation of a fault entry. Either way, this deletion is controlled via the Lock Flag, which can be viewed or modified using the Fault Entry Information display (for details, see Viewing fault entry information on page 129).

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 653).

Action-bar pull-down menus

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

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

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

In the following list of available pull-down menu options, the format used is:

```
menu_name->menu_option->menu_option...
```

where

menu_name

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

menu_option

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

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

File->Analyze MVS™ Dump Data Set

Used to initiate interactive analysis of a SYSMDUMP or SVC dump data set. Primarily intended for CICS® system dump analysis (see Performing CICS system abend dump analysis on page 247) or Java™ dump analysis (see Performing Java analysis on page 259).

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 88.

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 91.

File->Exit

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

File->Exit Fault Analyzer

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

File->Exit Interactive Reanalysis

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

File->Fault Entry Information

Used to open the Fault Entry Information display as shown in Figure 73: Sample Fault Entry Information display on page 130. Selecting this option is identical to issuing the INFO command (see INFO on page 106) 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 81).

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 84.

File->Format CICS® Auxiliary Trace Data Set

Used to format a CICS® auxiliary trace data set. See Formatting a CICS auxiliary trace data set on page 257.

File->Last Accessed Fault History File Entries

Used to display a list of up to 10 previously accessed history file entries. See Changing the history file or view displayed on page 59 for more information.

File->Last Accessed Fault History Files or Views

Used to display a list of up to 10 previously accessed history files or views. See Changing the history file or view displayed on page 59.

File->List Views

Used to display a list of all available views. See Changing the history file or view displayed on page 59.

File->New Fault History File Allocation

Permits the allocation of a new history file. See New history file allocation on page 87 for additional information.

Help->About Fault Analyzer

Used to display copyright and general usage information for Fault Analyzer. See Displaying product copyright, license, and version information on page 126.

Options->Batch Reanalysis Options

Used to set options for batch reanalysis. See Batch reanalysis options on page 149.

Options->Fault Analyzer Preferences

Used to set options that affect the behavior of the Fault Entry List display. See Fault Analyzer preferences on page 113.

Options->Interactive Reanalysis Options

Used to set options for interactive reanalysis. See Interactive reanalysis options on page 157 for more information.

Options->Options in Effect

Used to display the options that are in currently in effect and where they are specified. See Displaying options currently in effect on page 147 for details.

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 125.

Services->COBOL Explorer

Used to start the the COBOL Explorer dialog with prompt for program selection. See COBOL Explorer on page 241 for more information.

Services->IDIS Subsystem Information

Used to invoke the interactive IDIS subsystem interface. See Using the interactive IDIS subsystem interface on page 142 for more information.

Services-> LANGP Side File Formatting Utility

Used to invoke the LANGP side file formatting utility to display a pseudo compiler listing. For more information, see Formatting side files with the HFZLANGP utility.

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 212.

Services->Message ID Lookup

Used to show the explanation for a user-selected message, abend code, or other information. See Displaying user-selected message or abend code explanations on page 123 for more information.

Services->Service Information

Used to show information about installation and maintenance status. See Step 3: Verify the service level (optional) on page 460 in "Maintaining Fault Analyzer" for more information.

Services->Storage Map

Used to show the layout of the current address space. See Displaying the address space storage map (STGMAP) on page 218 for more information.

View->Add Blank Lines

Used to format displays by using blank lines when appropriate to separate the information that is shown. This value is the default. See Adding or removing blank lines on page 122.

View->Add Help Text

Used to add explanatory text to displays which might assist the inexperienced user. This value is the default. See Adding or removing help text on page 122 for more information.

View->Add Pseudo Assembler Instructions

Used to add pseudo-assembler instructions to the Compiler Listing display. See Displaying source code on page 206 for more information.

View->Collapse Duplicate Fault Entries

Used to remove duplicate fault instances that are displayed as separate fault entries from the Fault Entry List display. See Viewing the fault entry duplicate history on page 134.

View->Column Configuration

Used to change the columns of information that are shown for faults listed on the Fault Entry List display. See Fault entry list column configuration on page 65 for more information.

View->Expand Duplicate Fault Entries

Used to show duplicate fault instances as separate fault entries in the Fault Entry List display. See Viewing the fault entry duplicate history on page 134.

View->Preferred formatting Width

Used to set the preferred display formatting width. See Setting preferred formatting width on page 122.

View->Refresh

Used to reread all entries in the selected history file or view. See Refreshing fault entry information on page 92.

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 122 for more information.

View->Remove Help Text

Used to remove explanatory text from displays. See Adding or removing help text on page 122.

View->Remove Pseudo Assembler Instructions

Used to remove pseudo-assembler instructions from the Compiler Listing display. This value is the default. See Displaying source code on page 206.

Commands

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

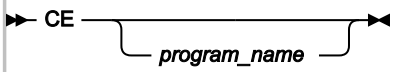
Note: Not all commands can be issued from all displays. Refer to the description of individual commands for details.

CE

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

Enter this command during interactive reanalysis.

Figure 27. Syntax



program_name

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

For more information, see COBOL Explorer on page 241.

CICSD

Use the CICSD command to display a CICS 3270 screen buffer that was captured when the fault entry was created. In contrast to the plain-text representation displayed by the Last CICS 3270 Screen Buffer on page 178 link in the interactive report display, the CICSD command displays all colors and attributes of the last CICS 3270 screen, including double-byte characters.

The CICSD command is valid only for fault entries captured using Fault Analyzer version 14.1.14 and later.

Figure 28. Syntax

CICSLINK

Brings up the CICS® Trace Link Analysis display that represents the execution path and hierarchy of CICS® transaction nested linked programs. See an example of the analysis display in Internal CICS Trace Link Analysis (CICSLINK) on page 183.

If you execute this command while you are viewing a formatted CICS® trace from a CICS® system dump or auxiliary trace data set, you will be prompted for the task number to analyze, unless the cursor is already positioned on the desired task number.

This command is only available while you are viewing a formatted CICS® trace.

Figure 29. Syntax

For more information, see CICS Trace Formatting on page 180.

CICSSTG

Use the CICSSTG command to filter and display CICS task storage areas. This command is available only from within the interactive report.

Figure 30. Syntax

For more information, see Displaying CICS transaction storage (CICSSTG) on page 217.

CMDS

The CMDS command displays the list of commands available in the ISPF interactive reanalysis session. A command can be executed by selecting it with an **S**.

Figure 31. Syntax

COLS

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

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

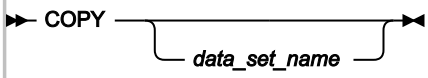
Figure 32. Syntax

For more information, see Fault entry list column configuration on page 65.

COPY

Copies the current display to the specified data set name.

Figure 33. Syntax



where:

data_set_name

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

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

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

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

For more information about the COPY command and an example of its use, see Copying interactive displays to a file on page 125.

CUROPTS

Use this command to display the options that are currently in effect and where the options are specified. This command is available only from the Fault Entry List display, and is identical to selecting the **Options->Options In Effect** option from the action-bar pull-down.

See Displaying options currently in effect on page 147 for details.

DISASM

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

Figure 34. Syntax

For information about how to use this command, see Disassembling object code on page 229.

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.

Figure 35. Syntax

For information about how to use this command, see Mapping storage areas using DSECT information on page 223.

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.

Figure 36. Syntax

For information about the display of duplicate fault details, see Viewing the fault entry duplicate history on page 134.

EDIT

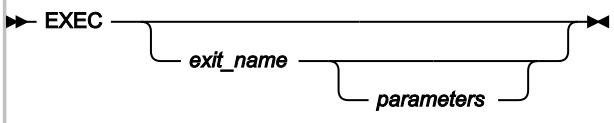
This command copies the current display to a temporary sequential data set and then opens this data set in an ISPF EDIT session.

Figure 37. Syntax

EXEC

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

Figure 38. Syntax



where:

exec_name

The name of the Formatting user exit to be executed.

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

parameters

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

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

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

```
EXEC MYEXEC A B
```

and the parameters can be received in the exit using:

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

Note that

```
PARSE ARG PARM1 PARM2
```

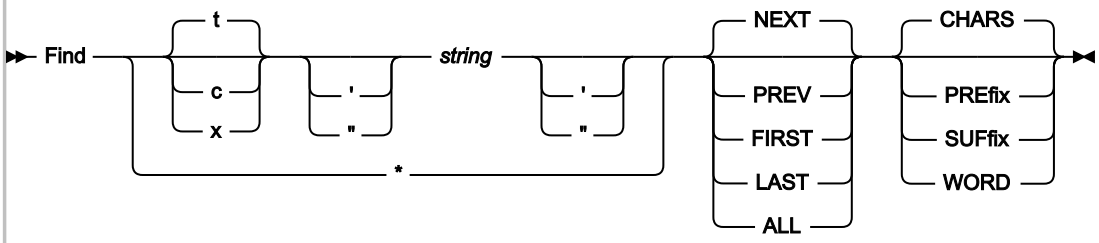
will not work as expected.

For information on creating your own Formatting user exits, and making these available to your environment, see User-specific report formatting on page 233.

FIND

Used to locate a text string in the current display.

Figure 39. Syntax



where:

t

Indicates that a text string is being searched for. Text strings are non-case-sensitive in all displays, except for the Dump Storage display where all strings are case-sensitive.

c

Indicates that a character string is being searched for. Character strings are case-sensitive.

x

Indicates that a hexadecimal value is being searched for. An even number of digits must be specified.

string

The character string to be searched for. If the string includes blanks, quotes, or tokens that could be mistaken for a FIND command keyword (for example, if searching for the string NEXT), the string must be enclosed in either single quotes or double quotes. The ending quote must be of the same type (single or double) as the starting quote. All other quotes are considered part of the search string.

*****

Indicates that the same string as previously searched for is to be used.

NEXT

Indicates that the search should start at the first position after the current cursor location and proceed ahead to find the next occurrence of the string. NEXT is the default.

PREV

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.

FIRST

Indicates that the search should start at the top of the display and proceed ahead to find the first occurrence of the string.

LAST

Indicates that the search should start at the bottom of the display and proceed backwards to find the last occurrence of the string.

ALL

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.

CHARS

Indicates that any occurrence of the sequence of characters searched for is considered a match. This value is the default.

PREfix

Indicates that one or more blank or attribute characters must precede the sequence of characters searched for in order to be considered a match. Either PRE or PREFIX can be specified.

SUFfix

Indicates that one or more blank or attribute characters must follow the sequence of characters searched for in order to be considered a match. Either SUF or SUFFIX can be specified.

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

Table 6. FIND command: differences between display types

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

Dump Storage display FIND ALL command progress

If the FIND *string* ALL command is issued from the Dump Storage display, and processing has not completed within approximately 10 seconds, then the FIND ALL Progress display is shown. An example of this display is shown in Figure 40: Command progress display on page 105.

Figure 40. Command progress display

```
FIND ALL Progress                                    Lin   CHARS X'ef' found
Command ===> _____________________________________   Scroll ===> CSR

Number of matches so far. : 201,967
Address of last match . . : 1639801C

Press ENTER to keep searching or PF3/PF12 to abort.

Seconds before next progress
  display . . . . . . . . . : 10 (1-99 or 0 to disable)

*** Bottom of data.

  00000030       +30  00000000 00000000 7FFFF000 7FFFF000 *........".0.".0.*
  00000040       +40  00000000 00000000 00000000 00FD7D08 *..............'.*
  00000050       +50  00000000 00000000 00000000 00000000 *................*
        Lines 00000060-00000070 (X'20' bytes) same as above
  00000080       +80  00000000 00001005 00020001 00040016 *................*
  00000090       +90  00000001 000E0000 00000000 00000000 *................*
  000000A0       +A0  00000001 00000000 000001EF C17FF400 *............A"4.*
  000000B0       +B0  00000000 00000F1F 00010087 027361C0 *...........g../{*
  000000C0       +C0  18000000 00000000 FBEBFFFB FEFFFF7C *...............@*
  000000D0       +D0  07FCE000 00000000 F7003B98 00800000 *..\.....7..q....*
```

The number of matches found so far is shown, along with the storage address of the most recent match.

To continue the search for matches, press **Enter**. Optionally, specify a different number of seconds between 1 and 99 before the next progress display is shown. Specify the value 0 to disable any further progress displays.

To abort the search, press **PF3** or P**F12**.

IDISINFO

The IDISINFO command can be used to invoke the interactive IDIS subsystem interface.

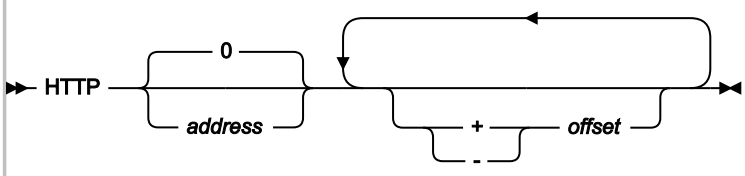
Figure 41. Syntax

For more information, see Using the interactive IDIS subsystem interface on page 142.

HTTP

The HTTP command, which is available only from within the interactive report, can be used to display formatted HTTP data located at a specified storage address.

Figure 42. Syntax



For example, to display the HTTP data starting at address 007F2300, enter:

```
HTTP 7F2300
```

The address parameter is 64-bit enabled. For details, see Specifying 64-bit addresses on page 142

Invoking this command displays a pop-up window where the Address field is pre-filled with the address you specified.

For detailed information about how to use this command and interact with the results, see Displaying formatted HTTP data (HTTP) on page 219.

INFO

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

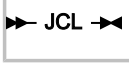
Issuing the INFO command is identical to selecting "Fault Entry Information" from the interactive reanalysis report action-bar "File" pull-down menu (see Action-bar pull-down menus on page 94), or entering the "?" line command against a fault entry from the Fault Entry List display (see Applying an action to a particular fault on page 81). In either case, the Fault Entry Information display, as shown in Figure 73: Sample Fault Entry Information display on page 130, is presented.

Figure 43. Syntax

JCL

After the JclCapture option successfully captures the JCL of an abending job during real-time processing, you can use the JCL command to display the JCL in an ISPF EDIT panel. See JclCapture on page 612.

Figure 44. Syntax



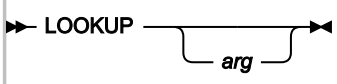
The JCL command is available only from within the interactive report.

Using the JCL command is equivalent to selecting the "Abend Job JCL" point-and-shoot field from the "Abend Job Information" section of the interactive reanalysis report.

LOOKUP

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

Figure 45. Syntax



where:

arg

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

If *arg* is not specified, a pop-up panel is displayed, from which you can select or search for the required information.

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

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

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

XYZ

Implicitly searches for a match on *XYZ*, which means that zero or more characters might precede XYZ, and zero or more characters might follow (for example, XYZ, AXYZ, XYZB, AXYZBBB).

A*

Matches everything starting with A, followed by zero or more characters (for example, A, AB, ABA000I).

A%

Matches everything starting with A, followed by exactly one character (for example, AA, AB, AC).

ABA%%%I

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

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

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

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

1. User-defined information in member IDIHUSRM of the IDI.SIDISAM1 data set. For details, see Providing application-specific explanations and descriptions on page 455.
2. IBM®-supplied information in the IDIDOC data set.
3. IBM-supplied information in the IDIVSxxx data set, where *xxx* is the 3-character language ID in effect for the Language option.
4. In the case of message or abend code explanations, the Message and Abend Code Explanation user exit might choose to supply an alternate explanation. For details, see Message and Abend Code Explanation user exit on page 485.

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 Modifying your ISPF environment on page 327.

For more information about the LOOKUP command, and an example of its use, see Displaying user-selected message or abend code explanations on page 123.

MATCH

Refer to Additional ways to match and select faults on page 77 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 displays

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

The Dump Storage display

The storage address displayed prior to entering a PREV command is shown again. The display is shown as it was when the PREV command was entered, with scroll and cursor position maintained.

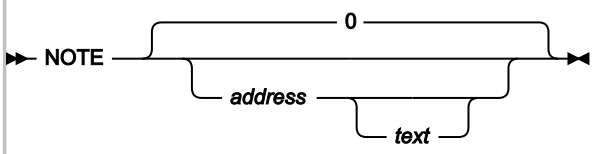
This command is usually assigned to the PF11 function key.

Figure 46. Syntax

NOTE

This command can be used to create a user note or edit an existing user note in an edit session. It is only available from within the interactive report.

Figure 47. Syntax



If a user note already exists for the address, that user note will be opened; otherwise, a new user note will be created.

When creating a new user note, this command can be used as an alternative to overtyping an area of storage on the Dump Storage display, for example if the note address is in a section of storage that is not shown due to *"same as above"* suppression.

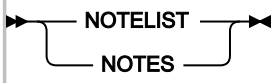
Using the NOTE command without parameters (defaulting to address 0) can be used to maintain general notes about the fault entry, such as a description of the problem or reminders about things still to be checked.

For more information, see Creating and managing user notes on page 212.

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.

Figure 48. Syntax



For more information, see Creating and managing user notes on page 212.

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 displays

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

The Dump Storage display

The previously displayed storage address (if any) is shown. The display is shown as it was when the previous address was selected, with scroll and cursor position maintained.

This command is usually assigned to the PF10 function key.

Figure 49. Syntax

QUIT

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

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

  This behavior is the equivalent to selecting "Exit Interactive Reanalysis" from the File pull-down menu.
- From the Fault Entry List display, the ISPF interface is terminated.

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

Figure 50. Syntax

REFRESH

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

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

Any MATCH command that might be active is reset.

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

Figure 51. Syntax

RESET

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

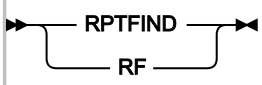
Causes the Fault Entry List column configuration to be changed to the configuration defined by the FAISPFopts(HistCols(...)) option in effect.

Figure 52. Syntax

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.

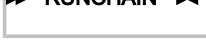
Figure 53. Syntax



RUNCHAIN

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

Figure 54. Syntax

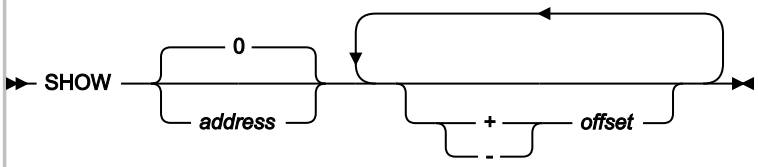


For information about how to use this command, see Displaying chained data areas on page 227.

SHOW

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

Figure 55. Syntax



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

```
SHOW 7F2300
```

The address parameter is 64-bit enabled. For details, see Specifying 64-bit addresses on page 142.

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

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

```
SHOW 142A0 + 1C0 + 1B - 8
```

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

SHOWFREE

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

Figure 56. Syntax

Sample output from the SHOWFREE command follows:

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

SIT

Displays the CICS® system initialization parameters during interactive reanalysis of a CICS fault entry.

Figure 57. Syntax

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.

Figure 58. Syntax

For information about how to use this command, see Converting STORE CLOCK values on page 231.

STGMAP

This command, which is only available from within the interactive report, can be used to show the Storage Map display.

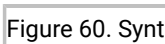
Figure 59. Syntax

For information about how to use this command, see Displaying the address space storage map (STGMAP) on page 218.

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 63).

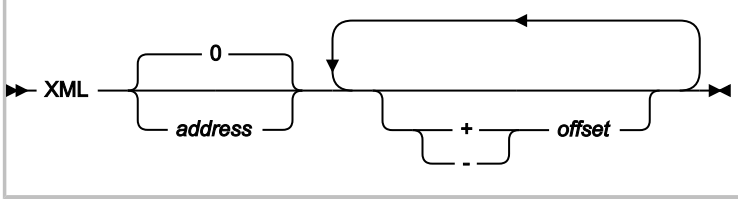
Figure 60. Syntax



XML

This command, which is only available from within the interactive report, can be used to display formatted XML data located at a specified storage address.

Figure 61. Syntax



For example, to display the XML data starting at address 007F2300, enter the following command:

```
XML 7F2300
```

The address parameter is 64-bit enabled. For details, see Specifying 64-bit addresses on page 142

Invoking this command displays a pop-up window where the Start Address field is pre-filled with the address you specified.

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

```
XML 142A0 + 1C0 + 1B - 8
```

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

For detailed information about how to use this command and interact with the results, see Displaying formatted XML (XML) on page 220.

Fault Analyzer preferences

To specify preferences that apply to the Fault Analyzer ISPF interface in general, select **Fault Analyzer Preferences** from the **Options** menu on the **Fault Entry List** display. This option opens the display shown in Figure 62: Sample Fault Analyzer Preferences display on page 113. (For general information about menu options, refer to Action-bar pull-down menus on page 94.)

Figure 62. Sample Fault Analyzer Preferences display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Fault Analyzer Preferences                                  Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR

Press PF3 to save options or PF12 to cancel.

Confirm Fault Entry Deletion: Y (Y/N)
Default Lock Flag Value . . : /
Locked Fault Entry
Highlighting. . . . . : Y (Y/N)
  Color . . . . . : R (B=Blue, G=Green, P=Pink, R=Red, T=Turquoise)
  Column. . . . . : L (F=Fault_ID, L=Lock, B=Both)
Expanded Duplicate Fault
Entry Highlighting. . . . . : Y (Y/N)
  Color . . . . . : P (B=Blue, G=Green, P=Pink, R=Red, T=Turquoise)
  Column. . . . . : B (F=Fault_ID, D=Dups, B=Both)

*** Bottom of data.
```

The options you can set are:

Confirm Fault Entry Deletion

N

The **Confirm Fault Entry Deletion** display is not shown each time fault entries are attempted to be deleted.

Y

The **Confirm Fault Entry Deletion** display is shown each time fault entries are attempted to be deleted.

You can also change this option from the confirmation display. See Deleting history file entries on page 126 for additional information about the **Confirm Fault Entry Deletion** display.

Default Lock Flag Value

One or two characters to be set as the lock flag value when using the L line command. If left blank, / is used.

The final value of the lock flag is subject to any changes made by the IDIXLOCK lock flag control exit. See Controlling fault entry lock flag values on page 128.

Locked Fault Entry Highlighting

A single character that controls the highlighting of locked fault entries in the Fault Entry List display. If left blank, N is used.

N

Locked fault entries are not highlighted.

Y

Locked fault entries are highlighted. Specifying Y displays additional options:

Color

A single character indicating the highlighting color of locked fault entries. If no color is specified, R is used.

B

Blue

G

Green

P

Pink

R

Red (default)

T

Turquoise

Column

A single character indicating the columns to highlight in the Fault Entry List display. If left blank, L is used.

F

The Fault_ID column

L

The Lock column (default)

B

Both the Fault_ID column and the Lock column

Expanded Duplicate Fault Entry Highlighting

A single character that controls the highlighting of expanded duplicate fault entries in the Fault Entry List display. If left blank, Y is used.

N

Expanded duplicate fault entries are not highlighted

Y

Expanded duplicate fault entries are highlighted. Specifying Y displays additional options:

Color

A single character indicating the highlighting color of expanded duplicate fault entries in the Fault Entry List display. If left blank, P is used.

B

Blue

G

Green

P

Pink (default)

R

Red

T

Turquoise

Column

A single character indicating the columns to highlight in the Fault Entry List display. If left blank, B is used.

F

The Fault_ID column

D

The Dups column

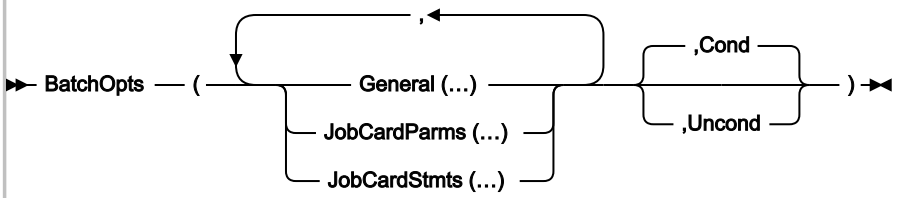
B

Both the Fault_ID column and the Dups column (default)

BatchOpts

Use the BATCHOPTS suboption to specify alternatives to the default settings in the Batch Reanalysis Options display. See Batch reanalysis options on page 149.

Figure 63. Syntax



BatchOpts COND and UNCOND suboptions

You can specify the COND and UNCOND suboptions at each level of the BATCHOPTS suboption:

- At the highest level, on the BATCHOPTS suboption
- At the intermediate level, on the GENERAL, JOBCARDPARMS, and JOBCARDSTMTS suboptions
- At the lowest level, on individual settings, for example JCLEDIT or MSGCLASS

COND (conditional) means the selected value is used as the default setting on the display if the user has not already specified a different setting. That is, it applies to a first-time user, or whenever the user clears the setting on the display.

UNCOND (unconditional) means the selected value is used as the current setting on the display at the start of any Fault Analyzer ISPF interface session, regardless of the previous setting. During the session, the setting can be changed to any other valid value, which remains in effect until the next session.

COND is the default at all levels, unless UNCOND is specified at a higher level. That is, the higher-level specification is by default propagated to each lower level.

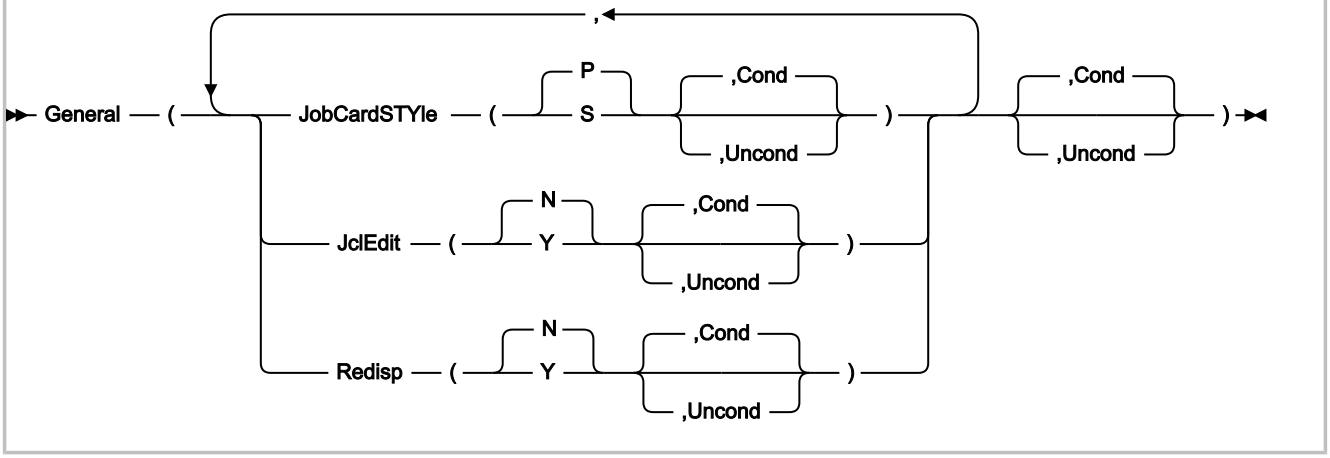
In the following example:

```
FAISPFOPTS(BATCHOPTS(GENERAL(JOBCARDSTYLE(P),JCLEDIT(Y),JOBCARDSTMTS(<stmt1>,<stmt2>),
    JOBCARDPARMS(MSGCLASS(A),COND),UNCOND))
```

- `JOBCARDPARMS(MSGCLASS(A))` is conditional.
- `GENERAL(JOBCARDSTYLE(P)`, `JCLEDIT(Y)`, and `JOBCARDSTMTS(<stmt1>,<stmt2>)` are unconditional.

BatchOpts General suboptions

Figure 64. Syntax



The GENERAL suboptions apply to the **General Options** section of the Batch Reanalysis Options display.

JOBCARDSTYLE(P | S [, COND | UNCOND]))

JCSTY(P | S [, C | U])

Specifies the value for the **Job card style** field in the Batch Reanalysis Options display:

P

Parameters

S

Statements

If not specified, the default is P.

JCLEDIT(Y | N [, COND | UNCOND]))

JE(Y | N [, C | U])

Specifies the value for the **Display panel to edit generated JCL** field in the Batch Reanalysis Options display:

Y

Yes

N

No

If not specified, the default is N.

REDISP(Y | N [, COND | UNCOND]))

R(Y | N [, C | U])

Specifies the value for the **Redisplay this panel before each reanalysis** field in the Batch Reanalysis Options display:

Y

Yes

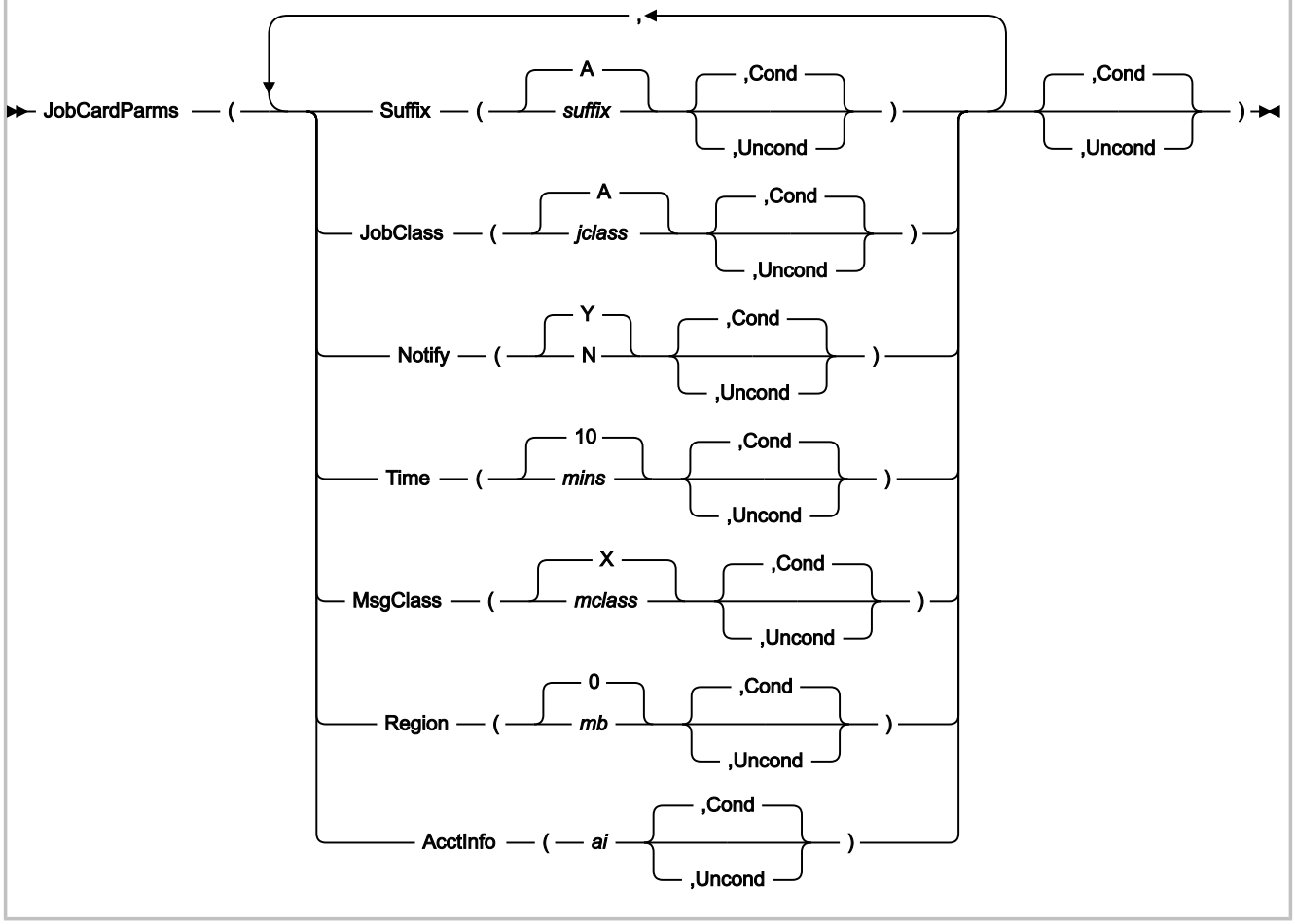
N

No

If not specified, the default is N.

BatchOpts JobCardParms suboptions

Figure 65. Syntax



The JOBCARDPARMS suboptions apply to the **Job Card Parameters** section of the Batch Reanalysis Options display.

SUFFIX(*suffix*[, COND | UNCOND]))

S(*suffix*[, C | U])

Specifies the value of the **Job name suffix** field in the Batch Reanalysis Options display:

suffix

A-Z, 0-9, @, #, or $

If not specified, the default is A.

JOBCLASS(*jclass*[, COND | UNCOND]))

JC(*jclass*[, C | U])

Specifies the value for the **Job class** field in the Batch Reanalysis Options display:

jclass

A-Z or 0-9

If not specified, the default is A.

NOTIFY(Y | N [, COND | UNCOND]))

N(Y | N [, C | U])

Specifies the value for the **Job notify** field in the Batch Reanalysis Options display:

Y

Yes

N

No

If not specified, the default is Y.

TIME(*mins*[, COND | UNCOND]))

T(*mins*[, C | U])

Specifies the value for the **Job time minutes** field in the Batch Reanalysis Options display:

mins

0-99

If not specified, the default is 10.

MSGCLASS(*mclass*[, COND | UNCOND]))

MC(*mclass*[, C | U])

Specifies the value for the **Message class** field in the Batch Reanalysis Options display:

mclass

A-Z or 0-9

If not specified, the default is X.

REGION(*mb*[, COND | UNCOND]))

R(*mb*[, C | U])

Specifies the value for the **Region megabytes** field in the Batch Reanalysis Options display:

mb

0-2047

If not specified, the default is 0.

ACCTINFO(*ai*[, COND | UNCOND]))

AI(*ai*[, C | U])

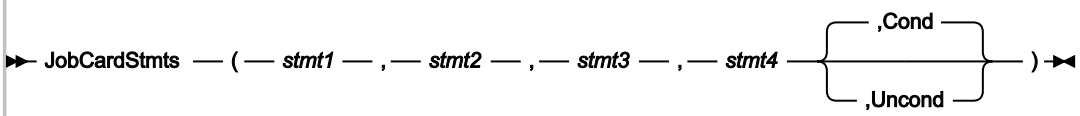
Specifies the value for the **Accounting info** field in the Batch Reanalysis Options display:

ai

Up to 50 characters.

BatchOpts JobCardStmts suboptions

Figure 66. Syntax



You can specify up to four comma-delimited JCL statements.

- If a statement is not specified (for example, if you need only 3 statements), you must still specify the delimiting comma of the omitted statement.
- If a statement contains blanks or commas, enclose it in either single quotes or double quotes. If the statement already contains quotes of the same type, double them.

  For example, to create the JCL statement:

  ```
  //MYJOB JOB (123),'X Y Z',
  ```

  You can specify any of the following:

  ```
  '//MYJOB JOB (123),"X Y Z",'
  ```

  ```
  "//MYJOB JOB (123),'X Y Z',"
  ```

  ```
  '//MYJOB JOB (123),''X Y Z'','
  ```

  In the last example, two single quotes surround `XYZ`.

Viewing a saved report

To view the saved report that is associated with a fault, enter **V** (or **S**) against the entry in the history file display. Figure 67: Sample Saved Report display on page 121 shows a sample Saved Report display.

Figure 67. 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® V16R1M4 (UI97160  2024/06/04)
*
* Copyright IBM Corp. 2000, 2017.  All rights reserved.
* Copyright HCL Technologies Ltd 2017, 2024.  All rights reserved.
*******************************************************************************

JOBNAME: IDIVPPL2  SYSTEM ABEND: 0C9              FAE1      2024/06/22  09:28:


+ <H1> S Y N O P S I S
+ <H1> E V E N T   S U M M A R Y
+ <H1> 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
```

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

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

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

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

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

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

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

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

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

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

Adding or removing blank lines

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

The View menu Add Blank Lines and Remove Blank Lines options control the insertion or elimination of blank lines respectively. These action-bar pull-down menu options are available from most Fault Analyzer displays. (For information about Fault Analyzer action-bar pull-down menus in general, see Action-bar pull-down menus on page 94.)

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 94).

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

The default setting is to add help text.

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

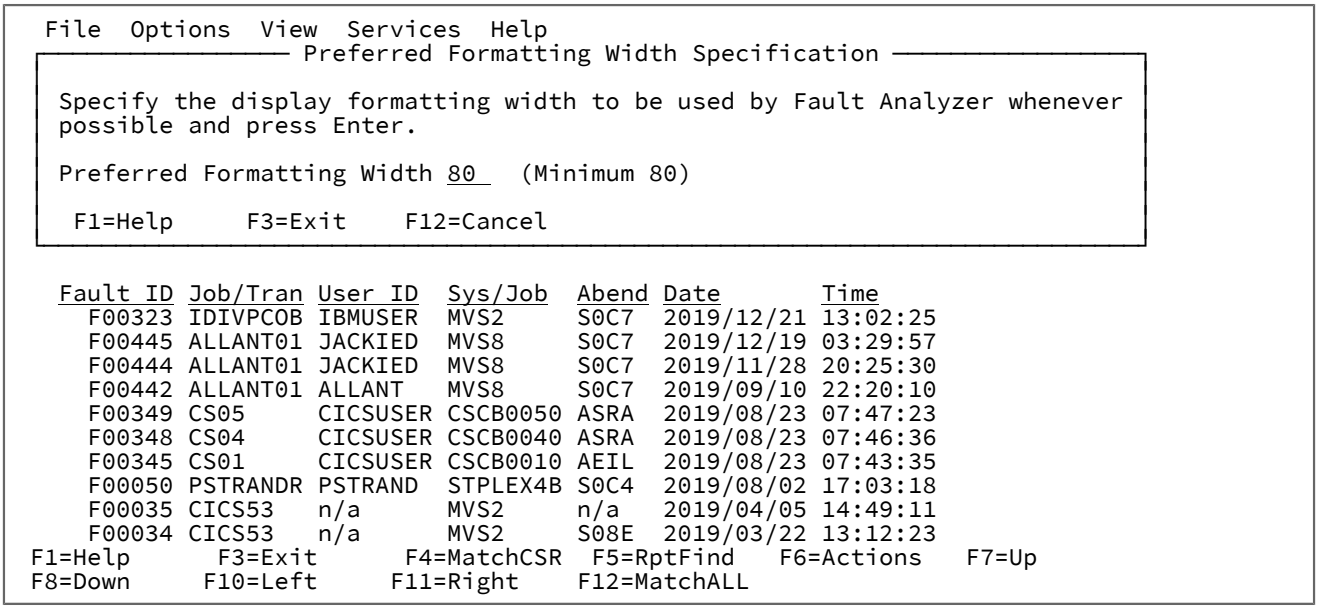
Setting preferred formatting width

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

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 94).

Figure 68. Sample Preferred Formatting Width display



The minimum accepted width is 80 characters, which is also the default. Although a value of 999 can be specified, the maximum width is at all times limited by the actual display width.

After typing the desired formatting width, press the Enter key to return to the previous display using the specified formatting width.

If the formatting width exceeds the physical width of your screen, then you can use the scroll commands, LEFT (PF10) and RIGHT (PF11) to view the entire display.

To return to the previous display without changing the formatting width, enter the EXIT (PF3) or CANCEL (PF12) command.

Displaying user-selected message or abend code explanations

The LOOKUP command can be issued from the interactive fault history file display or the interactive fault reanalysis report. The command can be used to display explanations for user-selected message IDs or abend codes. It can also be used to display other information, such as DB2® SQLCODEs or VSAM feedback codes. (For the command syntax, see LOOKUP on page 106.)

For example, if entering the command:

```
LOOKUP IEC141I
```

a panel containing the message explanation is displayed as shown in Figure 69: Sample Message ID Look-Up display on page 124.

Figure 69. Sample Message ID Look-Up display

```
Message IEC141I Explanation                                  Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR


  IEC141I

  IEC141I 013-rc,mod,jjj,sss, ddname[-#] [,dev,volser, dsname]

  Explanation:  An error occurred during the processing of an OPEN macro.
  System completion code 013, with the return code, accompanies this
  message.

  In the message text:

  rc
       The return code.

  mod
       The name of the module in which the error was detected.

  jjj
       The job name.

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

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:

Figure 70. Sample Lookup Search and Browse display

```
Lookup Search and Browse                                     Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR

Either search for abend codes, messages, and miscellaneous information by
typing a pattern, or browse such information using the expand/collapse browser
below.

Search. . . . . . . . . . . : ______________________________

+ Abend Codes
+ Messages
+ Miscellaneous Information









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

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 106), or you can navigate to the desired information by using the expand (+) and collapse (-) point-and-shoot fields.

By placing the cursor on a + (expand) point-and-shoot field, the available subcategories are shown. For example, if placing the cursor on the + sign next to Abend Codes in Figure 70: Sample Lookup Search and Browse display on page 124 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 provided by Fault Analyzer, or specified in member IDIHUSRM of the IDI.SIDISAM1 data set, can be entered.

Copying interactive displays to a file

The COPY command can be used to write a copy of the current display (which includes the entire scrollable area) to a sequential data set or a member in a partitioned data set. (For the command syntax, see COPY on page 100.)

For example, if entering the command:

```
COPY PRINT(A)
```

the current display is written to member A in the partitioned data set '*prefix*.PRINT', where *prefix* is the current TSO prefix.

If the COPY command is entered without any parameters, a pop-up panel from which the data set name can be specified is displayed.

Displaying product copyright, license, and version information

Fault Analyzer copyright and general usage information is available by selecting Help > About Fault Analyzer. (See Action-bar pull-down menus on page 94). The screen also displays the installed version, release, maintenance level, and applied fixes of the installed Fault Analyzer product.

Deleting history file entries

A fault entry can be deleted from a history file automatically or explicitly.

Automatic fault entry deletion

Fault Analyzer automatically deletes the oldest, unlocked fault entry from a history file when the history file contains its maximum number of fault entries and a new fault occurs. To use ISPF to configure the number of fault entries in a history file, click **File > Fault History File Properties**.

Explicit fault entry deletion

You can use the ISPF Fault Entry List display to explicitly delete a fault entry from a history file by entering the D line command against the entry. You can also delete a range of fault history entries. Enter DD next to the first and last fault history entries to delete the two entries and all of the entries displayed between them. You can use a match to first group together the entries you want to delete. (See Sorting and matching fault entries on page 71.) When you delete a range of a matched set of entries, only the entries displayed on the screen are deleted. You do not delete any entries that are interleaved with the displayed entries in the original history file but are not currently displayed.

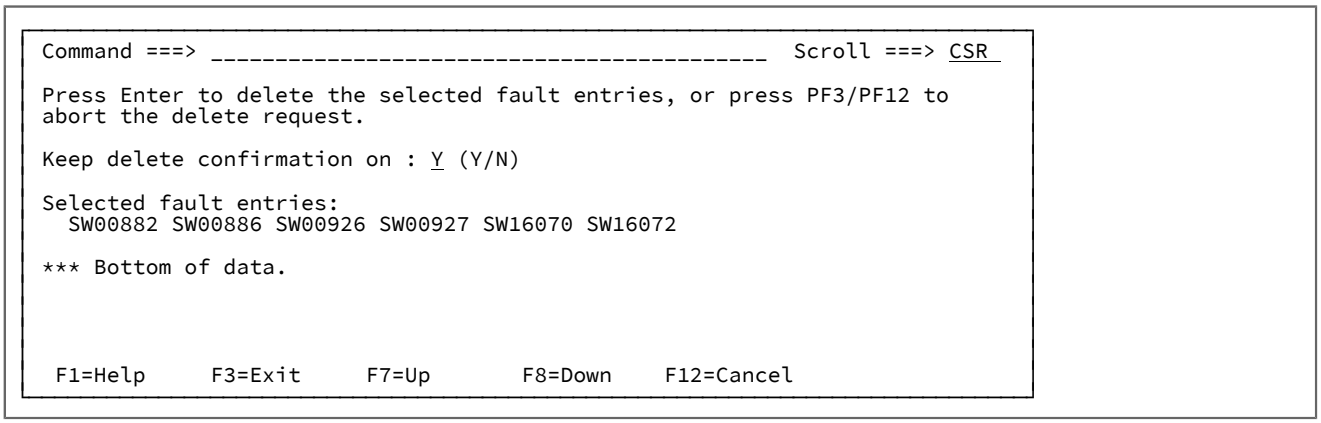
A fault entry is not deleted if the entry's lock flag is set to a non-numerical, non-blank value, or if the specified number of days since the fault entry's creation has not elapsed. If a fault entry cannot be deleted due to insufficient access authority, or because the fault entry is locked, the reason is displayed. See Locking fault entries on page 127 and Viewing fault entry information on page 129.

You can also delete a fault entry with the IDIUTIL utility. See the DELETE control statement on page 444.

Configuring confirmation of fault-entry deletion

By default, Fault Analyzer prompts for confirmation before deleting the selected fault entries.

Figure 71. Sample Confirm Fault Entry Deletion display



To bypass the confirmation display, set **Keep delete confirmation on** to N.

To restore the confirmation display, set the **Confirm Fault Entry Deletion** option to Y. See Fault Analyzer preferences on page 113.

Locking fault entries

You can lock a fault entry to prevent it from being deleted from a history file. You can specify the number of days that the fault entry is to remain locked, or specify that the lock never expires (the fault entry is never deleted). You can highlight locked fault entries in the Fault Entry List display. The optional IDIXLOCK exit can control fault entry flag values across an installation.

By default, fault entries are not locked. You must have UPDATE access to a fault entry to change its lock flag.

Locking a fault entry from the Fault Entry List display

From the Fault Entry List display, you can apply the L line command against a fault entry to lock the entry using the default lock configuration. See Fault Analyzer preferences on page 113 for information about setting the default lock configuration.

Locking a fault entry from the Fault Entry Information display

You can configure a fault entry lock from the Fault Entry Information display.

1. Do one of the following to display the Fault Entry Information:
    - In the Fault Entry List display, enter the ? line command against the fault entry.
    - In the Interactive Reanalysis report, do either of the following:
        - Issue the INFO command. See INFO on page 106 for details.
        - Select **File > Fault Entry Information** from the action-bar. See Action-bar pull-down menus on page 94.
2. Configure the Lock Flag. See Viewing fault entry information on page 129.

Locking a fault entry through a user exit

You can lock a fault entry through a user exit by using the ENV.LOCK_FLAG field. See ENV - Common exit environment information on page 653.

Controlling fault entry lock flag values

Optionally, a systems programmer can control fault entry locking throughout the installation. For example, they can limit the number of days fault entries can be locked or prevent fault entries from being locked indefinitely. When a fault entry lock flag changes (whether through an Analysis Control user exit or through the Fault Analyzer ISPF interface), control can pass to the optional IDIXLOCK exit routine. The IDIXLOCK exit can accept the change or replace it with a different value.

The IDIXLOCK exit can impose different restrictions based on user ID or security server group ID. For example, the IDIXLOCK exit could permit an administrator to set any lock flag value, while restricting others to installation-wide rules.

Member IDISLOCK in data set IDI.SIDISAM1 includes a job that creates a sample exit. The exit must be a load module named IDIXLOCK in IDI.SIDIAUTH or in another APF-authorized library. Include the library in LNKLST to ensure the IDIXLOCK load module can be found by all jobs.

Highlighting locked fault entries

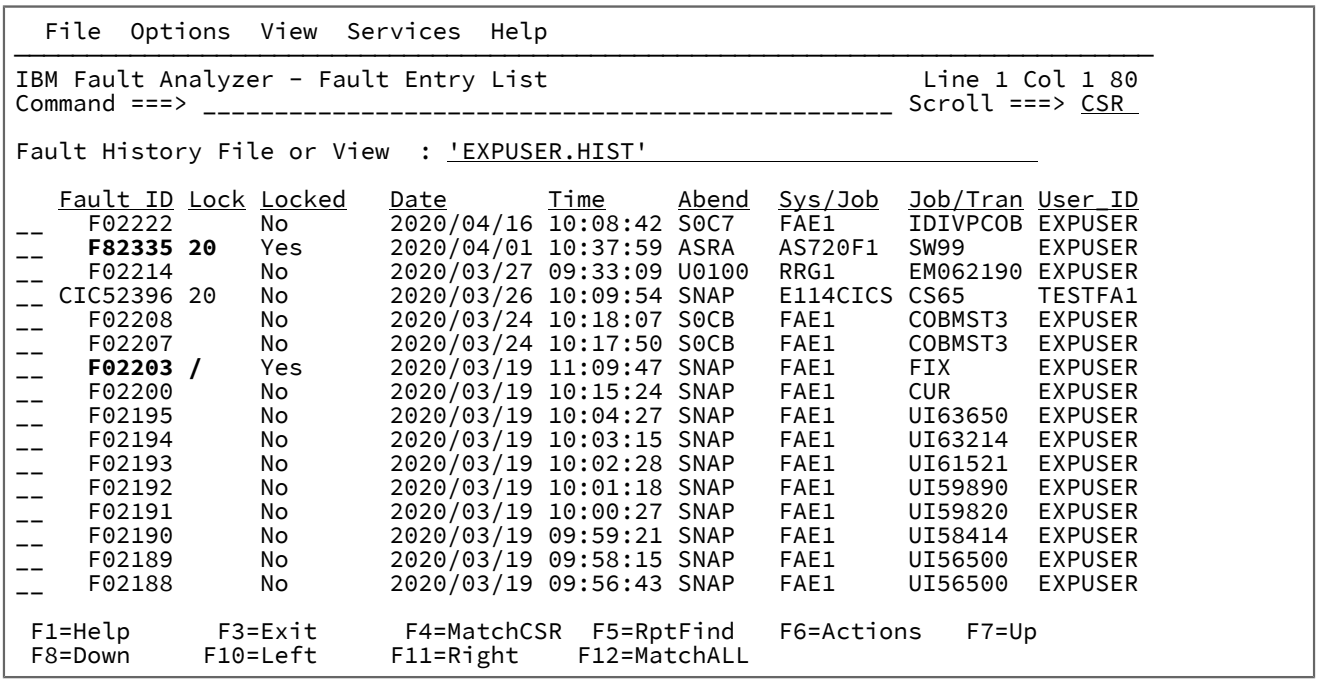
You can use text color to highlight the Fault ID and Lock columns of locked fault entries in the Fault Entry List display. Highlighting can make locked fault entries easier to find in the display. If you want to conserve screen space, you can indicate locked entries with highlighting and exclude the Locked column from the display. (See Fault entry list column configuration on page 65.) Conversely, the Locked column is a text alternative to highlighting that can be read by screen readers and those with color vision deficiency. You can sort fault entries based on the Locked column, but not based on highlighting.

For details about how to configure highlighting for locked fault entries, see Fault Analyzer preferences on page 113.

In the following example:

- Bold text indicates the highlighted fault-entry columns (which display the configured text color in a live screen).
- Highlighting is configured for both the Fault_ID column and the Lock column of locked fault entries.
- The Fault Entry List display shows that Fault IDs F82335 and F02203 are actively locked.
- The lock on Fault ID F82335 is active and extends for 20 days from the date when the fault occurred.
- The lock on Fault ID F02203 is active and has no expiration date.
- The 20-day lock on Fault ID CIC52396 has expired.

Figure 72. Example Fault Entry List with locked entries highlighted



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

Fault History File or View  : 'EXPUSER.HIST'_____________________________

   Fault ID Lock Locked   Date       Time     Abend  Sys/Job  Job/Tran User_ID
__   F02222      No       2020/04/16 10:08:42 S0C7   FAE1     IDIVPCOB EXPUSER
__   F82335 20   Yes      2020/04/01 10:37:59 ASRA   AS720F1  SW99     EXPUSER
__   F02214      No       2020/03/27 09:33:09 U0100  RRG1     EM062190 EXPUSER
__ CIC52396 20   No       2020/03/26 10:09:54 SNAP   E114CICS CS65     TESTFA1
__   F02208      No       2020/03/24 10:18:07 S0CB   FAE1     COBMST3  EXPUSER
__   F02207      No       2020/03/24 10:17:50 S0CB   FAE1     COBMST3  EXPUSER
__   F02203 /    Yes      2020/03/19 11:09:47 SNAP   FAE1     FIX      EXPUSER
__   F02200      No       2020/03/19 10:15:24 SNAP   FAE1     CUR      EXPUSER
__   F02195      No       2020/03/19 10:04:27 SNAP   FAE1     UI63650  EXPUSER
__   F02194      No       2020/03/19 10:03:15 SNAP   FAE1     UI63214  EXPUSER
__   F02193      No       2020/03/19 10:02:28 SNAP   FAE1     UI61521  EXPUSER
__   F02192      No       2020/03/19 10:01:18 SNAP   FAE1     UI59890  EXPUSER
__   F02191      No       2020/03/19 10:00:27 SNAP   FAE1     UI59820  EXPUSER
__   F02190      No       2020/03/19 09:59:21 SNAP   FAE1     UI58414  EXPUSER
__   F02189      No       2020/03/19 09:58:15 SNAP   FAE1     UI56500  EXPUSER
__   F02188      No       2020/03/19 09:56:43 SNAP   FAE1     UI56500  EXPUSER

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

Viewing fault entry information

The history file is displayed with one line of information per entry. Therefore, not all information pertaining to a fault can be displayed simultaneously.

To view more information, enter the **?** line command against the entry on the Fault Entry List display. From within the interactive reanalysis report, the same can be accomplished by either issuing the INFO command (see INFO on page 106) or by selecting **Fault Entry Information** from the action-bar **File** pull-down menu (see Action-bar pull-down menus on page 94).

An example of the fault entry information display follows:

Figure 73. Sample Fault Entry Information display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Fault Entry Information                                    Line 1 Col 1 80
Command ===> _____________________________________________ Scroll ===> CSR


Fault ID. . . . . . . . . . : BAT00009
User Name . . . . . . . . . : ________
User Title. . . . . . . . . : ____________________________________________
Lock Flag . . . . . . . . . : __ (Not locked)
Lock/Unlock User ID . . . . :
Abend Code. . . . . . . . . : S0C4
POF Module Name . . . . . . : IBMBLIIA
POF Program Name. . . . . . : IBMBLII1
POF Offset. . . . . . . . . : 96C
Abend Date. . . . . . . . . : 2019/09/22
Abend Time. . . . . . . . . : 15:03:02
Job Name. . . . . . . . . . : PLI23
Job ID. . . . . . . . . . . : JOB30836
Job Execution Class . . . . : A
Job Type. . . . . . . . . . : Batch
Job Step Name . . . . . . . : GO
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 2019/10/22 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
```

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

interactive reanalysis report. To change the contents of this field, remember to press the Enter key prior to returning with PF3.

Lock Flag

A user-maintained input/output field that provides a mechanism to prevent accidental deletion of the fault entry.

If this field contains a numeric value 0 - 99, then fault entry expiration control is active. The specified value is the number of days that the fault entry remains locked, following its creation. The time of the day is not considered. If, for example, a value of 1 is specified when the fault entry is created, it is unlocked at midnight on the same day. It follows that a value of 0 means that the fault entry is not locked.

If fault entry expiration control is active, but has not yet expired, or if setting this flag to any other non-blank character, then the following is not performed against the fault entry:

- IDIUTIL DELETE processing (for details, see DELETE control statement on page 444).

  Note: The default is to not delete the fault entry, but this default can be overridden by using an IDIUTIL user exit (for details, see IDIUTIL Delete user exit on page 510).

- 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 126).
- Automatic deletion due to the maximum number of fault entries for a history file having been reached, and the fault entry which is locked is the oldest fault entry in the history file. In this case, the search for a fault entry to delete continues in chronological order until the oldest fault entry that is not locked has been located.

By default, the lock flag is set to a blank, which does not prevent deletion of the fault entry.

Any printable character can be entered for this field:

- If a non-printable character is entered, then it is changed to a '/'.
- If a lowercase character is entered, then it is translated to uppercase.

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

The lock flag can also be modified by:

- Any user exit, by changing the value in the ENV.LOCK_FLAG field. See ENV - Common exit environment information on page 653.
- The optional IDIXLOCK exit. See Controlling fault entry lock flag values on page 128.

Lock/Unlock User ID

Identifies the user ID who last changed the Lock Flag setting.

Abend Code

The initial (if more than one) abend code.

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

POF Module Name

The point-of-failure module name.

POF Program Name

The point-of-failure program name.

POF Offset

The point-of-failure offset.

Abend Date

The date when the abend occurred.

Abend Time

The time when the abend occurred.

Job Name

The name of the batch job, started task, or TSO user ID that caused the entry to be written.

Job ID

The JES ID of the abending job.

Job Execution Class

The JES execution class of the abending job.

Job Type

The abending job type.

Job Step Name

The name of the job step that caused the entry to be written.

EXEC Program Name

The program name that is specified on the JCL EXEC statement of the abending job step.

User ID

The user ID associated with the abending job.

System Name

The system name on which the abend occurred.

CICS® Transaction ID

If a CICS® transaction fault, the ID of the CICS® transaction that caused the entry to be written.

CICS® Task Number

If a CICS® transaction fault, the task number that caused the entry to be written.

CICS® Terminal ID

If a CICS® transaction fault, the CICS® terminal ID.

CICS® Terminal Netname

If a CICS® transaction fault, the CICS® terminal netname.

IMS™ Program Name

IMS™ program name.

Accounting Information

Job accounting information.

Duplicate Count

The number of duplicate faults that have occurred, using the same fault history file, since the recording of this fault.

Note: This value might not always include all duplicate faults that have occurred as it is maintained in the volatile history file cache only for performance reasons.

If a non-zero value is displayed, and duplicate details are available for the fault, then the value becomes a point-and-shoot field, which can be selected by placing the cursor on it, and pressing Enter. This selection results in the Fault Entry Duplicate History display being shown (for details, see Viewing the fault entry duplicate history on page 134).

Minidump Pages

The number of minidump pages written to the history file for this fault.

History File Data Set Name

The name of the history file data set containing the viewed fault entry.

Java™ DTFJ Processing Status

For a fault entry for which Java™ activity was detected, this field shows the status of the asynchronous Java™ DTFJ processing as one of the following:

- Dump error
- Pending
- Started *date time*
- Finished, elapsed seconds *seconds*

MVS™ Dump Data Set Name

The associated dump data set name, if any.

MVS™ Dump Status

The MVS™ dump data set status.

The MVS™ dump data set status field shows you information about the dump that can be associated with the current fault. Possible values of this field are:

Available

An MVS dump data set is associated with the fault entry and the data set exists.

Not found

An MVS dump data set was associated with the fault entry but the data set no longer exists.

When you exit from the Fault Entry Information display, after you have changed either of the user-managed fields, a prompt is displayed to confirm that you want to update the fault entry with the new information. 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:

- Entering the H line command against a fault entry from the Fault Entry List display.
- Placing the cursor on the Fault Entry List display Dups column value for a fault entry, when this entry is presented as a point-and-shoot field, and pressing Enter.
- Issuing the DUPS primary command from within the interactive reanalysis report.
- Placing the cursor on the Fault Entry Information display Duplicate Count value, when this value is presented as a point-and-shoot field, and pressing Enter.

An example of a Fault Entry Duplicate History display follows:

Figure 74. Sample Fault Entry Duplicate History display

```
  File  View  Services  Help
_____________________________________________________________________________
Fault Entry Duplicate History                                  End of data
Command ===> _____________________________________________ Scroll ===> CSR

Most recent duplicate
  occurred. . . . . . . . . . : 2019/06/28 15:22:13
Initial abend occurred. . . : 2019/06/28 15:20:55
Total duplicate count . . . : 4

Duplicate details in reverse chronological order:

Date       Time     Jobname  Job_ID   System   Dup Type User_ID  Stepname
2019/06/28 15:22:13 CICS5FA1 JOB48697 FAE1     Normal   SIMCOC2  CICS5FA1
2019/06/28 15:22:02 CICS5FA1 JOB48697 FAE1     Normal   SIMCOC2  CICS5FA1

Date       Time     Jobname  Job_ID   System   Dup Type
2019/06/28 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
```

The following information is provided:

Most recent duplicate occurred

The local date and time when the most recent duplicate of this fault occurred. The formatting of the date and time is subject to the Locale option in effect.

Initial abend occurred

The local date and time when the initial fault that caused the entry to be written occurred. The formatting of the date and time is subject to the Locale option in effect.

Total duplicate count

The total number of duplicates that have been registered against this fault.

Duplicate details in reverse chronological order

Details of the duplicate occurrences.

Depending on whether the duplicates resulted from NoDup(Normal) or NoDup(CICSFast) processing, the information that is provided differs:

- NoDup(Normal) duplicates

For NoDup(Normal) duplicates, the information that is provided for each instance is comprised of:

Date

The date when the duplicate occurred. The formatting of the date is subject to the Locale option in effect.

Time

The time when the duplicate occurred. The formatting of the time is subject to the Locale option in effect.

Jobname

The name of the job that caused the duplicate fault.

Job ID

The JES job ID of the job that caused the duplicate fault.

System

The name of the system on which the job that caused the duplicate fault was executing.

Dup Type

The type of duplicate (always "Normal" for NoDup(Normal) duplicates).

User ID

The user ID that is associated with the job that caused the duplicate fault.

Stepname

The name of the job step that caused the duplicate fault.

Multiple consecutive NoDup(Normal) occurrences are grouped under a common heading.

- NoDup(CICSFast) duplicates

  For NoDup(CICSFast) duplicates, the information that is provided consists of:

  - A common section, with details that are applicable to all duplicate faults (one or more) that occurred within the recording interval:

    Date

    The date when the first duplicate occurred within the recording interval. The formatting of the date is subject to the Locale option in effect.

    Time

    The time when the first duplicate occurred within the recording interval. The formatting of the time is subject to the Locale option in effect.

    Jobname

    The name of the job that caused the duplicate faults.

    Job ID

    The JES job ID of the CICS® region that caused the duplicate faults.

    System

    The name of the system on which the CICS® region that caused the duplicate faults was executing.

Dup Type

The type of duplicates (always "Fast" for NoDup(CICSFast) duplicates).

- Detailed information of up to 10 unique faults that occurred within the recording interval:

User ID

The CICS® user ID that caused the duplicate faults.

Terminal ID

The CICS® terminal ID that caused the duplicate faults.

Count

The total number of occurrences of this unique combination of CICS® user ID and terminal ID that occurred within the recording interval.

Note: If the sum total of all "Count" values does not equal the total number of duplicates that occurred within the recording interval, then a note is provided with information about the number of duplicates for which details are not available.

This situation can happen if duplicate faults for more than 10 unique combinations of CICS® user IDs and terminal IDs occurred during the recording interval.

Note: If the sum total of all duplicate faults shown does not equal the total duplicate count for the fault entry, then a note is provided with information about the number of duplicates for which details are not available.

This situation can happen if more duplicate faults have occurred than can be recorded with details in the fault entry.

Expanding and collapsing duplicate fault entries

You can expand duplicate fault entry information into separate fault entries instead of viewing it as described in Viewing the fault entry duplicate history on page 134. In the expanded view, fault entries that have a duplicate count greater than 0 display the original fault entry and its duplicates as individual fault entries. With the expanded view, you can see the occurrences of each duplicate fault over time and in the context of other faults.

Figure 75: Sample Fault Entry List display showing collapsed duplicates on page 138 shows the default collapsed view. Fault entries F02417, F61327, and F82311 have duplicate instances.

Figure 75. Sample Fault Entry List display showing collapsed duplicates

```
  File  Options  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
IBM Fault Analyzer - Fault Entry List                        Refresh complete
Command ===> _____________________________________________ Scroll ===> CSR

Fault History File or View  : 'XMPUSER.HIST.ABC'

   Fault ID Dups  Lock MD Pages Date Time          System  Abend  Jobname  J
──   F02418                82 2020/09/14 11:26:31 FAE1    S0C7   IDIVPCOB B
──   F02422               100 2020/09/14 11:25:52 FAE1    S06F   IDISSE1  S
──   F02417     1          84 2020/09/08 10:12:13 FAE1    S0C7   IDIVPCOB B
──   F61327     4          89 2020/09/08 10:06:39 FAE1    S0C7   FADUP    B
──   F02416                85 2020/08/31 18:48:52 FAE1    S0C7   IDIVPCOB B
──   F02143       /        52 2020/02/10 13:14:46 FAE1    S0C7   S0C7A    B
──   F02140                45 2020/02/05 12:08:53 FAE1    S0C7   S0C7A    B
──   F82311    12 /       276 2020/02/05 08:54:45 FAE1    ASRA   AS720F1  C
──   F02139       25       52 2020/02/04 12:39:34 FAE1    S0C4   ASM64B   B
──   F02138       25       80 2020/02/04 12:36:19 FAE1    S0C1   ASMLE64  B
──   F02137       25       52 2020/02/04 09:44:54 FAE1    S0C4   ASM64B   B

**  Bottom of data.
```

To expand the duplicate fault entries as shown in Figure 76: Sample Fault Entry List display showing expanded duplicates on page 138, select **View > Expand Duplicate Fault Entries**.

To more easily identify expanded duplicate fault entries, select the desired highlighting options from the Fault Analyzer Preferences display. See Fault Analyzer preferences on page 113.

Figure 76. Sample Fault Entry List display showing expanded duplicates

```
  File  Options  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
IBM Fault Analyzer - Fault Entry List (Expanded)              Line 1 Col 1 80
Command ===> _____________________________________________ Scroll ===> CSR

Fault History File or View  : 'XMPUSER.HIST.ABC'

   Fault ID Dups  Lock MD Pages Date Time          System  Abend  Jobname  J
──   F02418                82 2020/09/14 11:26:31 FAE1    S0C7   IDIVPCOB B
──   F02422               100 2020/09/14 11:25:52 FAE1    S06F   IDISSE1  S
──   F02417     1          84 2020/09/08 10:12:37 FAE1    S0C7   FADUP    B
──   F02417     1          84 2020/09/08 10:12:13 FAE1    S0C7   IDIVPCOB B
──   F61327     1          89 2020/09/08 10:11:30 FAE1    S0C7   FADUP    B
──   F61327     1          89 2020/09/08 10:06:47 FAE1    S0C7   FADUP    B
──   F61327     1          89 2020/09/08 10:06:46 FAE1    S0C7   FADUP    B
──   F61327     1          89 2020/09/08 10:06:46 FAE1    S0C7   FADUP    B
──   F61327     4          89 2020/09/08 10:06:39 FAE1    S0C7   FADUP    B
──   F02416                85 2020/08/31 18:48:52 FAE1    S0C7   IDIVPCOB B
──   F02143       /        52 2020/02/10 13:14:46 FAE1    S0C7   S0C7A    B
──   F02140                45 2020/02/05 12:08:53 FAE1    S0C7   S0C7A    B
──   F82311    11 /       276 2020/02/05 08:57:13 FAE1    ASRA   AS720F1  C
──   F82311     1 /       276 2020/02/05 08:57:13 FAE1    ASRA   AS720F1  C
──   F82311    12 /       276 2020/02/05 08:54:45 FAE1    ASRA   AS720F1  C
──   F02139       25       52 2020/02/04 12:39:34 FAE1    S0C4   ASM64B   B
```

Copying history file entries

You copy by entering the line command C against the fault entry in the Fault Entry List display.

You can also copy a range of fault history entries, using two CC range markers. When you copy a range of a matched set of entries, you only copy the entries displayed on the screen. You do not copy any entries that are not displayed, but are interleaved with the displayed entries on the original history file.

After pressing the Enter key, a display showing the list of fault entries that are selected for copy with the C or CC commands is presented. Included are also any fault entries that are selected for move with the M or MM commands. The display provides a field for specification of a history file to which the fault entries should be copied. This field is initialized with the name of the history file last specified. The following is an example of the Specify Move/Copy Options display that is shown after pressing the Enter key.

A separate option, **Include Tightly Coupled Dump Data Sets**, has these possible settings:

All

Specifies that all associated dump data sets that uniquely belong to the selected fault entries will be duplicated.

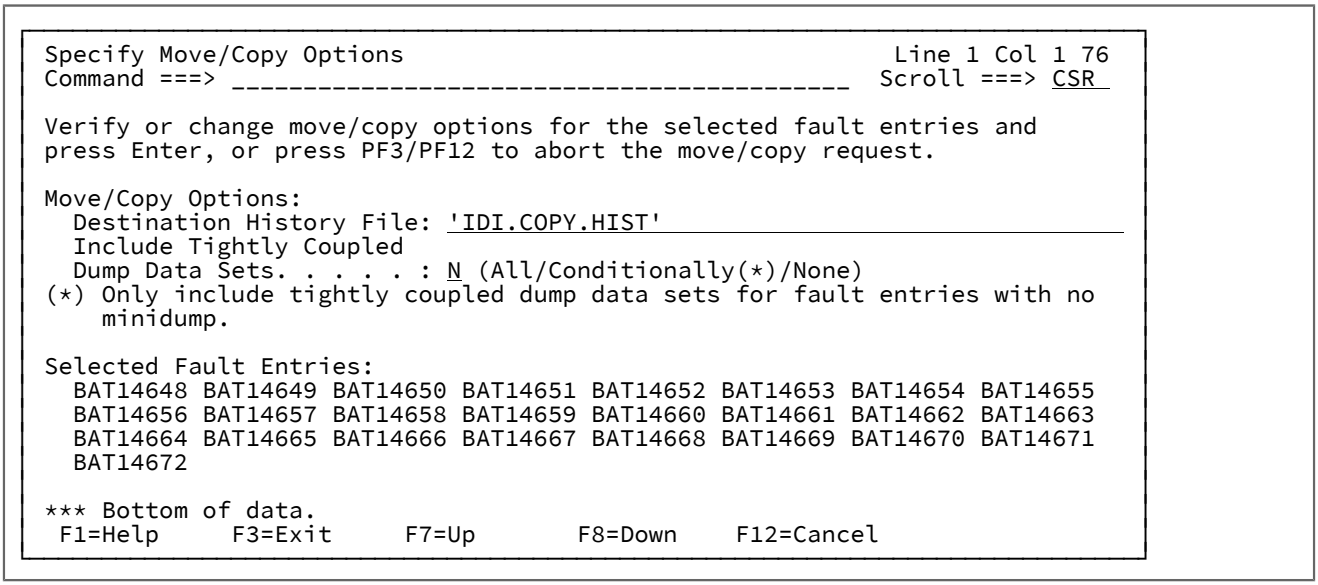
Conditionally

Specifies that all associated dump data sets that uniquely belong to the selected fault entries will only be duplicated if a fault entry does not include a minidump. This might be the case, if for example a Recovery Fault Recording (RFR) fault entry has not yet been analyzed.

None

Specifies that no associated dump data sets will be copied.

Figure 77. Sample Specify Move/Copy Options display



The destination history file that is specified is checked for UPDATE access authorization prior to commencing the move/copy processing. 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 specified history file does not exist, then the New History File Allocation display is shown (for details, see New history file allocation on page 87).

The original fault ID is preserved, if possible. However, if an identical fault ID already exists in the destination history file, then the fault number is incremented to the next available fault ID.

If the copy of one or more fault entries was not successful, then a display detailing the reasons is shown.

The TSO user's access authorization level required to copy a fault entry is READ.

Moving history file entries

Use the M line command in the Fault Entry List display to move a fault entry.

You can also move a range of fault history entries, using two MM range markers. When you move a range of a matched set of entries, you only move the entries displayed on the screen. You do not move any entries that are not displayed, but are interleaved with the displayed entries on the original history file.

After pressing the Enter key, a display showing the list of fault entries that are selected for move with the M or MM commands is presented. Included are also any fault entries that are selected for copy with the C or CC commands. The display provides a field for specification of a history file to which the fault entries should be moved. This field is initialized with the name of the history file last specified. See Copying history file entries on page 138 for an example of the Specify Move/Copy Options display that is shown after pressing the Enter key.

The destination history file that is specified is checked for UPDATE access authorization prior to commencing the move/copy processing. If the specified history file does not exist, then the New History File Allocation display is shown (for details, see New history file allocation on page 87).

The original fault ID is preserved, if possible. However, if an identical fault ID already exists in the destination history file, then the fault number is incremented to the next available fault ID.

If the move of one or more fault entries was not successful, then a display detailing the reasons is shown.

A locked fault entry cannot be moved. If you need to move a locked fault entry, then first issue the '?' line command and unlock the fault entry from the Fault Entry Information display by changing the Lock Flag value to blanks. If required, lock the moved fault entry again by changing the Lock Flag value to a non-blank value. For details of the values that are permitted for the Lock Flag, see Viewing fault entry information on page 129.

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

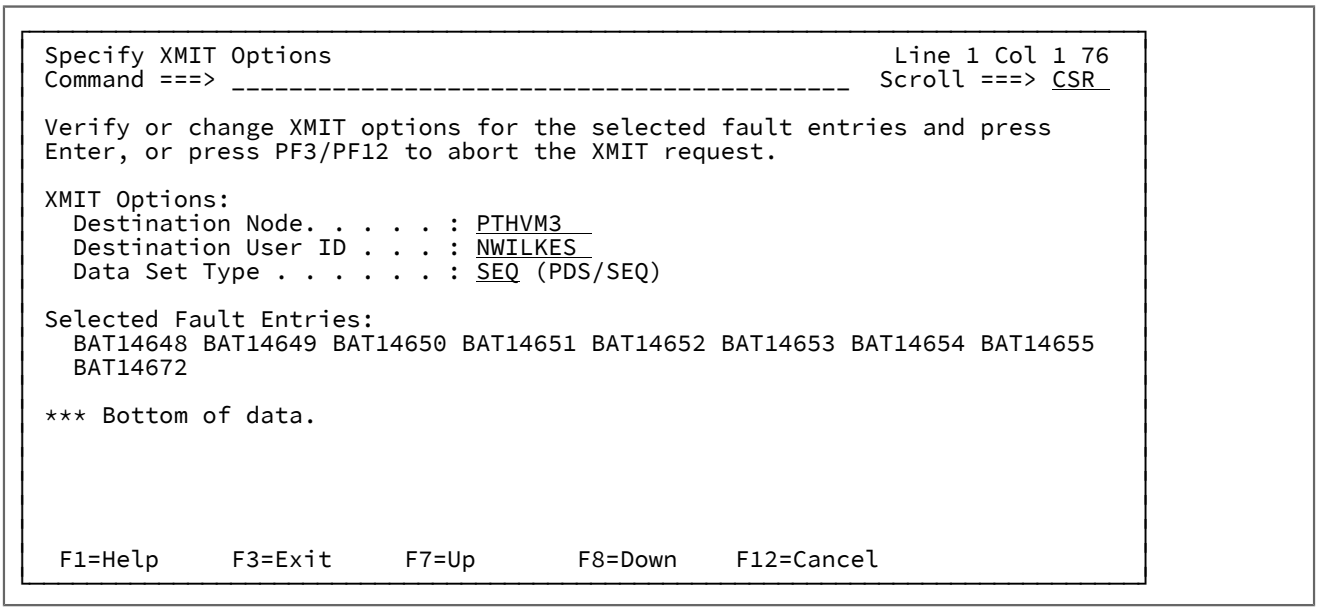
Transmitting history file entries

Use the X line command in the Fault Entry List display to XMIT a fault entry.

You can also XMIT a range of fault history entries, using two XX range markers. When you XMIT a range of a matched set of entries, you only XMIT the entries displayed on the screen. You do not XMIT any entries that are not displayed, but are interleaved with the displayed entries on the original history file.

After pressing the Enter key, a display showing the list of fault entries that are selected for XMIT with the X or XX commands is presented. You can specify the destination node and user ID to which the fault entries should be transmitted, as well as the transmitted data set type: either sequential (SEQ) or partitioned (PDS). These fields are initialized with the values last specified. The following is an example of the Specify XMIT Options display that is shown after pressing the Enter key.

Figure 78. Sample Specify XMIT Options display



Each selected fault entry is transmitted separately to the specified destination.

If the XMIT of one or more fault entries was not successful, then a display detailing the reasons is shown. Any associated dump data set is ignored.

The TSO user's access authorization level required to XMIT a fault entry is READ.

Packaging fault entries

To package fault entry data for sending to IBM® Support, enter P against the entry in the Fault Entry List display. This will generate JCL to package related Fault Entry data to a tersed data set suitable for transmission. The JCL is displayed in an edit session which can then be amended and submitted.

Tip: It is not necessarily sufficient to send just the fault entry. The IDIUTIL EXPORT step in the generated JCL appends the associated dump data set, if one exists, to ensure that Fault Analyzer Support receives all the appropriate data. While the Package option attempts to estimate space requirements, you might need to amend the space estimate to ensure successful execution.

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 363.

Alternatively, the use of Views (see Using views on page 59) in conjunction with data set profile security might be considered.

Specifying 64-bit addresses

The following applies to display input fields and command parameters where 64-bit addresses are supported.

A 64-bit enabled address field or command parameter can be specified by up to 16 hexadecimal digits, and might optionally include an underscore between bits 31 and 32. If an underscore is specified, then the value on the right hand side of the underscore is automatically padded with leading zeroes to form 8 digits. For example, the following are all considered identical address specifications:

```
100100000
1_100000
1_00100000
0000000100100000
00000001_00100000
```

Special consideration for 2GB-4GB addresses

Addresses in the range 80000000 to 8FFFFFFF are generally considered 31-bit addresses with the high-order bit (bit 0) ignored.

Whenever an address in this range, that does not include an underscore in the specification, is selected or entered, the high-order bit is implicitly set to 0 before the resulting 31-bt address is displayed.

To display storage in the 2GB-4GB range, the specified address must include an underscore. For example, entering the address as 0_80000000 or simply _80000000, will display the 64-bit address 80000000.

Using the interactive IDIS subsystem interface

The IDIS Subsystem Information option displays the status of the IDIS subsystem.

This display is shown as the result of entering the IDISINFO primary command, or selecting **Services** > **IDIS Subsystem Information** from the action bar.

Figure 79. Sample IDIS Subsystem Information display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
IDIS Subsystem Information                                 Line 1 Col 1 80
Command ===> _____________________________________________ Scroll ===> CSR

Started Task Name . . . . . : IDISSE1
ASID. . . . . . . . . . . . : X'0133'
Dispatching Priority. . . . : X'FE'
User ID . . . . . . . . . . : IDISS
PARM Field Options in Effect: UPDINDEX NOIMAGEFAST FASTEXCLUDE NOXCFGRPSUFFIX
Version . . . . . . . . . . : V16R1M4 (UI97160  2024/06/04)

Select one of the following options for additional information:
 1. Managed History Files
 2. Excluded History Files

*** Bottom of data.

 F1=Help      F3=Exit      F5=RptFind   F7=Up        F8=Down     F10=Left
F11=Right    F12=Cancel
```

If the IDIS subsystem has not been started, or is not responding as expected, then specific information about the subsystem status is displayed.

Otherwise, the following information is displayed:

Started Task Name / Job Name

The name of the IDIS subsystem started task or job.

ASID

The started task or job hexadecimal address space identifier.

Dispatching Priority

The IDIS subsystem dispatching priority.

User ID

The user ID associated with the IDIS subsystem.

PARM Field Options in Effect

The options in effect for the IDIS subsystem. These are either default values or are specified in the PARM field of the started task or job.

Version

The IDIS subsystem version, release, and maintenance level.

Two selectable options provide additional information:

1. **Managed History Files**

   Lists the history files that are currently managed by the IDIS subsystem.

2. **Excluded History Files**

   Lists the history files that are currently excluded by the IDIS subsystem.

Managed history files

The **Managed History Files** display lists history files that are currently managed by the IDIS subsystem.

History files that you have ALTER data set access to are shown in white. You can enter the S line command to select a history file. History files that you do not have ALTER data set access to are shown in blue, and do not have an input field for line commands.

```
IDIS Subsystem Managed History Files                        Line 1 Col 1 80
Command ===> _____________________________________________ Scroll ===> CSR

 Line commands: S (Select). Press Enter to refresh.

    History File Name
__  CTEST.RERUN50.FAX1
    DA.DCAT
__  DA.HIST.FAX1
__  DA.PSTORE.HISTORY
    OKHAN.HIST.BATCH
    OKHAN.HIST.BATCH.FAX2
    OKHAN.HIST.CICS.FAX2
    OKHAN.TEST.HIST
__  KCOSMIC.HISTORY

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

You can select multiple history files using one of the following methods:

- Use the SS line command to select the first and last in a block of history files.
- Use a 'repeat line command' (*c). For example, enter *s to repeat the S command on subsequent lines until terminated by * or *s on the last line in the block.

Press F1 from the **Managed History Files** display for more information about how to select history files.

Selected history files are shown in the IDIS Subsystem Selected History Files display, where you can modify their status. For example, you can stop the IDIS subsystem managing history files for the purpose of reallocation or some other activity that requires exclusive access. You can also sort and filter the list of history files. For more information, see Selected history files on page 145.

Excluded history files

The **Excluded History Files** display lists history files that are currently excluded by the IDIS subsystem.

History files that you have ALTER data set access to are shown in white. You can enter the S line command to select a history file. History files that you do not have ALTER data set access to are shown in blue, and do not have an input field for line commands.

```
IDIS Subsystem Excluded History Files                        Line 1 Col 1 80
Command ===> _____________________________________________ Scroll ===> CSR

 Line commands: S (Select). Press Enter to refresh.

    History File Name
    LJBERRY.DHIST
__  JGALAGA.DHIST
    NWILKES.HIST

 **  Bottom of data.

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

You can select multiple history files using one of the following methods:

- Use the SS line command to select the first and last in a block of history files.
- Use a 'repeat line command' (*c). For example, enter *s to repeat the S command on subsequent lines until terminated by * or *s on the last line in the block.

Press F1 from the **Excluded History Files** display for more information about how to select history files.

Selected history files are shown in the IDIS Subsystem Selected History Files display, where you can modify their status. For example, you can stop the IDIS subsystem managing history files for the purpose of reallocation or some other activity that requires exclusive access. You can also sort and filter the list of history files. For more information, see Selected history files on page 145.

Selected history files

This display is shown when selecting one or more history files from the IDIS Subsystem **Managed History Files** or IDIS Subsystem **Excluded History Files** display.

```
IDIS Subsystem Selected History Files                      Line 1 Col 1 80
Command ===> _____________________________________________ Scroll ===> CSR

 Line commands: R (Reset), X (Exclude). Press Enter to refresh.

    History File Name                          Status
__ CTEST.DAIOSMVS.DCAT                         Managed
__ CTEST.DAIPPMVS.DCAT                         Managed
__ CTEST.DAI3AMVS.DCAT                         Managed
__ CTEST.DAI3PMVS.DCAT                         Managed
__ CTEST.DALEXASM.FAX1                         Excluded
__ CTEST.DALEXCBE.FAX1                         Excluded
__ CTEST.DALEXIFP.FAX1                         Excluded
__ CTEST.DALEXILC.FAX1                         Excluded
__ CTEST.DALEXPLI.FAX2                         Managed
__ CTEST.DANLEASM.FAX2                         Managed
__ CTEST.DANLECBL.FAX2                         Managed
__ CTEST.DANLEPLI.FAX2                         Managed

**  Bottom of data.

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

The current status of each history file is shown as one of the following:

Managed

The history file is currently managed by the IDIS subsystem.

Excluded

The history file is currently excluded by the IDIS subsystem.

n/a

The history file is currently neither managed nor excluded by the IDIS subsystem.

You can change the status of selected history files, for example to make the IDIS subsystem stop managing history files for the purpose of reallocation or other activity that requires exclusive access.

Sorting and filtering the list of history files

The list of history files can be sorted or reduced by placing the cursor on the **History File Name** heading and pressing Enter. This brings up the Column Attributes display, from which the sort order can be selected, or filtering criteria can be entered using * and % characters as wildcards:

- An asterisk (*) can be specified to indicate zero, one, or more characters.
- A percent sign (%) can be specified to indicate a single character only.

Another method of filtering is to overtype characters in the list, including the above wildcards, and press Enter. For example, given the following list of history files:

HIST.DEPA.PROD
HIST.DEPA.TEST
HIST.DEPB.PROD
HIST.DEPB.TEST
HIST.DEPC.PROD
HIST.DEPC.TEST

In the history file HIST.DEPB.PROD, replace the last qualifier with an asterisk:

HIST.DEPA.PROD
HIST.DEPA.TEST
HIST.DEPB.*
HIST.DEPB.TEST
HIST.DEPC.PROD
HIST.DEPC.TEST

After pressing Enter, only the matching history file names are displayed:

HIST.DEPB.PROD
HIST.DEPB.TEST

To redisplay the complete list in the original sort order, enter RESET on the primary command line.

Press Enter to refresh the list of IDIS subsystem managed history files.

Press PF3 to return to the IDIS Subsystem Information display.

Line commands

Two line commands are available to change the status of selected history files:

R

Reset. If the current status is Managed or Excluded, entering the R line command will result in the status n/a, indicating that the history file is neither managed, nor excluded, by the IDIS subsystem.

If the current status is n/a, the R line command is ignored.

X

Exclude. If the current status is Managed or n/a, entering the X line command will result in the status Excluded, indicating that the history file is excluded by the IDIS subsystem.

If the current status is Excluded, the X line command is ignored.

Displaying options currently in effect

The Options in Effect display shows all Fault Analyzer options that are in effect when the ISPF interface is invoked, as well as where the option or a suboption is set. Use the CUROPTS primary command on page 100 or Options->Options in Effect on page 96 to show this display.

In the display, options that span more than one line are shown with continuation characters (a plus sign to the right of each line). This format is valid syntax for specification in, for example, the IDICNFxx parmlib member, so you can copy and paste options directly from this display.

Figure 80. Sample Options in Effect Display

```
File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Options in Effect                                            Line 1 Col 1 80
Command ===>                                                Scroll ===> CSR

Option                                     Where Last Set
DATASETS(IDIADATA(DA.SYSADATA),IDIBOJPN(J4. + SYS1.PARMLIB.FAE1.USER(IDICNF00)
BOOKS),IDIBOKOR(K4.BOOKS),IDIDOC(A4.DOC),ID +
IDOJPN(J4.DOC),IDIDOKOR(K4.DOC),IDIDSECT(DA +
.DSECTS),IDIEXEC(DA.EXEC,DA.EXITS.EXEC,DA.T +
EST.CLIST),IDIHIST(DA.DCAT),IDILC(DA.LISTIN +
G.C),IDILCOB(DA.LISTING.COBOL),IDILPLI(DA.L +
ISTING.PLI),IDIMAPS(DA.SIDIMAPS),IDIVIEWS(& +
TSOPFX..VIEWS,DA.VIEWS,CTEST.VIEWS),IDIVSEN +
U(A4.IDIHVENU),IDIVSJPN(A4.IDIHVJPN),IDIVSK +
OR(A4.IDIHVKOR))
──────────────────────────────────────────── ──────────────────────────────────
DEFERREDREPORT(CICS(FATASKS(1,20)),NOIMS,NO + Default
BATCH)
──────────────────────────────────────────── ──────────────────────────────────
DETAIL(MEDIUM)                             Default
──────────────────────────────────────────── ──────────────────────────────────
DUMPREGISTRATIONEXITS(CONTROL(REXX(SLIPCTL) + SYS1.PARMLIB.FAE1.USER(IDICNF00)
),NOTIFY(REXX(SLIPNFY)))
```

Recovering user notes

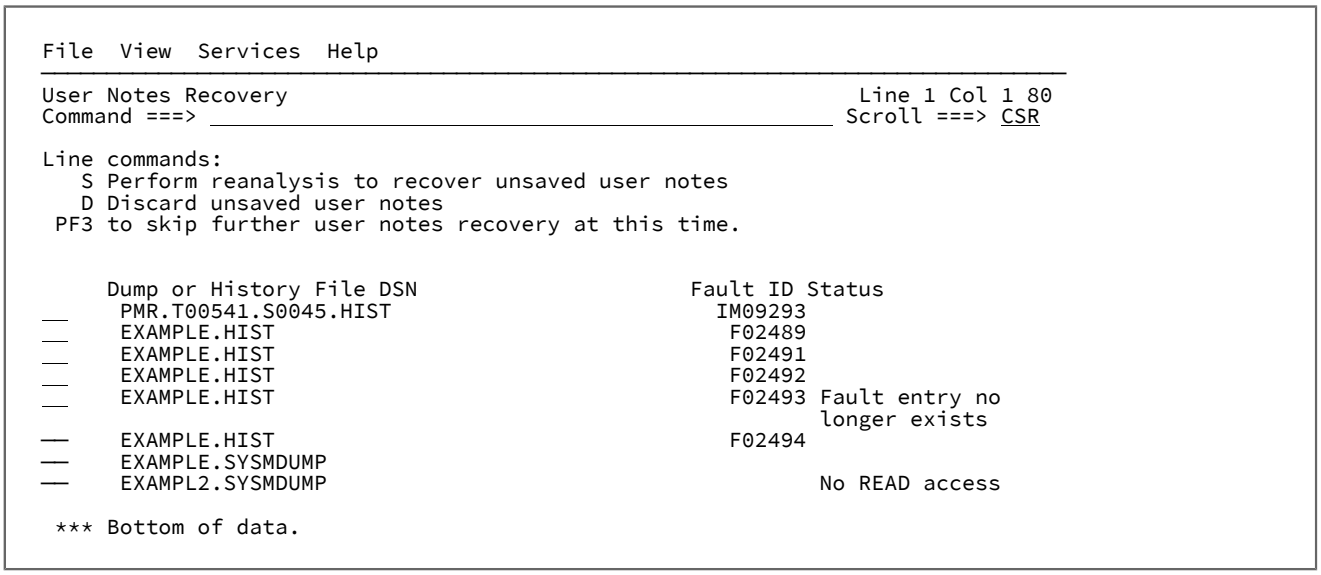
As described in Creating and managing user notes on page 212, you can record user notes against any storage location while performing interactive analysis. When interactive reanalysis ends normally, user notes are stored in the history file fault entry. When interactive reanalysis does not end normally, any user notes that you added, deleted, or modified during the session are lost unless you enable user notes recovery.

When you enable user notes recovery, Fault Analyzer records all user notes activity in a specified data set. If an interactive reanalysis session ends abnormally, Fault Analyzer automatically applies the user notes activity when you analyze the same history file fault entry or system dump again.

To enable user notes recovery, select **Options->Interactive Reanalysis Options** from the action-bar pull-down menu and specify a valid data set name in the **User Notes Recovery Data set name** field. (For additional details about interactive reanalysis options, see Interactive reanalysis options on page 157.)

When the user notes recovery data set contains information about unsaved user notes activity, the Fault Analyzer ISPF interface shows the User Notes Recovery display on startup. An example of this display is shown in Figure 81: User Notes Recovery display on page 148.

Figure 81. User Notes Recovery display



Each entry in the display is one interactive reanalysis session that has unsaved user notes.

- To recover the user notes, select the entry using the S line command (or use SS block line commands) and press Enter.
- To discard the user notes, use the D line command (or DD block line command).

When you issue the S or D line command against an entry, it drops off the display.

If you don't use the S or D line command to select an entry from this display (for example, if you press PF3), user notes are still recovered automatically when you perform interactive reanalysis of the same history file fault entry or system dump again. In this case, the User Notes Recovery display no longer shows the matching entry on startup of the Fault Analyzer ISPF interface.

Chapter 4. Performing batch reanalysis

Use batch reanalysis to:

- Rerun analysis on a number of faults
- Run the reanalysis 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 The Fault Analyzer report on page 272.

Batch reanalysis options

Many of the general options specified for an installation are also applicable to batch reanalysis. See Options on page 571 for information about all available options and the different ways in which they can be specified.

Some of the options that you might want to consider using to control the batch reanalysis report are:

Detail

Specify this option if you want to adjust the level of detail given in the batch reanalysis report. See Detail on page 591 for more detail.

PrintInactiveCOBOL

The PrintInactiveCOBOL option can be used to request that storage for inactive COBOL programs (programs that are not in the current save-area chain) is included in the reanalysis report.

To specify batch reanalysis options that apply to your batch jobs only, select **Batch Reanalysis Options** from the **Options** menu on the **Fault Entry List** display (for general information about menu options, refer to Action-bar pull-down menus on page 94). This opens the **Batch Reanalysis Options** display as shown in Figure 82: Sample Batch Reanalysis Options display on page 150.

Figure 82. Sample Batch Reanalysis Options display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Batch Reanalysis Options                                    Line 1 Col 1 80
Command ===> _____________________________________________ Scroll ===> CSR

Press PF3 to save options or PF12 to cancel.

General Options:
  Options line for batch
    reanalysis. . . . . : Detail(Long)_____________________________________
  Reanalysis report
    destination . . . . . : _______________________________________________
  Redisplay this panel
    before each reanalysis. : N (Y/N)
  Display panel to edit
    generated JCL . . . . . : Y (Y/N)
  Route to original system. : Y (Y/N)
  Job card style. . . . . : P (P=Parameters, S=Statements)

Job Card Parameters:
  Job name suffix . . . . . : A (A-Z, 0-9, @, #, or $)
  Job class . . . . . : A (A-Z or 0-9)
  Job notify. . . . . : Y (Y/N)
  Job time minutes. . . . . : 10 (0-99)
  Message class . . . . . : X (A-Z or 0-9)
  Region megabytes. . . . . : 0    (0-2047)
  Accounting info . . . . . : ___________________________________________

Reanalysis 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
```

The following can be specified using this display:

Options line for batch reanalysis

Options that apply to all batch reanalysis jobs that you submit can be specified here. These options, which are used in the PARM field of the generated batch reanalysis job, take precedence over any options that are specified through an options file (see *"Options data set name"* below).

The option Detail(Long) is shown as an example on the options line in Figure 82: Sample Batch Reanalysis Options display on page 150.

If you need to specify more options than fit on this line, then use one of the options *"Display panel to edit generated JCL"* or *"Edit the options data set before reanalysis"* instead (both are explained in the following).

Options that are specified on the options line are saved in the user profile.

Reanalysis report 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.

Redisplay this panel before each reanalysis

If this option is set to Y, then the Batch Reanalysis Options display is shown each time a batch reanalysis is requested, prior to generating the JCL stream.

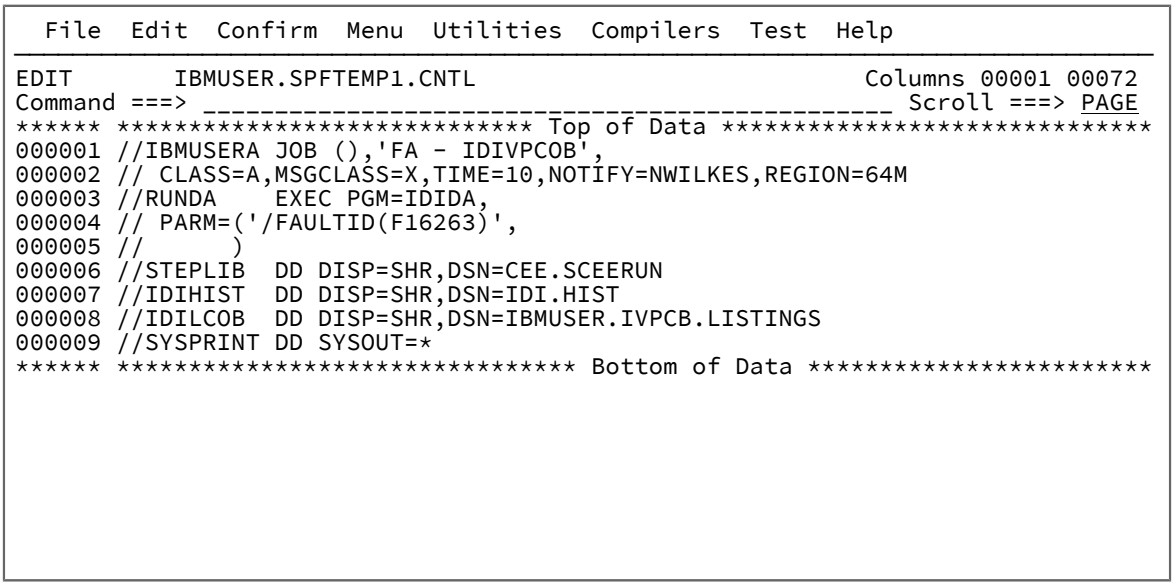
Make your changes. Then press PF3 (to continue with the current options) or PF12 (to undo any changes made). What happens next depends on the *"Display panel to edit generated JCL"* option below.

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

Display panel to edit generated JCL

If this option is set to Y, then you are presented with an ISPF EDIT display screen of the JCL stream generated by Fault Analyzer as the example shown in Figure 83: Sample batch reanalysis JCL stream EDIT on page 151.

Figure 83. Sample batch reanalysis JCL stream EDIT



Make your changes. Then issue the SUBMIT command to execute the job. Then enter the EXIT (PF3) or CANCEL (PF12) command to return to the Fault Entry List display.

For more information about the use of this option, see Data sets used for batch reanalysis on page 154.

If this option is set to N, then the generated JCL stream is submitted automatically without first displaying the JCL EDIT screen.

Route to original system

If the reanalysis job is submitted on a different system than where the original abend occurred, this option routes the job back to the originating z/OS system. This can be useful, for example, when required data sets are not shared across all systems in the sysplex.

If this option is set to Y and the system on which the original abend occurred is different from the current system, a /*JOBPARM S=*system-name* JCL statement is inserted into the generated job.

If this option is set to N, a /*JOBPARM statement is not added. This is the default setting.

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 82: Sample Batch Reanalysis Options display on page 150 follows the General Options section after the Enter key is pressed.
- If S is specified, then a Job Card Statements section follows the General Options section after the Enter key is pressed.

  This would alter the display from that shown in Figure 82: Sample Batch Reanalysis Options display on page 150 to the following:

Figure 84. Sample Batch Reanalysis Options display

```
  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. . . . . . . : ____________________________________________
  Reanalysis report
    destination . . . . . . : ____________________________________
  Redisplay this panel
    before each reanalysis. : N (Y/N)
  Display panel to edit
    generated JCL . . . . . : Y (Y/N)
  Route to original system. : Y (Y/N)
  Job card style. . . . . . : S (P=Parameters, S=Statements)

Job Card Statements:
  ===> ________________________________________________________________________
  ===> ________________________________________________________________________
  ===> ________________________________________________________________________
  ===> ________________________________________________________________________

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
```

Whichever style of job card specification is selected with this option determines the method used when generating the job card for all batch reanalysis jobs.

Job name suffix

This suffix is the character that is appended to your user ID to form the job name that is used for the generated batch reanalysis JCL stream. The default is A.

If the user ID is eight characters in length, it is truncated to seven characters before the suffix is appended.

Job class

This job class is the job class that is used on the CLASS parameter of the generated JOB card. The default is A.

Job notify

If this field is set to Y, then a NOTIFY=*userid* parameter is added to the generated JOB card. If it is set to N, then no NOTIFY parameter is added. The default is Y.

Job time minutes

This value is the number of minutes that are used on the TIME parameter of the JOB card. The valid range is 1 - 30. The default is 10.

Message class

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

Region megabytes

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

Accounting info

Anything specified in this field is used as accounting information on the generated JOB card. The default is not to provide any accounting info.

Options data set name

This field can optionally specify the name of a PDS or PDSE data set in which a member (see *"Options member name"*) contains Fault Analyzer options. The data set and member name are used as the IDIOPTS user options file. This data set can, for example, be used if more options than fit on the options line at the top of this display are required.

Note:

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

Options member name

This name is the member name of the data set specified in *"Options data set name"*.

Use this data set during reanalysis

If this option is set to Y, then the data set and member name specified above are used by Fault Analyzer during the batch reanalysis. If it is set to N, then the data set and member name are not used.

Edit the options data set before reanalysis

If this field is set to Y, then an ISPF EDIT display screen of the member in the options data set specified above is presented prior to generating the batch reanalysis JCL stream. An example is shown in Figure 85: Sample options file EDIT for batch reanalysis on page 154.

Figure 85. Sample options file EDIT for batch reanalysis

```
  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(l)
****** **************************** 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
```

Change the options data set (if you wish). Then enter the EXIT command (usually mapped to PF3).

Initiating batch reanalysis

To initiate batch reanalysis, enter B against the fault history entry.

Depending on your batch options, one or more displays might be shown prior to the submission of the Fault Analyzer-generated JCL stream. For details, see Batch reanalysis options on page 149.

Data sets used for batch reanalysis

When performing batch reanalysis through the ISPF interface, the generated JCL includes DD statements as required for any JOBLIB, STEPLIB, or Fault Analyzer compiler listing or side file data sets. DD statements are added for Fault Analyzer data sets even if they were not explicitly included in the real-time JCL, but were supplied through the DataSets option or an Analysis Control user exit. These data sets are added to the reanalysis job in an attempt to recreate the same execution environment as were used in real time.

DataSets options that are specified via the _IDI_OPTS user options file or the PARM field cause those data sets to be logically concatenated to the data sets from the real-time execution.

If the *"Display panel to edit generated JCL"* option on the Batch Reanalysis Options display is set to Y (see Batch reanalysis options on page 149), then it is possible to make changes to the real-time data set specifications before submitting the reanalysis job. Also, any data sets that were used in real time but do not exist in the reanalysis environment, or data sets with READ access prohibited, are identified by a comment as shown in the following example for IDILCOB:

```
//IDILCOB  DD DISP=SHR,DSN=CTEST.DUMPA.LISTING.CICS.COBOL
//         DD DISP=SHR,DSN=DA.LISTING.COBOL
//* The following IDILCOB data set is unavailable:
//*        DD DISP=SHR,DSN=CTEST.DUMPA.LISTING.CICS.COBOSVS
//* The following IDILCOB data set is READ protected:
//*        DD DISP=SHR,DSN=CTEST.PROTECT.LISTINGS
```

Note: If the GenerateSavedReport option is in effect, then all data set or path names that were used during real-time processing will automatically be included, even if these have been deleted from the generated JCL before performing reanalysis.

Creating your own batch reanalysis job

Batch mode can also be invoked via JCL that you have created or saved from a job previously generated by Fault Analyzer and later modified if necessary. If you are performing reanalysis of a fault entry, specify the FaultID option. If you are performing analysis of an MVS dump data set, specify the DumpDSN option.

JCL can be generated using the B line command from the Fault Entry List display. This is the preferred method of performing batch reanalysis, because it will automatically add JCL DD statements for side file data sets and any STEPLIB concatenation that was in effect during real-time processing.

Your job might, for example, look like this:

Figure 86. Sample batch reanalysis job

```
//RTURNERA JOB (),'FAULT ANALYZER',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//*
//* Allocate a PDSE for compiler listings
//*
//ALLOC    EXEC PGM=IEFBR14
//DD1      DD  DISP=(,CATLG),DSN=&SYSUID..COBLIST,SPACE=(CYL,(1,1,5)),
//             DCB=(RECFM=FBA,LRECL=133),DSNTYPE=LIBRARY
//*
//* Recompile MYPGMA
//*
//CBLCOMP  EXEC IGYWC,PARM.COBOL='LIST,MAP,Source,XREF'
//COBOL.SYSIN  DD DISP=SHR,DSN=MY.COBOL.Source(MYPGMA)
//COBOL.SYSPRINT DD DISP=SHR,DSN=&SYSUID..COBLIST(MYPGMA)
//*
//* Recompile MYPGMB
//*
//CBLCOMP  EXEC IGYWC,PARM.COBOL='LIST,MAP,Source,XREF'
//COBOL.SYSIN  DD DISP=SHR,DSN=MY.COBOL.Source(MYPGMB)
//COBOL.SYSPRINT DD DISP=SHR,DSN=&SYSUID..COBLIST(MYPGMB)
//*
//* Reanalyze SYSMDUMP data set
//*
//RUNDA    EXEC PGM=IDIDA,PARM=('/DumpDSN(MY.DUMPDS)')
//SYSPRINT DD  SYSOUT=*
//IDILCOB  DD  DISP=SHR,DSN=&SYSUID..COBLIST
//*
//* Delete temporary compiler listings PDSE
//*
//DELETE   EXEC PGM=IEFBR14
//DD1      DD  DISP=(OLD,DELETE),DSN=&SYSUID..COBLIST
```

The job includes the recompilation of two COBOL programs, MYPGMA and MYPGMB. These are assumed involved in the fault being reanalyzed, but their compiler listings might not have been available to Fault Analyzer during real-time analysis. Providing them to the fault reanalysis enables identification of the line of source code where the error occurred.

Note that only one program is compiled in each job step. This limitation is to facilitate the naming of the compiler listing data set member in accordance with the rules outlined in Naming compiler listings or side files on page 378.

You can optionally add an IDIOPTS DD statement to your JCL. This statement supplies the name of a sequential file containing Fault Analyzer options, providing job step overrides of product and installation defaults.

Other DD statements that you can add into your JCL are described in Pointing to listings with JCL DD statements on page 35.

Any options that are specified in the JCL EXEC statement PARM field override options set via the IDIOPTS file.

For more information, see User options file IDIOPTS on page 575.

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 MVS dump data set can be displayed, regardless of whether it is included in the Fault Analyzer report.
- Source code information (if provided via compiler listing or side file) can be viewed in its entirety.
- This method is the only way to analyze CICS® system abends.

Note: Whereas the information in this chapter assumes that the interactive reanalysis is performed under ISPF, it is possible to instead perform this under CICS®. When doing so, restrictions might apply. These restrictions are described in Performing interactive reanalysis under CICS on page 284.

Interactive reanalysis options

Many of the general options specified for an installation are also applicable to interactive reanalysis. See Options on page 571 for information about all available options and the different ways in which they can be specified.

To specify interactive reanalysis options that apply to your interactive reanalysis sessions only, select **Interactive Reanalysis Options** from the **Options** menu on the **Fault Entry List** display (for general information about menu options, refer to Action-bar pull-down menus on page 94). This option opens the Interactive Reanalysis Options display as shown in Figure 87: Sample Interactive Reanalysis Options display on page 158.

Figure 87. Sample Interactive Reanalysis Options display

```
  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. . : GSR(O)
  Redisplay this panel
    before each reanalysis. . : N (Y/N)
  Display panel to alter
    allocated data sets . . . : N (Y/N)
  Prompt before opening an
    MVS dump data set . . . . : Y (Y/N)
  Always prompt to select
    TCB for MVS dump data
    set analysis. . . . . . . : N (Y/N)
  Prompt for missing side
    files . . . . . . . . . . : Y (Y/N)
  Current list of excluded programs ( Edit ):
  (Empty)

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)

Deferred Breakpoints Repository:
  Data set name (PDS/E) . . : PRINT.PDS

User Notes Recovery:
  Data set name . . . . . . : 'EXAMPLE.NOTES.RECOVERY'

*** Bottom of data.
```

The following possibilities can be specified by using this display:

Options line for interactive reanalysis

Options that apply to all interactive reanalysis sessions that you initiate can be specified here. These options take precedence over any options that are specified through an options file (see Options data set name on page 162). The options are the equivalent of the **PARM** field options that are used by batch reanalysis jobs.

The option `GSR(O)` is shown as an example on the options line in Figure 87: Sample Interactive Reanalysis Options display on page 158.

Redisplay this panel before each reanalysis

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

Make your changes. Then press PF3 (to continue with the current options), or press PF12 (to undo any changes). What happens next depends on the **Display panel to alter allocated data sets** option.

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

Display panel to alter allocated data sets

If this option is set to Y, then you are presented with an ISPF EDIT display screen of the pseudo JCL stream that is generated by Fault Analyzer.

Figure 88. Sample interactive reanalysis pseudo JCL stream EDIT

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       IBMUSER.SPFTEMP1.CNTL                          Columns 00001 00072
Command ===> _____________________________________________ Scroll ===> PAGE
****** ***************************** Top of Data ******************************
====== Type "RESET" on the command line and press Enter to see edit
====== instructions.
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 50 Line(s) not Displayed
000051 //IDIADATA DD DISP=SHR,DSN=DA.SYSADATA
000052 //IDILCOB  DD DISP=SHR,DSN=DA.LISTING.COBOL
000053 //* The following IDILCOB data set is unavailable:
000054 //*        DD DISP=SHR,DSN=DA.LISTING.COBOL.OLD
000055 //IDIJAVA  DD PATH='/u/temp/payroll/directory171/DEPT64directory/accoun
000056 //             tingDIR1'
000057 //         DD PATH='/u/temp/payroll/directory171/DEPT64directory/accoun
000058 //             tingDIR2'
000059 //* The following IDIJAVA zFS path is unavailable:
000060 //*        DD PATH='/u/temp/payroll/directory171/DEPT64directory/accoun
000061 //*            tingDIR3'
000062 //         DD PATH='/u/a+'
000063 //IDILC    DD DISP=SHR,DSN=DA.LISTING.C
000064 //IDILPLI  DD DISP=SHR,DSN=DA.LISTING.PLI.PROD
000065 //         DD DISP=SHR,DSN=DA.LISTING.PLI.TEST
****** **************************** Bottom of Data ****************************
```

Type RESET on the command line and press Enter. Make your changes in accordance with the instructions that are displayed. Then enter the EXIT (PF3) or CANCEL (PF12) command as appropriate to initiate the interactive reanalysis.

For more information about this option, see Data sets used for interactive reanalysis on page 240.

If this option is set to N, then the interactive reanalysis commences without first displaying the pseudo JCL EDIT screen.

Prompt before opening an MVS dump data set

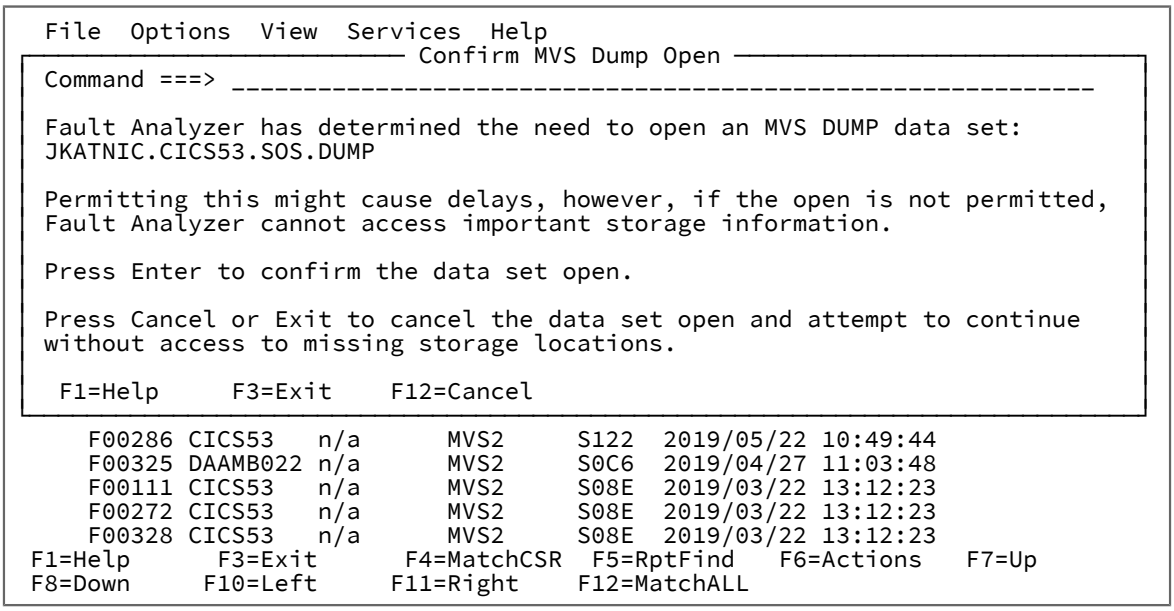
Sometimes access is required to a storage location that is not contained in the saved minidump:

- During the interactive reanalysis.
- As a result of displaying storage locations from within the interactive report.

When this situation happens, if the field is set to Y, a display is shown before an associated MVS dump data set is opened, to look for the missing storage.

An example of this display is shown in Figure 89: Sample Confirm MVS Dump Open display on page 160.

Figure 89. Sample Confirm MVS Dump Open display



You are only prompted at most one time during any interactive reanalysis session. If the open is canceled by entering CANCEL or EXIT, no further attempts are made to open the MVS dump data set. Likewise, if the open is confirmed, Fault Analyzer checks the MVS dump for all references to storage locations not contained in the minidump.

If this field is set to N, then the associated dump data set is opened if required without prompting you first.

Always prompt to select TCB for MVS dump data set analysis

A single character ('Y' or 'N') that controls whether or not the TCB selection display is shown before analysis of an MVS dump data set is performed.

If this option is set to 'N', the TCB selection display is shown only if Fault Analyzer is unable to select a TCB. This might, for example, happen with console dumps when no tasks have abended.

If this option is set to 'Y', the TCB selection display is always shown. If Fault Analyzer is able to select a TCB, then that TCB will be preselected on the display. Accept this TCB, or change to another, then press Enter to perform the analysis.

Prompt for missing side files

A single character (Y or N) that controls whether the Compiler Listing Not Found display is shown when no compiler listing or side file is found for a program.

The default setting of this field is based on whether any DataSets option specification exists in the IDICNFxx parmlib member for any DDnames from the following list:

IDIADATA
IDILANGX
IDILC

IDILCOB
IDILCOBO
IDILPLI
IDILPLIE
IDISYSDB

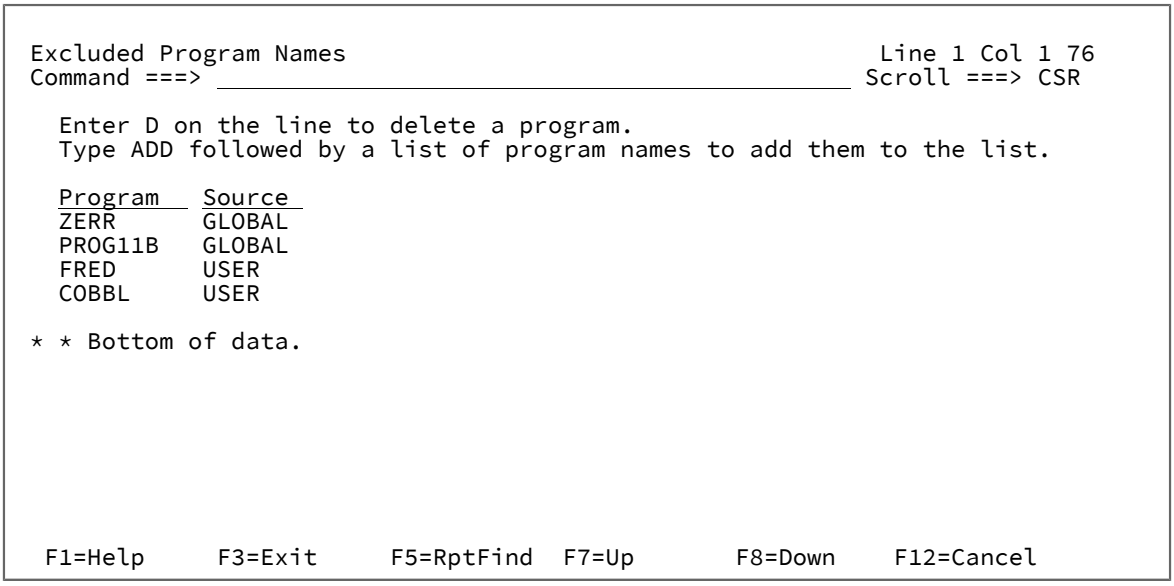
If no data sets for any of these DDnames were specified in the IDICNFxx parmlib member, then the default setting of this field is 'N'. Otherwise, it is 'Y'.

By clearing this field, and pressing Enter, the value is reinitialized to its default setting.

Current list of excluded programs (Edit)

If the "Prompt for missing side files" option is set to *"Y"*, then the current list of excluded program names, for which compiler listing or side file searching is not performed, is provided. The list can be modified by placing the cursor on the **Edit** point-and-shoot field and pressing Enter. This results in the Excluded Program Names display being shown. An example of this display is shown in Figure 90: Excluded Program Names display on page 161.

Figure 90. Excluded Program Names display



Press PF3 to exit the display and save any changes made.

Type ADD on the command line, followed by one or more blank-delimited program names that you wish to add as user-specified exclusions, and press Enter. If the program name does not adhere to required naming conventions, then a message is issued.

Program names must be valid PDS or PDSE member names, with the addition of the optional wildcard characters '*' (zero, one or more characters) and '%' (a single required character) to make the name more generic. The program names specified are not case sensitive.

The following are examples of valid program name specifications:

```
*XMAI*
PAYROLL0
SELOPT%
SUBRTN*
TZ%%C*
```

Use the D line command to delete individual entries from the user-specified list. Individual entries cannot be deleted from the global list via this display.

The program name exclude list can also be edited from the Compiler Listing Not Found display. For details, see Prompting for compiler listing or side file on page 235.

Options data set name

This field can optionally specify the name of a PDS or PDSE data set in which a member (see *"Options member name"*) contains Fault Analyzer options. The data set and member name are used as the IDIOPTS user options file. This data set can, for example, be used if more options than fit on the options line at the top of this display are required.

Note:

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

Options member name

This field is the member name of the data set specified in *"Options data set name"*.

Use this data set during reanalysis

If this option is set to Y, then the data set and member name that are previously specified are used by Fault Analyzer during the interactive reanalysis. If it is set to N, then the data set and member name are not used.

Edit the options data set before reanalysis

If this field is set to Y, then an ISPF EDIT display screen of the member in the options data set specified above is presented prior to commencing the interactive reanalysis. An example is shown in Figure 91: Sample options file EDIT for interactive reanalysis on page 163.

Figure 91. Sample options file EDIT for interactive reanalysis

```
  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(lx)))
****** **************************** 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
```

Change the options data set (if required). Then enter the EXIT command (usually mapped to PF3).

Data set name (PDS/E)

Specifies the name of the data set used for z/OS® Debugger Deferred Breakpoints set via COBOL Explorer.

Standard ISPF data set name specification rules apply, that is, if the data set name is not enclosed within single quotes, then it is prefixed by the current TSO prefix.

The data set name specified need not exist, as it is allocated automatically when required. If an existing data set name is specified, then it must be the name of a PDS data set allocated RECFM=VB and LRECL=255.

A member name must not be specified.

User Notes Recovery Data Set Name

Specifies the name of a data set to be used for user notes recovery.

- Standard ISPF data set name specification rules apply: If the data set name is not enclosed within single quotes, then it is prefixed by the current TSO prefix.
- If you specify the name of an existing data set, it must be a sequential data set allocated with RECFM=VB and LRECL=1024. If the data set is not empty, it must contain valid user notes recovery information.

- If the specified data set does not exist, a User Notes Recovery Data Set Create display like the example in Figure 92: User Notes Recovery Data Set Create display on page 164 is shown.
- Each TSO/ISPF user should use a unique user notes recovery data set. If a user's notes might contain sensitive information, protect data set access accordingly.

For more information, see Recovering user notes on page 147.

Figure 92. User Notes Recovery Data Set Create display

```
  File  View  Services  Help
 ┌──────────────────────────────────────────────────────────────────────┐
 │ User Notes Recovery Data Set Create                  Line 1 Col 1 76 │
 │ Command ===> _____________________________________ Scroll ===> CSR  │
 │                                                                      │
 │ The specified user notes recovery data set does not exist. Press Enter to
 │ create, or press PF3/PF12 to cancel.                                 │
 │                                                                      │
 │ Data Set Name . . . . . : 'EXAMPLE.NOTES.RECOVERY'                   │
 │ Primary Space . . . . . : 5     Cylinders                            │
 │ Secondary Space . . . . : 5     Cylinders                            │
 │                                                                      │
 │ *** Bottom of data.                                                  │
 └──────────────────────────────────────────────────────────────────────┘
    MVS dump data set . . . . : Y (Y/N)
  Always prompt to select
    TCB for MVS dump data
    set analysis. . . . . : N (Y/N)
  Prompt for missing side
    files . . . . . : Y (Y/N)
  Current list of excluded programs ( Edit ):
  (Empty)

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)

Deferred Breakpoints Repository:
  Data set name (PDS/E) . . : PRINT.PDS

User Notes Recovery:
  Data set name . . . . . : 'EXAMPLE.NOTES.RECOVERY'

*** Bottom of data.
```

You can change the default primary and secondary space values as needed.

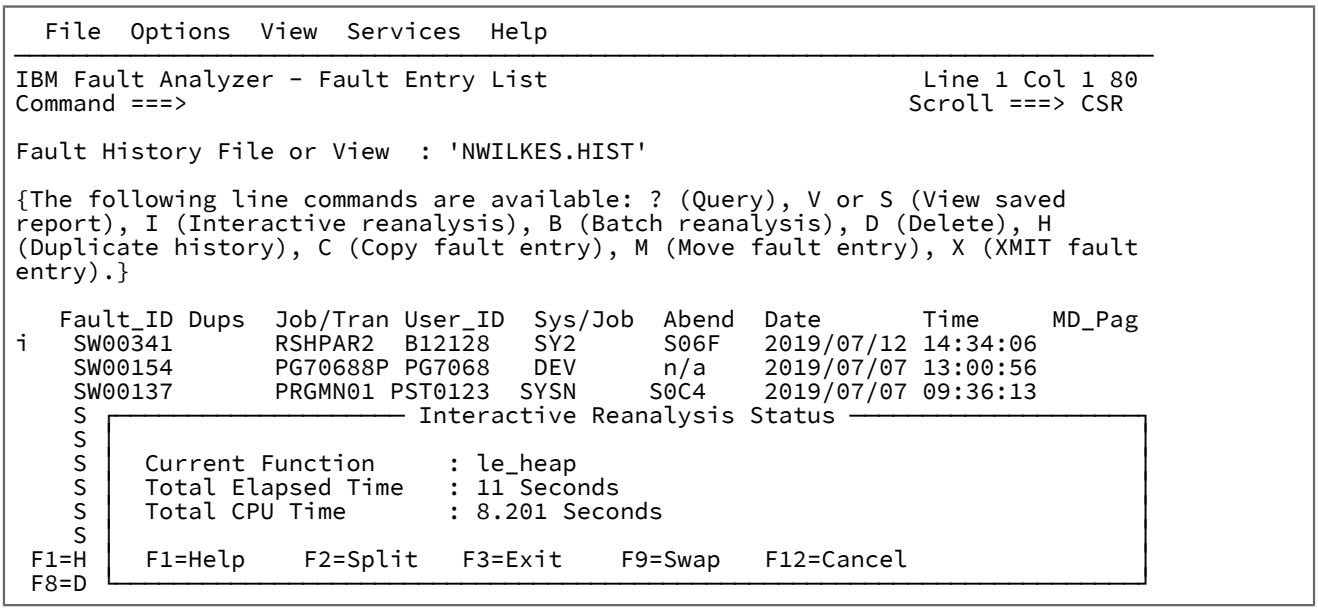
Initiating interactive reanalysis

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.

Figure 93. 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 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 94: Sample Interactive Reanalysis Report display on page 166 shows an example of the first interactive report display from where all other parts of the interactive report can be selected:

Figure 94. Sample Interactive Reanalysis Report display

```
  File  View  Services  Help
_____________________________________________________________________________
Interactive Reanalysis Report                                 Line 1 Col 1 80
Command ===> _____________________________________________ Scroll ===> CSR
JOBNAME: IDIVPCOB  SYSTEM ABEND: 0C7              MVS2     2019/12/21 13:02:25

Fault Summary:
Module IDISCBL1, program IDISCBL1, entry point IDISCBL1, source line 35 :
Abend S0C7-X'0' (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.
```

A fault summary is provided at the top of the initial display, which is equivalent to the summary provided in message IDI0002I on page 695 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 122), then information about the maximum amount of allocated storage that Fault Analyzer used during the analysis is included at the bottom of the display. This amount of storage is for explicit allocations only and does not include storage for loaded modules, and so on.

The interactive report is formatted differently depending on the logical screen size used. All examples in this book are based on a screen size which is 24 lines by 80 columns, however, if your screen is larger, Fault Analyzer formats the report accordingly. This situation is true also if the screen size is dynamically resized; just press the Enter key and the report section viewed is reformatted to match the screen size.

Anywhere in the interactive report, you can use the UP (PF7), DOWN (PF8), LEFT (PF10), or RIGHT (PF11) commands to see the entire report section that is currently selected. (Note that PF10 and PF11 in the Dump Storage display are instead mapped to the PREV and NEXT commands respectively, as this display does not necessitate horizontal scrolling.)

Throughout the interactive report are tabbable fields in yellow. These are point-and-shoot fields which respond to placing the cursor on them and pressing the Enter key. What is displayed next depends on the type of information selected. Some take you to other parts of the report while others display detailed information about the item selected. For example:

Source code line or statement number

Displays the source code for the entire program as obtained from the compiler listing or side file with the selected line or statement number highlighted. In addition, disassembly of machine instructions is provided. For more information and an example of this display, refer to Displaying source code on page 206.

Storage address

Displays the storage at this location in both hexadecimal and translated EBCDIC. For more information and an example of this display, refer to Displaying storage locations on page 209.

Program Status Word (PSW)

PSWs are displayed with two halfs:

- If the high word is selected, the information about the PSW bit settings is displayed. For more information, see Displaying PSW information on page 222.
- If the low word is selected, the storage at the PSW address is displayed. For more information, see Displaying storage locations on page 209.

Although the point-and-shoot fields are defined using the ISPF color attribute YELLOW, they might actually be displayed with a different color depending on user settings. However, in this book, they are referred to as yellow fields.

Apart from storage addresses, all point-and-shoot fields can also be entered on the command line of the display. This point-and-shoot ability is especially convenient when selecting an item from a list of options.

Exit from the interactive report

Upon exit from the interactive report using the EXIT (PF3) or CANCEL command from the Interactive Reanalysis Report display, you might be presented with a confirmation prompt as the example shown in Figure 95: Sample Confirm Exit display on page 167.

Figure 95. Sample Confirm Exit display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Interactive Reanalysis Report                                  Line 1 Col 1 80
Command ===>                                                  Scroll ===> CSR
JOBNAME: IDIVPCOB  SYSTEM ABEND: 0C7             MVS2     2019/12/21 13:02:25

┌──────────────────────────────── Confirm Exit ────────────────────────────────┐
│                                                                              │
│ Are you sure you want to exit the interactive report ? Press Enter to        │
│ confirm exit or press PF12 to return to the interactive report.              │
│                                                                              │
│  F1=Help    F12=Cancel                                                       │
└──────────────────────────────────────────────────────────────────────────────┘
  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.
```

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 610.

An extra prompt might be displayed if user information has been modified. For details, see Refresh processing on page 240.

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 96: Sample Synopsis display on page 168.

Figure 96. Sample Synopsis display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Synopsis                                                      Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
JOBNAME: COBUNI    SYSTEM ABEND: 0C9            FAE1      2019/07/21  14:36:43

A system abend 0C9 occurred in module COBUNI program COBUNI at offset X'5C8'.

A program-interruption code 0009 (Fixed-Point-Divide Exception) is associated
with this abend and indicates that:

  The divisor was zero in a signed binary division.

The cause of the failure was program COBUNI in module COBUNI.  The COBOL source
code that immediately preceded the failure was:

  Source
  Line #
  000036            IF UTF-16 NOT EQUAL ' ' AND V1 / V2 > 0

The COBOL source code for data fields involved in the failure:

  Source
  Line #
  000018        01  UTF-16 PIC N(30) USAGE NATIONAL.
  000020        01  V1 PIC 9(9) BINARY VALUE 357.
  000021        01  V2 PIC 9(9) BINARY VALUE 0.

Data field values at time of abend:

  UTF-16 = UTF-16 UNICODE DATA
  V1     = 357
  V2     = 0  *** Cause of error ***

*** Bottom of data.
```

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 97: Sample Event Summary display on page 169.

Figure 97. Sample Event Summary display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Event Summary                                                  Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
Full Application only - TRANID: FRED   CICS ABEND: AEIL      2019/06/13  10:42

{The following events are presented in chronological order.}

Event            Fail  Module   Program  EP
#  Type          Point Name     Name     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   F6=Actions   F7=Up
 F8=Down     F10=Left     F11=Right
```

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 170.

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 209.
- 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 206.

Two point-and-shoot fields are also provided under the panel title to toggle the state of the display:

- **Full** displays all application and non-application events.
- **Application only** limits the event summary to events originating from user applications. When this option is selected, individual event numbering is likely to differ from the full event summary.

The selected option is remembered between interactive analysis sessions.

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 disappears, but reappears if the display is scrolled to the top. For general information about help text, see Adding or removing help text on page 122.

The column headings are never scrolled out of view. However, if scrolling horizontally, the column headings scroll with the data below them.

Detailed Event Information

An Event Details display is presented when an event is selected from the Event Summary display. An example is shown in Figure 98: Sample Event Details display for point of failure (Part 1 of 2) on page 171.

Figure 98. Sample Event Details display for point of failure (Part 1 of 2)

```
  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      2019/07/21  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: ❶
  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
LJBERRY.LISTING.COBOL(COBPERF6).

Load Module Name. . . . . : SYS09202.T161638.RA000.COBPERF6.GOSET.H01(COBPER
  At Address. . . . . : 16B00988
  Load Module Length. . . . : X'1678'
  Link-Edit Date and Time . : 2009/07/21  16:16:40

Program and Entry Point Name: COBPERF6
  At Address. . . . . : 16B00988 (Module COBPERF6 offset X'0')
  Program Length. . . . . : X'A4A'
  Program Language. . . . . : COBOL (Compiled using IBM Enterprise COBOL for
                              z/OS and OS/390 V4 R1 M0 on 2009/07/21 at
                              16:16:39)
  Compiler Options Used . . : NOADATA ADV APOST ARITH(COMPAT) NOAWO
                              BUFSIZE(4096) NOCICS CODEPAGE(1140) NOCOMPILE(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 99. Sample Event Details display for point of failure (Part 2 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
  Binary Optimizer. . . . . : Automatic Binary Optimizer for z/OS V1 R0 M0    ❷
                              optimized COBPERF6 on 2015/09/03 at 12:57:14
                              using ARCH(10)

Machine Instruction . . . . : 6D008094      DD    FR0,148(,R8)
  At Address. . . . . . . . : 16B010D8 (Entry point COBPERF6 offset X'750')
  AMODE . . . . . . . . . . : 31
  Failing Operand . . . . . : Second operand
  First Operand (FR0) . . . : 42DE0000 00000000
  Second Operand Address. . : 16BB4164 (Module COBPERF6 program COBPERF6
                              WORKING-STORAGE SECTION BLW=0000 + X'94', symbol
                              ELEMENT-2, source line # 16 - 433820 bytes of
                              storage addressable)
  Second Operand Length . . : 8
  Second Operand Storage. . : 00000000 00000000   *........*

Program Status Word (PSW) . : 078D2000 96B010DC

General Purpose Registers (AMODE: 64 31 24 , Bytes: Dec Hex ):  ❸
   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.
```

Notes:

❶

The COBOL PERFORM traceback shows the sequence of PERFORMs, including nested PERFORMs, that were executed to arrive at the event source line. The traceback will not appear if the compiler inlined the procedure, which can happen when OPT is in effect.

❷

Automatic Binary Optimizer for z/OS® (ABO) information will only appear here if a COBOL program has been optimized by ABO.

❸

The general purpose registers are initially displayed in accordance with the event AMODE, or defaults to AMODE 31 if no event AMODE has been determined. However, by selecting the AMODE 64, 31 or 24 point-and-shoot fields, the register display changes accordingly. In addition, the default display of the number of bytes in the register description is decimal, but this can be changed to hexadecimal or back to decimal again by selecting the Dec or Hex point-and-shoot fields.

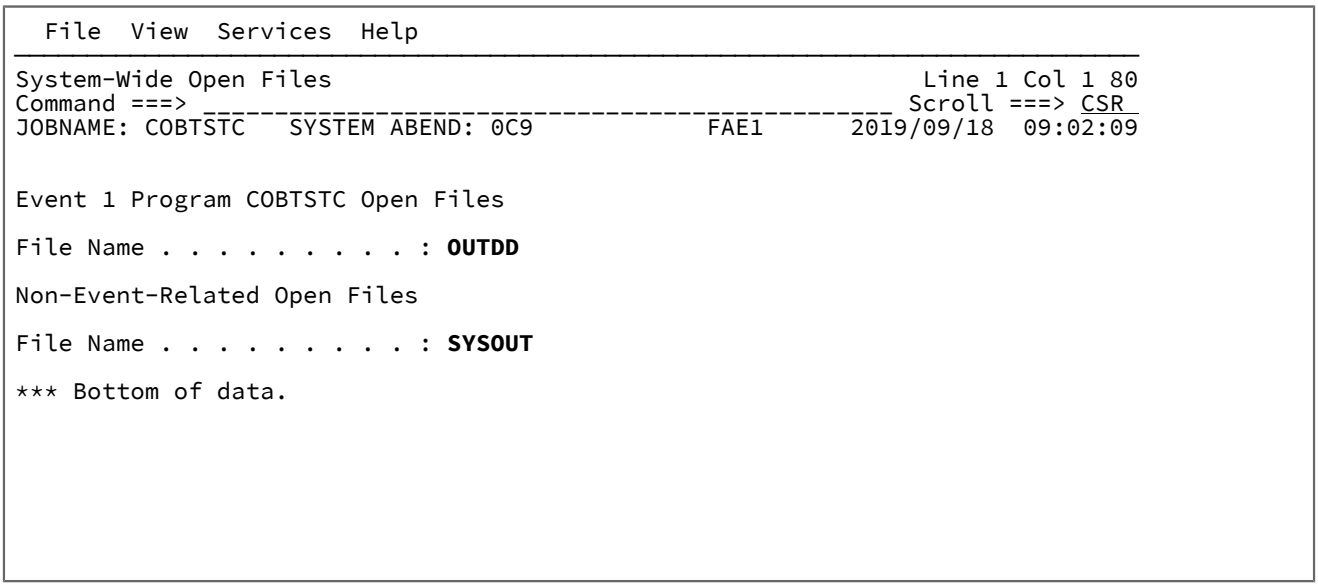
All information that is associated with the currently selected event is either already included in the displayed information, or a point-and-shoot link to the information is provided in yellow. Such links include message and abend codes for which an explanation can be provided if selected (refer to Expanding messages and abend codes on page 206) and associated storage areas (refer to Displaying associated storage areas on page 202).

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

Primary option: Open Files

Selecting the *"Open Files"* option provides you with a display that lists all open files which could not be associated with any particular event, as well as files that might be listed in the detail section of the report for individual events. An example of the open file list is shown in Figure 100: Sample System-Wide Open Files display on page 174.

Figure 100. 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 101: Sample File Information display on page 175.

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

Figure 101. Sample File Information display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
File Information                                              Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
JOBNAME: COBTSTC   SYSTEM ABEND: 0C9            FAE1      2019/09/18  09:02:09

File Name . . . . . . . . . : OUTDD
  Data Set Name . . . . . . : NWILKES.OUT80S ❶
  File Attributes . . . . . : ORGANIZATION=SEQUENTIAL, ACCESS MODE=SEQUENTIAL,
                              RECFM=VARIABLE BLOCKED  SPANNED
  Last I/O Function . . . . : WRITE
  Open Status . . . . . . . : OUTPUT
  File Status Code. . . . . : 0

  Last Record Written -2. . : RDW=001C0100 Segment data length 24, variable re
  Address  Offset    Hex                                  EBCDIC / ASCII
  16599FC0           C9C9C9F9 F9F94040 40404040 40404040 *III999           *
  16599FD0       +10 40404040 40404040                   *                 *

  Last Record Written -1. . : RDW=00240300 Segment data length 32, variable re
  Address  Offset    Hex                                  EBCDIC / ASCII
  16599FE0           40404040 40404040 40404040 40404040 *                 *
        Line 16599FF0 same as above

  Last Record Written . . . : RDW=001C0200 Segment data length 24, variable re
  Address  Offset    Hex                                  EBCDIC / ASCII
  16599EF0           40404040 40404040 40404040 40404040 *                 *
  16599F00       +10 40404040 40404040                   *                 *

  Current Record Build Area : RDW is zero, no length assigned yet
  Address  Offset    Hex                                  EBCDIC / ASCII
  1659AFA4           D1D1D1C1 C1C14040 40404040 40404040 *JJJAAA           *
  1659AFB4       +10 40404040 40404040 40404040 40404040 *                 *
        Lines 1659AFC4-1659AFD4 same as above
  1659AFE4       +40 40404040 40404040 40404040 40406E6E *               >>*
  1659AFF4       +50 40404040                            *                 *

  Associated File Control Blocks ❷

*** Bottom of data.
```

If the data set shown at ❶ is QSAM, VSAM, or IAM, and exists, then it is presented as a point-and-shoot field, that if selected, shows a selection menu from where either Edit, View or Browse can be chosen. If File Manager for z/OS® is available, then it is used to perform the requested function against the data set. Otherwise, if the data set is QSAM non-spanned record format, then ISPF is invoked.

For more information about the presentation of open file buffers, see Open file record information on page 272.

An "Associated File Control Blocks" point-and-shoot field might be available at the bottom of the File Information display, as shown at ❷ 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 102: Sample Associated File Control Blocks display on page 176.

Figure 102. Sample Associated File Control Blocks display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Associated File Control Blocks                                 Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
JOBNAME: COBTSTC   SYSTEM ABEND: 0C9            FAE1      2019/09/18  09:02:09

Data Extent Block (DEB) at Address 008BCD84 :
Address  Offset   Hex                                      EBCDIC / ASCII
008BCD84          038C2CF0 10000000 E8000000 *    ...0....Y...*
008BCD90        +C 0F001100 01000000 FF000000 8F01A038 *...............*
008BCDA0       +1C 048BCD60 10F41AE0 000000FF 000000FF *...-.4.\........*
008BCDB0       +2C 000E000F 00010001 00000000 00000000 *...............*
008BCDC0       +3C 00000054 F3C2C1D9 C2D70000 00000000 *....3BARBP......*
008BCDD0       +4C 00000000 00000000 00000000 00000000 *...............*
008BCDE0       +5C 00000000 00000000 E2E6C1D4 00000000 *........SWAM....*
008BCDF0       +6C 00000218 008BCE08 500000E6 008BD6B0 *........&..W..O.*
008BCE00       +7C 008BD140                            *..J             *

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

Primary option: CICS Information

Selecting the *"CICS Information"* option provides you with a display of information that is related to CICS®, as the example shown in Figure 103: Sample CICS Information display on page 176.

Figure 103. Sample CICS Information display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
CICS Information                                               Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
TRANID: MXT1      CICS ABEND: ABC1              FAE1      2022/01/16  12:06:24

CICS Release. . . . . . . . : 0730 (TS 5.6)
Application ID. . . . . . . : C73E1FA1
CICS Transaction ID . . . . : MXT1
CICS Task Number. . . . . . : 02432
CICS Terminal ID. . . . . . : 1006
CICS Terminal Netname . . . : TCPE1006

Select one of the following:
   1. CICS Control Blocks
   2. CICS Transaction Storage Summary
   3. CICS Transaction Storage
   4. Last CICS 3270 Screen Buffer
   5. Summarized CICS Trace
   6. CICS Trace Formatting
   7. CICS Task Trace Table
   8. CICS Recovery Manager
   9. CICS Levels, Commareas, and Channels
  10. CICS Monitoring Data
  11. CICS Event Program Information
  12. CICS Asynchronous Information
  13. CICS System Initialization Parameters
*** Bottom of data.
```

From the CICS Information display, you can display additional CICS information by selecting an option with the cursor or by entering it on the command line.

CICS Control Blocks

Select this option to display CICS® control blocks that are relevant to the transaction abend.

CICS Transaction Storage Summary

Selecting the "CICS® Transaction Storage Summary" link provides you with a display which provides information about the storage areas owned by the current CICS® transaction, as the example shown in Figure 104: Sample CICS Transaction Storage Summary display on page 177.

Figure 104. Sample CICS Transaction Storage Summary display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
CICS Transaction Storage Summary - All Tasks                  Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
Current All - TRANID: CFA       CICS ABEND: FLT1              FAE1      2019/07

                          Tran
Address   Length   Type   ID   Task #   Possible Overlay
00100008  00002D60        NEWC 0002915C n/a
00103008  00000650        CECI 0002916C n/a
00104000  000029F0 USER24              None detected
  Total:  000029F0 USER24
19320008  00000030        NEWC 0002915C n/a
193201E8  000000C0        NEWC 0002915C n/a
193202B8  00003AD0        NEWC 0002915C n/a
19323D98  0000A770        NEWC 0002915C n/a
19330008  000001C0        CECI 0002916C n/a
193301E8  000000C0        CECI 0002916C n/a
193302B8  00007D00        CECI 0002916C n/a
19337FC8  00000120        CECI 0002916C n/a
193380F8  000004F0        CECI 0002916C n/a
1934B170  00000FD0 USER31              None detected
19340660  000000E0 USER31              None detected
19340910  0000A860 USER31              None detected
```

The CICS® Transaction Storage Summary display can be important when a transaction's storage has been overlaid by another task, as it helps identify transactions whose storage was near or adjacent to the overlay. When Fault Analyzer is invoked to analyze a CICS® transaction abend, the storage allocations of all other concurrently running tasks are gathered. This information can then be displayed, during interactive reanalysis, in the CICS® transaction storage summary display.

The display has two point-and-shoot fields, "Current" and "All", which toggle the display accordingly.

You also can display CICS® transaction storage by using the CICSSTG command. See CICSSTG on page 99.

CICS Transaction Storage

Select this option to view CICS® storage allocated by the abending transaction. This display shows a composite view of the same storage areas that you can select individually in the CICS® Transaction Storage Summary display.

Alternatively, you can use the CICS®STG command to display CICS® transaction storage. See CICSSTG on page 99.

Last CICS 3270 Screen Buffer

Selecting the *"Last CICS 3270 Screen Buffer"* link displays the last 3270 screen buffer written by CICS, as in the example shown in Figure 105: Sample Last CICS 3270 Screen Buffer display on page 178.

Figure 105. Sample Last CICS 3270 Screen Buffer display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Last CICS 3270 Screen Buffer                                  Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
Normal Hex - TRANID: SK00   CICS ABEND: ASRA      CICS02   2019/06/07 11:10:32

    Column
Row ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+
  1 sk00*** CICB0CB0 - START OF PROGRAM.....................................
  2 .......................................................................
  3 .......................................................................
  4 .......................................................................
  5 .......................................................................
  6 .......................................................................
  7 .......................................................................
  8 .......................................................................
  9 .......................................................................
 10 .......................................................................
 11 .......................................................................
 12 .......................................................................
 13 .......................................................................
 14 .......................................................................
 F1=Help      F3=Exit      F5=RptFind   F6=Actions   F7=Up        F8=Down
F10=Left     F11=Right
```

All non-printable characters are shown as periods (.).

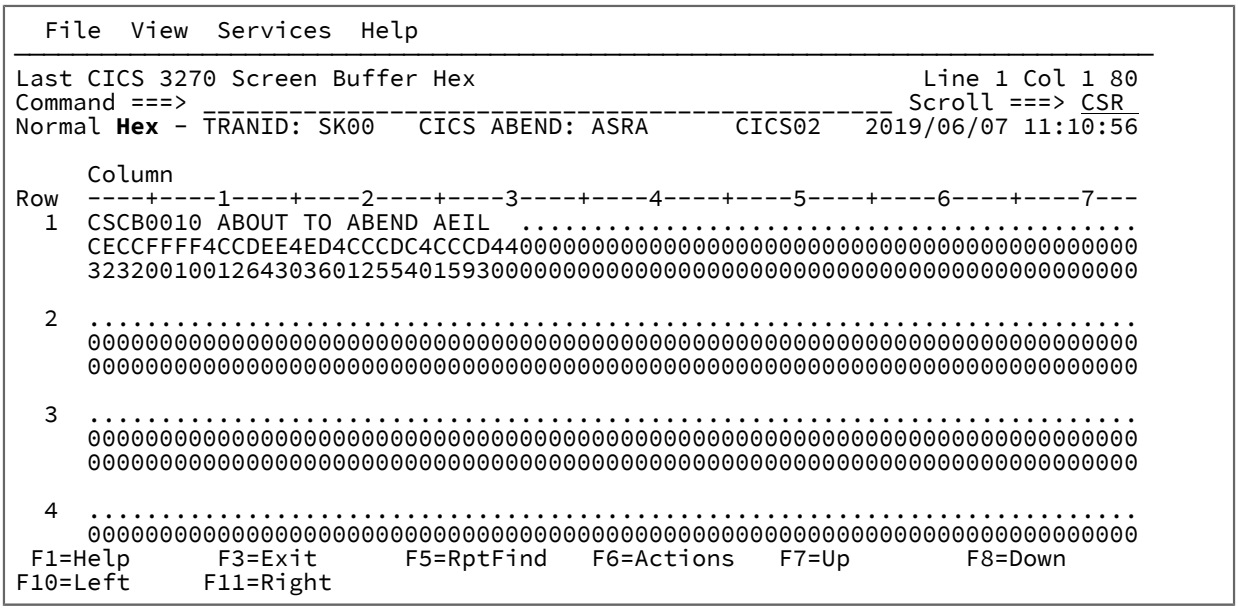
It is not always possible for Fault Analyzer to provide CICS 3270 screen buffer information. For example, if there are indications of VTAM® session errors on the terminal, or if one of the following CICS® abends have occurred (* indicates a wildcard character), then no 3270 screen buffer information is provided:

AEXZ
AKC3
AKCT
AKK*
ATC*
ATN*
AZCT
AZI*
AZTS

Two point-and-shoot fields are provided under the panel title to toggle the state of the display:

- **Normal** displays only the EBCDIC characters of the last screen buffer.
- **Hex** also displays the hex values of each character below each row:

Figure 106. Sample Last CICS 3270 Screen Buffer Hex display



The selected option is remembered between interactive analysis sessions.

When the CICS 3270 screen buffer is available, it is displayed to all users by default. You can restrict its display to users who have READ access or higher to an XFACILIT profile named IDI_FEATURE_DISPLAY_CICS_3270BUFFER. For example, the following RACF commands restrict the 3270 screen buffer display to members of group `DEV1`:

```
RDEFINE XFACILIT IDI_FEATURE_DISPLAY_CICS_3270BUFFER UACC(NONE)
PERMIT IDI_FEATURE_DISPLAY_CICS_3270BUFFER CLASS(XFACILIT) ID(DEV1) ACCESS(READ)
```

Note: The contents of the CICS 3270 screen buffer are visible if a user views a fault entry outside of Fault Analyzer (for example, using ISPF BROWSE). To control this level of access, define the history file data-set-profile access as UACC(NONE), and define an XFACILIT profile that grants access to the history files. See Using the XFACILIT resource class for history file fault entries on page 363.

Summarized CICS Trace

Selecting the *"Summarized CICS® Trace"* link provides you with a display of the CICS® trace table as the example shown in Figure 107: Sample Summarized CICS Trace display on page 180.

Figure 107. Sample Summarized CICS Trace display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Summarized CICS Trace                                        Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
TRANID: CS31      CICS ABEND: ASRA              FAE2      2019/08/16  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_DUMPCODE/EXCEPTION DUMPCODE_NOT
00027 QR    DU 0601 DUTM  EXIT  LOCATE_SYSTEM_DUMPCODE/OK FFFFFFFF,1,0,0,NO,YE
00027 QR    DU 0601 DUTM  EXIT  COMMIT_SYSTEM_DUMPCODE/OK
```

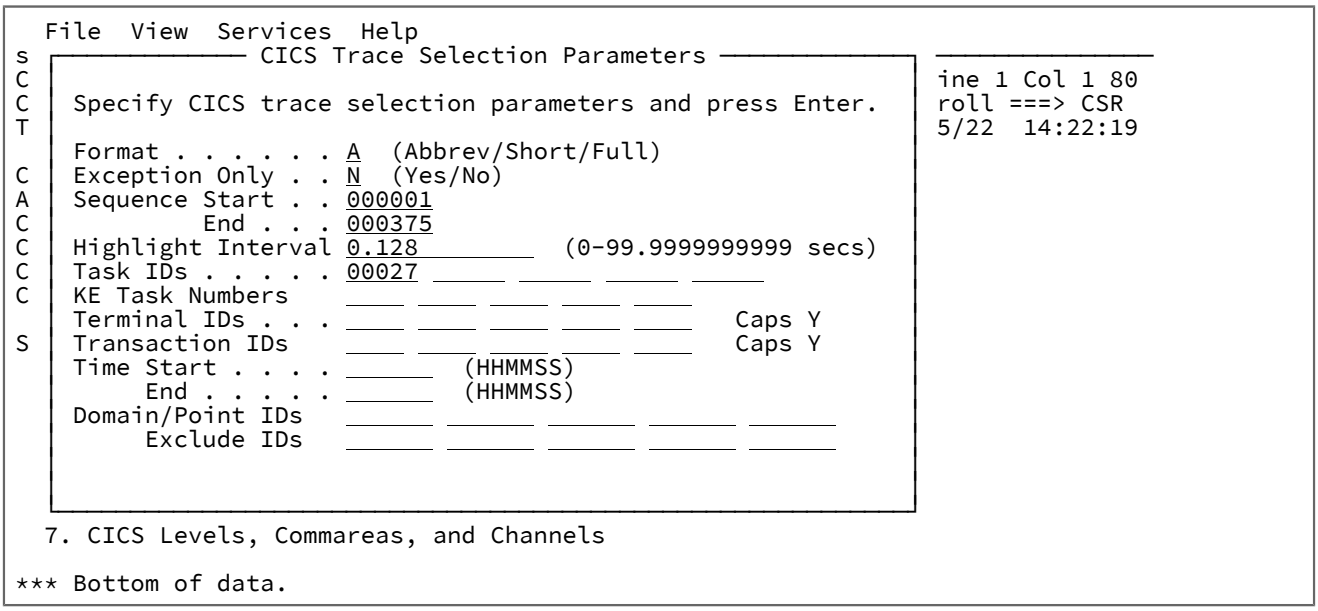
The standard CICS® trace table is enhanced by Fault Analyzer for greater ease of use. Information is added to indicate the call point origin addresses, including the module name, CSECT name, offset within CSECT, and source line or listing statement number if available.

As in all other sections of the interactive report, source line numbers or listing statement numbers are selected by placing the cursor on them and pressing Enter. This provides a full source listing display as shown in Displaying source code on page 206.

CICS Trace Formatting

Selecting the *"CICS® Trace Formatting"* link provides you with a display that permits specific selection parameters to be entered for the CICS® trace, as the example shown in Figure 108: Sample CICS Trace Selection Parameters display on page 181.

Figure 108. Sample CICS Trace Selection Parameters display



The following parameters can be specified:

Format

Specifies the level of trace formatting:

A

The abbreviated form of the trace, with one line per entry.

S

The short formatted display, consisting of the abbreviated entry plus fully formatted parameter list, return address, time, and interval.

F

The fully formatted display of all the data in each entry.

Exception Only

Specifies that only exception entries in the internal trace are to be displayed.

Sequence Start/End

Specifies which sequence numbers are to be selected (sequence numbers start at 1 and are up to 6 digits in length). You can specify Start, End or both.

Highlight Interval

Specifies the interval between internal trace entries after which entries are highlighted (with an asterisk).

Task IDs

Specifies up to five task identifiers whose trace entries are to be displayed. An ID value can be one of:

- Any number up to 5 digits in length.
- Any of JAS, J01 through J99.
- III, TCP, or DST.
- A two character domain ID of the attaching domain (for non-TCA tasks).

KE Task Numbers

Specifies up to 5 four hexadecimal digit kernel task numbers to be displayed.

Terminal IDs

Specifies the terminal identifiers of up to five terminals for which trace entries are to be displayed. If any terminal IDs contain lowercase, set Caps to N and enter all IDs as case-sensitive. When Caps is Y, all entries are translated to uppercase.

Transaction IDs

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

Time Start/End

Specifies the time period for which trace entries are to be displayed. You can specify Start, End or both.

Domain/Point IDs

Specifies up to 5 two-character domain IDs, and optionally followed by a trace point ID within that domain. These IDs are included. Here are the domain IDs:

```
AP BA BR CC DD DE DH DM DP DS DU EJ EM EX GC IE II IS KE LC
LD LG LM ME ML MN NQ OT PA PG PI PT RL RM RS RX RZ SH SJ SM
SO ST TI TR TS US WB W2 XM XS
```

The trace point ID can be up to four hexadecimal digits in length (if fewer than 4 digits are entered, then it is treated as a generic value).

If no domain or trace point IDs are specified in this selection parameter field, then the default is to include all trace entries. The exception is any trace entries that are excluded by way of the Exclude IDs selection parameter. Otherwise, only trace records that match the specified IDs (which are not also excluded via the Exclude IDs selection parameter) are shown.

Exclude IDs

Specifies up to 5 two-character domain IDs (same values as for Domain/Point IDs), optionally followed by a trace point ID within that domain. These IDs are excluded. The trace point ID can be up to four hexadecimal digits in length (if fewer than 4 digits are entered, then it is treated as a generic value).

After you make any optional changes to parameters, press Enter to view the CICS® trace. An example of the CICS® Trace display is shown in Figure 109: Sample CICS Trace display on page 183.

Figure 109. Sample CICS Trace display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
CICS Trace                                                    Entry 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
Abbrev Short Full - TRANID: CS65      IDISNAP CALL                FAE2      2
00027 QR    SM 0D02 SMMF  EXIT  FREEMAIN/OK          CICS storage at 174E7748
00027 QR    SM 0C01 SMMG  ENTRY GETMAIN              2000,NO,00,CICS
00027 QR    SM 0C02 SMMG  EXIT  GETMAIN/OK           174CF028
00027 QR    SM 0D01 SMMF  ENTRY FREEMAIN             174CD018,CICS
00027 QR    SM 0D02 SMMF  EXIT  FREEMAIN/OK          CICS storage at 174CD018
00027 QR    SM 0C01 SMMG  ENTRY GETMAIN              2000,NO,00,CICS
00027 QR    SM 0C02 SMMG  EXIT  GETMAIN/OK           174D1038
00027 QR    SM 0D01 SMMF  ENTRY FREEMAIN             174CF028,CICS
00027 QR    SM 0D02 SMMF  EXIT  FREEMAIN/OK          CICS storage at 174CF028
00027 QR    SM 0C01 SMMG  ENTRY GETMAIN              2000,NO,00,CICS
00027 QR    SM 0C02 SMMG  EXIT  GETMAIN/OK           174CD018
00027 QR    SM 0D01 SMMF  ENTRY FREEMAIN             174D1038,CICS
00027 QR    SM 0D02 SMMF  EXIT  FREEMAIN/OK          CICS storage at 174D1038
00027 QR    SM 0C01 SMMG  ENTRY GETMAIN              2000,NO,00,CICS
00027 QR    SM 0C02 SMMG  EXIT  GETMAIN/OK           174CF028
00027 QR    SM 0D01 SMMF  ENTRY FREEMAIN             174CD018,CICS
00027 QR    SM 0D02 SMMF  EXIT  FREEMAIN/OK          CICS storage at 174CD018
00027 QR    SM 0C01 SMMG  ENTRY GETMAIN              2000,NO,00,CICS
00027 QR    SM 0C02 SMMG  EXIT  GETMAIN/OK           174E7748
```

Initially, the trace is formatted in accordance with the settings made on the CICS® Trace Selection Parameters display. However, at any time while the trace is displayed, the format can be changed by selecting one of the point-and-shoot fields provided at the top of the display. The top-most trace entry that is displayed remains at the top, regardless of the format selected.

Note: Fault Analyzer uses CICS® supplied trace formatting modules to format CICS® trace entries. These modules are named DFHTR*nnn*, where *nnn* is the CICS® version. If fault entries contain CICS® trace records from multiple versions of CICS®, you must ensure that all the necessary versions of DFHTR*nnn* are available to Fault Analyzer.

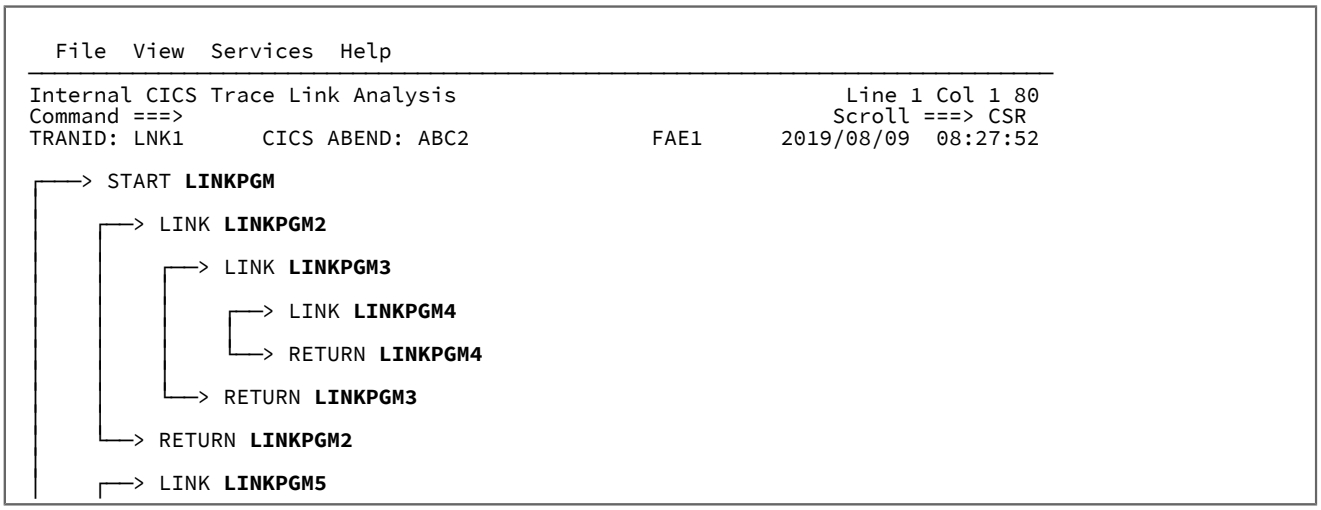
Internal CICS Trace Link Analysis (CICSLINK)

If you use the CICSLINK command when you view a formatted CICS® trace, a hierarchical representation of CICS® LINK entries and exits is provided. The CICSLINK command defaults to PF4 on CICS® formatted trace displays. For more information about the CICSLINK command, see CICSLINK on page 99.

You can also use this feature for the formatted CICS® traces that are provided by CICS® system dumps or auxiliary trace data sets. When you use the CICSLINK command for a formatted trace from one of these sources, you are prompted to provide the task number to be analyzed. When you place the cursor on a task number in the formatted trace and press **PF4**, the CICS® Trace Link Analysis for that task number is opened.

An example of the display is shown in the following figure.

Figure 110. Sample CICS Trace Link Analysis Display



This diagram is generated by parsing the CICS® trace for EXEC CICS® LINK entries. The following headings appear in the hierarchy that is displayed.

START

This entry shows the start of the CICS® trace and always appears at the beginning of the Link Analysis where the first program begins execution.

LINK

This entry shows the point at which a LINK CICS® trace entry occurs, marking the start of a nested program execution in the CICS® transaction.

RETURN

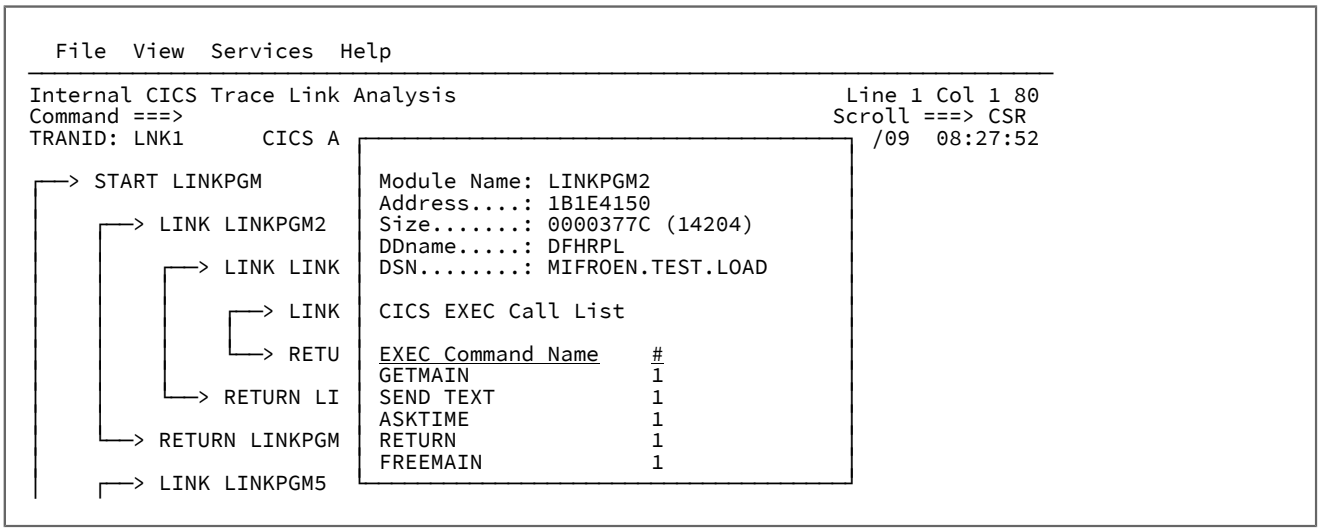
This entry shows the point at which the current highest level linked module exits.

ABEND

This entry indicates the point at which an ABEND occurs in the transaction.

When you select one of the point-and-shoot program name fields in the hierarchy, further details on the selected program and events that occurred during its execution are displayed. See the following figure as an example.

Figure 111. Sample CICS Trace Link Analysis additional information display



Module Name

The name of the linked load module.

Address

The address of the program. The address is unavailable when a CICS® auxiliary trace data set or CICS® system dump is being analyzed.

Size

The size of the program. The size is unavailable when a CICS® auxiliary trace data set or CICS® system dump is being analyzed.

DDname

The load module origin DDname. The DDname is unavailable when a CICS® auxiliary trace data set or CICS® system dump is being analyzed.

DSN

The load module origin data set name. The DSN is unavailable when a CICS® auxiliary trace data set or CICS® system dump is being analyzed.

CICS® EXEC Call List

A list of all CICS® EXEC commands that are recorded in the CICS® Trace during the execution of this program. The list includes the EXEC command names and the number of individual command invocations. The list does not include EXEC commands that are called by nested programs.

Storage Violations Detected

A list of all CICS® storage violations that are detected from the SM trace entries in the CICS® Trace during the execution of this program. The list includes the type of violation and the number of individual violations. The list does not appear if no storage violations occurred and does not include storage violations that are caused by nested programs.

CICS Task Trace Table

Select this option to display the CICS® task trace block. This trace block contains the last 20 trace entries created by the task.

CICS Recovery Manager

Select this option to view information specific to the CICS® Recovery Manager.

CICS Levels, Commareas, and Channels

Selecting the *"CICS® Levels, Commareas, and Channels"* point-and-shoot field provides you with a display as the example shown in Figure 112: Sample CICS Levels, Commareas, and Channels display on page 186.

Figure 112. Sample CICS Levels, Commareas, and Channels display

```
  File  View  Services  Help
_____________________________________________________________________________
CICS Levels, Commareas, and Channels                          Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
TRANID: CONT      CICS ABEND: CVER              FAE1      2019/09/17  14:39:58

 Lnk Ev                 Commarea          Created                    Passed f
  #  # Type Program  Address  Len  Channel(s)       Container name   Channel
  1  4 EXEC CONTEST                PASS1001         CONT1002
                                                    CONT1001
                                   LOCAL102         CONT1002
                                                    CONT1001
                                   COMMONCHANNEL    COMMONCONTAINER
                                   LOCAL101         CONT1002
                                                    CONT1001
  2  8 EXEC CONTEST2               PASS2001         CONT2002         PASS1001
                                                    CONT2001
                                   LOCAL202         CONT2002   ❶
                                                    CONT2001
                                   COMMONCHANNEL    COMMONCONTAINER
                                   LOCAL201         CONT2002
                                                    CONT2001
  3 12 EXEC CONTEST3               PASS3001         CONT3001-1MB     PASS2001
```

Selecting the container name point-and-shoot field provides information about the container type. For example, selecting the CONT2002 point-and-shoot field at ❶ in Figure 112: Sample CICS Levels, Commareas, and Channels display on page 186 might display:

Figure 113. Sample CICS container display

```
 CICS Container                                                Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
TRANID: CONT      CICS ABEND: CVER              FAE1      2019/09/17  14:39:58


Container CONT2002 at address 177E33A0 has a length of X'1024'

Segment             Segment   Segment Preview  Segment Preview
Address             Offset    (EBCDIC)         (Hex)
17912028            X'000000' CONT2002/L202... C3D6D5E3 F2F0F0F2 61D3F2F0 F24B
1790C328            X'000FD8' ---------------- 60606060 60606060 60606060 6060
```

For some standard CICS® containers, a description of the container is also provided. For an example, see Figure 114: Sample standard CICS container display on page 187.

Figure 114. Sample standard CICS container display

```
_____________________________________________________________________________
Container Data                                                Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
TRANID: COBA      CICS ABEND: ATCV              FAE1      2019/04/06  12:32:44

DFHWS-BODY Contains the body section of the SOAP envelope. Typically, the
application will modify the contents.
                   ❷
Data Length X'3FE' Format XML Data

Address  Offset    Hex                                  ASCII / EBCDIC
16A4B788                          3C534F41 502D454E *        <SOAP-EN*
16A4B790        +8  563A426F 64793E20 20202020 20202020 *V:Body>         *
16A4B7A0       +18  20202020 20202020 20202020 20202020 *                *
         Lines 16A4B7B0-16A4B7C0 same as above
16A4B7D0       +48  20202020 20200D0A 3C434943 53564552 *      ..<CICSVER*
16A4B7E0       +58  324F7065 72617469 6F6E3E20 20202020 *2Operation>     *
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
```

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 ❷ shows the formatted XML data:

Figure 115. Sample CICS XML formatter display

```
_____________________________________________________________________________
XML Formatter                                                 Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
TRANID: COBA      CICS ABEND: ATCV              FAE1      2019/04/06  12:32:44

   <SOAP-ENV:Body>
     <CICSVER2Operation>
       <commarea>
         <DisplayOrUpdate>
           T
         </DisplayOrUpdate>
         <LinkRC>
           0
         </LinkRC>
         <msg1>
           Message 1
         </msg1>
         <msg2>
           Message 2
         </msg2>
         <msg3>
           Message 3
         </msg3>
```

CICS Monitoring Data

Select this option to view detailed information from the CICS® Transaction Monitoring Area (TMA) control block.

CICS Event Program Information

Select this option to view a list of all entries found in the CICS® processing program table (PPT) for this fault entry.

CICS System Initialization Parameters

You can display the values of the CICS® system initialization parameters during interactive reanalysis of a CICS fault entry.

- From the "CICS Information" section of the interactive report, you can select the "CICS System Initialization Parameters" option.
- From the ISPF command line, you can enter the SIT command.

The values are listed alphabetically. You can use a point-and-shoot field to display the values by category.

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 116: Sample System-Wide Messages display on page 188.

Figure 116. Sample System-Wide Messages display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
System-Wide Messages                                          Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
JOBNAME: P00398    SYSTEM ABEND: 0CB              MVS2     2019/03/24 13:54:05


Event 5 Messages

CEE3211S Job-specific text not available

Non-Event-Related Messages

$HASP375 P00398   ESTIMATED  LINES EXCEEDED

$HASP375 P00398   ESTIMATE EXCEEDED BY              10,000  LINES

$HASP375 P00398   ESTIMATE EXCEEDED BY              20,000  LINES

*** Bottom of data.


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

Primary option: DB2 Information

Selecting the *"DB2 Information"* option provides you with a display of information that is related to DB2® as the example shown in Figure 117: Sample DB2 Information display (Part 1 of 3) on page 189.

Figure 117. Sample DB2 Information display (Part 1 of 3)

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
DB2 Information                                                Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR_
JOBNAME: DAC2DH07  SYSTEM ABEND: 806              MVS2     2019/11/26 09:44:50

DB2 Subsystem DBT6

DB2 Version . . . . . : V6R1M0
Plan Name . . . . . : DAC2DH (Bound 2019/04/25 12:13:14)
Plan Owner. . . . . : HARRIDA
Package Name. . . . . : DAPNDH11 (Created 2019/07/30 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
                        2019/04/25 12:07:57)
Consistency Token . . . . . : X'177522E41D026D08'
Primary Authorization ID. . : HARRIDA
Current SQL ID. . . . . : HARRIDA
```

Figure 118. Sample DB2 Information display (Part 2 of 3)

```
Last Executed SQL Statement:
  List
  Stmt #
  000227                    EXEC SQL FETCH HVAR1 INTO :HVTABLE

Fault Analyzer Event #. . . : 7 (Program DAPNDH11)
Declare Cursor Stmt No. . . : 133
Declare Cursor Stmt . . . . : DECLARE HVAR1 CURSOR FOR SELECT HVARKEY ,
                              VCHAR300 , DEC9 , LINT , CHARHEX , TIMESTMP
                              FROM HVARS
Open Cursor Stmt No . . . . : 166
Open Cursor Stmt. . . . . . : OPEN HVAR1

Browse Table Contents in File Manager:
  HVARS

Output Host Variables:
  Name and Data Type. . . . : HVTABLE.HVARKEY CHARACTER(6)
    At Address. . . . . . . : 0007EA26
    Data Value. . . . . . . : 000003

  Name and Data Type. . . . : HVTABLE.VCHAR300 VARCHAR(300)
    At Address. . . . . . . : 0007EA2C
    Data Value. . . . . . . : This is record 3
                              ----------------------------------------------
                              ==================================101
                              ----------------------------------------------
                              ----------------------------------------------
                              201
                              ----------------------------------------------
                              ----------------------------------------------->

  Name and Data Type. . . . : HVTABLE.DEC9 DECIMAL(9,4)
    At Address. . . . . . . : 0007EB5A
    Data Value. . . . . . . : 12345.6789

  Name and Data Type. . . . : HVTABLE.LINT INTEGER
    At Address. . . . . . . : 0007EB74
    Data Value. . . . . . . : 214748364

  Name and Data Type. . . . : HVTABLE.CHARHEX CHARACTER(21)
    At Address. . . . . . . : 00082A8C
    Data Value. . . . . . . : *** Data format error: Character string contains
                              non-printable character at offset X'B' ***
    Host Variable Storage:
    Address  Offset    Hex                                EBCDIC / ASCII
    00082A8C           E4959799 8995A381 82938522 83888199 *Unprintable.char*
    00082A9C       +10 8183A385 99                         *acter           *

  Name and Data Type. . . . : HVTABLE.TIMESTMP CHARACTER(26)
    At Address. . . . . . . : 000869C9
    Data Value. . . . . . . : 2019-09-30-10.44.59.000001
```

When you select the SQL table point-and-shoot field from the display shown in Figure 118: Sample DB2 Information display (Part 2 of 3) on page 190, the File Manager DB2 Browse panel displays with the details of the selected table prefilled. The SQL table point-and-shoot field does not display unless File Manager is installed.

Figure 119. Sample DB2 Information display (Part 3 of 3)

```
DB2 Control Blocks

SQL Communications Area (SQLCA) for Event # 7 Program DAPNDH11 at Address
1747E9F8 :
Offset     Field        Value
Dec  Hex   Name         Hex                                      EBCDIC / ASCII
   0   (0) SQLCAID      E2D8D3C3 C1404040                        *SQLCA           *
   8   (8) SQLCABC      00000088                                 *...h            *
  12   (C) SQLCODE      00000064                                 *....            *
                        SQLCODE 100 Explanation
  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      FFFFFF92 00000000 00000000 FFFFFFFF      *...k...........*
 112  (70)              00000000 00000000                        *........        *
 120  (78) SQLWARN      40404040 40404040 404040                 *                *
 131  (83) SQLSTATE     F0F2F0F0 F0                              *02000           *
                        SQLSTATE 02000 Explanation

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
```

Information for one or more DB2® subsystems are provided. If the SQLCA data area for a DB2® subsystem is identical to one available from an Event Details display, then only a reference to the event is provided here, as shown in the example above. Otherwise, the contents of the SQLCA are shown in this display.

Primary option: IMS Information

Selecting the *"IMS™ Information"* option provides you with a display of information that is related to IMS™. There is an example display in Figure 120: Sample IMS Information display (Part 1 of 4) on page 191.

Figure 120. Sample IMS Information display (Part 1 of 4)

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
IMS Information                                              Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
JOBNAME: IBCB0030  USER ABEND: 4036              MVS2     2019/11/29 13:51:55

Summary Detail - JOBNAME: IDCB0070  SYSTEM ABEND: 0CB            FAE1 2015

IMS Version . . . . . : V13R1M0
IMS Region Type . . . . . : Batch Message Processing Region
IMS Subsystem Name. . . . . : IB81
Application Program Name. . : IDCB0070
PSB Name. . . . . : DFHSAM25
```

Figure 121. Sample IMS Information display (Part 2 of 4)

```
Last DL/I Call Parameter List

Note that storage addressed by individual parameters might no longer be valid.

Parameter 1 . . . . . : 0002BD68
  DL/I Call Function. . . . : GU (Get Unique)

Parameter 2 . . . . . : 0001A098
  (See "IMS Control Blocks" for details of this PCB)

Parameter 3 . . . . . : 0002BCA8

Parameter 4 . . . . . : 8002BD48
  SSA # 1 . . . . . : PARTROOT(         =          )

IMS Control Blocks

Input/Output Program Communications Blocks (IOPCBs)

IOPCB:
  At Address. . . . . : 0001A020
  PCB Name. . . . . : IOPCB
  Relative PCB Number . . . : 1
  PCB Type. . . . . : I/O
  Logical Terminal ID . . . : n/a
  Status Code . . . . . : '  ' (Normal status)
  User ID . . . . . : DFHSAM25
  Group Name. . . . . : n/a
  Formatting Module Name. . : n/a

Data Base Program Communications Blocks (DBPCBs)

DBPCB (*** Current/Last Used ***):
  At Address. . . . . : 0001A098
  PCB Name. . . . . : n/a
  Relative PCB Number . . . : 2
  PCB Type. . . . . : Data Base or Online
  Data Base Name. . . . . : DI21PART
  Segment Level . . . . . : 01
  Status Code . . . . . : '  ' (Normal status)
  Processing Options. . . . : A
  Segment Name. . . . . : PARTROOT
  Number of Segments. . . . : 5
  Key Feedback Length . . . : 17

  Key Feedback Data:
    Address  Offset    Hex                                     EBCDIC / ASCII
    0001A0F4           F0F2C1D5 F9F6F0C3 F1F04040 40404040 *02AN960C10      *
    0001A104      +10  40                                      *               *
```

Figure 122. Sample IMS Information display (Part 3 of 4)

```
JCB DL/I Call Trace (Most recent call first):
      Call                        Status
    # Code Description             Code    Description
    1 01   GHU or GU               '  '    Status good.
    2 03   GHN or GN               AB      Segment I/O area required; none specified
                                           in call. Only applies to full-function DEQ
                                           calls.
                                   -or-
                                   BB      Call could not be completed because data
                                           was unavailable and updates are backed out
                                           only since the last commit point.
                                   -or-
                                   GB      End of database.
    3 03   GHN or GN               '  '    Status good.
    4 03   GHN or GN               '  '    Status good.
    5 03   GHN or GN               '  '    Status good.
    6 03   GHN or GN               '  '    Status good.

IMS Accounting Information

DL/I Data Base Calls:
  GU Calls. . . . . . . . . : 151
  GN Calls. . . . . . . . . : 10050
  GNP Calls . . . . . . . . : 0
  GHU Calls . . . . . . . . : 0
  GHN Calls . . . . . . . . : 0
  GHNP Calls. . . . . . . . : 0
  ISRT Calls. . . . . . . . : 0
  DLET Calls. . . . . . . . : 0
  REPL Calls. . . . . . . . : 0
  Total Calls . . . . . . . : 10201
DL/I Message Calls:
  GU Calls. . . . . . . . . : 0
  GN Calls. . . . . . . . . : 0
  ISRT Calls. . . . . . . . : 0
  PURG Calls. . . . . . . . : 0
  CMD Calls . . . . . . . . : 0
  GCMD Calls. . . . . . . . : 0
  CHNG Calls. . . . . . . . : 0
  AUTH Calls. . . . . . . . : 0
  SETO Calls. . . . . . . . : 0
```

Figure 123. Sample IMS Information display (Part 4 of 4)

```
DL/I System Service Calls:
  APSB Calls. . . . . . . . : 0
  DPSB Calls. . . . . . . . : 0
  GMSG Calls. . . . . . . . : 0
  ICMD Calls. . . . . . . . : 0
  RCMD Calls. . . . . . . . : 0
  CHKP Calls. . . . . . . . : 0
  XRST Calls. . . . . . . . : 0
  ROLB Calls. . . . . . . . : 0
  ROLS Calls. . . . . . . . : 0
  SETS Calls. . . . . . . . : 0
  SETU Calls. . . . . . . . : 0
  INIT Calls. . . . . . . . : 0
  INQY Calls. . . . . . . . : 0
  LOG Calls . . . . . . . . : 0
  DB DEQ Calls. . . . . . . : 0
  VSAM I/O Count (READs). . : 1800
  VSAM I/O Count (WRITEs) . : 0
  OSAM I/O Count (READs). . : 0
  OSAM I/O Count (WRITEs) . : 0
  Total DL/I I/O Count
    (OSAM+VSAM) . . . . . . : 1800
  Total ESAF Calls. . . . . : 0
  FP FLD Calls. . . . . . . : 0
  FP POS Calls. . . . . . . : 0
  RLSE Calls. . . . . . . . : 0
  SAVE Calls (XQUERY) . . . : 0
  RSTR Calls (XQUERY) . . . : 0
  COPY Calls (XQUERY) . . . : 0
  DL/I ICAL Calls . . . . . : 0

IMS Parameter Modules

Module DFSPRPX0 Address . . : 000078B0

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

The IMS™ Information display provides:

- General information about the IMS™ region
- Last DL/I call parameter list
- Information about all PCBs

  All PCBs are shown in the order of their relative PCB number with identification of current (or most recently used) PCBs.

  If available, JCB call trace information follows each data base PCB, showing the most recent call and up to five previous calls.
- IMS™ accounting information
- The address of the DFSPRPX0 parameter module

  Selecting the address point-and-shoot field permits you to view the module storage in hex-dump format.

By default, the IMS™ Information display is shown in the *"Detail"* format. In this format, all PCBs are shown fully expanded. By instead selecting the *"Summary"* point-and-shoot field from the display header, PCBs are instead provided condensed in a table format, as in the example shown in Figure 124: Summary IMS Information display (Part 1 of 2) on page 195.

Figure 124. Summary IMS Information display (Part 1 of 2)

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
IMS Information                                                 Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
JOBNAME: IBCB0030  USER ABEND: 4036              MVS2     2019/11/29 13:51:55

Summary Detail - JOBNAME: IDCB0070  SYSTEM ABEND: 0CB           FAE1 2015

IMS Version . . . . . : V13R1M0
IMS Region Type . . . . . : Batch Message Processing Region
IMS Subsystem Name. . . . . : IB81
Application Program Name. . : IDCB0070
PSB Name. . . . . : DFHSAM25

Last DL/I Call Parameter List

Note that storage addressed by individual parameters might no longer be valid.

Parameter 1 . . . . . : 0002BD68
  DL/I Call Function. . . . : GU (Get Unique)

Parameter 2 . . . . . : 0001A098
  (See "IMS Control Blocks" for details of this PCB)

Parameter 3 . . . . . : 0002BCA8

Parameter 4 . . . . . : 8002BD48
  SSA # 1 . . . . . : PARTROOT(         =          )

IMS Control Blocks

Input/Output Program Communications Blocks (IOPCBs)

PCB          Cur PCB      Terminal Status User     Group    Module
#   Address  PCB Name     ID       Code   ID       Name     Name
  1 0001A020     IOPCB    n/a              DFHSAM25 n/a      n/a

Data Base Program Communications Blocks (DBPCBs)

PCB          Cur PCB      DataBase Status Segment Process Segment  Key    # of
#   Address  PCB Name     Name     Code   Level   Options Name     Length Segments
  2 0001A098 *** n/a      DI21PART              01    A     PARTROOT     17        5
```

Figure 125. Summary IMS Information display (Part 2 of 2)

```
IMS Accounting Information

DL/I Data Base Calls:
  GU Calls. . . . . . . . . : 151
  GN Calls. . . . . . . . . : 10050
  GNP Calls . . . . . . . . : 0
  GHU Calls . . . . . . . . : 0
  GHN Calls . . . . . . . . : 0
  GHNP Calls. . . . . . . . : 0
  ISRT Calls. . . . . . . . : 0
  DLET Calls. . . . . . . . : 0
  REPL Calls. . . . . . . . : 0
  Total Calls . . . . . . . : 10201
DL/I Message Calls:
  GU Calls. . . . . . . . . : 0
  GN Calls. . . . . . . . . : 0
  ISRT Calls. . . . . . . . : 0
  PURG Calls. . . . . . . . : 0
  CMD Calls . . . . . . . . : 0
  GCMD Calls. . . . . . . . : 0
  CHNG Calls. . . . . . . . : 0
  AUTH Calls. . . . . . . . : 0
  SETO Calls. . . . . . . . : 0
DL/I System Service Calls:
  APSB Calls. . . . . . . . : 0
  DPSB Calls. . . . . . . . : 0
  GMSG Calls. . . . . . . . : 0
  ICMD Calls. . . . . . . . : 0
  RCMD Calls. . . . . . . . : 0
  CHKP Calls. . . . . . . . : 0
  XRST Calls. . . . . . . . : 0
  ROLB Calls. . . . . . . . : 0
  ROLS Calls. . . . . . . . : 0
  SETS Calls. . . . . . . . : 0
  SETU Calls. . . . . . . . : 0
  INIT Calls. . . . . . . . : 0
  INQY Calls. . . . . . . . : 0
  LOG Calls . . . . . . . . : 0
  DB DEQ Calls. . . . . . . : 0
  VSAM I/O Count (READs). . : 1800
  VSAM I/O Count (WRITEs) . : 0
  OSAM I/O Count (READs). . : 0
  OSAM I/O Count (WRITEs) . : 0
  Total DL/I I/O Count
    (OSAM+VSAM) . . . . . . : 1800
  Total ESAF Calls. . . . . : 0
  FP FLD Calls. . . . . . . : 0
  FP POS Calls. . . . . . . : 0
  RLSE Calls. . . . . . . . : 0
  SAVE Calls (XQUERY) . . . : 0
  RSTR Calls (XQUERY) . . . : 0
  COPY Calls (XQUERY) . . . : 0
  DL/I ICAL Calls . . . . . : 0

IMS Parameter Modules

Module DFSPRPX0 Address . . : 000078B0

*** Bottom of data.
```

From this display, individual PCBs can be selected by placing the cursor on the PCB # point-and-shoot field, and pressing Enter. A sample detailed PCB display is shown in Figure 126: Detailed PCB display on page 197.

Figure 126. Detailed PCB display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
IMS Input/Output Program Communications Block (IOPCB)..        Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
JOBNAME: IDCB0070  SYSTEM ABEND: 0CB            FAE1      2019/01/19 14:17:15

IOPCB:
  At Address. . . . . . . . : 0001A020
  PCB Name. . . . . . . . . : IOPCB
  Relative PCB Number . . . : 1
  PCB Type. . . . . . . . . : I/O
  Logical Terminal ID . . . : n/a
  Status Code . . . . . . . : '  ' (Normal status)
  User ID . . . . . . . . . : DFHSAM25
  Group Name. . . . . . . . : n/a
  Formatting Module Name. . : n/a

*** Bottom of data.
```

Selecting the address point-and-shoot field from this display results in the PCB data area being shown in the Dump Storage display, as in the example shown in Figure 145: Sample Dump Storage suppressed display on page 212.

Primary option: Storage Areas

Selecting the *"Storage Areas"* option provides you with links to any event-related formatted storage areas, any hex-dumped storage ranges, and in case of COBOL, information about any static storage for programs that are no longer on the DSA chain, as the example shown in Figure 127: Sample System-Wide Storage Areas display on page 197.

Figure 127. Sample System-Wide Storage Areas display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
System-Wide Storage Areas                                   Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
TRANID: CD01      CICS ABEND: DSNC               CICS41   2019/04/28 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.
```

Select any event-related point-and-shoot links from this display (such as the "Event 1 Program COBMST3 Storage Areas" link above), by one of these actions:

- Enter the option number on the command line.
- Place the cursor on the option number point-and-shoot field and press Enter.

The result is a display that is similar to the one presented if the "Associated Storage Areas" link is selected from the detailed section for the event. See Displaying associated storage areas on page 202 for more information and example.

The "Hex-Dumped Storage" link provides a display of relevant unformatted storage areas which might be for one or more events. An example of the hex-dumped storage display is shown in Figure 128: Sample Hex-Dumped Storage display on page 198.

Figure 128. Sample Hex-Dumped Storage display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Hex-Dumped Storage                                          Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
TRANID: CD01      CICS ABEND: DSNC               CICS41   2019/04/28 13:22:19

Address  Offset    Hex                                  EBCDIC / ASCII
         Event 4 Program CDCB0010 BLL=0001 (Address 001410D0)
         See Event 4 Program COBMAIN Storage Areas for address range
         001410D0-001420CF formatted storage

         Event 4 Program CDCB0010 GPR 12 (Address 18206138)
18206138                          00000800 00000000 *        ........*
18206140       +8  18206C90 18207C90 00000000 00000000 *..%...@.........*
18206150      +18  00000000 00000000 00000000 00000000 *................*
         Lines 18206160-18206190 same as above
182061A0      +68  00000000 00000000 00000000 80028B20 *................*
182061B0      +78  00000000 00000000 00000000 00000000 *................*
         Lines 182061C0-18206240 same as above
18206250     +118  00000000 00000000 18203FF0 00000000 *...........0....*
18206260     +128  00000000 00000000 00000000 00000000 *................*
         Lines 18206270-182062C0 same as above
 F1=Help      F3=Exit      F5=RptFind   F6=Actions   F7=Up        F8=Down
F10=Left     F11=Right
```

Primary option: Java Information

See Performing Java analysis on page 259.

Primary option: Language Environment Heap Analysis

Selecting the *"Language Environment® Heap Analysis"* option displays a summary of information that is related to the LE heap. An example of the Summary display is shown in Figure 129: Sample Language Environment Heap Analysis summary display on page 199.

Figure 129. Sample Language Environment Heap Analysis summary display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Language Environment Heap Analysis                          Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
Summary Full - JOBNAME: LEHEAP    SYSTEM ABEND: 0C4          FAE1      2019/0

Segment  Segment       Segment  Root     Root     No.Free  No.Alloc Alloc
Address  Type          Length   Address  Length   Elements Elements Bytes
17E59018 User Heap     00008000 17E59E68 000071B0        1        5 00000D50 E
17E35000 Anywhere Heap 00004000 17E37580 00000E20        4       30 00002DA8
17E7B000 Anywhere Heap 00002000 17E7B4E8 00001A80        3       10 00000440
17E7D000 Anywhere Heap 00004A50 00000000 00000000        0        1 00004A30

Additional Heap Control Blocks

There are no additional heaps

*** Bottom of data.
```

From this display, use the **Segment Address** point-and-shoot field to display an individual heap segment. Use the **Segment Type** point-and-shoot field to display all segments of a given type.

Select the **Full** point-and-shoot field to display all information about the LE heap:

Figure 130. Sample Language Environment Heap Analysis full display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Language Environment Heap Analysis                          Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
Summary Full - JOBNAME: LEHEAP    SYSTEM ABEND: 0C4          FAE1      2019/0

Errors were found in one or more segments

Enclave-Level Storage
  Management (ENSM) Address : 17613268
Heap allocation
  initialization value
  specified . . . . . . . . : No
Heap free initialization
  value specified . . . . . : No

User Heap Analysis

Heap Anchor Node (HANC) . . : 17E59018
Heapid. . . . . . . . . . . : 00000000
Root Address. . . . . . . . : 17E59E68
Segment Length. . . . . . . : 00008000
Root Length . . . . . . . . : 000071B0
```

Use the scroll commands, UP (PF7), DOWN (PF8), LEFT (PF10), and RIGHT (PF11), as necessary to view the entire display.

Primary option: MTRACE Records

Selecting the *"MTRACE Records"* option provides you with a display showing the MVS™ master trace, as the example shown in Figure 131: Sample MTRACE Records display on page 200.

Figure 131. Sample MTRACE Records display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
MTRACE Records                                              Line 1 Col 1 80
Command ===> _____________________________________________ Scroll ===> CSR

SLIP DUMP ID=F092                                FAE1      2019/08/21  18:11:11
M 0100000 FAE1     06233 18:10:40.30 STC21507 00000090  IST530I AM GBIND PENDI
D                                        016 00000090  IST1051I EVENT CODE =
D                                        016 00000090  IST1062I EVENT ID = 00
E                                        016 00000090  IST314I END
M 0100000 FAE1     06233 18:10:40.30 STC21507 00000090  IST530I AM GBIND PENDI
D                                        017 00000090  IST1051I EVENT CODE =
D                                        017 00000090  IST1062I EVENT ID = 00
E                                        017 00000090  IST314I END
N 4000000 FAE1     06233 18:10:40.63 JOB08017 00000090  IEF677I WARNING MESSAG
N 0020000 FAE1     06233 18:10:40.67 JOB08017 00000090  ICH70001I ANDYMEL  LAS
N 4000000 FAE1     06233 18:10:40.68 JOB08017 00000090  $HASP373 DACBB012 STAR
N 0000000 FAE1     06233 18:10:40.71 JOB08017 00000090  IEF403I DACBB012 - STA
N 0004000 FAE1     06233 18:10:40.99 JOB08017 00000290  -
-PAGING COUNTS---
N 0004000 FAE1     06233 18:10:40.99 JOB08017 00000290  -JOBNAME  STEPNAME PRO
AGE   SWAP    VIO SWAPS STEPNO
N 0004000 FAE1     06233 18:10:41.01 JOB08017 00000290  -DACBB012          V00
  0      0      0     0     1
N 0004000 FAE1     06233 18:10:41.25 JOB08017 00000290  -DACBB012          D00
```

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 132: Sample Abend Job Information display on page 201.

Figure 132. Sample Abend Job Information display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Abend Job Information                                        Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
TRANID: FRED      CICS ABEND: AEIL                    2019/05/24  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
```

This display provides information about the environment that existed when the fault was analyzed in real time. The information that is shown depends on the type of fault analyzed.

Primary option: User Notes

Selecting the "User Notes" option from the initial interactive report is equivalent to issuing the NOTELIST command (for details, see NOTELIST on page 109), and results in the User Note List display being shown (see Figure 146: Sample User Note List display on page 216).

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 *"Fault Analyzer Options"* from the initial interactive report displays the Fault Analyzer section of the report as shown in Figure 133: Sample Fault Analyzer Options display on page 202.

Figure 133. Sample Fault Analyzer Options display

```
  File  View  Services  Help
_____________________________________________________________________________
Fault Analyzer Options                                      Line 1 Col 1 80
Command ===> _____________________________________________ Scroll ===> CSR
TRANID: FRED      CICS ABEND: AEIL               CICS04   2019/09/25 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
```

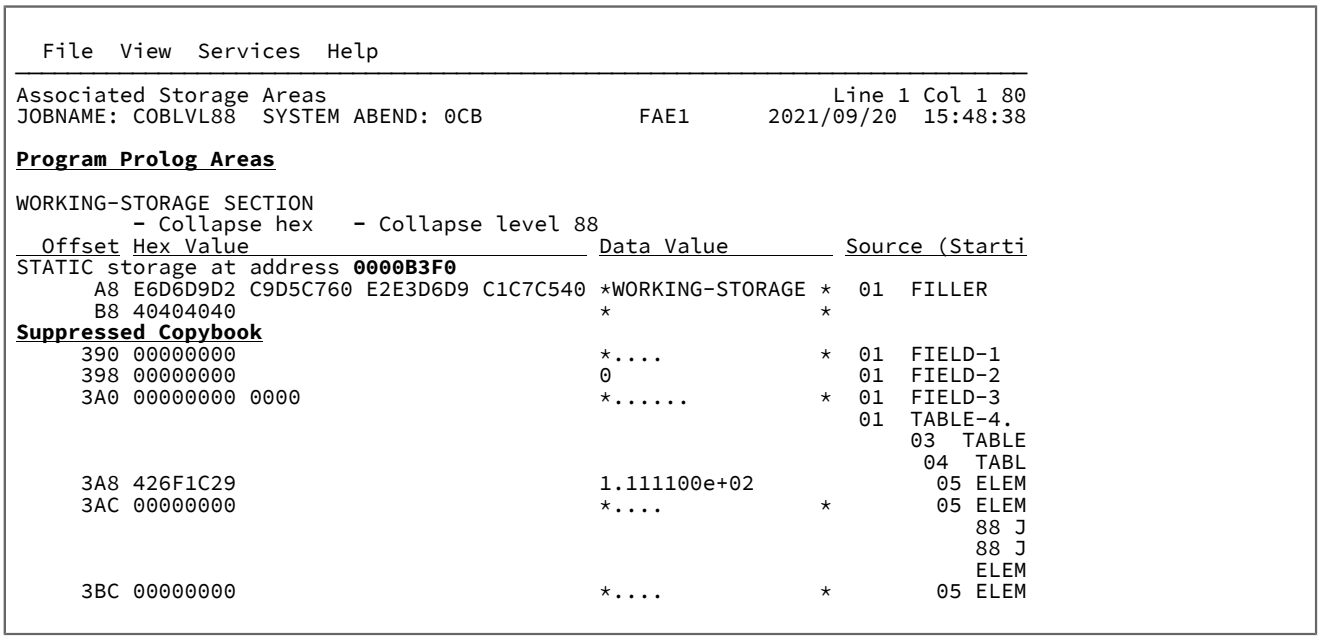
Displaying associated storage areas

In the interactive report, the link *"Associated Storage Areas"* is provided in the detailed event display depending on the programming language used. What is displayed when selecting this link depends on both the programming language and whether or not a compiler listing or side file is available to Fault Analyzer for the program:

- For COBOL programs without a compiler listing or side file, base locators are displayed in hexadecimal dump format.
  - For COBOL V4 and earlier, the TGT is formatted.
  - For COBOL V5 and later, PPAs are formatted.
- For COBOL programs where a compiler listing or side file is supplied, the source declaration of all fields along with their current content is provided.

An example of the associated storage areas display for a COBOL program with source listing provided is shown in Figure 134: Sample Associated Storage Areas display on page 203.

Figure 134. Sample Associated Storage Areas display



```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Associated Storage Areas                                    Line 1 Col 1 80
JOBNAME: COBLVL88  SYSTEM ABEND: 0CB            FAE1      2021/09/20  15:48:38

Program Prolog Areas

WORKING-STORAGE SECTION
        - Collapse hex   - Collapse level 88
  Offset Hex Value                          Data Value          Source (Starti
STATIC storage at address 0000B3F0
      A8 E6D6D9D2 C9D5C760 E2E3D6D9 C1C7C540 *WORKING-STORAGE *  01  FILLER
      B8 40404040                            *                 *
Suppressed Copybook
     390 00000000                            *....             *  01  FIELD-1
     398 00000000                            0                    01  FIELD-2
     3A0 00000000 0000                       *......           *  01  FIELD-3
                                                                  01  TABLE-4.
                                                                      03  TABLE
                                                                       04  TABL
     3A8 426F1C29                            1.111100e+02              05 ELEM
     3AC 00000000                            *....             *       05 ELEM
                                                                         88 J
                                                                         88 J
                                                                         ELEM
     3BC 00000000                            *....             *       05 ELEM
```

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

Some features are unique to the interactive reanalysis report:

- The ability to hide the hex-value column.
- The ability to collapse level 88 items.
- The ability to show all COBOL base locators.

These are explained in the following topics.

Hiding the hex-value column

Hiding the hex-value column might be useful for users of narrow (80-column) screens. By placing the cursor on the minus sign above the "Hex Value" heading in Figure 134: Sample Associated Storage Areas display on page 203 and pressing Enter, the display is changed to that shown in Figure 135: Sample Associated Storage Areas display with hex value column collapsed on page 204:

Figure 135. Sample Associated Storage Areas display with hex value column collapsed

```
  File  View  Services  Help
_____________________________________________________________________________
Associated Storage Areas                                    Line 1 Col 1 80
JOBNAME: COBLVL88  SYSTEM ABEND: 0CB           FAE1      2021/09/20  15:48:38

Program Prolog Areas

WORKING-STORAGE SECTION
        + Expand hex   - Collapse level 88
  Offset Data Value       Source (Starting at Line # 000010 )
STATIC storage at address 0000B3F0
      A8 *WORKING-STORAGE *  01  FILLER                PIC X(20)  VALUE 'WORK
      B8 *                *
Suppressed Copybook
     390 *....            *  01  FIELD-1               PIC 999999 COMP-3.
     398 0                   01  FIELD-2               PIC 999999 COMP-4.
     3A0 *......          *  01  FIELD-3               PIC 999999.
                             01  TABLE-4.
                                 03  TABLE-8 OCCURS 6 TIMES.
                                  04  TABLE-8A OCCURS 3 TIMES.
     3A8 1.111100e+02             05 ELEMENT-1                          COMP
     3AC *....            *       05 ELEMENT-2 OCCURS 4 TIMES          PIC
                                     88 JACK VALUE 'JACK'.
                                     88 JILL VALUE 'JILL'.
                                     ELEMENT-2(1,1,2) to ELEMENT-2(1,1,4) sam
     3BC *....            *       05 ELEMENT-3                PIC 999999 CO
```

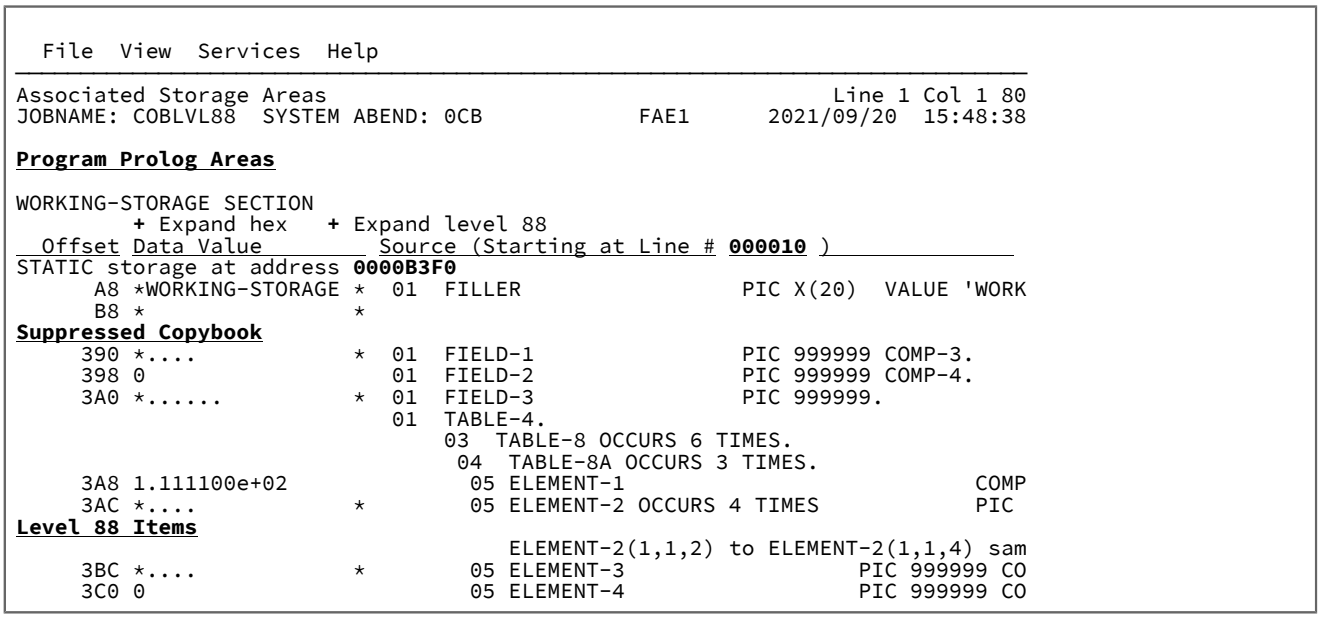
Note that the minus sign point-and-shoot field, now above the "Data Value" column, has changed to a plus sign, and that if placing the cursor on this point-and-shoot field, the display changes back to the first display with "Hex Value" visible.

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 134: Sample Associated Storage Areas display on page 203 and pressing Enter, the display is changed to that shown in Figure 136: Sample Associated Storage Areas display with level 88 items collapsed on page 205:

Figure 136. Sample Associated Storage Areas display with level 88 items collapsed

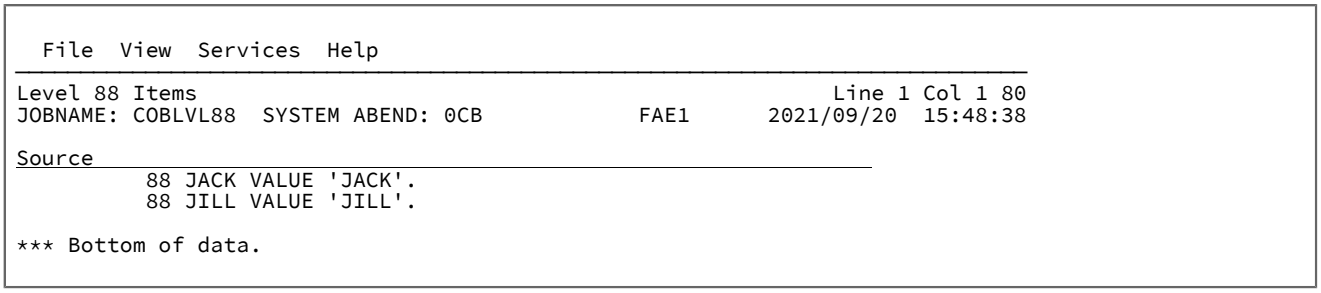


```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Associated Storage Areas                                    Line 1 Col 1 80
JOBNAME: COBLVL88  SYSTEM ABEND: 0CB            FAE1      2021/09/20  15:48:38

Program Prolog Areas

WORKING-STORAGE SECTION
        + Expand hex   + Expand level 88
  Offset Data Value        Source (Starting at Line # 000010 )
STATIC storage at address 0000B3F0
      A8 *WORKING-STORAGE *  01  FILLER                PIC X(20)  VALUE 'WORK
      B8 *                *
Suppressed Copybook
     390 *....            *  01  FIELD-1               PIC 999999 COMP-3.
     398 0                   01  FIELD-2               PIC 999999 COMP-4.
     3A0 *......          *  01  FIELD-3               PIC 999999.
                             01  TABLE-4.
                                 03  TABLE-8 OCCURS 6 TIMES.
                                  04  TABLE-8A OCCURS 3 TIMES.
     3A8 1.111100e+02             05 ELEMENT-1                          COMP
     3AC *....            *       05 ELEMENT-2 OCCURS 4 TIMES          PIC
Level 88 Items
                                    ELEMENT-2(1,1,2) to ELEMENT-2(1,1,4) sam
     3BC *....            *       05 ELEMENT-3                PIC 999999 CO
     3C0 0                        05 ELEMENT-4                PIC 999999 CO
```

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 137: Sample level 88 items display on page 205.

Figure 137. Sample level 88 items display



```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Level 88 Items                                              Line 1 Col 1 80
JOBNAME: COBLVL88  SYSTEM ABEND: 0CB            FAE1      2021/09/20  15:48:38

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

*** Bottom of data.
```

The last selected "Collapse/Expand level 88" option is saved in the user's ISPF profile and is used as the default display mode on subsequent Associated Storage Areas displays.

Showing all COBOL base locators

If source code is not available for an event, and multiple contiguous base locators exist of the same storage type (for example, WORKING-STORAGE), then these are initially displayed summarized. In this case, the starting address is shown and the length is the total length of all the contiguous base locators that follow.

Place the cursor on the plus sign that precedes "Show all BLs," and press Enter. The display changes to show all base locators separately with each their own starting address and length. At the same time, the plus sign at the top of the display changes to a minus sign, and the text changes to "Summarize BLs."

To switch back to the default summarized base locator display, place the cursor on the minus sign that precedes "Summarize BLs," and press Enter.

Expanding messages and abend codes

Messages or abend codes are initially never expanded when using the interactive dump reanalysis feature of Fault Analyzer. This lack of expansion is to prevent the need to scroll through potentially very long explanations to see report items that might follow. Instead, to view the explanation for messages or abend codes in the interactive report, one can place the cursor on the message identifier or abend code and press the Enter key. This opens a display similar to what you see in the batch report as the example shown in Figure 138: Sample Message Explanation display on page 206.

Figure 138. Sample Message Explanation display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Message Explanation                                          Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
JOBNAME: DACBB045  USER ABEND: 4038              MVS2     2019/10/10 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. ❶

  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
```

When a message is displayed in the report, the first occurrence of the message contains the actual text from when it was issued (see ❶ in the above example). If the instance-specific text is not available, then a comment *"job-specific text not available"* replaces it.

In the expansion that immediately follows the issued message or abend code is its explanation that is generally obtained from the softcopy books provided with Fault Analyzer. If an explanation for the message or abend code cannot be found, then the text *"explanation not available"* replaces it.

Displaying source code

To display the source code for an entire program, place the cursor on any yellow point-and-shoot source line number or listing statement number and press Enter. For example, if line number 69 was selected from an event in the interactive report, the display that is shown in Figure 139: Sample Compiler Listing display on page 207 might be displayed.

Figure 139. Sample Compiler Listing display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Program CICFRED Compiler Listing                              Line 63 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
TRANID: FRED      CICS ABEND: AEIL               CICS04   2019/01/09 15:37:46
000060             Move length of MSG1 to dfhb0020
000061             Call 'DFHEI1' using by content x'0404300007000000810000400
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'0602f000070000800f0f0f0f2
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'0404300007000000010000400
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=AddBkp    F7=Up        F8=Down
F10=Left     F11=Right
```

The source line or statement number that is initially selected is highlighted.

The example that is shown here assumes that the display of pseudo-assembler instructions is suppressed. For details on this suppression, see Displaying source code on page 206.

Information that is displayed in blue (all lines that start in column 1) pertains to the program source code. On the left side of the display is information about the source line or listing statement number of the source displayed. This information is followed by the actual source code at this location in the program.

To add machine instruction information to the listing, select the **View** menu Add Pseudo Assembler Instructions option (for information about selecting this option in general, refer to Action-bar pull-down menus on page 94). This option causes the Compiler Listing display to be reformatted with pseudo-assembler instructions inserted into the program source code as shown in Figure 140: Sample Compiler Listing display: Pseudo-assembler instructions enabled on page 208.

Figure 140. Sample Compiler Listing display: Pseudo-assembler instructions enabled

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Program CICFRED Compiler Listing                          Line 316 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
TRANID: FRED      CICS ABEND: AEIL              CICS04   2019/01/09 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=AddBkp    F7=Up        F8=Down
F10=Left     F11=Right
```

Information that is displayed in green (all lines that start in column 15) pertains to the disassembly of machine instructions. This information is shown interspersed with the source code information, following the line of code to which they belong.

Note:

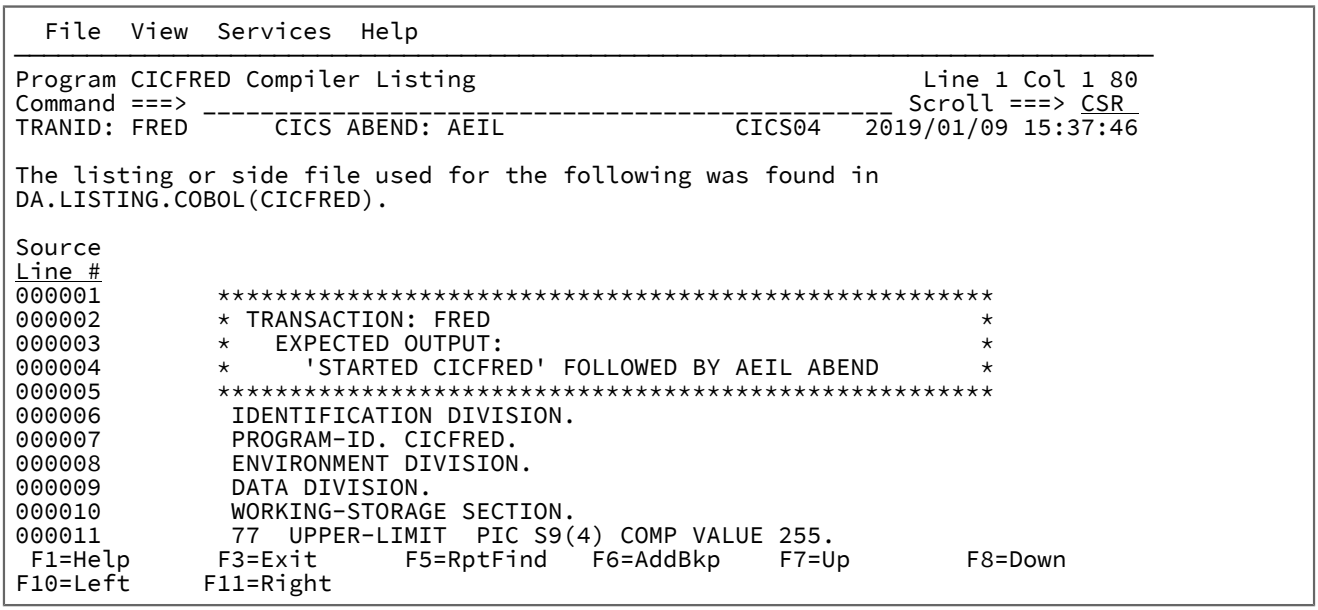
For COBOL programs compiled with TEST(NONE,SYM,SEPARATE), all pseudo-assembler instructions are placed following the last line of source code.

For all other programs, ensure that the source line of interest is scrolled to the top of the display before you add pseudo-assembler instructions to the listing display. The extra display lines might cause the source line to disappear from the current view.

To remove the pseudo-assembler instructions from the display, select the **View** menu Remove Pseudo Assembler Instructions option.

Information at the top of the listing indicates the source of the compiler listing or side. An example is shown in Figure 141: Sample Compiler Listing display: Origin information on page 209. You can scroll to the top of the side file listing by entering, for example, the UP MAX command.

Figure 141. Sample Compiler Listing display: Origin information



Deferred Breakpoints Feature

The compiler listing display supports the setting of z/OS® Debugger Deferred Breakpoints, similar to IPVLANGP. For details, see *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*.

To set Deferred Breakpoints, ADFz Common Components must be installed.

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 111 for details).

An example of the storage display is shown in Figure 142: Sample Dump Storage display on page 210.

Figure 142. Sample Dump Storage display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Dump Storage                                                  17C01380-17C013D7
Command ===> ____________________________________________ Scroll ===> CSR
JOBNAME: IDIVPCOB  SYSTEM ABEND: 0C7              MVS2     2019/08/12 13:46:58

Address  Offset    Hex                                  EBCDIC / ASCII
         Module IDISCBL1 program IDISCBL1 + X'640', source line # 32
         Event 1 Program IDISCBL1 GPR 1 + X'1051'
         Event 1 Program IDISCBL1 GPR 10 + X'1300'
         Event 1 Program IDISCBL1 GPR 11 + X'1008'
         Event 1 Program IDISCBL1 GPR 12 + X'1348'
         Event 1 Program IDISCBL1 GPR 14 + X'680'
         Event 1 Program IDISCBL1 GPR 3 + X'600'
         Event 1 Program IDISCBL1 GPR 4 + X'1544'
17C01380            20004160 61345060 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
```

The character-interpreted section on the right-hand side of the hex data is generally displayed as EBCDIC. To instead display the data as ASCII, place the cursor on the ASCII point-and-shoot field and press Enter. With the cursor now on the EBCDIC point-and-shoot field, pressing Enter a second time reverts to the EBCDIC display.

Placing the cursor anywhere in the hexadecimal storage display area, and pressing the Enter key, takes you to the selected address. If the point-and-shoot field in which the cursor is placed is fewer than 8 digits, then it is padded with leading zeroes to form an 8-digit 31-bit address.

Overtyping the first two digits of an 8-digit address point-and-shoot field with zeroes, immediately prior to pressing the Enter key, ensures that the address is interpreted as a 24-bit address.

A 64-bit address is selected by 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 process logically 'joins' the point-and-shoot field closest to the underscore with the field in which it is placed. For example, given the following two adjacent address point-and-shoot fields

```
00000001 80109020
```

and either overtyping the last digit of the first field like this:

```
0000000_ 80109020
```

or the first digit of the second field like this:

```
00000001 _0109020
```

results in the 64-bit address 00000001_80109020 being displayed. As with 31-bit addresses, the second half of the 64-bit address is padded with leading zeroes to 8-digits.

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

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

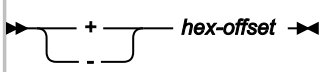
If available, a description of the initially selected address is provided, along with descriptions of the beginning of other storage areas, such as modules and programs, and any user notes (see Figure 142: Sample Dump Storage display on page 210).

Note that event numbers in descriptions that include these, such as `Event 1 Program IDISCBL1 GPR 1 + X'1051'` in Figure 142: Sample Dump Storage display on page 210, refer to event numbers in the Full event summary. If the Application Only setting is in effect for the event summary, event numbers might not match the storage descriptions. For additional information about the different event summary settings, see Primary option: Event Summary on page 168.

The FIND command used from this display behaves differently from 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 104.

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:

Figure 143. Syntax



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

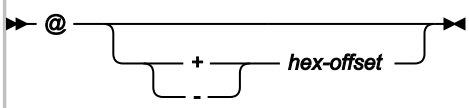
```
+10C
```

or

```
-D4
```

Another way to display addresses while in the Dump Storage display is by using the @ ("at" sign) command, optionally followed by a valid hexadecimal offset:

Figure 144. Syntax



If @ is entered by itself, then the original address at offset 0 is displayed.

If @ is entered, followed by either + or -, and a valid hexadecimal offset, then the specified offset from the original address at offset 0 is displayed.

Displaying data areas

When selecting address point-and-shoot fields from within the interactive reanalysis report, the result is generally the display of all available storage but with the selected address at offset 0. However, in some cases, where the address is associated with a data area of a given length, the resulting display initially shows only the storage for the implied length and with all

storage prior to, or following, shown as "suppressed". An example of this is shown in Figure 145: Sample Dump Storage suppressed display on page 212.

Figure 145. Sample Dump Storage suppressed display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Dump Storage                                                  17C01380-17C013D7
Command ===> ____________________________________________ Scroll ===> CSR
ShowAll - JOBNAME: IDCB0070  SYSTEM ABEND: 0CB            FAE1 2019/01/19

Address  Offset    Hex                                  EBCDIC /  ASCII
         Address range 00000000-0001A01F suppressed
         PCB #1 - Start
0001A020           00400038 00010018 40404040 00000000 *. ......    ....*
0001A030       +10 008A1054 00000000 12A74084 C9D6D7C3 *.........x dIOPC*
0001A040       +20 C2404040 00000000 00000000 00000000 *B   ............*
0001A050       +30 00000000 0001A020                   *........        *
         Event 5 Program IDCB0070 BLL=00001
0001A058       +38                  40404040 40404040 *                *
0001A060       +40 10004040 0000000F 0000000F 00000000 *..  ............*
0001A070       +50 00000000 00000000 C4C6C8E2 C1D4F2F5 *........DFHSAM25*
0001A080       +60 40404040 40404040 00000000          *        ....    *
         PCB #1 - End
         Address range 0001A08C-FFFFFFFF_FFFFFFFF suppressed
```

To see all available storage, place the cursor on the ShowAll point-and-shoot field in the display title, and press Enter.

Alternatively, selecting any hexadecimal address point-and-shoot field results in all available storage being shown.

If the ShowAll method is used, then it is not possible to again suppress the surrounding storage, other than by pressing PF3 and reselecting the original address point-and-shoot field.

However, if an address is instead selected, then the address is *"stacked"* as per usual in the Dump Storage display, and if pressing PF10 until the original address is again displayed, the surrounding storage is again be suppressed. This is the case unless more than 10 subsequent addresses have been selected.

Creating and managing user notes

User notes are comments that the interactive user can add against any storage location. They are saved in the history file fault entry and are available to all users.

User notes can be created by using the NOTE command from any interactive reanalysis report display. For details, see NOTE on page 108.

Alternatively, user notes can be created by overtyping these displays:

- Dump Storage
- Associated Storage Areas
- Event-Related Storage Areas (Event *n* Program *name* Storage Areas)
- Hex-Dumped Storage

In the following, these are simply referred to as "storage areas" displays.

The user notes are created by placing the cursor on the area of storage to which the note applies, typing one or more characters that are distinguishable from hexadecimal digits, and pressing Enter. For example, given the Dump Storage display shown in Figure 142: Sample Dump Storage display on page 210, 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     2019/08/12 13:46:58

Address  Offset    Hex                                  EBCDIC / ASCII
         Module IDISCBL1 program IDISCBL1 + X'640', source line # 32
         Event 1 Program IDISCBL1 GPR 1 + X'1051'
         Event 1 Program IDISCBL1 GPR 10 + X'1300'
         Event 1 Program IDISCBL1 GPR 11 + X'1008'
         Event 1 Program IDISCBL1 GPR 12 + X'1348'
         Event 1 Program IDISCBL1 GPR 14 + X'680'
         Event 1 Program IDISCBL1 GPR 3 + X'600'
         Event 1 Program IDISCBL1 GPR 4 + X'1544'
17C01380            20004160 61345060 51This i s10C310A *...-/.&-..K.....*
17C01390       +10  D2035110 310E5860 20085060 5114D20B *K......-..&-..K.*
17C013A0       +20  51183112 58602004 50605128 58602008 *.....-..&-...-..*
17C013B0       +30  5060512C D2175130 311E98EC D00C07FE *&-..K.....q.}...*
         Module IDISCBL1 CSECT CEESG005
17C013C0       +40  E2F0F0F5 00140001 00000000 00000000 *S005............*
17C013D0       +50  00000000 00000000                   *........        *
         Module IDISCBL1 CSECT CEEBETBL
 F1=Help   F3=Exit   F7=Up     F8=Down  F10=Prev  F11=Next
```

Pressing Enter results in an ISPF edit panel being presented, initialized with the text typed:

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
──────────────────Volume: ──────────────────────────────────────────────────────
EDIT       Note.17C01389                                 Columns 00001 00072
Command ===> ____________________________________________ Scroll ===> PAGE
****** ***************************** Top of Data ******************************
000001 This is
****** **************************** Bottom of Data ****************************









 F1=Help      F2=Split     F3=Exit      F4=:tf       F5=Rfind     F6=Rchange
 F7=Up        F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

From here the note can be completed, adding as many lines as required. The first line should be treated as a heading as it is the only line of the note shown if the display of the note is later collapsed.

Assuming that the final note is as follows:

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------Volume: ------------------------------------------------------
EDIT       Note.17C01389                                 Columns 00001 00072
Command ===> _____________________________________________ Scroll ===> PAGE
****** **************************** Top of Data ******************************
000001 This is an important reminder!
000002 The contents of storage at this offset into this module could be
000003 significant for the understanding of the error that caused this fault.
****** ***************************** Bottom of Data ****************************









 F1=Help      F2=Split     F3=Exit      F4=:tf       F5=Rfind     F6=Rchange
 F7=Up        F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Press PF3 to return to the storage areas display, which shows the newly created user note, inserted immediately ahead of the storage to which it belongs.

```
  File  View  Services  Help
------------------------------------------------------------------------------
Dump Storage                                                 17C01380-17C013D7
Command ===> _____________________________________________ Scroll ===> CSR
JOBNAME: IDIVPCOB  SYSTEM ABEND: 0C7              MVS2     2019/08/12 13:46:58

Address  Offset    Hex                                  EBCDIC / ASCII
         Module IDISCBL1 program IDISCBL1 + X'640', source line # 32
         Event 1 Program IDISCBL1 GPR 1 + X'1051'
         Event 1 Program IDISCBL1 GPR 10 + X'1300'
         Event 1 Program IDISCBL1 GPR 11 + X'1008'
         Event 1 Program IDISCBL1 GPR 12 + X'1348'
         Event 1 Program IDISCBL1 GPR 14 + X'680'
         Event 1 Program IDISCBL1 GPR 3 + X'600'
         Event 1 Program IDISCBL1 GPR 4 + X'1544'
17C01380           20004160 61345060 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 is *"collapsed"* as shown below:

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Dump Storage                                                  17C01380-17C013D7
Command ===> ____________________________________________ Scroll ===> CSR
JOBNAME: IDIVPCOB  SYSTEM ABEND: 0C7             MVS2     2019/08/12 13:46:58

Address  Offset    Hex                                  EBCDIC / ASCII
         Module IDISCBL1 program IDISCBL1 + X'640', source line # 32
         Event 1 Program IDISCBL1 GPR 1 + X'1051'
         Event 1 Program IDISCBL1 GPR 10 + X'1300'
         Event 1 Program IDISCBL1 GPR 11 + X'1008'
         Event 1 Program IDISCBL1 GPR 12 + X'1348'
         Event 1 Program IDISCBL1 GPR 14 + X'680'
         Event 1 Program IDISCBL1 GPR 3 + X'600'
         Event 1 Program IDISCBL1 GPR 4 + X'1544'
17C01380            20004160 61345060 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 IDISCBL1 CSECT CEESG005
17C013C0       +40  E2F0F0F5 00140001 00000000 00000000 *S005............*
 F1=Help   F3=Exit   F7=Up     F8=Down  F10=Prev  F11=Next
```

The preceding point-and-shoot field now indicates "collapsed" by a plus sign instead. By simply placing the cursor on this point-and-shoot field, the user can toggle between the collapsed and expanded view.

It is also possible to over-type the point-and-shoot field with two more action characters (not case-sensitive):

D

Used to delete the user note.

E

Used to edit the user note.

To see all user notes that exist for the current fault entry, enter the NOTELIST command from the command line of any display within the interactive report, or select the "List User Notes" option from the View action-bar pull-down menu. The result is a display like the example shown in Figure 146: Sample User Note List display on page 216:

Figure 146. Sample User Note List display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
User Note List                                                 Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
JOBNAME: IDIVPCOB  SYSTEM ABEND: 0C7              MVS2     2019/08/12 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
```

As indicated in the optional help text on this display, the point-and-shoot field preceding each user note can be overtyped to request a specific action in the same way as in the storage areas displays.

Additionally, the User Note List display permits all user notes to be expanded or collapsed simultaneously by selecting the "expand all" or "collapse all" point-and-shoot fields at the top of the display. The expand/collapse state of any note is common between the User Note List display and the Dump Storage display, so that any changes made in one display is reflected in the other.

To display the storage that is associated with a user note, use the S line command, or place the cursor on the address point-and-shoot field, and press Enter.

User notes are saved in the history file fault entry when the user exits from the interactive report. At this time, if user notes have been added or modified, the user is prompted to acknowledge the update of the fault entry with a display as the example shown in Figure 147: Sample User Notes Update prompt on page 217:

Figure 147. Sample User Notes Update prompt

```
  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     │
  │ PF3/PF12 to exit from the interactive report without updating the fault   │
F │ entry.                                                                    │
M │                                                                           │
  │ History file DSN  . : 'IDI.HIST'                                          │
S │ Fault ID  . . . . . : F00264                                              │
i │                                                                           │
  │  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
```

Pressing Enter from this display permits the history file fault entry to proceed, while pressing PF3 or PF12 exits from the interactive report without any updates to user notes.

Displaying CICS transaction storage (CICSSTG)

As an alternative method to selecting the CICS Transaction Storage Summary or CICS Transaction Storage options from the CICS Information display (see Primary option: CICS Information on page 176), you can use the CICSSTG command (see CICSSTG on page 99 for details).

An example of the CICSSTG command CICS Transaction Storage display is shown in Figure 148: Sample CICS Transaction Storage display on page 217.

Figure 148. Sample CICS Transaction Storage display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
CICS Transaction Storage                                      Line 1 Col 1 80
Command ===>                                                 Scroll ===> CSR
Previous Next Filter

 Element: 1 of 3 (9), address 0_1AE00000, length 0x870 (USER31)
 Filter:  Storage classes(USER31)


Address  Offset     Hex                              EBCDIC / ASCII
1AE00000            E4F0F0F0 F0F1F0F8 00E86EC4 C6C8C5C9 *U0000108.Y>DFHEI*
1AE00010       +10  E4E24040 40404040 00000000 00000000 *US      ........*
1AE00020       +20  00000000 00000000 00000000 00000000 *................*
1AE00030       +30  00000000 00000000 00000000 1AE00100 *.............\..*
1AE00040       +40  00000000 1AE008A0 00000000 00000000 *.....\..........*
1AE00050       +50  00000000 0008B614 00000000 19C7D429 *.............GM.*
1AE00060       +60  00000000 00000420 00000000 1AE0003C *.............\..*
1AE00070       +70  00000000 00041800 00000000 1941DA98 *................q*
1AE00080       +80  00000000 1AC96DF0 00000000 1AC9725C *.....I_0.....I.**
1AE00090       +90  00000000 19419C02 00000000 1AC97438 *.............I..*
1AE000A0       +A0  00000000 7F466948 00000000 1AC93000 *....".........I..*
1AE000B0       +B0  00000000 1AE00008 00000000 1AFFB4A4 *.....\.........u*
1AE000C0       +C0  00000000 00882000 00000000 00000000 *.....h..........*
```

Use the "Previous" and "Next" point-and-shoot fields to navigate through individual storage areas. Use the "Filter" point-and-shoot field to filter which storage areas are included based on CICS storage class, storage area length, and areas that contain specific text. You can display additional help by using PF1 from the CICSSTG display.

Displaying the address space storage map (STGMAP)

To show information about the layout of the current address space, invoke the Storage Map display by either:

- Issuing the STGMAP command from within the interactive reanalysis report.
- Selecting **Services > Storage Map** from the action-bar.

There are two variations of the display: Summary and Details.

Initially the Summary display is shown. The Summary display includes the main storage areas of the address space, as well as information about user region sizes and remaining free space.

A sample Storage Map Summary display is shown in figure 147.

Figure 149. Sample Storage Map Summary display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Storage Map                                                    Line 1 Col 1 80
Command ===> ──────────────────────────────────────────────── Scroll ===> CSR
Summary Details - JOBNAME: S0C7A     SYSTEM ABEND: 0C7            FAE1      202

┌──────────────────────────┐ < 100000000_00000000
│ High Private             │
├──────────────────────────┤ <---- 20000_00000000
│ Shared Area              │
├──────────────────────────┤ <------ 200_00000000
│ High Common              │
├──────────────────────────┤ <------ 1EF_80000000
│ Low Private              │
├──────────────────────────┤ <------- 10_00000000
│ Compressed references    │
│ area                     │
╞══════════════════════════╡ <---------- 80000000 "The Bar"
│ Extended Private         │ <---------- 18B00F7E Current Address
│ (Free 31-Bit User Region │
│ = 1006.70 MB)            │
├──────────────────────────┤ <---------- 18B00000
│ Extended Common          │
╞══════════════════════════╡ <---------- 01000000 "The Line"
│ Common                   │
├──────────────────────────┤ <---------- 00900000
│ Private                  │
│ (Free 24-Bit User Region │
│ = 436.00 KB)             │
├──────────────────────────┤ <---------- 00002000
│ Common                   │
└──────────────────────────┘ <---------- 00000000

USER REGION SIZES

Requested

REGIONX(400K,1G)

Actual

64-Bit. . . . . . . . . . . : X'FFFFFFFF00000000' (16.00 EB)
31-Bit. . . . . . . . . . . : X'40000000' (1.00 GB)
24-Bit. . . . . . . . . . . : X'00074000' (464.00 KB)

*** Bottom of data.
```

You can select the Details display from the point-and-shoot field at the top of the Storage Map display. The Details display provides a further breakdown of the main storage areas and the sizes of each area.

When you show the Storage Map from the Dump Storage display, both the Summary display and the Details display indicate the current address.

Storage area names are shown as point-and-shoot fields in both the Summary display and the Details display. Place the cursor on a storage area name and press Enter to select the storage area and display its dump storage. For details, see Displaying data areas on page 211.

Displaying formatted HTTP data (HTTP)

The HTTP command is used to parse raw data at a specified storage address and format it into a readable, indented HTTP structure. See HTTP on page 105 command for the complete command syntax.

If the HTTP command is entered from a dump storage display without specifying an address, the address in the first position in the current display is used.

When invoked, a pop-up panel similar to the following is displayed:

Figure 150. Sample Storage HTTP Command entry display

```
┌──────────────────────── Storage HTTP Mapping ─── ──────────────────────┐
│                                                                       │
│ To view formatted HTTP data, specify the storage address in the Address │
│ field and press Enter.                                                │
│                                                                       │
│ Address    1B118000                                                   │
│                                                                       │
│  F1=Help     F3=Exit    F12=Cancel                                    │
└───────────────────────────────────────────────────────────────────────┘
```

You can configure the following option on this panel:

Address

The explicit memory address where the HTTP formatting begins. It must identify the first byte of the HTTP Request or Status line. If you specify a different address, the output might be truncate or the system might not detect the HTTP data. The address parameter is 64-bit enabled. For details, see Specifying 64-bit addresses on page 142.

Press **Enter** to process the storage. The following output is displayed:

Figure 151. Example of formatted HTTP output

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────
HTTP Formatter
   Line 1 Col 1 132
Command ===>
   Scroll ===> CSR

Request/Status Line . . . . : POST /Demo HTTP/1.1
Host. . . . . . . . . . . . : 192.0.2.0:18900
Connection. . . . . . . . . : keep-alive
Content-Length. . . . . . . : 14
Cache-Control . . . . . . . : max-age=0
Origin. . . . . . . . . . . : http://192.0.2.0:18900
Content-Type. . . . . . . . : application/x-www-form-urlencoded
Upgrade-Insecure-Requests . : 1
User-Agent. . . . . . . . . : Mozilla/5.0 (Windows NT 10.0; Win64; x64)
                              AppleWebKit/537.36 (KHTML, like Gecko)
                              Chrome/144.0.0.0 Safari/537.36
Accept. . . . . . . . . . . : text/html,application/xhtml+xml,
                              application/xml;q=0.9,image/avif,image/webp,
                              image/apng,*/*;q=0.8,
                              application/signed-exchange;v=b3;q=0.7
Referer . . . . . . . . . . : http://192.0.2.0:18900/Demo
Accept-Encoding . . . . . . : gzip, deflate
Accept-Language . . . . . . : en-US,en;q=0.9
Body. . . . . . . . . . . . : int1=50&int2=0

*** Bottom of data.


F1=Help     F3=Exit     F5=RptFind  F7=Up       F8=Down     F10=Left    F11=Right   F12=Cancel
 *IDIPVIE
```

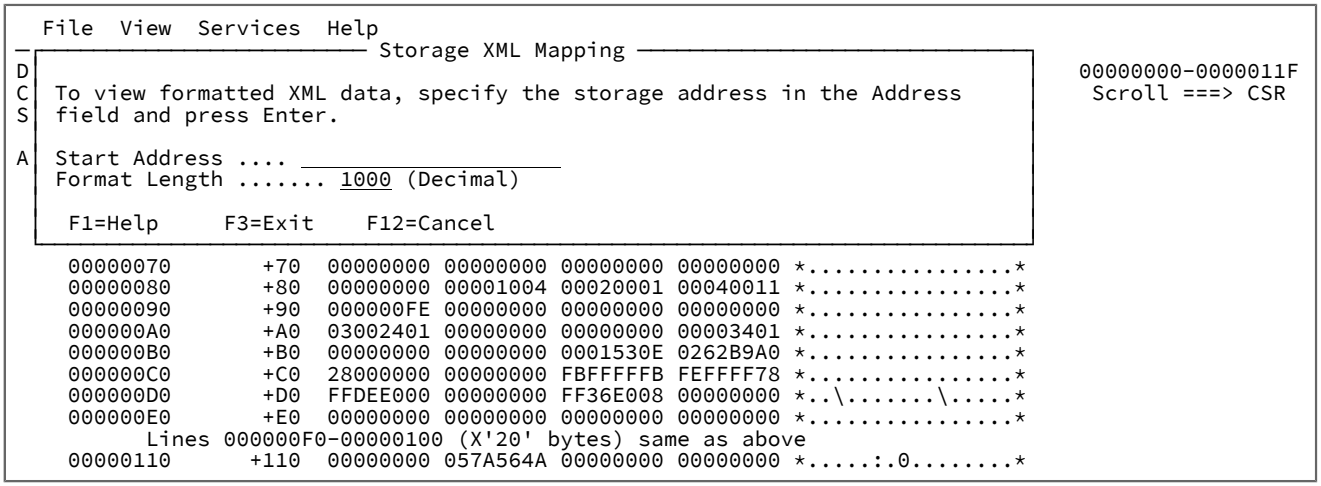
To exit from the HTTP display and return to the previous screen, enter EXIT (PF3).

Displaying formatted XML (XML)

The XML command is used to parse raw data at a specified storage address and format it into a readable, indented XML tree structure. See XML on page 112 command for the complete command syntax.

When invoked, a pop-up panel similar to the following is displayed:

Figure 152. Sample Storage XML Command entry display



Note: If the XML command is issued from the Dump Storage display, the Address field is automatically initialized to the current address.

You can configure the following options on this panel:

Start Address

The explicit memory address where the XML formatting begins. The address parameter is 64-bit enabled. For details, see Specifying 64-bit addresses on page 142.

Format Length

The maximum number of bytes to read from the specified address. The default is 1000.

Press **Enter** to process the storage. The following output is displayed:

Figure 153. Example of formatted XML output

```
  File  View  Services  Help
_____________________________________________________________________________________________
XML Formatter                                                                  Line 1 Col 1 132
Command ===> _____________________________________________________________________Scroll ===> CSR
SYSTEM=CICSOIA1 CODE=ZC2479   ID=65/0005        RNW1      2024/12/09  05:03:42

   <?xml
    version="1.0" encoding="UTF-8"?>.
     <soapenv:Envelope
      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www
.w3.org/2001/XMLSchema-instance">.
       <soapenv:Body>
         <TXLife>
           <UserAuthRequest>
             <UserLoginName>
              LPSM244
             </UserLoginName>
             <BatchID>
              303
             </BatchID>
           </UserAuthRequest>
           <TXLifeRequest>
             <TransRefGUID>
              X
             </TransRefGUID>
F1=Help     F3=Exit     F5=RptFind  F6=Actions   F7=Up       F8=Down    F10=Left    F11=Right
```

To exit from the XML display and return to the previous screen, enter EXIT (PF3).

Displaying PSW information

The following figure shows an example of a PSW in an interactive reanalysis report:

Figure 154. Sample Program Status Word display

```
  File  View  Services  Help
_____________________________________________________________________________
Event 1 of 3: Abend S0C7 *** Point of Failure ***              Line 69 Col 1 80
Command ===> _____________________________________________ Scroll ===> CSR
JOBNAME: IDIVPCOB  SYSTEM ABEND: 0C7              FAE1     2019/05/31  12:04:53
  Second Operand Storage. . : C1C2C3CF     *ABC.*

Associated Messages

CEE3207S The system detected a data exception (System Completion Code=0C7).
                                ❶     ❷
Program Status Word (PSW) . : 078D2000 98601172
PSW Summary . . . . . . . . : Primary Space Mode, PSW Key 8, AMODE 31, Problem
                              State

General Purpose Registers (AMODE: 64 31 24 , Bytes: Dec Hex ):
   R0:     10000_186970F0 (405,264 bytes of storage addressable)
   R1:         0_18600FA1 (Module IDISCBL1 program IDISCBL1 + X'209')
   R2:         0_000077FC (83,972 bytes of storage addressable)
   R3:         0_18601136 (Module IDISCBL1 program IDISCBL1 + X'39E', source
                          line # 26 )
   R4:         0_18600DD0 (Module IDISCBL1 program IDISCBL1 + X'38')
   R5:         0_1860EE90 (962,928 bytes of storage addressable)
   R6:         0_00000000 (2,048 bytes of storage addressable)
```

The PSW contains two point-and-shoot fields. One field is for the high word ❶ and another is for the low word ❷.

If you select the point-and-shoot field for the low word, the storage at that address is shown. For more information, see Displaying storage locations on page 209.

If you select the point-and-shoot field for the high word, the Program Status Word Breakdown Table display is shown. See the following figure as an example.

Figure 155. Sample Program Status Word Breakdown Table display

```
Program Status Word Breakdown Table                          Line 1 Col 1 76
Command ===> _____________________________________________ Scroll ===> CSR

PSW: 078D1000 A68011AA (ESA/390)

PSW Format:
 ┌─┬─┬───┬─┬─┬─┬──────┬─┬─┬─┬─┬──┬──┬────────┬────────┐
 │0│R│000│T│I│E│PSW KEY│1│M│W│P│AS│CC│PGM MASK│00000000│
 └─┴─┴───┴─┴─┴─┴──────┴─┴─┴─┴─┴──┴──┴────────┴────────┘
  0       5     8       12      16 18 20       24      31
 ┌─┬──────────────────────────────────────────────────┐
 │A│Instruction Address                               │
 └─┴──────────────────────────────────────────────────┘
  32 33                                              63

Actual Values
 ┌─┬─┬───┬─┬─┬─┬────┬─┬─┬─┬─┬──┬──┬────┬────────┐
 │0│0│000│1│1│1│1000│1│1│0│1│00│10│0000│00000000│
 └─┴─┴───┴─┴─┴─┴────┴─┴─┴─┴─┴──┴──┴────┴────────┘
  0       5     8       12      16 18 20       24      31
 ┌─┬──────────────────────────────────────────────────┐
 │1│18601172                                          │
 └─┴──────────────────────────────────────────────────┘
  32 33                                              63

  Bit    Meaning
 ─────  ──────────────────────────────────
  1      Program-Event-Recording Mask         (R)
  5      DAT Mode                             (T = 1)
  6      Input/Output Mask                    (I)
  7      External Mask                        (E)
  12     One indicates ESA/390
  13     Machine-Check Mask                   (M)
  14     Wait State                           (W = 1)
  15     Problem State                        (P = 1)
  16-17  Address Space Control                (AS)
         xx Real Mode                         (T = 0)
         00 Primary-Space Mode                (T = 1)
         01 Access-Register Mode              (T = 1)
         10 Secondary-Space Mode              (T = 1)
         11 Home-Space Mode                   (T = 1)
  18-19  Condition Code                       (CC)
  20     Fixed-Point-Overflow Mask
  21     Decimal-Overflow Mask
  22     HFP-Exponent-Overflow Mask
  23     HFP-Signficance Mask
  32     31-Bit Addressing Mode               (A = 1)
  33-63  Instruction Address                  (Hexadecimal)

**** Bottom of data.
```

This display shows a breakdown of the various bit fields found in the selected PSW.

The display contains two pairs of tables. The first one shows the general format of a PSW, and the second one shows the contents of the selected PSW. The remaining part of the display contains the bit fields that are referenced in the formatting table and associated bit offsets.

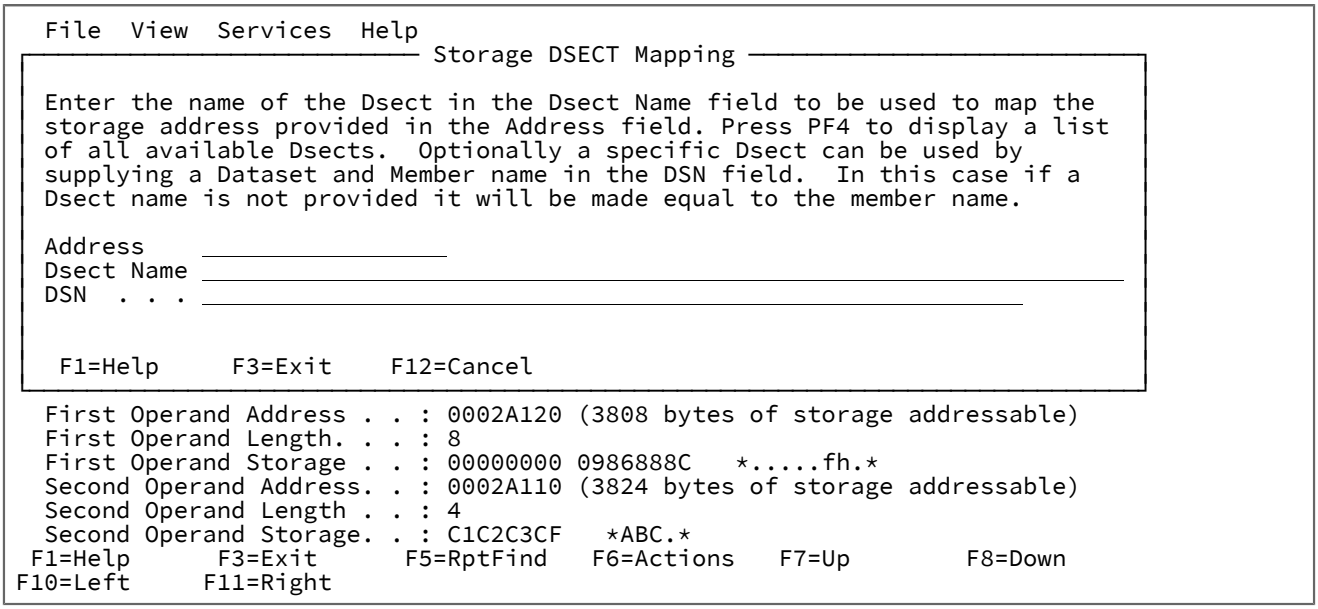
Mapping storage areas using DSECT information

By using the DSECT command from within the interactive report, storage areas can be mapped based on PDS or PDSE data set members containing assembler macro or DSECT copybooks.

The DSECT command (see DSECT on page 101 for syntax), can be entered from the command line of any display, or via PF key assignment. By default, the DSECT command is assigned to PF4.

When invoked, you are shown a popup window similar to the following:

Figure 156. Sample Storage DSECT Mapping Entry display



Note: If the DSECT command is issued while the cursor is on an address point-and-shoot field. then the popup display address is automatically initialized to that address.

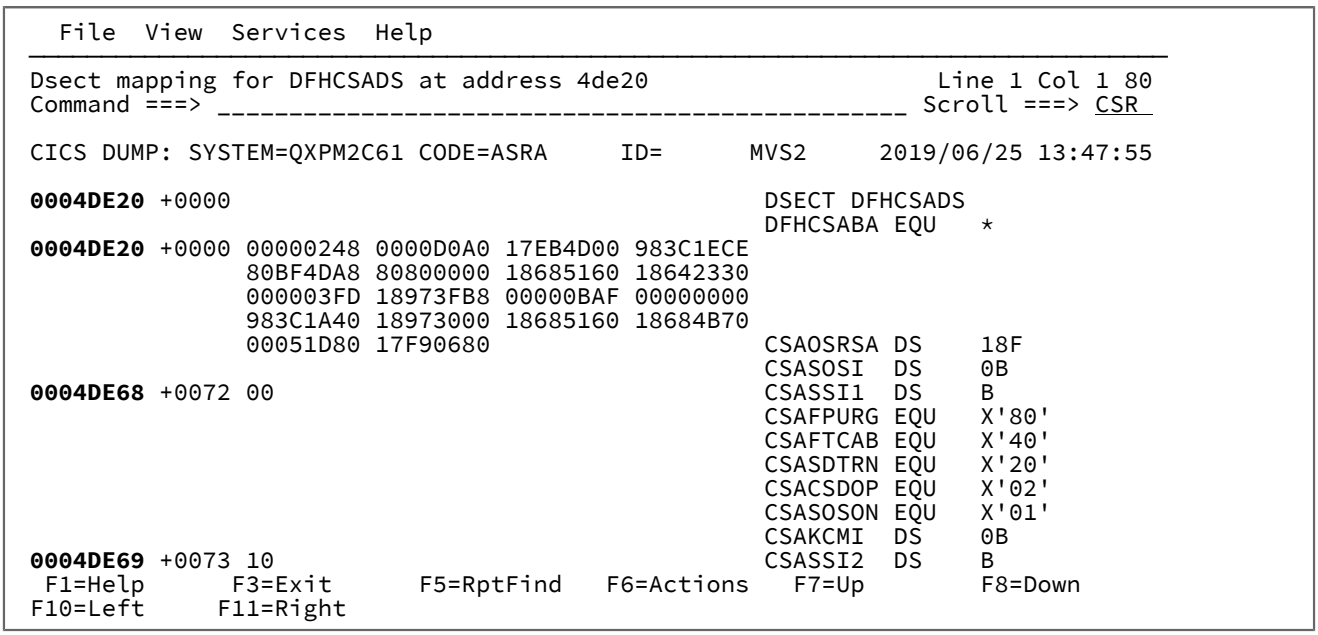
You can now supply the start address (if not already filled in), and the name of the DSECT to use when mapping the storage area. The Address field is 64-bit enabled. For details, see Specifying 64-bit addresses on page 142.

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 226 for details). 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 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 or PDSE member name.

Once a valid storage address and DSECT name have been specified, the storage is mapped and displayed, similar to the following example:

Figure 157. Sample Storage DSECT Mapping Map display



Scroll up/down or right/left as needed to display more DSECT information.

Press PF3 to return from the Storage DSECT Mapping Map display.

IDISYSLB concatenation

The DSECT command and IDIPDSCU utility both use, internally, the assembler to generate ADATA which is then used to determine field names and offsets. The SYSLIB DD, used by the assembler, is dynamically allocated to the same data set as the source DSECT. However this may be insufficient where, for example, an assembler macro is used to generate the DSECT and that macro does not exist in the DSECT data set. To allow for this situation additional SYSLIB data sets can be specified by allocating them to an IDISYSLB DD.

For example, data set MY.DSECTS has 1 member in it, CICSSIT, for generating the CICS DFHSIT DSECT:

```
000001 * GENERATE THE CICS DFHSIT DSECT
000002          DFHSIT TYPE=DSECT
000003          END
```

Using this data set or member in the DSECT command or IDIPDSCU utility will not generate a usable DSECT as the DFHSIT macro will not be resolved. Therefore, the IDISYSLB DD can be used. For example:

```
//UTIL1    EXEC PGM=IDIPDSCU,PARM=('MY.DSECTS')
//SYSPRINT DD   SYSOUT=*
//SYSPUNCH DD   SYSOUT=*
//IDISYSLB   DD DISP=SHR,DSN=CICS.TS61.CICS.SDFHMAC
```

When using the DSECT command during interactive reanalysis, or when using IDIPDSCU in ISPF, use the TSO ALLOCATE command:

ALLOCATE FILE(IDISYSLB) DS('CICS.TS61.CICS.SDFHMAC') SHR

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 or PDSE data sets, which contain DSECT files to be used when processing the DSECT command.

The IDIDSECT data set attributes must be one of the following:

- RECFM=FB and LRECL≥80
- RECFM=VB and LRECL≥84

Regardless of LRECL, only the first 80 bytes of each record are read. The recommended IDIDSECT data set attributes are RECFM=FB and LRECL=80.

When in an interactive report, the first time the DSECT command is issued, it processes each data set in the IDIDSECT concatenation. If the data set contains a $DINDEX member (see Indexing your DSECT data sets ($DINDEX member) on page 226), 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. When all the DSECT details have been determined for a data set, the process is repeated for the next data set, until all the data sets in the IDIDSECT concatenation have been processed.

Note that this process only happens once per interactive report session. If new DSECTs are added to a data set in the IDIDSECT concatenation, or if the $DINDEX member is updated, then you must do one of these actions:

- Restart the interactive report.
- Explicitly identify the new DSECT by specifying the data set and member name it is contained in.

Indexing your DSECT data sets ($DINDEX member)

To allow for DSECT names of up to the maximum of 63 characters, and individual members that contain multiple DSECTS, the IDIPDSCU utility can be used to create a $DINDEX member.

The $DINDEX member should contain a line for every DSECT in each member of the PDS or PDSE. Each line should consist of 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, 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. 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 need to call the macro in question, providing the required keywords, and need to be stored in the same data set as the macro they invoke. For example, CICS® provides in its SDFHMAC data set a member called DFHTCTZE, which provides multiple terminal-related DSECTs. If this member is processed directly by the IDIPDSCU utility, then it does not detect the TCTENIB DSECT, as detection requires special macro keywords to be specified. In this case, if a member is created in the SDFHMAC data set (or a copy of it), which contains the following source line, then all DSECTs are detected, including TCTENIB:

```
DFHTCTZE CICSYST=YES
```

The IDIPDSCU utility can be used either by entering IDIPDSCU next to a data set name in ISPF, or as a batch utility, in which case the data set to process is passed as a parameter. See the following example:

```
//UTILJOB1 JOB ...
//RUNUTIL EXEC PGM=IDIPDSCU,PARM=('fully_qualified_PDS(E)_data_set_name')
//SYSPRINT DD SYSOUT=*
```

The IDIPDSCU utility creates a $DINDEX member in the target data set, so you must have write access to this data set.

See IDISYSLB concatenation on page 225 for information about specifying SYSLIB data sets for macro resolution.

Displaying chained data areas

Using the RUNCHAIN command (see RUNCHAIN on page 110, storage can be scanned for a chain of linked control blocks. The RUNCHAIN command can be invoked either by entering RUNCHAIN on any interactive report command line, or by assigning RUNCHAIN to a PF key. When invoked, you are shown a popup panel similar to the following:

Figure 158. Sample Storage RUNCHAIN Command entry display

```
  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: All    │
│                                                 0's and all F's)             │
│ 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
```

The Start Address is 64-bit enabled. For details, see Specifying 64-bit addresses on page 142.

If a 31-bit Start Address is specified:

- The forward pointer at the Forward Pointer Offset is assumed to be a 31-bit address.
- The End of Chain Identifier is assumed to be a 32-bit hexadecimal value. If less than 8 hexadecimal digits are specified, then the specified value is padded with leading zeroes to form a 32-bit hexadecimal value.

If a 64-bit Start Address is specified:

- The forward pointer at the Forward Pointer Offset is assumed to be a 64-bit address.
- The End of Chain Identifier is assumed to be a 64-bit hexadecimal value. If less than 16 hexadecimal digits are specified, then the specified value is padded with leading zeroes to form a 64-bit hexadecimal value.

For a given Start Address and Forward Pointer Offset, the RUNCHAIN command follows the chain of control blocks until one of the following end conditions is met:

1. The number of control blocks scanned has exceeded the maximum number set by the user (the default value is 9999).
2. The forward pointer of the current control block contains one of the 'End of Chain' values. These values are:
    - 00000000 for a 31-bit Start Address or 0000000000000000 for a 64-bit Start Address.
    - FFFFFFFF for a 31-bit Start Address or FFFFFFFFFFFFFFFF for a 64-bit Start Address.
    - The initial start address, implying the chain has looped
    - A user supplied End Of Chain Identifier:
        - The value specified is automatically padded with leading zeroes to a length of 8 digits for a 31-bit Start Address or 16 digits for a 64-bit Start Address. A 64-bit address might include an underscore (see Start Address above for details).
        - If more than 8 digits are specified for a 31-bit Start Address, then the value is left truncated to 8 digits.
        - If more than 16 digits are specified for a 64-bit Start Address, and the address does not include an underscore, then the value is left truncated to 16 digits.
3. The forward pointer of the current control block points to invalid or unavailable storage.

For each control block, its address and the first 32 bytes of data are shown.

Optionally, you can provide an eyecatcher and its offset in the control block, in which case, for each control block, the text at the specified offset is compared against the supplied text, and if they do not match, then a warning message is issued.

As an example of the RUNCHAIN command, the Storage RUNCHAIN Command entry display might be specified as follows:

```
  File  View  Services  Help
┌──────────────────────── Storage RUNCHAIN Command ────────────────────────┐
│                                                                          │
│ 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: All│
│                                                 0's and all F's)         │
│ Eyecatcher Text  . . . . . . DFHSMQPH                                    │
│ Eyecatcher Offset  . . . . . 2     (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
```

Pressing Enter, the following display is presented:

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Runchain starting at address 0005C000. 1 Control blocks        Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
CICS DUMP: SYSTEM=QXPM2C61 CODE=ASRA     ID=      MVS2     2019/06/25 13:47:55

Address   Hex (First 32 Bytes Of Data)
0005C000  00306EC4 C6C8E2D4 D8D7C840 40404040  C1D76DE3 C3C1F2F4 002AA000 0005
0005B000  00306EC4 C6C8E2D4 D8D7C840 40404040  C1D76DE3 C3C1F2F4 0005C000 0005
0005A000  00306EC4 C6C8E2D4 D8D7C840 40404040  C1D76DE3 C3C1F2F4 0005B000 0005
00059000  00306EC4 C6C8E2D4 D8D7C840 40404040  C1D76DE3 C3C1F2F4 0005A000 2E80
2E80CCCC  Invalid eyecatcher.  Expected: >DFHSMQPH
                               Found   : 24.....Q.
2E80CCCC  C3C1F2F4 2E80CC14 1ED82728 C0010100  00000000 00000600 00059000 002A
002AA000  00306EC4 C6C8E2D4 D8D7C840 40404040  C1D76DE3 C3C1F2F4 2E80CCCC 0005
End of RUNCHAIN 6 Control Blocks processed

*** Bottom of data.




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

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

Disassembling object code

Using the DISASM command (see DISASM on page 100), you can disassemble object code at a given address. The DISASM command can be invoked either by entering DISASM on any interactive report command line, or by assigning DISASM to a PF key. When invoked, you are shown a popup panel similar to the following:

Figure 159. Sample Storage Disassemble display

```
  File  View  Services  Help
┌──────────────────────────── Storage Disassemble ─────────────────────────────┐
│                                                                              │
│ WARNING Before using this function you must be aware of and respect the      │
│ intellectual property rights of others. You are not authorized to use this   │
│ function to disassemble, copy or create assembly listings or disassembled    │
│ Assembler Language source code in violation of any contractual or other      │
│ legal obligation. You are authorized to use this function only for object    │
│ code for which you have verified you have the right to perform               │
│ disassembly.                                                                 │
│                                                                              │
│ Start Address  . . . . . . . . . . _________                                 │
│ Origin Address (optional)  . . . . _________                                 │
│                                                                              │
│                                                                              │
│  F1=Help     F3=Exit    F12=Cancel                                           │
└──────────────────────────────────────────────────────────────────────────────┘

{Fault Analyzer maximum storage allocated: 1.71 megabytes.}

*** Bottom of data.

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

The DISASM command attempts to disassemble code from a given start address. Optionally, an origin address can be provided in which case the offset of each disassembled instruction is calculated relative to the origin address, rather than the start address. If an origin address is not provided, then it defaults to the same as the start address.

As an example of the DISASM command, the Storage Disassemble display might be specified as follows:

```
  File  View  Services  Help
┌──────────────────────────── Storage Disassemble ─────────────────────────────┐
│                                                                              │
│ WARNING Before using this function you must be aware of and respect the      │
│ intellectual property rights of others. You are not authorized to use this   │
│ function to disassemble, copy or create assembly listings or disassembled    │
│ Assembler Language source code in violation of any contractual or other      │
│ legal obligation. You are authorized to use this function only for object    │
│ code for which you have verified you have the right to perform               │
│ disassembly.                                                                 │
│                                                                              │
│ Start Address  . . . . . . . . . . 18D00326                                 │
│ Origin Address (optional)  . . . . 18D00300                                 │
│                                                                              │
│                                                                              │
│  F1=Help     F3=Exit    F12=Cancel                                           │
└──────────────────────────────────────────────────────────────────────────────┘

{Fault Analyzer maximum storage allocated: 1.71 megabytes.}

*** Bottom of data.

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

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              SYS01     2019/12/01 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.

FIND/RPTFIND only finds data on the currently displayed screen.

Scrolling UP MAX/DOWN MAX is not supported for this display.

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

Converting STORE CLOCK values

Using the STCK command (see STCK on page 112), you can convert binary STORE CLOCK values to human-readable date and time format. The STCK command can be invoked either by entering STCK on any interactive report command line, or by assigning STCK to a PF key. When invoked, you are shown a popup panel similar to the following:

Figure 160. Sample STCK Conversion display

```
  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
```

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 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:

```
  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   D6C0AE3BC09E3146                                          │
│ Date Time  : 2019/09/20 11:45:23.462627                                │
│                                                                        │
│                                                                        │
│  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
```

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

User-specific report formatting

Using the EXEC command (see EXEC on page 101), 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 490.

Non-REXX Formatting user exits cannot be executed through the EXEC command.

If an exit name is specified with the EXEC command, then that exit is executed and a display containing any information that the exit has provided is presented. However, if no exit name is specified, then a list of all REXX Formatting user exits which are available from the IDIEXEC concatenation of data sets is presented in a Formatting User Exit Selection List display like the following example:

Figure 161. Sample Formatting User Exit Selection List display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Formatting User Exit Selection List                          Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
JOBNAME: IDIVPPLI  SYSTEM ABEND: 0C9          FAE1      2019/05/07  11:29:07

{The following line commands are available: S (Select), B (Browse), E (Edit).}

   Name     Comment/Arguments
__ IDISUFM1 Sample Formatting user exit to display TCB information
__ IDISUFM2 Sample Formatting user exit for CICS CWA
__ IDISUFM3 Sample Formatting user exit to illustrate the use of formatting tags









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

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 162. Sample $$UFMTX member

```
 BROWSE    FRED.EXEC($$UFMTX) - 01.01               Line 00000000 Col 001 080
 Command ===> ____________________________________ Scroll ===> CSR
******************************** Top of Data ********************************
IDISUFM1 Sample Formatting user exit to display TCB information
IDISUFM2 Sample Formatting user exit for CICS CWA
IDISUFM3 Sample Formatting user exit to illustrate the use of formatting tags
******************************* Bottom of Data ******************************









  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
  F7=UP        F8=DOWN      F9=SWAP nex F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

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 action should present a display similar to that in Figure 161: Sample Formatting User Exit Selection List display on page 234 from where the sample exits can be run.

From the Formatting User Exit Selection List display, a line command can be issued against individual exits:

S

Executes the exit.

B

Enters ISPF browse against the exit.

E

Enters ISPF edit against the exit.

Often, exits require one or more parameters to be passed. Passing these parameters can be done by clearing (if necessary) and overtyping the "Comments/Arguments" field of the Formatting User Exit Selection List display to the right of the exit name. The field changes color when it has been overtyped to indicate that the data is used as parameters for the exit. By clearing the field, and pressing Enter, the original comment can be redisplayed instead.

Prompting for compiler listing or side file

If no satisfactory compiler listing or side file was found for a COBOL, PL/I, C/C++, or assembler program, then a prompting display, as the example shown in Figure 163: Sample Compiler Listing Not Found display on page 236, is presented. Prompting does not occur for C/C++ if the side file was originally located in zFS.

The term *side file* refers to one of the following:

- An IBM® Application Delivery Foundation for z/OS (ADFz) LANGX side file
- A COBOL SYSDEBUG side file that was generated by using the TEST(NONE,SYM,SEPARATE) compiler option
- An Enterprise PL/I side file that was generated by using the TEST(STMT,SYM,NOHOOK,SEPARATE) compiler option

Figure 163. Sample Compiler Listing Not Found display

```
Compiler Listing Not Found                                Line 1 Col 1 76
Command ===> _____________________________________________ Scroll ===> CSR

Module SYS13029.T142716.RA000.IDIVPCOB.GOSET.H01(IDISCBL1) containing
COBOL program IDISCBL1 entry point IDISCBL1 compiled date 2019/01/29 time
14:27:30 does not have a matching listing/side-file.
This is the point of failure program.  ❶
Select one of the following options and press Enter:
  1. (F3) Continue without compiler listing or side file for IDISCBL1
  2. Specify compiler listing or side file to use for IDISCBL1
  3. Retry search for compiler listing or side file for IDISCBL1
  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. Add IDISCBL1 to your side file search exclude list

The trace of the listing/side-file search follows.

  Rejected - NWILKES.IVPCB.LISTINGS(IDISCBL1)
    Failed -
        OPEN error, member not found.

  Rejected - DA.LISTING.COBOL(IDISCBL1)
```

The prompt provides you with these choices:

1. **(F3) Continue without compiler listing or side file for *program-name***

   If a compiler listing or side file cannot be supplied, select this option to continue without program source code information. Alternatively, enter the EXIT (PF3) or CANCEL (PF12) command.

2. **Specify compiler listing or side file to use for *program-name***

   This option displays a pop-up panel in which you can provide the data set and member name (if a PDS or PDSE data set) of a compiler listing or side file (or, in the case of Enterprise COBOL version 5, a program object containing DWARF debugging information) that should be used for the current program as the example shown in Figure 164: Sample Specify Compiler Listing or Side File display on page 237.

Figure 164. Sample Specify Compiler Listing or Side File display

```
Specify Compiler Listing or Side File                      Line 1 Col 1 76
Command ===> _____________________________________________ Scroll ===> CSR

Specify the data set and member name containing the compiler listing or
side file and press Enter.
Data Set Name . . . : 'TDEV003.LISTING.PLI'
Member. . . . . : IDISCBL1

Alternatively, place cursor on choice and press Enter to use previously
specified compiler listing or side file data set name.
==> 'TDEV003.LISTING.PLI'
==> 'TDEV003.@SDSF'
==> 'PMR.P03527.B370.C000.IDILANG'
==> 'TDEV003.JCLLIB'
==> 'TDEV003.$$TEMP$$.LISTING'
==>
==>
==>
==>
==>

*** Bottom of data.
```

The data set name is specified in accordance with the ISPF convention of prefixing with the current TSO prefix, unless enclosed in single quotes.

The last 10 data set names that were specified are stored in the ISPF profile for your application ID and are used for initialization of the display. The most recent data set name is used to initialize the data set name field, but any one of the listed data set names can be selected by placing the cursor on the desired data set name and pressing Enter.

The member name defaults to the program name for which the listing or side file is required. If the actual member name for your listing or side file differs from the program name, you need to change this field.

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

Having specified or selected the desired data set and member name, press the Enter key.

If Fault Analyzer has determined that the specified compiler listing or side file is not a good match, then another prompt as the example shown in Figure 165: Sample Listing/Side File Mismatch display on page 238 is presented.

Figure 165. Sample Listing/Side File Mismatch display

```
                                                              Line 1 Col 1 76
Command ===> _____________________________________________ Scroll ===> CSR

Listing/Side File . . . . . : LJBERRY.SYSDEBUG.COBOL(COBSEP)

Compile Date/Time:
  Load Module . . . . . . . : 2019/01/30 12:05:11
  Listing/Side File . . . . : 2019/01/30 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
```

If pressing Enter, and thus accepting the provided mismatching compiler listing or side file, then it is possible that some incorrect information is presented, for example, data fields with incorrect values, or incorrect source lines or statements.

In case a side file fails validation to such a degree that it is not even possible to attempt using it, then an Enter response to the prompt in Figure 165: Sample Listing/Side File Mismatch display on page 238 instead shows the display in Figure 163: Sample Compiler Listing Not Found display on page 236 again, which allows for a different side file to be provided.

3. Retry search for compiler listing or side file for *program-name*

Selecting this option causes Fault Analyzer to repeat the search for the compiler listing or side file via the standard search path. This option can be selected after, for example, having recompiled the current program via a split screen ISPF session and provided the compiler listing or side file to Fault Analyzer in, for example, the IDILCOB data set concatenation.

This repeated search is only performed once. The user is not prompted a second time for the same program, even if the listing or side file is still not found.

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

If you select this option, then Fault Analyzer does not prompt you again for a missing compiler listing or side file for any program for the duration of the current interactive reanalysis session.

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

If you select this option, then Fault Analyzer only prompts you again for a missing compiler listing or side file for a program, if that program has been determined as belonging to the point-of-failure event. If the initial prompt is already for the point-of-failure program, then a message is added to the display to indicate this (as shown at ❶ in Figure 163: Sample Compiler Listing Not Found display on page 236).

6. Add *program-name* to your side file search exclude list

If you select this option, then a display that enables you to add the current program name to your side file search exclude list, as the example shown in Figure 166: Sample Exclude Program from Side File Search display on page 239, is presented.

Figure 166. Sample Exclude Program from Side File Search display

```
Exclude Program from Side File Search                    Line 1 Col 1 76
Command ===> _____________________________________________ Scroll ===> CSR

Press PF3 to continue without updates.

Press Enter to add IDISCBL1 to your side file search exclude list below.
Optionally, edit the name using wildcards '*' and/or '% before pressing
Enter.
Program Name. . . . : IDISCBL1

Current list of excluded programs ( Edit ):
(Empty)

*** Bottom of data.
```

Press PF3 to continue without updates. You might be prompted again to provide a compiler listing or side file for this program.

Press Enter to add the program name to your side file search exclude list. Optionally, edit the name first using the wildcard characters '*' (zero, one or more characters) or '%' (a single required character) to make the name more generic.

Your current side file search exclude list is provided. The list can be modified by placing the cursor on the Edit point-and-shoot field and pressing Enter. The program names must be valid PDS or PDSE member names, but can include the wildcard characters described above. The specified program names are not case sensitive.

The following are examples of valid program name specifications:

XMAI
PAYROLL0
SELOPT%
SUBRTN*

The program name exclude list can also be edited from the Interactive Reanalysis Options display. For details, see Interactive reanalysis options on page 157.

Following the list of options is the trace of the listing/side file search.

This trace is equivalent to what can be obtained when using the IDITRACE DDname, as shown in IDITRACE information on page 383.

When the Compiler Listing Not Found display (Figure 163: Sample Compiler Listing Not Found display on page 236) is first shown, then the entire search trace up until that point in time is provided. From then on, the trace only contains the records that were written since the last time when the Compiler Listing Not Found display was shown.

Controlling prompting

The interactive reanalysis option, "Prompt for missing side files", determines if prompting occurs or not. For details about this option, see Interactive reanalysis options on page 157.

Data sets used for interactive reanalysis

When performing interactive reanalysis through the ISPF interface, pre-allocation is performed as required for any Fault Analyzer compiler listing or side file data sets that were used in real time. Allocations are performed for Fault Analyzer data sets if they were explicitly included in the real-time JCL, or supplied through the DataSets option or an Analysis Control user exit. These data sets are used in the reanalysis in an attempt to recreate the same execution environment as were used in real-time.

DataSets options that are specified via the IDIOPTS user options file or the PARM field cause those data sets to be logically concatenated to the data sets from the real-time execution.

If the *"Display panel to alter allocated data sets"* option on the *"Interactive options"* display is set to Y (see Interactive reanalysis options on page 157), 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
```

Note: If the GenerateSavedReport option is in effect, then all data set or path names that were used during real-time processing will automatically be included, even if these have been deleted from the pseudo JCL display before performing reanalysis.

Refresh processing

Refresh processing is what occurs when a fault entry is being rewritten due to user information, having been updated during interactive reanalysis. The user information which might cause this refresh processing is any of the following:

- User name, user title, or lock flag changed via the INFO command or the "File->Fault Entry Information" action-bar pull-down menu option.
- User notes against dump storage addresses added, deleted, or modified.

When either or both of these situations occur

- User information has changed
- A minidump with a size that exceeds the MaxMinidumpPages option in effect is first added to a fault entry

then a display like the example shown in Figure 161: Sample refresh processing exit prompt on page 241 is presented upon exiting the interactive reanalysis report.

Figure 167. Sample refresh processing exit prompt

```
Fault Entry Refresh                                      Line 1 Col 1 76
Command ===> _____________________________________________ Scroll ===> CSR

Press Enter to continue using the fault entry refresh options below.

* User information has been added or modified.

  Save Updated User
    Information . . . . . . : Y (Y/N)

  NOTE: If user information is not saved, the minidump will not be saved
        either.

* A minidump does not yet exist for this fault entry, but the minidump
size exceeds the current limit.

MaxMinidumpPages Option . : 5
Current Minidump Pages. . : 100
Save Minidump . . . . . . : N (Y/N)

*** Bottom of data.
```

If the display is prompting for both the saving of updated user information, and the writing of a minidump, then the minidump can only be written if the user information is also saved.

To return to the Fault Entry List display using the selected options, press the Enter key.

To return to the Fault Entry List display using the default options, which are to save the updated user information but suppress the minidump, enter the CANCEL command.

It is not possible to end this display using the END or EXIT commands.

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 630).

COBOL Explorer

COBOL Explorer uses the event source line to create a branch analysis that shows a procedure traceback and use of selected variables. References that modify selected variables are highlighted. This set includes modify by group or REDEFINES.

The traditional editor-based source program presentation is replaced by a collapsed or expanded source view where you see just the parts of the program that are relevant. This approach makes working with large COBOL programs a snap.

There are three ways to start COBOL Explorer:

- From the point-and-shoot field that is in the Event Details display for a COBOL program event.
- From the Services pull-down menu.
- By entering the CE command anywhere during interactive reanalysis.

If you enter the CE command without a parameter, or start COBOL Explorer from the Services pull-down menu, a pop-up list is displayed. Select a program from the pop-up.

COBOL Explorer example

The following is an example of how to use the COBOL Explorer feature of Fault Analyzer.

A fault entry for an abend in a COBOL program is analyzed interactively using the 'I' line command from the Fault Entry List display. The "Event Summary" is selected, and from there the COBOL program event is selected, resulting in the "Event Details" display shown in Figure 168: Sample COBOL Explorer Event Details display on page 243.

Figure 168. Sample COBOL Explorer Event Details display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Event 1 of 1: Abend S0CB *** Point of Failure ***              Line 1 Col 1 80
Command ===> _____________________________________________ Scroll ===> CSR
JOBNAME: COBPERF6  SYSTEM ABEND: 0CF            FAE1      2019/07/21  16


Abend Code. . . . . . . . . : S0CB
Program-Interruption Code . : 000B (Decimal-Divide Exception)
  The divisor was zero in a signed decimal division.

The source code below was executed via the following sequence of PERFORM
statements:
  Source
  Line #
  000105             PERFORM READ-FILE UNTIL END-OF-FILE = '1'.
  000122               NOT AT END PERFORM PROCESS-VEHICLE
  000133                 PERFORM PROCESS-CAR
  000151               PERFORM CALC-TAX.

COBOL Source Code:
  Source
  Line #
  000175               COMPUTE BASE-AMOUNT = PRICE / CC

Data Field Declarations:
  Source
  Line #
  000033            03 PRICE             PIC 9(6).
  000049                  07 CC          PIC 9(4).
  000094         01 BASE-AMOUNT          PIC 9(3)V99 COMP-3.

Data Field Values:
  BASE-AMOUNT = 0.00
  CC          = 0  *** Cause of error ***
  PRICE       = 50000

COBOL Explorer  ❶

The listing file used for the above was found in
MY.LISTING.COBOL(COBPERF6).

⋮
```

By selecting the "COBOL Explorer" point-and-shoot field shown at ❶, the "Source Line and Data Values" display in Figure 169: Sample COBOL Explorer Source Line and Data Values display on page 244 is shown.

Figure 169. Sample COBOL Explorer Source Line and Data Values display

```
Source Line and Data Values                                Line 1 Col 1 76
Command ===> _____________________________________________ Scroll ===> CSR

Source:
  000175             COMPUTE BASE-AMOUNT = PRICE / CC

Data Field Declarations:
  000033          03 PRICE             PIC 9(6).
  000049                07 CC          PIC 9(4).
  000094       01 BASE-AMOUNT          PIC 9(3)V99 COMP-3.

Data Field Values:
s BASE-AMOUNT = 0.00
s CC          = 0  *** Cause of error ***  ❷
s PRICE       = 50000

Select data fields to use with source line to create view.

*** Bottom of data.
```

Because the data field "CC" at ❷ has been identified as the cause of the error (divide by zero), we want to look at this data field more closely. To do this, unselect the other two data fields ("BASE-AMOUNT" and "PRICE") by over-typing with blanks, and press Enter. The result is the display shown in Figure 170: Sample COBOL Explorer Debug View display on page 244.

Figure 170. Sample COBOL Explorer Debug View display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Exploring COBOL Program COBEX1                              Line 1 Col 1 80
Command ===> _____________________________________________ Scroll ===> CSR
JOBNAME: COBEX1    SYSTEM ABEND: 0CB            FAE1      2019/04/14  20:40:31
+ Expand all / - Collapse all    - Comments  + Level 88  - Redefines

+ 000001 IDENTIFICATION DIVISION.
+ 000003 ENVIRONMENT DIVISION.
+ 000011 FILE SECTION.
+ 000017 WORKING-STORAGE SECTION.
+ 000026 01 VEHICLE-RECORD.
  000038    03 SPECIFICATION.
  000042       05 ENGINE.
  000049          07 CC          PIC 9(4).
+ 000081 01 HEADINGS REDEFINES VEHICLE-RECORD.
+ 000084 01 WS-REC REDEFINES VEHICLE-RECORD.
+ 000098 LINKAGE SECTION.
+ 000101 PROCEDURE DIVISION USING PARMS.
  000105     PERFORM READ-FILE UNTIL END-OF-FILE = '1'.
+ 000119 READ-FILE.
  000122       NOT AT END PERFORM PROCESS-VEHICLE
+ 000125 PROCESS-VEHICLE.
  000126     MOVE FS-REC TO WS-REC.
  000133       PERFORM PROCESS-CAR
+ 000149 PROCESS-CAR.
  000151       PERFORM CALC-TAX.
+ 000173 CALC-TAX.
  000175       COMPUTE BASE-AMOUNT = PRICE / CC
  000178       COMPUTE BASE-AMOUNT = CC / CYLINDERS
```

The above Debug View display consists of a branch-analysis showing how the program got to the source line, and any source lines that reference the selected variable (or variables, if more than one had been selected from the previous Source Line and Data Values display).

This view shows that the abending source line 175 is in procedure CALC_TAX, and the branch analysis reveals the execution path to the source line.

In addition to selected variables, any references to containing group items or redefines are also shown. Any source lines that modify a reference are highlighted (often these are used to create a new view as variables that modify referenced variables might be of particular interest, use PF10/PF11 to scroll views).

You can select the following by placing the cursor on the appropriate line and pressing Enter (or just double-click, if you have selected the "Fnn" and "ENTER at cursor position" Personal Communications options available through Settings > Hotspots):

- Select any highlighted source line to create a new view.
- Select a DATA DIVISION section (for example, WORKING-STORAGE SECTION) to invoke the Associated Storage Areas display for that section (this shows all variables and their values). The FILE SECTION also allows you to edit/browse/view open files with ISPF or File Manager.
- Select a data item to show all references to it and the section/paragraph in which the reference occurs. For group items and group members, an aggregate map is also shown.
- Select a procedure name to show all branches to it and the procedure in which the branch occurs.
- Select an executable source line to show the value of its variables and optionally create a new view for that source line and selected variables. If the source line is a branch (for example, PERFORM, GO TO), the target procedure is expanded (subsequent collapse returns to the branch statement).
- Select an MQI call (MQOPEN/MQGET/MQPUT/MQCLOSE) to edit/browse the queue or list queue managers with File Manager WebSphere® MQ.

Section Expand/Collapse, by means of placing the cursor on the '+' or '-' signs on the left of the display and pressing Enter, is used to explore the program. Sections comprise FILE SECTION, WORKING-STORAGE SECTION and so on, plus any sections and paragraphs in the PROCEDURE DIVISION.

DATA DIVISION level 01 group items are initially collapsed. These can be expanded to show group members.

Expand the PROCEDURE DIVISION to reveal all procedures. These can then be expanded individually to show source.

Global Expand/Collapse, by means of placing the cursor on the '+' or '-' signs at the top of the display and pressing Enter, provides more filtering of the source program:

- "Expand all" expands all sections and group items. (Note that this causes all source records to be searched when using the Find command.)
- "Collapse all" restores the view's initial display.
- "Comments" expands or collapses comment lines.
- "Level 88" expands or collapses level 88 items showing only the high-level item when collapsed.
- "Redefines" expands or collapses REDEFINES showing only the high-level item when collapsed.

View the video introducing COBOL Explorer at https://www.youtube.com/watch?v=ZXwsaBnfk2Q.

Deferred Breakpoints Feature

COBOL Explorer supports the setting of z/OS® Debugger Deferred Breakpoints, similar to IPVLANGP. For details, see *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*.

To set Deferred Breakpoints, ADFz Common Components must be installed.

Chapter 6. Performing CICS system abend dump analysis

A feature unique to the interactive component of Fault Analyzer is the ability to analyze information that is related to CICS® system abends.

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 topics 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 157).

User exits

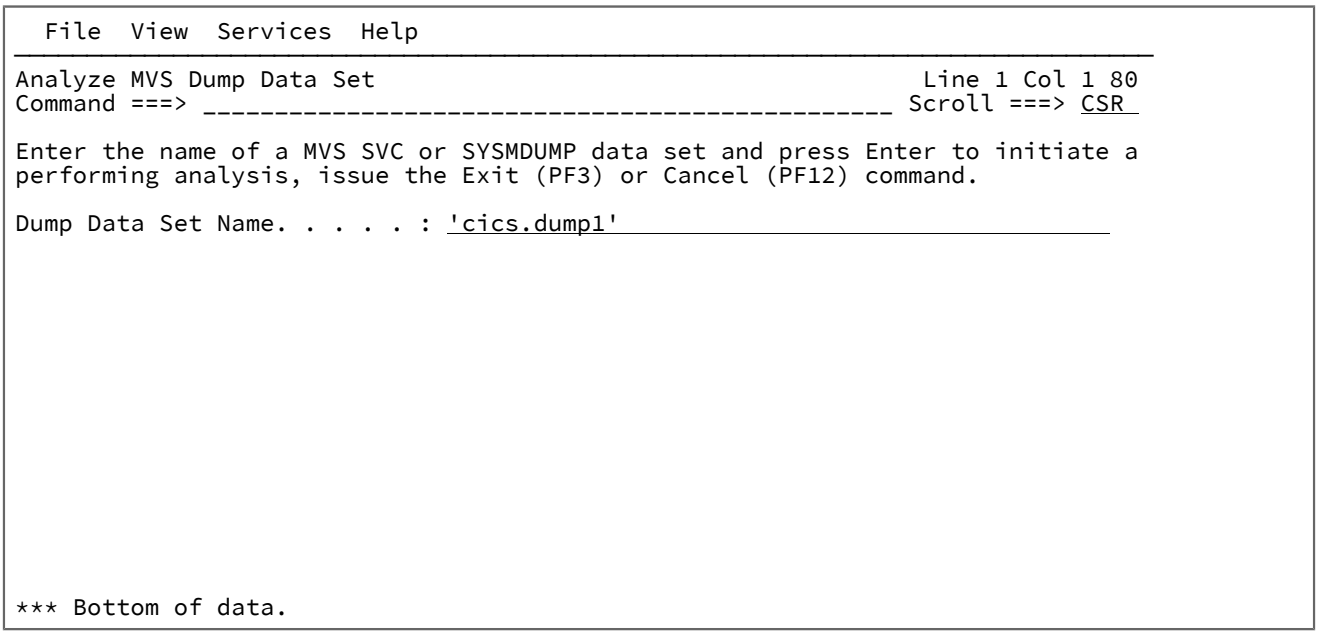
Since CICS® system abend analysis is performed as reanalysis against an MVS™ SVC dump or SYSMDUMP data set, only the following user exits can be used:

Analysis Control user exit on page 475
Compiler Listing Read user exit on page 480
Message and Abend Code Explanation user exit on page 485
Formatting user exit on page 490

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 94). This option brings up the Analyze MVS Dump Data Set display as shown in Figure 171: Sample Analyze MVS Dump Data Set display on page 248.

Figure 171. Sample Analyze MVS Dump Data Set display

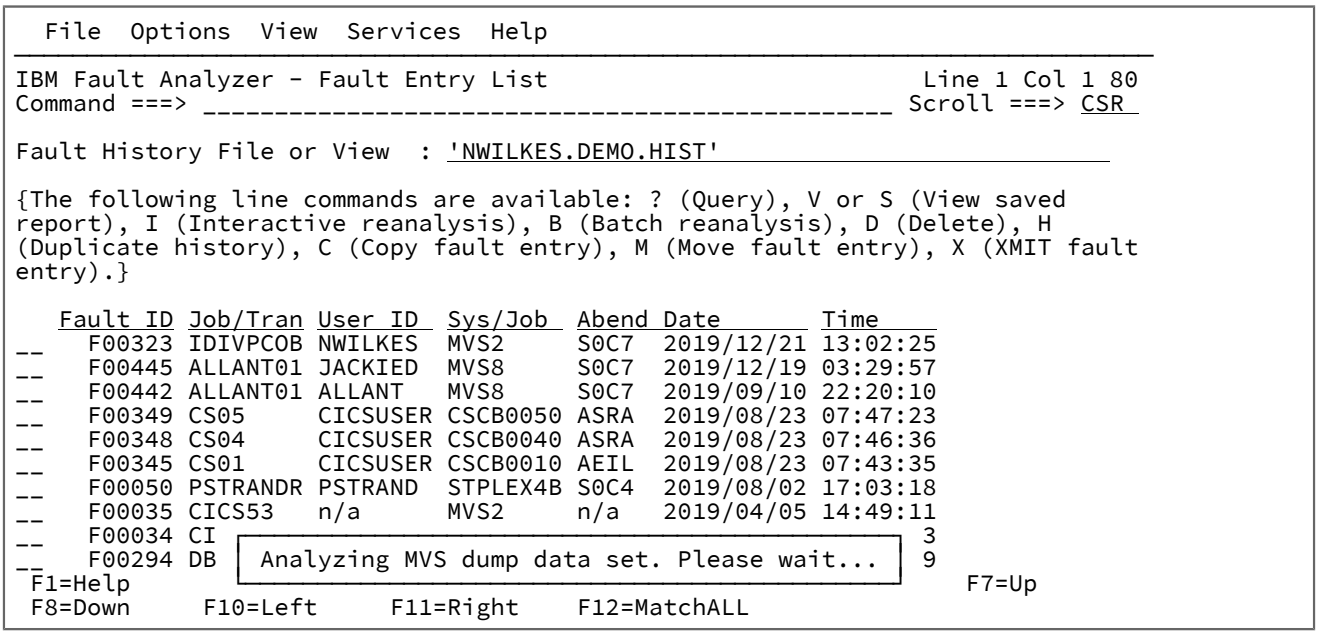


On this display, type the name of the SVC dump or SYSMDUMP data set containing the CICS® system abend dump to be analyzed. The data set name-specification follows the ISPF data set name rules, that is, a data set name that is not enclosed in single quotes is prefixed by the current TSO profile prefix. The data set specified is checked for existence before being accepted. In this example, data set CICS.DUMP1 is being analyzed.

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

When the specified MVS™ dump data set name has been validated, the Fault Analyzer CICS® system abend analysis commences as indicated by the *"Analyzing MVS dump data set. Please wait..."* message being displayed (Figure 172: Recognizing that analysis has commenced on page 249).

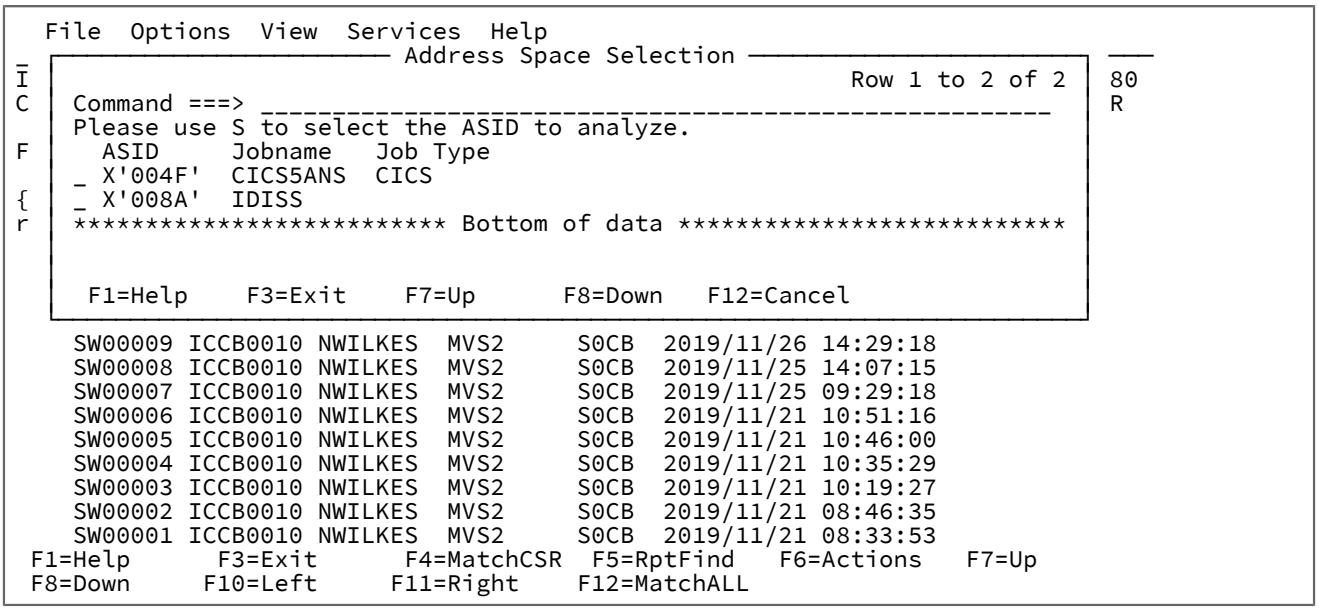
Figure 172. Recognizing that analysis has commenced



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 173: CICS system dump analysis address space selection on page 249, from which a selection can be made by typing an S against the desired address space, and pressing Enter. Any address spaces in which CICS® was found to be active are marked with a job type of "CICS".

Figure 173. CICS system dump analysis address space selection



If the dump contains only one address space, then no address space selection display is presented.

Displaying the CICS system abend interactive report

When the analysis of the CICS® system abend has completed, the CICS® system abend interactive report is shown (Figure 174: Sample CICS System Abend Interactive Reanalysis Report display on page 250).

Figure 174. Sample CICS System Abend Interactive Reanalysis Report display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
CICS System Abend Interactive Reanalysis Report               Line 1 Col 1 80
Command ===> _____________________________________________ Scroll ===> CSR
CICS DUMP: SYSTEM=CICSDI   CODE=SM0002   ID=      D381     2019/08/13 09:55:03

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
```

The initial interactive report display for a CICS® system abend is different from that of any other fault type. The content is logically divided into the following sections:

Available options

At the top of the display are options that can be selected by placing the cursor on the option number and pressing the Enter key. The tab key can be used to position the cursor to the option that you want. The individual options are explained in the sections that follow.

Dump reason

If available, the list of options is followed by the reason why the analyzed dump was taken.

Analysis observations

If any, the dump reason is followed by observations made during the analysis. These are loosely divided into three categories based on severity:

Severity	Description
1	Problems that are likely to be reasons for the fault.
2	Problems that might be reasons for the fault.

If ten or more severity 2 observations are available, then a point-and-shoot link that takes you to a separate display is provided instead.

3

Problems that generally would not be considered contributing reasons for the fault, but merely items of interest.

If five or more severity 3 observations are available, then a point-and-shoot link that takes you to a separate display is provided instead.

Fastpath navigation

To make navigation easier in the CICS® system dump analysis displays, fastpath commands can be entered on any command line. The valid fastpath commands are all of the highlighted fields shown in the CICS® System Information display (an example is shown in Figure 177: Sample CICS System Information display on page 253).

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 175: Sample CICS System Abend Synopsis display on page 251.

Figure 175. Sample CICS System Abend Synopsis display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Synopsis                                                      Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR_
CICS DUMP: SYSTEM=CICSDI   CODE=SM0002   ID=      D381     2019/08/13 09:55:03

Fault Information:

  CICS Product Level . . . . : V5 R3 M0
  Dump ID. . . . . . . . . . : 4/0007
  Dump Code. . . . . . . . . : SM0002
  Date/Time. . . . . . . . . : 2019/08/13 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
```

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, similar to the example shown in Figure 176: Sample Abend Job Information display on page 252.

Figure 176. Sample Abend Job Information display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Abend Job Information                                          Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
SYSTEM=C72E1FA1 CODE=CHAN ID=1/0002 FAE1 2019/10/10 08:42:25

IBM Fault Analyzer Abend Job information:

  Dump Date (Local) . . . . . : 2019/10/10
  Dump Time (Local) . . . . . : 08:42:25
  Dump Date (GMT) . . . . . : 2019/10/10
  Dump Time (GMT) . . . . . : 00:42:25
  System Name . . . . . : FAE1
  Subsystem Info. . . . . : CICS V7 R2 M0  (TS 5.5)
  Job Name. . . . . : AS720F1
  Job Step Name . . . . . : AS720F1
  Exec Program Name . . . . . : DFHSIP
  Requested Region Size . . . : 0M
  User id . . . . . : COSMICK

Execution Environment:

  Operating System. . . . . : z/OS     V02R03M00
  Data Facility Product . . . : DFSMS z/OS V2R3M0
  CPU Model . . . . . : 2965
 F1=Help      F3=Exit      F5=RptFind   F6=Actions   F7=Up        F8=Down
F10=Left     F11=Right
```

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 177: Sample CICS System Information display on page 253.

Figure 177. Sample CICS System Information display

```
  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.
```

This display provides options to select CICS® system information of interest.

Sorting and matching table displays

Whenever table headings are tabbable and shown in reverse highlight mode, then this highlighting indicates that the table column attributes are modifiable. By placing the cursor on the heading, and pressing Enter, a Column Attributes display is presented. This display allows you to sort the column data in ascending or descending order, or to show only the table rows that satisfy a given MATCH criteria.

The MATCH attribute is case insensitive and permits the use of wildcards. The supported wildcard characters are an asterisk (*) to indicate zero, one or more characters, and a percent sign (%) to indicate a single required character. Only table data rows that match the specified string are shown.

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

The reset can be performed by placing the cursor on the reset point-and-shoot field and pressing the Enter key. Alternatively, a RESET command can be entered on the command line to reset all tables in a given display simultaneously.

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

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

Option 4: Options in Effect

Selecting option 4 from the CICS® System Abend Interactive Reanalysis Report display results in the presentation of the Options in Effect display, of which an example is shown in Figure 178: Sample Options in Effect display on page 254.

Figure 178. Sample Options in Effect display

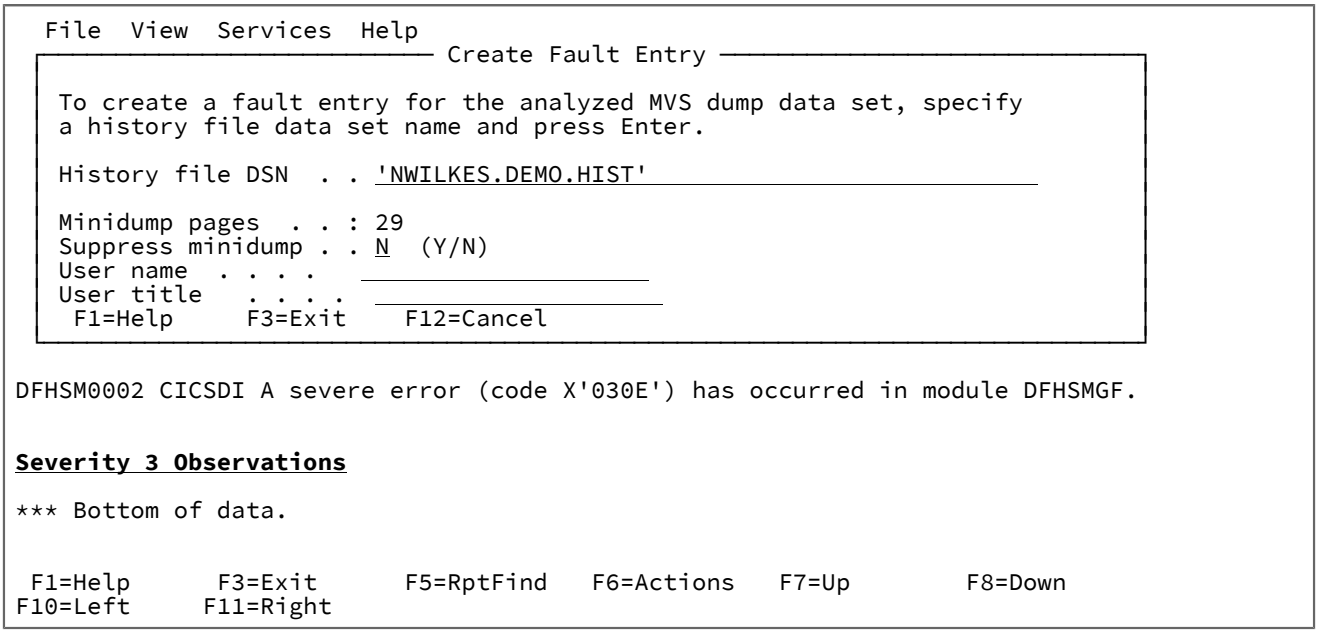
```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Options in Effect                                          Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
CICS DUMP: SYSTEM=CICSDI   CODE=SM0002   ID=      D381     2019/08/13 09:55:03

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)
```

This display provides information about Fault Analyzer options in effect for this reanalysis report.

Creating a fault entry

When you exit from the CICS® System Abend Interactive Reanalysis Report, you are presented with the option to create a fault entry as shown in Figure 179: Sample Create Fault Entry display on page 255.

Figure 179. Sample Create Fault Entry display



To create a fault entry:

1. If necessary, change the supplied name of the history file (obey the ISPF data set name specification rules).
2. Optionally:
   - Change the *"Suppress minidump"* option to 'Y' to show that a minidump with the indicated number of pages must not be created.
   - Provide a user name or title.
3. Press Enter.

The "History file DSN" field is by default initialized with the current history file name that is selected on the Fault Entry List display. If a view is being used, then the first history file name that is specified in the view member is used. To avoid creating a fault entry, press either **F3** or **F12**.

Instead of the default, the user can specify a different history file name. 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 specified history file does not exist, then the New History File Allocation display is shown (for details, see New history file allocation on page 87).

The user must have at least UPDATE access to the specified history file.

If a fault entry was created, message IDI0003I on page 696 is issued to inform you of the assigned fault ID. The history file in which the fault entry was created is automatically selected as the current history file. A `MATCH FAULT_ID fault_id` command is issued so that only the newly created fault entry is displayed. This command is issued because the newly created fault entry might not be at the top of the chronologically ordered Fault Entry List display, and thus difficult to locate. To again see all fault entries in the history file, enter the `MATCH ALL` command (normally mapped to PF12).

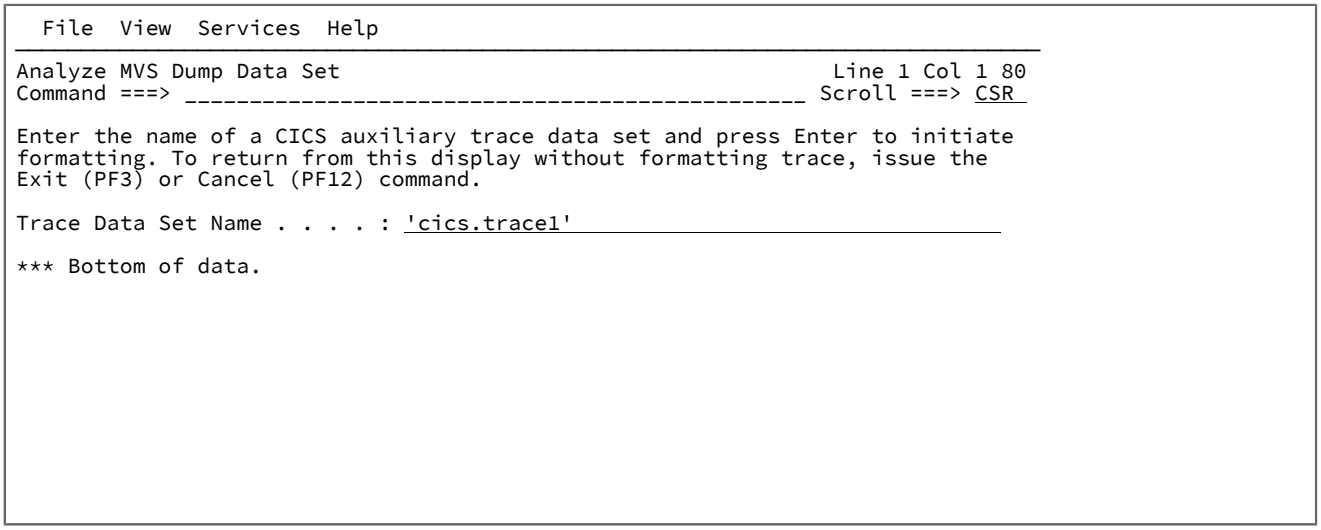
The performance of subsequent reanalysis of the fault is improved by creating a fault history entry for a CICS® system abend, and writing a minidump.

Chapter 7. Formatting a CICS auxiliary trace data set

Selecting a CICS auxiliary trace data set

To select a CICS® auxiliary trace data set, first select the Format CICS Auxiliary Trace Data Set option from the Fault Entry List display File menu (for information about selecting this option in general, refer to Action-bar pull-down menus on page 94). This opens the Format CICS Auxiliary Trace Data Set display as shown in Figure 180: Sample Format CICS Auxiliary Trace Data Set display on page 257.

Figure 180. Sample Format CICS Auxiliary Trace Data Set display

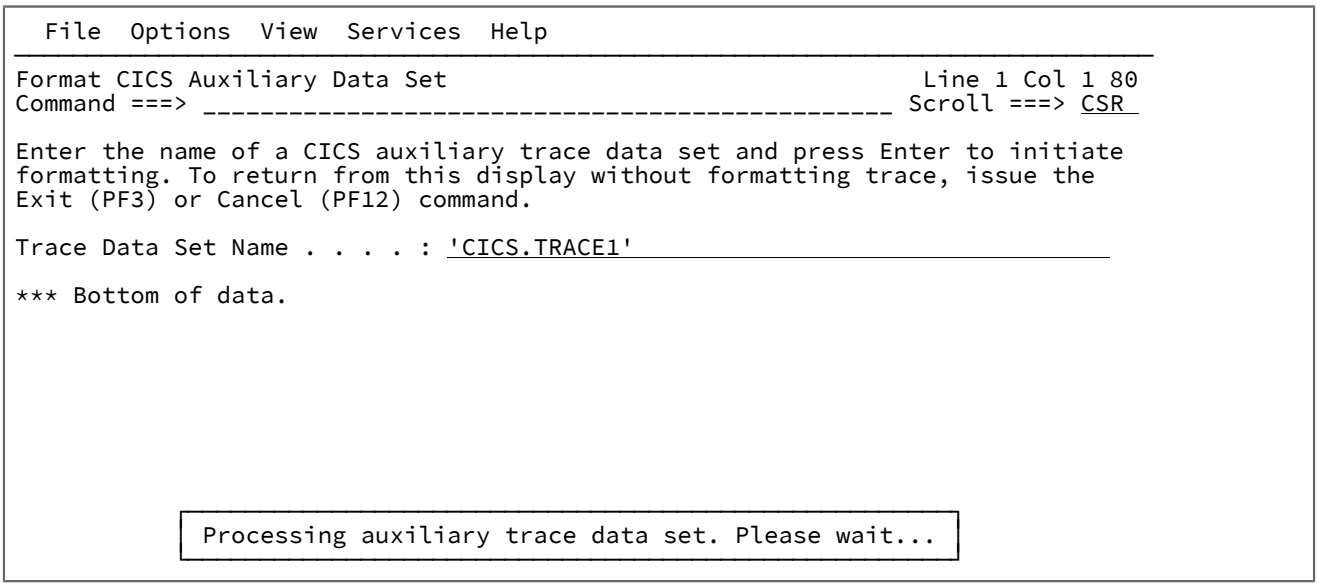


On this display, type the name of the data set containing the CICS® auxiliary trace to be formatted. The data set name-specification follows the ISPF data set name rules, that is, a data set name that is not enclosed in single quotes is prefixed by the current TSO profile prefix. The data set specified is checked for existence before being accepted. In this example, data set CICS.TRACE1 is being formatted.

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

When the specified trace data set name has been validated, the Fault Analyzer CICS auxiliary trace formatting commences as indicated by the *"Processing auxiliary trace data set. Please wait..."* message being displayed (Figure 181: Recognizing that trace processing has commenced on page 258).

Figure 181. 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 180, allows you to specify which internal trace entries are to be displayed and what level of formatting is required.

Chapter 8. Performing Java analysis

A feature unique to the interactive component of Fault Analyzer is the ability to present information that is related to Java™. The Java™ execution might be under WebSphere® Liberty, CICS®, or Unix System Services on MVS™. Typically, the environment is Java™ calling legacy programs. These topics explain how to:

- Set options for Java™ analysis
- Select a Java™ dump data set, or an existing Java™ fault entry, for analysis
- Display the resulting Java™ information in the interactive report

The steps outlined in the following assume that you have already started the interactive component of Fault Analyzer from an ISPF session.

Note: Java™ analysis is only compatible with the JVMs shown in Fault Analyzer supported application environments on page 26.

Setting options for Java™ analysis

The general interactive reanalysis options are also used for Java™ analysis (see Interactive reanalysis options on page 157).

An optional IDIJVM DD statement added to the IDIS subsystem start-up JCL allows for a default JVM to be used for performing Java™ analysis (for details, see IDIS subsystem requirements for Java on page 324).

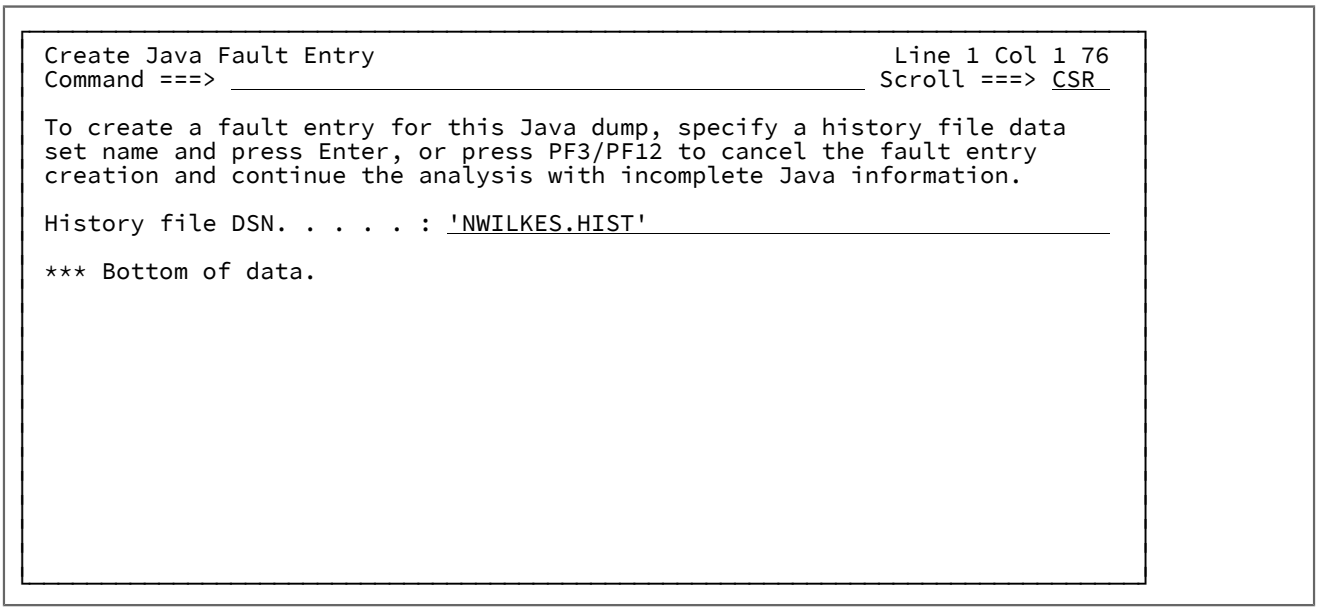
Selecting a Java dump data set

Specification of a Java™ dump data set to be analyzed is done by first selecting **File > Analyze MVS Dump Data Set** from the **Fault Entry List** display.

This option brings up the Analyze MVS™ Dump Data Set display on which the MVS™ dump data set name can be entered. An example of this display is shown in Selecting a CICS dump data set on page 247.

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 shown in Figure 182: Sample Create Java Fault Entry display on page 260.

Figure 182. Sample Create Java Fault Entry display



Specify a history file that is managed by the IDIS subsystem and to which you have at least UPDATE access. 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 specified history file does not exist, then the New History File Allocation display is shown (for details, see New history file allocation on page 87).

Press **Enter** to create a fault entry and return to the **Fault Entry List** display. A MATCH is automatically performed to show only the fault entry that was just created.

Note: Fault Analyzer Java™ DTFJ processing requires that the MVS™ post-dump exit IDIXTSEL is installed (for details, see Installing the MVS post-dump exit IDIXTSEL on page 412) and the IDIS subsystem is started (for details, see Using the Fault Analyzer IDIS subsystem on page 318).

If the DTFJ_Status column has been selected for display in the **Fault Entry List**, then you can keep pressing Enter to refresh the value and wait for DTFJ_Status to show **Finished**. By then, the asynchronous DTFJ processing has added information to the fault entry so that a subsequent reanalysis using the I line command will present a complete analysis report that includes all available Java information.

If PF3 or PF12 is pressed on the **Create Java Fault Entry** display, the early fault entry creation is skipped but the analysis still continues. In this case, the dump analysis is performed as if no Java™ activity was detected, and the user is prompted to create a fault entry when exiting the analysis instead.

Java fault entry reanalysis

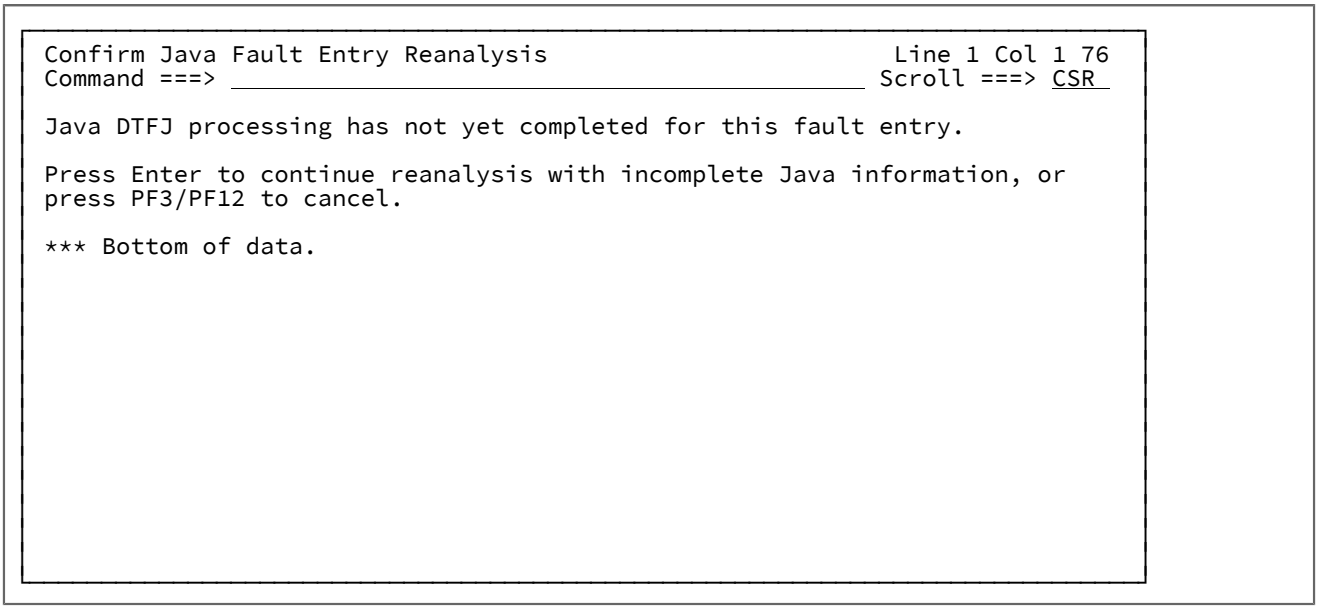
Only fault entries that were created from analysis of Java™ dumps, or from real-time analysis of Java™ abends, include Java™ information in the reanalysis report.

Note: Information about real-time invocation of Fault Analyzer from Java™ applications is provided in Real-time analysis on page 32.

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 183: Sample Confirm Java Fault Entry Reanalysis display on page 261.

Figure 183. Sample Confirm Java Fault Entry Reanalysis display



By pressing the Enter key, reanalysis continues, but the Java™ information is incomplete.

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

If the asynchronous DTFJ processing for the fault entry failed to complete within a reasonable time, then message IDI0177E on page 735 is issued instead of the Confirm Java™ Fault Entry Reanalysis display.

Note: If DTFJ processing never completes, then the reason might be one of the following:

- The MVS™ post-dump exit IDIXTSEL has not been installed (for details, see Installing the MVS post-dump exit IDIXTSEL on page 412).
- The IDIS subsystem has not been started (for details, see Using the Fault Analyzer IDIS subsystem on page 318).

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 184: Sample Java Interactive Reanalysis Report display on page 262.

Figure 184. Sample Java Interactive Reanalysis Report display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Interactive Reanalysis Report                                  Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
JOBNAME: BPXBATC3  SYSTEM ABEND: 0CB            FAE3      2010/03/16  19:25:50
User Title: Java

Fault Summary:
Module /u/rturner/kenichiExample/j2c2cob/libMyDllLib.so, program MYCOB1,
offset X'3CE': Abend S0CB (Decimal-Divide Exception).

Select one of the following options to access further fault information:
  1. Synopsis
  2. Event Summary
  3. Java Information
  4. Storage Areas
  5. Messages
  6. Language Environment Heap Analysis
  7. Abend Job Information
  8. Fault Analyzer Options

{Fault Analyzer maximum storage allocated: 9.12 megabytes.}

*** Bottom of data.
```

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 185: Sample Java Event Summary display on page 263.

Figure 185. Sample Java Event Summary display

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Event Summary                                                    Top of data
Command ===> ____________________________________________________ Scroll ===> CSR
Full Application only - JOBNAME: BPXBATC3  SYSTEM ABEND: 0CB  FAE3   2010/03/16

{The following events are presented in chronological order.}

Event            Fail  Module    Program   EP
#  Type          Point Name      Name      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                libMyDllLib.so
                                 *Java_Ja  Java_Java2C2CobolExample_callCobol
                                                     P+3F8 E+90              /u/rtu
13 Call                libMyDllLib.so
                                 *Java_Ja  doSomething1
                                                     P+340 E+90              /u/rtu
14 Call                libMyDllLib.so
                                 *Java_Ja  doSomething2

                                                     P+288 E+90              /u/rtu
15 Call                libMyDllLib.so
                                 *Java_Ja  doSomething3
                                                     P+142 E+92              /u/rtu
16 Abend S0CB   ***** libMyDllLib.so
                                 MYCOB1    MYCOB1    P+3CE E+3CE             /u/rtu


NOTE: Program names prefixed '*' are pseudo CSECT names created to help
      determine in which compile unit an entry point belongs.

(*) One or more of the following abbreviations might appear in the "Event
    Location" column:

    F#n  Source file number (refer to detailed event information for file
         identification)
    L#n  Source file line number
    S#n  Listing file statement number (refer to detailed event information
         for file identification)
    M+x  Offset from start of load module
    P+x  Offset from start of program
    E+x  Offset from start of entry point
```

Selecting the Java™ event # 6 from the above example, results in the Java™ event details display, as the example shown in Figure 186: Sample Java Event Details display on page 264.

Figure 186. Sample Java Event Details display

```
  File  View  Services  Help
------------------------------------------------------------------------------
Event Summary                                                    Top of data
Command ===> ____________________________________________ Scroll ===> CSR

Previous Event Details


This event occurred in Class Java2C2CobolExample Method main.

Java source from /u/rturner/kenichiExample/j2c2cob/Java2C2CobolExample.java:
  Source
  Line #
      -5   }
      -4
      -3   public static void main(String args[]) {
      -2     System.out.println("Hello World from Java program!");
      -1     Java2C2CobolExample e = new Java2C2CobolExample();
  000010     e.callCobol();
      +1   }
      +2 }
      +3
      +4


The class static variable information is not available.
The object instance variable information is not available.


Next Event Details

*** Bottom of data.
```

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 187: Sample Java Information display (Part 1 of 2) on page 265.

Figure 187. Sample Java Information display (Part 1 of 2)

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Java Information                                              Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
JOBNAME: BPXBATC2  SYSTEM ABEND: 0CB            FAE2      2009/10/02  06:53:54

Java Version. . . . . . . . : JRE 1.8.0 z/OS s390-31 (build 8.0.5.36 -
                              pmz3180sr5fp36-20190510_01(SR5 FP36))

Java environment variables

_ envvar ____________________________________________________________________

  IBM_JAVA_COMMAND_LINE=java Java2C2CobolExample
  LIBPATH=/apc/java800-UK35911/usr/lpp/java/J8.0/lib/s390/j9vm:/apc/java800-UK
  1/usr/lpp/java/J8.0/../lib/s390:/apc/java800-UK35911/usr/lpp/java/J8.0/lib/s
  31-UK18621/usr/lpp/java/J8.0/bin:/apc/java800-UK18621/usr/lpp/java/J8.0/bin/
  /j9vm:/apc/java800-UK35911/usr/lpp/java/J8.0/lib/s390
  _BPX_SHAREAS=NO
  JAVA_HOME=/apc/java800-UK18621/usr/lpp/java/J8.0
  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/java800-UK18621/usr/lpp/java/J8.0/bin/:.:/apc/java800-UK35911/usr/
  _=/apc/java800-UK35911/usr/lpp/java/J8.0/bin/java
```

Figure 188. Sample Java Information display (Part 2 of 2)

```
Java VM init args

_ args

  arg=-Xjcl:jclscar_24
  arg=-Dcom.ibm.oti.vm.bootstrap.library.path=/apc/java800-UK35911/usr/lpp/jav
  arg=-Dsun.boot.library.path=/apc/java800-UK35911/usr/lpp/java/J8.0/lib/s390
  arg=-Djava.library.path=/apc/java800-UK35911/usr/lpp/java/J8.0/lib/s390:/apc
  c/java800-UK35911/usr/lpp/java/J8.0/lib/s390:/apc/java800-UK35911/usr/lpp/ja
  a/J8.0/lib/s390::.:/u/rturner/kenichiExample/j2c2cob::/apc/java800-UK18621/u
  va/J8.0/bin/j9vm:/apc/java800-UK35911/usr/lpp/java/J8.0/lib/s390/j9vm:/apc/j
  arg=-Djava.home=/apc/java800-UK35911/usr/lpp/java/J8.0
  arg=-Djava.ext.dirs=/apc/java800-UK35911/usr/lpp/java/J8.0/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
```

Java event replacement in Fault Analyzer event list

When an abend includes Java™ events, Fault Analyzer requests a system dump of the JVM. This Java™ system dump is analyzed by Fault Analyzer using Diagnostic Tool Framework for Java (DTFJ) APIs.

The DTFJ processing occurs asynchronously and does not delay the initial real-time abend analysis. This means that the real-time Fault Analyzer report contains only internal JVM events, not Java™ events.

When DTFJ processing has completed, the additional Java™ information is added into the fault entry so when reanalysis of the fault entry is performed, the more concise and useful Java™ event information is shown instead.

For example, the real-time event summary might show the following:

```
Event           Fail  Module   Program   EP
#  Type         Point Name     Name      Name     Event Location (*) Description
-- ------------ ----- -------- --------  -------- ------------------ ---------------------------------------------
 1 Call               JVMLDM71 CEEROOTA  n/a      P+D4               BOOTSTRAP MODULE FOR LE; From CTEST.AUTHLOAD
 2 Call               CEEPLPKA n/a       CEEBBEXT E+1D2              BOOTSTRAP MODULE FOR Language Environment; From LPA
 3 Call               CELHV003 n/a       EDCZHINV E+B4               CRTL Main invocation event XPLINK; From CEE.SCEERUN2
 4 >>> XPLink         CEEPLPKA n/a       CEEVROND E+127E             Run on down stack swap; From LPA
 5 Call               JVMLDM71 JzosVM#C  main     E+A6               From CTEST.AUTHLOAD
 6 Call               JVMLDM71 JzosVM#C  JzosVM::run(int,char**)
                                                  E+34A              From CTEST.AUTHLOAD
 7 Call               JVMLDM71 JzosVM#C  JzosVM::invokeMain()
                                                  E+50A              From CTEST.AUTHLOAD
```

```
 8 Call               JVMLDM71 JzosVM#C  JNIEnv_::CallStaticVoidMethod(_jclass*,_jmethodID*,...)
                                         E+30               From CTEST.AUTHLOAD
 9 Call               libj9vm27.so
                                n/a      callStaticVoidMethodV
                                         E+4E
 From /apc/java710/31bit/usr/lpp/java/J7.1/lib/s390/default/
10 Call               libj9vm27.so
                                n/a      gpCheckCallin
                                         E+5C
 From /apc/java710/31bit/usr/lpp/java/J7.1/lib/s390/default/
11 Call               n/a      n/a      gpProtectAndRun
                                         n/a                From not determined
12 Call               libj9prt27.so
                                n/a      j9sig_protect_ceehdlr
                                         E+176
 From /apc/java710/31bit/usr/lpp/java/J7.1/lib/s390/default/
13 Call               libj9vm27.so
                                n/a      signalProtectAndRunGlue
                                         E+14
 From /apc/java710/31bit/usr/lpp/java/J7.1/lib/s390/default/
14 Call               n/a      n/a      gpProtectedRunCallInMethod
                                         n/a                From not determined
15 Call               n/a      n/a      RUNCALLINMETHOD
                                         n/a                From not determined
16 <<< XPLink         CEEPLPKA n/a      CEEVRONU E+10CE            CEL Common Runtime; From LPA
17 Abend S0C4   ***** libHelloWorld.so
                                n/a      Java_HelloWorld_badArrayAccess
                                         E+112              From /u/ctest/javatest-j2c/bin/
```

After DTFJ processing has completed, Java™ events are displayed:

```
The following events are presented in chronological order.

 Event           Fail  Module   Program  EP
 #  Type         Point Name     Name     Name     Event Location (*) Description
 -- ------------ ----- -------- -------- -------- ------------------ -------------------------------------------
  1 Call               JVMLDM71 CEEROOTA n/a      P+D4               BOOTSTRAP MODULE FOR LE; From CTEST.AUTHLOAD
  2 Call               CEEPLPKA n/a      CEEBBEXT E+1D2              BOOTSTRAP MODULE FOR Language Environment; From LPA
  3 Call               CELHV003 n/a      EDCZHINV E+B4               CRTL Main invocation event XPLINK; From CEE.SCEERUN2
  4 >>> XPLink         CEEPLPKA n/a      CEEVROND E+127E             Run on down stack swap; From LPA
  5 Call               JVMLDM71 JzosVM#C main     E+A6               From CTEST.AUTHLOAD
  6 Java               n/a      n/a      com.ibm.j9ddr.vm27.view.dtfj.java.DTFJJavaStackFrame@300676d
                                         L#1                From not determined
  7 Call               n/a      n/a      RUNCALLINMETHOD
                                         n/a                From not determined
  8 <<< XPLink         CEEPLPKA n/a      CEEVRONU E+10CE            CEL Common Runtime; From LPA
  9 Abend S0C4   ***** libHelloWorld.so
                                n/a      Java_HelloWorld_badArrayAccess
                                         E+112              From /u/ctest/javatest-j2c/bin/
```

Java information reporting limitations

In fault entries created by the Fault Analyzer wrapper utility, variable information is not shown.

This is due to the way the Java™ virtual machine processes exceptions:

- The JVM searches each stack frame for a handler (that is, a try-catch block) for an exception.
- If there is no handler in a given frame, the JVM discards the stack frame variables and searches the next frame. This search-and-discard repeats until the JVM finds a handler for the exception.

The Fault Analyzer wrapper utility acts as the handler for the otherwise unhandled exception. This means that none of the user-application variable values exist at the point when Fault Analyzer is invoked.

For details on how the JVM handles exceptions, see the section on the athrow instruction in the JVM Specification.

Java application samples

The JCL samples provided in the IDI.SIDISAM1 sample data set demonstrate how to use Java™ and interlanguage calls with Fault Analyzer.

To run the samples, you must:

- Tailor the JCL job card to meet your site-specific requirements.
- Specify where Fault Analyzer, Java™, and the required compilers are installed.
- Run the samples with a region size of 300 MB or larger.
- Ensure that the user ID that runs the samples has an OMVS segment, because the samples create hierarchical file structure (zFS) files.

The samples create several temporary files that you must manually delete.

Sample 1 (IDISJAV1): Enterprise PL/I invoking Java in batch

The JCL in sample IDI.SIDISAM1(IDISJAV1) demonstrates how to use Fault Analyzer when Enterprise PL/I invokes Java™ in batch. It includes Java™ calling an Enterprise PL/I DLL that is located in a PDSE data set member.

These are the sample results expected in the Fault Analyzer report for the SNAP dump call:

- The event summary includes only Java™ events with class and method information.
- The individual Java™ events include source code but no Java™ variable information, because the Java™ virtual machine (JVM) does not provide a mechanism to obtain variable information.

These are the sample results expected in the Fault Analyzer report for the PL/I abend:

- The synopsis section includes the source code information from the PL/I program and variable declarations and values.
- Because the PL/I code was compiled with TEST(SEP) option, the PL/I source code information is obtained from SYSDEBUG.
- The event list includes both the PL/I events and the Java™ events in a single view. For Java™, the class and method names are shown only during reanalysis. (The information is not available to the real-time report.)
- In the Java™ events, both source code information and variable values are displayed. When a variable name is not available to the Diagnostic Tool Framework for Java™ (DTFJ) API, the variable address is reported instead.

Sample 2 (IDISJAV2): Enterprise COBOL invoking Java in batch

The JCL in sample IDI.SIDISAM1(IDISJAV2) demonstrates how to use Fault Analyzer when Enterprise COBOL invokes Java in batch. It includes Java calling an Enterprise COBOL DLL that is located in a PDSE data set member.

These are the sample results expected in the Fault Analyzer report for the SNAP dump call:

- The event summary includes only Java events with class and method information.
- The individual Java events include source code but no Java variable information, because the Java virtual machine (JVM) does not provide a mechanism to obtain variable information.
- The Java information section includes information such as the Java virtual machine (JVM) version, Java environment variables, and JVM initialization arguments (`initArgs`).

These are the sample results expected in the Fault Analyzer report for the COBOL abend:

- The synopsis section displays COBOL source code information with associated variable declarations and values.
- The event list includes COBOL and Java events with class names, method names, and line numbers.
- The COBOL events use DWARF source information due to the TEST(SOURCE) compiler option.

Sample 3 (IDISJAV3): 64-bit Enterprise PL/I invoking Java in batch

The JCL in sample IDI.SIDISAM1(IDISJAV3) demonstrates how to use Fault Analyzer when 64-bit Enterprise PL/I invokes 64-bit Java™ in batch. It includes Java™ calling an Enterprise PL/I DLL that is located in a PDSE data set member.

The sample results expected in the Fault Analyzer report are essentially the same as the results in Sample 1 (IDISJAV1): Enterprise PL/I invoking Java in batch on page 268.

In this sample, the LANGX side file is used to display source code information, because the PL/I program was compiled with the NOTEST option.

Sample 4 (IDISJAV4): IMS Java batch processing sample

The JCL in sample IDI.SIDISAM1(IDISJAV4) demonstrates how to use Fault Analyzer with IMS™ Java™ batch processing (JBP) regions. In the sample, IMS™ calls a Java™ program, which in turn calls COBOL and assembler programs.

The sample uses the IMS™ installation verification program (IVP) program specification block (PSB). Ensure that the PSB is started before running the sample. If the PSB is stopped, abend 0456 occurs instead of the expected abend U0123.

These are the sample results expected in the Fault Analyzer report:

- The report includes information about the COBOL program, the assembler program, Java™, and IMS™.
- The synopsis section displays the COBOL source and variable information.
- In the event summary, the IMS™, Java™, COBOL, and assembler calls appear in a single overview.
- For the assembler event, source-code information is obtained from the assembler ADATA compiler option.

Sample 5 (IDISJAV5): C++ program invoking Java in batch

The JCL in sample IDI.SIDISAM1(IDISJAV5) demonstrates how to use Fault Analyzer when using a 31-bit or 64-bit C++ program to invoke Java™ in batch. It includes Java™ calling a C++ DLL located in a PDSE data set member.

For both the 31-bit and the 64-bit Snap.dump(), the Fault Analyzer report displays only Java™ information. Both include source code but no Java™ variable information, because the Java™ virtual machine (JVM) does not provide a mechanism to obtain variable information.

These are the sample results expected in the Fault Analyzer report for both the 31-bit and 64-bit C++ abend reports:

- The synopsis section includes the line of C++ source code where the abend occurred.
- The event summary includes both Java™ and C++ calls in a single view.
- In the C++ events, compiler listings are used for source code information, due to the NODEBUG compiler option.
- In the Java™ events, source code information is displayed along with stack frame reference information.

Sample 6 (IDISJAV6): Call a Java batch application with the Fault Analyzer wrapper utility

The JCL in sample IDI.SIDISAM1(IDISJAV6) demonstrates how to use the Fault Analyzer Java™ wrapper utility to invoke a Java™ class with a main() method within a try-catch block. Unhandled exceptions are automatically caught and fault entries are created using the Snap.dump() method.

When you use the wrapper utility to catch unhandled Java™ exceptions, only a limited amount of Java™ information is available to Fault Analyzer:

- The report overview includes details about the exception type and message.
- The event list includes Java™ events for the active thread, including class and methods names.
- In individual Java™ events, class-method variable information and stack-frame reference variable information are not available to Fault Analyzer, so they are not displayed.

See Java information reporting limitations on page 267 for additional information.

See Invoking the Fault Analyzer wrapper on page 422 for details about invoking the wrapper utility.

Example output from the Fault Analyzer wrapper utility

The wrapper utility reports the class being invoked and any command line arguments that were specified. For example:

```
Fault Analyzer Java Wrapper
Invoking class: com.example.JavaUnhandled
with args: '[pureJavaUnhandledException]'
When FA catches an unhandled exception, the FA wrapper displays the following
messages:
Fault Analyzer was invoked to handle a Java Throwable:
.
---------------------------------------
java.lang.ClassCastException: Cannot cast class java.util.concurrent.atomic.AtomicLong to
class java.at java.lang.Class.cast(Class.java:2614)
at com.example.JavaUnhandled.computeTheAnswer(JavaUnhandled.java:20)
at com.example.JavaUnhandled.pureJavaUnhandledException(JavaUnhandled.java:30)
```

```
at com.example.JavaUnhandled.main(JavaUnhandled.java:41)
---------------------------------------
Calling Snap.dump to create a fault entry:
DA.DCAT(F39992)
FA: Passing the exception to JVM...
...
```

Chapter 9. The Fault Analyzer report

This chapter describes the contents of the report that is written by Fault Analyzer.

General report information

The analysis report produced by batch reanalysis has the same structure as the real-time report. The interactive reanalysis report has a similar structure, however, you are able to select a particular section for viewing, rather than having to view the entire report in sequence. In addition, the interactive report provides specific information that is related to CICS® system abends or Java™ analysis, that is not included in the real-time or batch analysis reports. For further details, see Performing CICS system abend dump analysis on page 247.

The Fault Analyzer report is written to DDname IDIREPRT for real-time analysis, or to SYSPRINT for batch reanalysis.

Most significant abend code

If the analysis of a fault results in more than one abend code being presented, then Fault Analyzer always considers the first abend code (chronologically) the most significant. However, if the Language Environment® CEEWUCHA user condition handler is used (registered via the LE runtime option USRHDLR(CEEWUCHA)), then the final or last abend code is usually the one that provides the most detailed information about the error.

Open file record information

The records for open sequential files are shown in chronological order.

For input files, a 'previous—current—next' sequence is used with the current record being the record that was provided for application use prior to the fault.

For output files, "last record written" and "current record build area" descriptions are used.

Spanned record interpretation

When viewing the open file record information, where variable spanned records might be present, the user is required to reassemble the logical view of the record if a spanned record is involved. This requirement is because the buffers on which the record information is based might not still contain all of the segments for a particular spanned record. When spanned records are involved, the record information heading for each record contains “first segment”, “intermediate segment”, and “last segment” according to the data available and the structure of the particular spanned record (there might be zero to many intermediate segments).

Following are two real-time report extracts of open file information.

Figure 189: Sample real-time report extract of open input file information (Part 1 of 2) on page 273 shows file information and buffers for an input file:

Figure 189. Sample real-time report extract of open input file information (Part 1 of 2)

```
Open Files

File Name . . . . . . . . . : INDD
  Data Set Name . . . . . . : LJBERRY.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
```

Figure 190. Sample real-time report extract of open input file information (Part 2 of 2)

```
  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 191: Sample real-time report extract of open output file information on page 274 shows file information and buffers for an output file:

Figure 191. Sample real-time report extract of open output file information

```
Open Files

File Name . . . . . . . . . : OUTDD
  Data Set Name . . . . . . : LJBERRY.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                            *....            *
```

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 maintenance installed. Following this information is the Fault Analyzer copyright statement and a header for the report.

If performing batch reanalysis of a dump data set, or a fault entry created from interactive reanalysis of a dump data set, and the dump data set or fault entry contains CICS® system or Java™ environment information, then a note is added here. This note tells you that the better way to analyze the current fault is by using the appropriate ISPF interface reanalysis method. This approach is better 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> 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> E V E N T   S U M M A R Y
```

With each event entry in the list is also provided key information, such as module name, program name, and failing line or statement number. This information is a summary of the information that is provided in the details section for the event.

The following is provided for each entry in the list:

Event #

This section is a unique sequence number in ascending chronological order assigned to the event. The event number is used as reference to the detailed information for the event in the Event Details section of the report.

Event Type

The event type as one of the following:

Abend *abend-code*

An abend event occurs when system hardware or software terminates a program because it attempted to perform an invalid action, or because a valid action could not be performed for some reason. A system or user abend code generated by the operating system or other software is associated with the abend.

Multiple abends can be associated with a fault because an abend can be "trapped" by software and converted to an abend of another type.

AMODE64->31

An event representing the Language Environment switch from AMODE 64 to AMODE 31.

BALR

A BALR event occurs when control is transferred from one program to another via a branch and link mechanism and information about the status of the first program is not saved in a standard save area by the second program.

Branch

A branch event occurs when the Breaking Event Address Register (BEAR) hardware feature provides relevant information about the instruction that caused a wild branch.

Call

A call event occurs when control is transferred from one program to another via a branch and link mechanism and information about the status of the first program is saved in a standard save area by the second program. With link events, call events show an execution trail through the fault.

Fault Analyzer reconstructs the call events from the save-area information obtained at the time of the abend. There might be none, one, or many call events associated with each abend event for a fault.

Current PRB

A current Program Request Block (PRB) event.

Debug Tool

A z/OS® Debugger event.

EXEC CICS

A CICS EXEC CICS statement event.

EXEC DLI

A CICS EXEC DLI statement event.

EXEC SQL

A CICS EXEC SQL statement event.

Execute

An EXecute event summarizes the execution of an EX machine instruction that occurred prior to an abend event.

IDISNAP

An IDISNAP event.

Interrupt

An Interrupt Request Block (IRB) event.

Java™

A Java™ event. Usually, a call to a Java method.

LE Condition

An Language Environment® unhandled condition event.

Link

A link event occurs when control transfers from one program to another through a supervisor call mechanism, and information about the status of the first program is saved in a standard save area by the second program. With call events, link events show an execution trail through the fault.

Fault Analyzer reconstructs the link events from the save area information obtained at the time of the abend. There might be none, one, or many link events associated with each abend event for a fault.

ONCODE *code*

A PL/I 2.3 non-LE run-time ONCODE event. The ONCODE condition is shown.

PC *number*

A PC event occurs when a PC (Program Call) instruction is executed that causes control to be transferred to a system service.

SVC *number*

An SVC event occurs when an SVC instruction (other than a link) is executed that causes control to be transferred from an application program to a system service.

SVC events appear in the analysis when an abend occurs in a system service routine invoked by an application program. There can be at most one SVC event associated with each abend event for a fault.

Fail Point

If the event has been identified as the point of failure, then this identification is indicated by `*****` in this column.

Module Name

If available, the name of the load module that is associated with the event. Otherwise, `n/a`.

Program Name

If available, the name of the program or CSECT that is associated with the event. Otherwise, `n/a`.

EP Name

If available, the name of the entry point that is 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

Description

The following information is provided:

- If available, a brief description of the program or load module function.
- 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` is 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 events are shown, with an indication of the events which have been suppressed due to the maximum call depth having been exceeded.

The event details section

The event details section provides detailed information about each event. It is preceded by the heading:

```
<H1> E V E N T   D E T A I L S
```

The types of events included in the details section is subject to the Detail option in effect. Included in the detailed event section is also more information that is associated with the event, such as message descriptions (extracted by Fault Analyzer so that you do not need to look this up in a manual) and the contents of the program's working storage. When appropriate, you also find information such as abend code explanations and open file buffers here.

Source code information that is shown in the details section of the report includes up to 5 source lines or statements ahead of, and following, the source location for the event. If the event source location is within an expanded assembler macro, then the extra source statements are in addition to the statements caused by the macro expansion.

To change the default extra five lines or statements, use the Detail option (see Detail on page 591) or the Analysis Control user exit (see Analysis Control user exit on page 475).

If no matching compiler listing or side file was provided for the point-of-failure event, then the failing machine instruction is shown. In addition, up to 12 instructions ahead of, and 6 following, the failing instruction are provided. The following example listing is based on the IDIVPCOB COBOL installation verification program (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 IDISCBL1.

Load Module Name. . . . . . : SYS05291.T124505.RA000.IDIVPCOB.GOSET.H02(IDISCBL1)
  At Address. . . . . . . . : 16700D88
  Load Module Length. . . . : X'1278'
  Link-Edit Date and Time . : 2019/10/18  12:45:07

Program and Entry Point Name: IDISCBL1
```

```
  At Address. . . . . . . . : 16700D88 (Module IDISCBL1 offset X'0')
  Program Length. . . . . . : X'638'
  Program Language. . . . . : COBOL (Compiled using COBOL for OS/390 & VM V2 R2
                              M2 on 2019/10/18 at 12:45:06)

Machine Instruction . . . . : FD73D0B8D0A8  DP    184(8,R13),168(4,R13)
  At Address. . . . . . . . : 1670115C (Entry point IDISCBL1 offset X'3E2')
  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 user section

This section of the report is only present if one or more Formatting user exits were used to write information to the report.

The name of this section is by default "U S E R" in the batch report or "User" in the interactive reanalysis report, but can be changed by any Formatting user exit via the UFM.USEROPTIONTITLE data area field. It is preceded by the heading:

```
<H1>  U S E R
```

The AI analysis report

When IBM Fault Analyzer for z/OS Abend Intelligence is enabled, this section includes:

- **Executive Abend Summary:** Plain-language description of why the application abnormally ended.
- **Root Cause Analysis:** Technical correlation between failing offsets, storage registers, and application program logic.
- **Remediation and Code Modification:** Recommended code adjustments and sample source fixes to resolve the abend.

The content of the AI analysis report is determined by the third-party Large Language Model (LLM) in use.

If the background AI processing job has not yet completed when viewing the Fault Analyzer report, a status message is displayed.

The abend job information section

This section provides information about the abending job that is associated with the real-time invocation of Fault Analyzer or, in the case of reanalysis, the minidump or MVS™ dump analyzed. It is preceded by the heading:

```
<H1> A B E N D   J O B   I N F O R M A T I O N
```

The options section

This section is a list of the Fault Analyzer options that were in effect at the time of the analysis. It is preceded by the heading:

```
<H1> O P T I O N S
```

The epilog section

The epilog section consists of everything from the bottom of the options section until the end of the report.

For a real-time analysis report, information is provided about the invocation exit used and the approximate amount of above-the-line storage that is allocated during the analysis, followed by the fault ID assigned. A batch reanalysis report instead provides information about the fault ID and history file that is used.

The last line of the report provides information about the time and date when the report was created.

Sample reports

Fault Analyzer includes installation verification programs (IVPs) and sample reports produced by the IVPs. The IVPs are members of the IDI.SIDISAM1 data set, and the sample reports are members of the IDI.SIDIDOC1 data set. See the following table for the member names of the IVPs and sample reports.

Table 7. Member names of IVPs and sample reports

IVP	IVP member name in IDI.SIDISAM1	Sample report member name in IDI.SIDIDOC1
COBOL	IDIVPCOB	IDISRP01
PL/I	IDIVPPLE	IDISRP02
Assembler	IDIVPASM	IDISRP03
C DB2®	IDIVPDB2	IDISRP04
MQSeries®	The MQSeries® report was produced by performing reanalysis of a fault entry created for an MQSeries® abend.	IDISRP05
C	IDIVPC	IDISRP06

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

In addition to invoking Fault Analyzer interactively under TSO/ISPF, it is possible to access it through the following non-ISPF interfaces:

- Through an Eclipse plug-in
- Through a Microsoft Visual Studio (VS) Code extension client that works in conjunction with the Zowe Explorer for Visual Studio Code™
- Under CICS®
- Through an AI assistant using a workstation-based Model Context Protocol (MCP) server

Fault Analyzer plug-in for Eclipse

Fault Analyzer supports the IBM® Fault Analyzer plug-in for Eclipse. The plug-in is a workstation-based alternative to the FA ISPF interface. Used with the IBM Explorer for z/OS® or IBM Developer for z/OS Eclipse-based platforms, the plug-in provides a graphical interface to FA fault history files and views. With the plug-in, you can:

- Work with multiple fault history files and views from multiple systems.
- Browse fault entries that were created during real-time analysis of abending programs.
- Browse dump storage associated with a fault entry.
- Display the program source where a matching side file was found.

Information about the plug-in is available from the Help menu in the plug-in interface.

Refer to Installing the IBM Fault Analyzer plug-in for Eclipse on page 561 for information about installing this optional feature.

Fault Analyzer on VS Code

IBM Fault Analyzer for z/OS® is a host component that provides a Microsoft Visual Studio Code extension client.

The IBM Fault Analyzer for z/OS® Microsoft Visual Studio Code extension works in conjunction with Zowe Explorer for Visual Studio Code™. IBM Fault Analyzer for z/OS® helps developers analyze and fix application system failures.

When applications abnormally end (ABEMD), it is crucial to understand the root-cause. IBM Fault Analyzer for z/OS® can discover ABENDs faster, more confidently, and in less time by automatically harvesting real-time information of the ABEND and its environment at the time of failure.

For more detailed information about the IBM Fault Analyzer for z/OS® Microsoft Visual Studio Code extension, see the IBM Fault Analyzer for z/OS VS Code extension on Visual Studio Marketplace.

Refer to Installing IBM Fault Analyzer for z/OS on VS code on page 562 for information about installing this optional feature.

Performing interactive reanalysis under CICS®

Fault Analyzer uses a special component to display ISPF panels that can allow it to operate as a CICS® transaction to view history files and perform interactive reanalysis.

Note: The CICS interface is designed to function optimally with a terminal configured to support extended data streams, such as color and field underlining.

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 The Fault Analyzer ISPF interface on page 55), with the following restrictions and variations:

1. Batch reanalysis of the fault entry is not supported.
2. Functions which invoke ISPF EDIT are not supported:
    - Editing the options data set prior to interactive reanalysis.
    - Altering allocated data sets prior to interactive reanalysis.
    - EDIT a User Formatting EXEC from list of available EXECs.
    - EDIT of a DSECT from the DSECT list display.
3. Since this component is not running under TSO, no prefixing of data set names is performed. That is, where a data set name can be entered, for example a Fault History File or View, the data set name needs to be fully qualified with or without quotes.
4. ISPF profile changes made while in the Interactive Reanalysis Report, for example changing the location of the command line, might not be immediately reflected upon return to the Fault Entry List display. However, the profile changes are detected on the next invocation of the main CICS® transaction.

Refer to Enabling interactive reanalysis under CICS on page 563 for information about installing this interface.

Asynchronous AI analysis using the Model Context Protocol (MCP) server

You can connect Model Context Protocol (MCP) compliant AI assistants to Fault Analyzer to automate abend diagnosis through z/OSMF REST APIs.

The MCP server acts as a secure, workstation-based bridge between your AI assistant and z/OS. Running on Windows®, the server communicates with the host exclusively through z/OS Management Facility (z/OSMF) REST APIs over HTTPS. Because the server connects remotely using standard z/OSMF interfaces, you do not need to install additional software agents on your mainframe system. All host operations execute under your existing RACF user identity and permissions. Windows Credential Manager is the recommended authentication path for this scenario. See Configuring local authentication using Windows Credential Manager on page 565.

The five core diagnostic operations

Your AI assistant can chain five atomic operations to analyze abends autonomously.

The MCP server also exposes these operations as REST endpoints. You can call these operations from a script without an AI client if the server is run in HTTP mode. You can access the exposed REST API services through the Swagger UI sending HTTP requests directly to the REST API from you browser by connecting to the following URL: http://localhost:8080/swagger-ui.html.

The following are the five operations available:

- `find-history-datasets`: Searches the system catalog to list available Fault Analyzer history data sets.
- `search-history-file`: Invokes the IDIUTIL utility with the **LISTHF** statement to list and filter fault entries.
- `report-from-fault-id`: Executes interactive reanalysis on the host to retrieve the complete diagnostic report. To control report granularity and to optimize LLM token consumption, use the **detailLevel** settings as follows:

  ESSENTIAL

  Minimal token footprint. Includes:
  - synopsis
  - abend code
  - point-of-failure event

  MEDIUM

  This is the default setting. Standard report sections up to the point of failure.

  FULL

  Exhaustive details including: full stack trace, registers, and storage dumps.
- `get-job-spool`: Queries SDSF spool files directly to inspect job execution logs and DD statements.
- `extract-source-sidefile`: Reads debug side-files (LISTING, RAW_SOURCE, COBOL, LANGX, DWARF, SYSDEBUG) and extracts clean, numbered source code statements.

When connected, the AI assistant reads the diagnostic report, correlates the abend offsets with your application source code, and proposes code modifications to resolve the failure.

For workstation setup and client configuration instructions, see Configuring the Model Context Protocol (MCP) Server on page 564.

Part II. Fault Analyzer installation and administration

Chapter 11. Migrating from an earlier version of Fault Analyzer

Information about migrating from an earlier version of Fault Analyzer is provided in these topics.

See Maintaining Fault Analyzer on page 459 for information about the SMP/E installation and subsequent activation of the new version of Fault Analyzer.

The following information is applicable to all versions of Fault Analyzer.

LPA module compatibility

Fault Analyzer provides downward compatibility between load modules in the IDI.SIDIALPA data set. Hence, you can install a later version of Fault Analyzer, perform IPL with CLPA, and use these LPA modules with an earlier version of Fault Analyzer in LNKLST or STEPLIB.

All data sets in the STEPLIB concatenation must be APF-authorized for the Fault Analyzer data sets to retain their APF-authorization.

Sharing of history files across a sysplex

There are no issues with sharing of history files across a sysplex with a mix of supported levels of Fault Analyzer installed.

IDICHKI report

To ensure that important settings and exits are carried forward from an earlier version, you might want to create an IDICHKI report on the earlier version, and again after the later version has been installed. Compare the reports to identify any differences. For details, see Step 3: Verify the service level (optional) on page 460.

Migrating from V16.1 to V17.1

There are no specific actions required when migrating from V16.1 to V17.1.

Migrating from V15.1 to V16.1

This section provides information about changes to Fault Analyzer V17.1 that you should be aware of if migrating from V15.1.

If using the IDIOPTLM configuration-options module, ensure it is updated to include the PREVEXIT field and the required ending D'0' field. For details, see Customize Fault Analyzer by using an IDIOPTLM configuration-options module on page 337.

Migrating from V14.1 to V15.1

This section provides information about changes to Fault Analyzer version 15.1 that you should be aware of if migrating from version 14.1.

- Installation of the IDIXDCAP pre-dump exit is no longer supported via the IEAVTABX CSECT exit list. The IDIXDCAP must be installed as an IEAVTABX_EXIT dynamic exit. For details, see Installing the MVS change options/suppress dump exit IDIXDCAP on page 333.
- XDUMPs are enabled by default. For details, see Extended minidump data set (XDUMP) on page 53.
- Previously deprecated options have been removed:
  - HistCols
  - InteractiveExitPromptSeconds
  - RDZClient
  - UseIDISTime
- The IDIOPT1 SMP/E USERMOD (using a nonstandard LE parameter list separator character) has been deprecated. There is currently no replacement.
- The IDIOPT2 SMP/E USERMOD (suppressing IDIXDCAP real-time analysis if prior exit RC=8) has been deprecated. The replacement is to use the IDIOPTLM configuration-options module PREVEXIT options field. For details, see Suppressing IDIXDCAP real-time analysis if prior exit RC=8 (PREVEXIT) on page 342.
- Data set IDI.SIDILPA1 is no longer provided. If this data set exists from an earlier version of Fault Analyzer, then it should be deleted, including any entry for it in the LPALSTxx parmlib member. An IPL is recommended to remove this data set from LPA. Instead, use the SETPROG command to place module IDIDA in LPA, as described in Storage recommendations on page 296.

Migrating from V13.1 to V14.1

This section provides information about changes to Fault Analyzer version 14.1 that you should be aware of if migrating from version 13.1.

- Specify a suitable pattern for naming extended minidump data sets in the XDUMPDSN option in the IDIOPTLM configuration-options module. For details, see Changing the default extended minidump (XDUMP) data set name pattern (XDUMPDSN) on page 340.
- If you are using the IDIXDLOC function of the IDIXUFMT load module Formatting user exit to access large amounts of virtual storage there is a risk of exhausting all available storage, resulting in abnormal termination of the analysis. To prevent that from happening, change all usage of IDIXDLOC to IDIXXLOC instead. For details of the IDIXXLOC function, see IDIXXLOC – Locate dump storage using own buffering on page 558.

Migrating from V12.1 to V13.1

This section provides information about changes to Fault Analyzer version 13.1 that you should be aware of if migrating from version 12.1.

- The following Fault Analyzer SMP/E ++USERMODs are no longer available:
  IDILEDS
  IDISCNF
  IDISRFR
  IDISXCUM

These have been replaced by equivalent settings in the IDIOPTLM configuration-options module. For details, see Customize Fault Analyzer by using an IDIOPTLM configuration-options module on page 337.

- If the Fault Analyzer IDIOPTLM configuration-options module is currently used to specify an alternative data set name for IDICNFxx, the ADFz Common Components IPVOPTLM configuration-options module might also be required. Otherwise, security server access violations might occur. For more information, see Specifying an alternative parmlib data set for IDICNFxx (CNFDSN) on page 339.
- The Fault Analyzer Locale option *locale-name* can now alternatively be specified in the ADFz Common Components IPVCNF00 parmlib member. For more information, see Locale on page 614.
- The HistCols and InteractiveExitPromptSeconds options have been deprecated and replaced by the FAISPFopts option. For details, see FAISPFopts on page 603.
- The MVS™ post-dump exit IDIXTSEL is no longer an optional installation item. It is required to support additional Fault Analyzer functions, such as Java™ fault capture. For details, see Checklist for installing and customizing Fault Analyzer on page 291.
- Java™ dump capture requires access to the IDI_SDUMP_ACCESS XFACILIT profile. For details, see Invoking Fault Analyzer from a Java try-catch block on page 418.
- It is a requirement for CICS® open TCB users to include the Fault Analyzer IDIPLT program in the CICS® startup PLT. For details, see Adding the required programs to the startup PLT on page 398.
- The UseIDISTime option has been deprecated and is now always in effect. For details, see UseIDISTime.
- The Fault Analyzer plug-in for Eclipse is no longer provided as member IDIGUIP in data set IDI.SIDIDOC2. However, the plug-in is still available via the web, as explained in Installing the IBM Fault Analyzer plug-in for Eclipse on page 562.
- Changes have been made to the IDIXFXIT user exit parameter list:
  - The sixth parameter was previously a pointer to the fault entry ID for which the exit was invoked. Now, the sixth parameter is the address of an HD segment data area, which includes not only the fault entry ID, but also various other information that is related to the fault entry.
  - The security server user and default group ID, provided as the second and third parameters, are now always those of the exit caller, whereas previously they were at times obtained from the fault entry itself. Note that the HD segment data area now pointed to by the sixth parameter includes the fault entry creator security server user and default group ID information, when available.

  For details, see Using the IDIXFXIT user exit on page 366.
- The current value in ENV.VERSION has changed to 0005. This is due to ENV.FORMATTING_EXIT having changed from always specifying 'C' for a CICS® transaction fault, to now specifying 'C' if Fault Analyzer was invoked via the XPCABND exit or 'D' if invoked via the XDUREQ exit.

Migrating from V11.1 to V12.1

This section provides information about changes to Fault Analyzer version 12.1 that you should be aware of if migrating from version 11.1.

- ADFz Common Components must be installed in order for Fault Analyzer to provide source level support when using compiler listings or SYSADATA files.
- As BookManager® softcopy books are no longer shipped with Fault Analyzer, the **Help > Fault Analyzer User's Guide and Reference** action bar 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 IDI.SIDIAUT2 must be added as STEPLIB in the IDIS subsystem start-up JCL. For details, see Starting the IDIS subsystem on page 320.
- Users of Message and Abend Code Explanation load module user exits should note that the offset to the XPL.ABEND_CODE field has changed.
- In the past, prompting for missing compiler listings or side files during interactive reanalysis depended on whether any compiler listing or side file data sets had been provided to Fault Analyzer, implicitly or explicitly, by the time when these were required during the reanalysis of a fault entry. As a result, prompting could occur for some fault entries, but not for others.

  With the user-specified option now available to control prompting (see Interactive reanalysis options on page 157), it is easier to ensure a consistent behavior. If users have not yet set this option explicitly, then the default setting can be controlled by one of these actions:

  - Specify one or more compiler listing or side file data sets in the IDICNFxx parmlib member (in which case the default is to prompt).
  - Ensure that no such data sets are specified in the IDICNFxx parmlib member (in which case the default is to not prompt).

Migrating from V9.1 to V10.1

This section provides information about changes to Fault Analyzer version 10.1 that you should be aware of if migrating from version 9.1.

- The IDIJ subsystem is no longer used.

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. Real-time analysis on page 32 and The Fault Analyzer ISPF interface on page 55 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 IDICNFxx.

As part of the analysis process, Fault Analyzer attempts to find compiler listings or side files. Providing compiler listings or Fault Analyzer side files on page 374 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.

Notes:

- Copy all members of data set IDI.SIDISAM1 to another data set before proceeding, and make all changes to the copies only.
- You can use the IDICHKI utility to check the state of the Fault Analyzer installation or to query service information. See Step 3: Verify the service level (optional) on page 460.

1. **Make Fault Analyzer modules available via LINKLIST and LPA**

   For details, see Making Fault Analyzer modules available on page 306.

2. **Allocate a history file**

   Although multiple history files might eventually be used at your site, a single history file is sufficient to verify the installation of Fault Analyzer.

There are no restrictions on the name of the history file, but the default name searched for by Fault Analyzer is IDI.HIST. If a different name is used, then the IDICNFxx parmlib member is used to provide the name through the DataSets option. You review the IDICNFxx 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 Setting up history files on page 343, 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 IDICNFxx parmlib member**

   For details, see Setting and changing default options for the site on page 369.

   Ensure that a

   ```
   DataSets(IDIHIST(dsn))
   ```

   option is included if you allocated a history file with a name other than IDI.HIST in step 2 on page 291.

   Likewise, if Fault Analyzer was installed using a high-level qualifier other than IDI, use the DataSets option 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 308.

5. **Install the MVS™ change options/suppress dump exit IDIXDCAP**

   For details, see Installing the MVS change options/suppress dump exit IDIXDCAP on page 333.

   Information about the characteristics of this exit are provided in Exits for invoking Fault Analyzer on page 298.

   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 exits IDIXCEE and IDIXCE64**

   For details, see Installing the Language Environment abnormal termination exits IDIXCEE and IDIXCE64 on page 333.

   For information to help you determine the applicability of this exit at your site, see Exits for invoking Fault Analyzer on page 298 and Language Environment options required for invocation of Fault Analyzer on page 303.

7. **Install the SVC dump registration exit IDIXTSEL**

   For details, see Installing the MVS post-dump exit IDIXTSEL on page 412.

8. **Customize the CICS® environment**

   This step is only applicable if you are using CICS®.

   For details, see Customizing the CICS environment on page 396.

9. **Customize the DB2® environment**

   This step is only applicable if you are using DB2®.

   For details, see Customizing the DB2 environment on page 415.

   Create the DB2® table index discussed in Improving Fault Analyzer DB2 performance on page 415; otherwise, severe Fault Analyzer performance degradation might result when accessing DB2® catalog information.

10. **Customize the IMS™ environment**

    This step is only applicable if you are using IMS™.

    For details, see Customizing the IMS environment on page 417.

11. **Customize the Java™ environment**

    This step is only applicable if you are using Java.

    For details, see Customizing the Java environment on page 418.

12. **Customize for ISPF**

    For details, see Modifying your ISPF environment on page 327.

13. **Start the Fault Analyzer IDIS subsystem**

    For details, see Using the Fault Analyzer IDIS subsystem on page 318.

14. **Add entry in IFAPRDxx parmlib member**

    For details, see Registering Fault Analyzer in the IFAPRDxx parmlib member on page 316.

15. **Review and optionally change the default extended minidump (XDUMP) data set name pattern using an IDIOPTLM configuration-options module.**

    For details, see Changing the default extended minidump (XDUMP) data set name pattern (XDUMPDSN) on page 340.

Optional installation steps

1. **Add BPX security server program control profile for Fault Analyzer programs**

   This step is only required if program control has been activated for your installation.

   For details, see Defining program control access to Fault Analyzer programs on page 307.

2. **Install USERMOD IDISPLI or IDISPLIA to enable implicit Fault Analyzer invocation from PL/I V2R3 applications**

   For details, see Enabling implicit Fault Analyzer invocation from PL/I V2R3 applications (++IDISPLI/++IDISPLIA) on page 335.

3. **Change the default recovery fault recording IEATDUMP data set name pattern using an IDIOPTLM configuration-options module**

   For details, see Changing the default recovery fault recording IEATDUMP data set name (RFRDSN) on page 339.

4. **Define XFACILIT resource classes to manage recovery fault recording data sets.**

   For details, see Managing recovery fault recording data set access on page 309.

5. **Change the default extended minidump (XDUMP) data set name pattern using an IDIOPTLM configuration-options module**

   For details, see Changing the default extended minidump (XDUMP) data set name pattern (XDUMPDSN) on page 340.

6. **Define XFACILIT resource class to manage XDUMP data sets**

   For details, see Managing XDUMP data set access on page 313.

7. **Customize for Japanese language support**

   This step is only required if the Japanese feature of Fault Analyzer is installed.

   For details, see Customizing the Fault Analyzer Japanese feature on page 426.

8. **Install optional non-ISPF interfaces to access Fault Analyzer history files and to run an automated host-based reanalysis pipeline**

   To enable workstation-based AI assistant access, verify Java 21 on your Windows® system, ensure your z/OSMF REST API endpoint is active over HTTPS, and configure user RACF credentials in Windows Credential Manager.

   IBM Fault Analyzer for z/OS Abend Intelligence runs automated, real-time, host-based renalysis pipeline on z/OS,

   For details, see Asynchronous AI analysis using the Model Context Protocol (MCP) server on page 284 and Installing and configuring IBM Fault Analyzer for z/OS Abend Intelligence on page 567.

9. **Grant history file administrator authorization for change of settings via ISPF interface**

   For details, see Restricting change of history file settings on page 307.

10. **Review the chapter "Quick start guide for compiling and assembling programs for use with the ADFz family of products" in *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*** for information about steps required to prepare your programs for use with ADFz family of products.
11. **Set SLIP traps to capture documentation for selected Fault Analyzer error messages**

    For details, see Setting Fault Analyzer SLIP traps on page 305.

12. Enable IBM Fault Analyzer for z/OS Abend Intelligence.

    For details, see Installing and configuring IBM Fault Analyzer for z/OS Abend Intelligence on page 567.

Additional customization can optionally be performed using user exits, as described in Customizing Fault Analyzer by using user exits on page 464. However, no user exits are required for Fault Analyzer to run.

Installation verification

1. **Perform assembler IVP**

   For details, see Verifying the use of Fault Analyzer with assembler on page 428.

2. **Perform COBOL IVP**

   Only perform this step if COBOL is installed at your site.

   For details, see Verifying the use of Fault Analyzer with COBOL on page 429.

3. **Perform PL/I IVP**

   Only perform this step if PL/I is installed at your site.

   For details, see Verifying the use of Fault Analyzer with PL/I on page 430.

4. **Perform IDIXCEE Language Environment® exit IVP**

   For details, see Verifying the IDIXCEE Language Environment exit enablement on page 432.

5. **Perform CICS® IVP**

   Only perform this step if CICS® is installed at your site.

   For details, see Verifying Fault Analyzer customization under CICS on page 432.

6. **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 434.

7. **Perform ISPF IVP**

   For details, see Verifying the use of Fault Analyzer through ISPF on page 438.

Library names after you finish installing

The following data sets should exist after you have completed the SMP/E APPLY of Fault Analyzer:

Data Set Name	Containing
IDI.SIDIALPA	Load modules that must be made available from LPA.
IDI.SIDIAUTH	Authorized load modules that must be made available from LINKLIST.
IDI.SIDIDOC1	Message and abend code explanation override files, sample reports, and Fault Analyzer User's Guide and Reference PDF.

Data Set Name	Containing
IDI.SIDIDOC2	Message and abend code explanation repository input data. The IDI.SIDIDOC2 data set is input only to the IDISVENU job, described in Setting up the message and abend code explanation repository on page 308. The IDISVENU job allocates and initializes the message and abend code repository.
IDI.SIDIEXEC	REXX execs.
IDI.SIDIMAPS	Control block maps.
IDI.SIDIMLIB	ISPF message members.
IDI.SIDIPLIB	ISPF panels.
IDI.SIDISAM1	Softcopy samples and installation jobs.
IDI.SIDISLIB	ISPF skeletons.
IDI.SIDITLIB	ISPF tables.

If the Japanese feature of Fault Analyzer is installed, the following extra data sets should exist:

Data Set Name	Containing
IDI.SIDIDJPN	Japanese message and abend code explanation override files, sample reports, and Fault Analyzer User's Guide and Reference PDF.
IDI.SIDIMJPN	Japanese ISPF message members.
IDI.SIDIPJPN	Japanese ISPF panels.
IDI.SIDISJPN	Japanese ISPF skeletons.
IDI.SIDITJPN	Japanese ISPF tables.
IDI.SIDIXJPN	Japanese softcopy samples and installation jobs.

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 **minimum** requirements for the 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 37 megabytes above-the-line (31-bit) storage for CICS® transactions.
- A minimum of 35 megabytes above-the-line (31-bit) storage for programs other than CICS transactions.

Depending on the type of fault being analyzed, and the environment in which this fault occurs, more storage might be required.

The storage requirements under CICS are for MVS GETMAIN-managed storage, not CICS DSA-managed storage. So, to increase below-the-line MVS GETMAIN-managed storage, you would need to decrease CICS below-the-line DSA-managed storage (and similarly for above-the-line storage).

Information about the actual storage used by Fault Analyzer is available at the end of the real-time analysis report. However, the amount of storage provided in the report accounts for the explicit allocations performed by Fault Analyzer only and does not include, for example, Language Environment® heap and stack storage or storage used for load modules.

In post-abend situations, where the minidump or associated MVS dump data set 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 15 megabytes, reducing the storage requirement for a typical non-CICS program to around 12 megabytes.

If the necessary below-the-line (24-bit) size is not available, message IDI0086E on page 714 is issued and processing terminates.

If the necessary above-the-line (31-bit) size is not available prior to commencing analysis, message IDI0055E on page 707 is issued and processing terminates. Message IDI0087I on page 714 might also 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 module IDIDA and IPVLANGX. To place these modules in LPA, and save approximately 16 megabytes above-the-line (31-bit) storage, issue the following operator command:

```
SETPROG LPA,ADD,MOD=(IDIDA,IPVLANGX),DSN=LNKLST
```

The above SETPROG command can be added to the COMMNDxx parmlib member to persist across an IPL.

Note: Ensure ECSA size is sufficient to accommodate the additional storage required by the SETPROG LPA,ADD command.

In the case of IPVLANGX, alternatively add the IPV.SIPVLPA1 data set to the LPALSTxx parmlib member (IDIDA is not in IPV.SIPVLPA1 and would need to be added via SETPROG regardless).

Note:

- If Fault Analyzer modules are loaded into LPA, then it is important that step Step 3: Verify the service level (optional) on page 460 is performed after applying any Fault Analyzer maintenance. Failure to perform this step following the installation of maintenance prevents the update of Fault Analyzer LPA modules. Because not all Fault Analyzer modules are in LPA, the result can be a mismatch between the old and the new code, which might lead to undefined behavior.
- 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.
- IPV.SIPVLPA1 and IPVLANGX are installed as part of the installation of ADFz Common Components.

The MVS IEFUSI exit can be used as a general way to provide a larger region size if JCL change is not practical for all jobs. A sample IEFUSI exit is provided as member IDISUSI in the IDI.SIDISAM1 data set.

The exit increases the region size of all jobs by 16 megabytes. Refer to the comments in the sample for details about how to install the exit.

If storage is exhausted during analysis, message IDI0005S on page 696 is issued.

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 invoke Fault Analyzer for real-time analysis when an abend other than a CICS® transaction abend occurs (for example, batch and IMS™).

MVS™ IEAVTABX change options/suppress dump exit IDIXDCAP

- This exit can be used with Language Environment®-based and non-Language Environment-based batch application programs. IDIXDCAP is installed as an IEAVTABX_EXIT dynamic exit (z/OS 2.2 or later is required) and is invoked for all abends requesting an ABDUMP, regardless of whether the job step has allocated a SYSMDUMP, SYSUDUMP, or SYSABEND DDname. Note that IMS pseudoabends that do not request an ABDUMP, will not invoke Fault Analyzer through this exit.

The use of SLIP with ACTION=NODUMP (for example, a CANCEL command resulting in an Sx22 abend, for which most MVS™ systems have a matching SLIP TRAP) might not prevent Fault Analyzer from being invoked through the IEAVTABX_EXIT. To prevent Fault Analyzer invocation, ensure that the IDICNFxx PARMLIB configuration member includes an Exclude option specification with a list of appropriate abend codes for your installation. A sample list of standard abend codes are included in the sample IDICNFxx parmlib member provided as member IDICNF00 in data set IDI.SIDISAM1.

- Reanalysis of faults that were captured using this exit can be performed if a minidump was written.
- LE-enabled abends need to run with TERMTHDACT, specifying the suboption UATRACE, UADUMP, UAONLY, or UAIMM, in the LE options so that LE calls a system dump to activate this exit. All other TERMTHDACT suboption settings skip the IEAVTABX exit invocation and invoke the CEEEXTAN exit (described below) instead.
- This exit can extract WTO console messages related to the abending job from the master trace table and include these messages in the analysis report.

For information about installing this exit, see Installing the MVS change options/suppress dump exit IDIXDCAP on page 333.

Batch Language Environment (LE) abnormal termination CEEEXTAN CSECT exit IDIXCEE and CELQXTAN CSECT exit IDIXCE64

- These exits are effective only with Language Environment®-based batch (non-CICS) application programs.
- There is no requirement for a JCL SYSMDUMP DD statement to be allocated for these exits to be invoked.
- Reanalysis of faults that were captured using these exits can be performed if a minidump was written.
- Permits instance-specific LE message inserts to be obtained and included in the analysis report.
- These exits can extract WTO console messages related to the abending job from the master trace table and include these messages in the analysis report.
- **64-bit LE applications:** Because the MVS IEAVTABX change options/suppress dump exit is never invoked for LE-based AMODE64 batch application programs, the IDIXCE64 exit is required in that environment.
- **31-bit LE applications:** If the LE option TERMTHDACT is used with the UATRACE, UADUMP, UAONLY, or UAIMM suboption, then the MVS™ IEAVTABX change options/suppress dump exit is invoked instead of the IDIXCEE exit. In all other cases, Fault Analyzer will only be invoked for LE-based AMODE31 applications if the IDIXCEE exit is installed.
- If you run both AMODE31 and AMODE64 LE-based applications, you must install both the IDIXCEE and IDIXCE64 exits.

For information about installing this exit, see Installing the Language Environment abnormal termination exits IDIXCEE and IDIXCE64 on page 333.

Both MVS IDIXDCAP and Batch LE IDIXCEE exits are installed

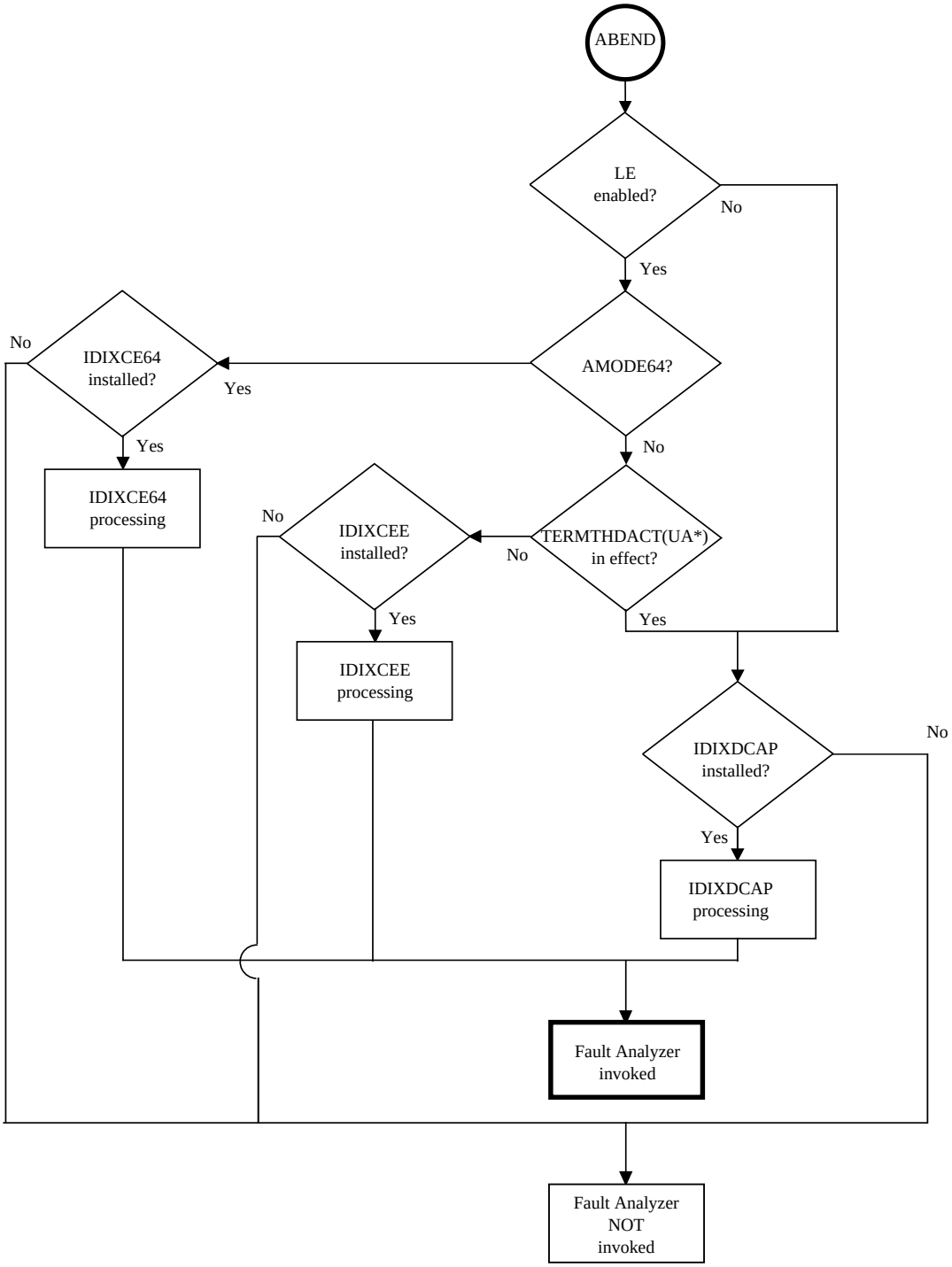
If both the batch Language Environment® abnormal termination exit IDIXCEE and the MVS™ change options/suppress dump exit IDIXDCAP are installed, then the IDIXDCAP exit intercepts the abend instead of the LE exit if one of the following LE options is in effect:

- TERMTHDACT(UATRACE)
- TERMTHDACT(UADUMP)
- TERMTHDACT(UAONLY)
- TERMTHDACT(UAIMM)

Summary of exit usage

Figure 192: Summary of Fault Analyzer non-CICS (batch) invocation exit usage on page 301 shows the exit used to invoke Fault Analyzer, depending on execution environment and options in effect.

Figure 192. Summary of Fault Analyzer non-CICS (batch) invocation exit usage



Invocation for CICS transaction abends

The following exits all invoke Fault Analyzer for real-time analysis when a CICS® transaction abend occurs.

CICS XPCABND and XDUREQ global user exit IDIXCX53

Characteristics:

- This exit is provided to invoke Fault Analyzer for CICS® transaction abend analysis.

  All transaction abends can be captured using this exit, except for U1xxx or U4xxx-type abends in Language Environment® based applications. These transaction abend types can be handled by also installing the CICS LE abnormal termination CEECXTAN CSECT exit, IDIXCCEE, described below.
- There is no requirement for a JCL SYSMDUMP DD statement to be allocated for this exit to be invoked.
- Reanalysis of faults that were captured using this exit can be performed if a minidump was written.

For information about installation of this exit, refer to Customizing the CICS environment on page 396.

CICS LE abnormal termination CEECXTAN CSECT exit IDIXCCEE

Characteristics:

- This exit is only effective with Language Environment® based CICS® application programs.
- There is no requirement for a JCL SYSMDUMP DD statement to be allocated for this exit to be invoked.
- Reanalysis of faults that were captured using this exit can be performed if a minidump was written.
- The LE option TERMTHDACT does not affect the invocation of this exit.

For information about installation of this exit, refer to Configuring Language Environment for CICS to invoke Fault Analyzer on page 397.

SVC dump registration

An exit is provided with Fault Analyzer for SVC dump registration into a history file.

MVS™ post-dump IEAVTSEL CSECT exit IDIXTSEL

Characteristics:

- This exit is invoked whenever an SVC dump is written by the DUMPSRV address space.
- The use of this exit requires the Fault Analyzer IDIS subsystem to be active. For information on this, see Using the Fault Analyzer IDIS subsystem on page 318.
- 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 412.

Language Environment options required for invocation of Fault Analyzer

The need for specific Language Environment (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 for z/OS® to be invoked for an abend.

Note:

- If the IDIXCEE exit is not installed, then abends in the LE-based 31-bit applications are only captured if the IDIXDCAP exit is installed, and the LE TERMTHDACT option with any of the UA* values (such as, UATRACE or UADUMP) is in effect.
- If both the IDIXCEE and IDIXDCAP exits are installed, then Fault Analyzer for z/OS® resolves the processing to perform only one analysis of a fault.
- To invoke Fault Analyzer for abends in LE-based 64-bit applications, the IDIXCE64 exit must be installed, regardless of LE options used or whether the IDIXDCAP exit is installed.

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 410.

Running Fault Analyzer with similar third-party products

In general, Fault Analyzer works with other similar third-party products without problems, with the possible exception being Language Environment® batch jobs. Fault Analyzer uses the Language Environment® CEEEXTAN facility (IDIXCEE), as does potentially other similar third-party products. If more than one exit is specified in the CEEEXTAN list, then Fault Analyzer should be the first exit specified.

The analysis of LE jobs by Fault Analyzer can also be done using the IEAVTABX MVS™ change options/suppress dump exit (IDIXDCAP). Ensure that the LE options include the TERMTHDACT option, with the UATRACE, UADUMP, UAONLY, or UAIMM suboption, to make LE request an MVS™ dump for an abend. This way, the third-party product can use the CEEEXTAN exit while Fault Analyzer can run from the IEAVTABX MVS™ change options/suppress dump exit. The Fault Analyzer IDIXDCAP exit analyzes LE abends exactly the same way as the IDIXCEE exit. This similarity is because the exits do not actually do the analysis, they simply provide the method of invoking Fault Analyzer. It is the same Fault Analyzer analysis engine that runs for all Fault Analyzer exits, including CICS®.

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 IDICNFxx 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.

The Fault Analyzer pre-abend (XPCABND) exit affects the CICS pre-transaction dump global user exit (XDUREQ). When invoked through the XPCABND exit, Fault Analyzer suppresses transaction dumps by default, and CICS does not invoke XDUREQ exit programs when transaction dumps are suppressed. If you require CICS to invoke the XDUREQ exit, use the Fault Analyzer RetainCICSdump(ALL) option.

MVS dump data set size

With Fault Analyzer installed, you expect an increase in the size of any MVS™ system dump taken. It might be necessary to review your dump data set allocation size parameters.

Application-handled error conditions

You can write application error handlers to completely suppress any indications of an abend, or on their completion, allow abend processing to continue.

Generally, abend processing is allowed to continue after the error handler has completed its task. However, if error handlers for application programs are not letting normal abend termination occur, and you want to invoke Fault Analyzer for such applications, then it might be necessary to either disable the application error handling, or add a call to IDISNAP (for details, see Using the program SNAP interface (IDISNAP) on page 37).

An example of an application error handling routine that does not invoke Fault Analyzer is a PL/I "ON ERROR" block, that either calls PLIDUMP with the 'S' option, or issues STOP.

A PL/I USERMOD is available to always invoke Fault Analyzer when calling PLIDUMP, even if using the 'S' option—for details, see Always invoking Fault Analyzer from PL/I PLIDUMP (++IDISPDM) on page 335.

Setting Fault Analyzer SLIP traps

As documentation for certain situations where Fault Analyzer issues an error message, an MVS™ dump is often required by IBM® Support.

In most cases, Fault Analyzer automatically writes a Recovery Fault Recording (RFR) dump, which serves two purposes:

- To enable reanalysis of the RFR fault entry, effectively making it transparent to the user that an error has occurred.
- To provide IBM Support with information about the error, if suspected of being a Fault Analyzer defect.

If such RFR dumps are being written (which requires the Fault Analyzer IDIS subsystem to be started and the IDIXTSEL exit installed - see Recovery fault recording on page 51 for more information), then there is generally no need to set SLIP traps for message IDs other than those marked with '*' below.

The following shows a sample SLIP trap to capture an SVC dump in case a particular Fault Analyzer message is issued:

```
SL SET,ID=xxxx,MSGID=zzzzzzzz,ACTION=SVCD,END
```

where `xxxx` is a unique SLIP trap identifier (for example, F047) `zzzzzzzz` is the message ID.

Refer to "MVS™ System Commands" for the complete syntax of the SLIP command and for additional parameters that might be considered, such as MATCHLIM.

SLIP traps for the following message IDs should be considered:

IDI0047S on page 705
IDI0092S on page 715
IDI0105S on page 718
IDI0123S on page 722 *
IDI0144E on page 727 *
IDI0168E on page 733 *

* These message IDs are not subject to Recovery Fault Recording (RFR) processing and MVS™ dumps are therefore never written automatically by Fault Analyzer for these.

Chapter 13. Customizing the operating environment for Fault Analyzer

This chapter provides information about customizing the operating environment required to run Fault Analyzer.

Making Fault Analyzer modules available

To make Fault Analyzer modules available, the following steps must be performed following the SMP/E installation of Fault Analyzer.

1. **Authorize IDI.SIDIAUTH and add to the LINKLIST**

   Fault Analyzer modules that require APF-authorization are placed in target library, IDI.SIDIAUTH. You must APF-authorize IDI.SIDIAUTH by adding it to the IEAAPF*xx* or PROG*xx* member (if available on your system) in SYS1.PARMLIB. IDI.SIDIAUTH must also reside in the LINKLIST. Add IDI.SIDIAUTH to your concatenated LINKLIST by using the LNKLST*xx* or PROG*xx* member in your SYS1.PARMLIB.

   Note: MVS requires that data sets in LINKLIST either be in the master catalog or specified with the volume serial number where the data set resides.

   The load modules IDIDA, IDIPDDIR, and IDIUTIL in IDI.SIDIAUTH do not execute correctly unless they are loaded from an APF-authorized library.

2. **Add IPV.SIPVMODA to the LINKLIST**

   Data set IPV.SIPVMODA, which is created as part of the ADFz Common Components installation, must be added to LINKLIST for Fault Analyzer to provide source level support when using compiler listings or SYSADATA files.

   Note: MVS™ requires that data sets in LINKLIST either be in the master catalog or specified with the volume serial number where the data set resides.

   ADFz Common Components also provides a data set IPV.SIPVMOD1, which is not required by Fault Analyzer.

3. **Add IDI.SIDIALPA to the LPALIST**

   Fault Analyzer modules that must be loaded into the LPA reside in the target library, IDI.SIDIALPA. Add IDI.SIDIALPA to your concatenated LPALIST via the LPALST*xx* 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 LPALST*xx* change must be implemented by performing an IPL with CLPA (step 5 on page 307), or by doing the following if an IPL cannot be scheduled:

   a. Issue the operator command:

      ```
      SETPROG LPA,ADD,MOD=(IDICSVCR, IDICSRBL),DSN=IDI.SIDIALPA
      ```

Note: Ensure ECSA size is sufficient to accommodate the additional storage required by the SETPROG LPA,ADD command.

b. Submit a batch job containing the following EXEC JCL statement to dynamically install the Fault Analyzer SVC 109 ESR code 53 on your system.

```
// EXEC PGM=IDICZSVC
```

The *xxx* variable specifies the SVC table entry index number to use if entry 53 is unavailable. It can be any value from 200 to 255.

If no PARM value is specified, IDICZSVC will dynamically locate a free ESR code.

If the IDI.SIDIALPA modules are not added to LPA via an IPL with CLPA (step 5 on page 307) or the above dynamic update using SETPROG and IDICZSVC, then errors are expected, including abend S16D.

4. **Add Fault Analyzer and IPVLANGX to LPA**

   This is an optional but recommended step to provide additional storage in the abending region for Fault Analyzer real-time analysis.

   For more information see, Storage recommendations on page 296.

5. **Perform IPL with CLPA**

   If any of the previous steps were not successfully implemented dynamically, then it is necessary to IPL your system with CLPA.

Defining program control access to Fault Analyzer programs

If security server program control is activated for your installation, for example due to z/OS Unix System Services (BPX) server requirements, then a PROGRAM class profile that identifies all Fault Analyzer programs as being controlled programs can be defined using the command:

```
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 BPXP014I, being issued, if an abend occurs in a z/OS Unix System Services server region.

Additionally, messages CSV024I or IEF450I might be issued, possibly indicating abend S306-30.

Restricting change of history file settings

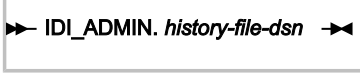
By default, all users with UPDATE access to a history file can change the history file prefix or the minimum or maximum number of fault entries.

Either of the following methods can be used:

- The action-bar option **File > Change Fault History File Settings** in the Fault Analyzer ISPF interface. For details, see Change fault history file settings on page 88.
- The SetFaultPrefix, SetMaxFaultEntries, and SetMinFaultEntries batch utility functions. For details, see IDIUTIL control statements on page 441.

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.

Figure 193. Syntax



where *history-file-dsn* is the fully qualified data set name of the history file.

To change history file settings after the IDI_ADMIN XFACILIT profile is 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, through either normal security server data set profiles, or through XFACILIT (for details, see Managing history file fault entry access on page 363).

The following sample RACF® commands define an IDI_ADMIN XFACILIT profile for history file MY.HIST and grant Fault Analyzer administrator authorization to change settings for users who are members of group PAYROLL:

```
RDEFINE XFACILIT IDI_ADMIN.MY.HIST UACC(NONE)
PERMIT IDI_ADMIN.MY.HIST CLASS(XFACILIT) ID(PAYROLL) ACCESS(UPDATE)
```

Setting up the message and abend code explanation repository

To enable Fault Analyzer message and abend code explanations, a VSAM cluster must be defined and initialized. This definition and initialization is done by submitting the sample job IDISVENU in the IDI.SIDISAM1 data set. If a name, other than the default name of IDISVENU is used, then it is necessary to specify the DataSets option with the IDIVSENU suboption containing the name to be used. For details, see DataSets on page 581.

In installations that run different versions of both z/OS® and Fault Analyzer, consider using the &SYSR1 substitution symbol to specify the IDISVENU data set. For example:

```
DataSets(IDIVSENU(IDI.IDIVSENU.&SYSR1.))
```

Then allocate these VSAM data sets with the SYSRES VOLSER on the respective SYSRES volumes. That way, the IDICNFxx parmlib member DataSets specification of IDIVSENU is always correct, no matter which SYSRES volume is being IPL'ed from.

Ensure that all users have READ access to the data sets defaulted to, or specified using, the IDIDOC and IDIVSENU DDnames. Additionally, due to the occasional automatic updating of this file by the IDIS subsystem with corrections or new explanations, the IDIS subsystem should be granted UPDATE access to the IDISVENU VSAM KSDS data set.

For the Japanese feature of Fault Analyzer, an extra repository must be set up using sample job IDISVJPN in theIDI.SIDIXJPN data set.

If a name, other than the default name of IDI.IDIVSJPN is used, then it is necessary to use the IDIVSJPN suboption of the DataSets option to specify the name to be used. For details, see DataSets on page 581.

Ensure that all users have READ access to the data set defaulted to, or specified using, the IDIVSJPN DDname. Additionally, due to the occasional automatic updating of this file by the IDIS subsystem with corrections or new explanations, the IDIS subsystem should be granted UPDATE access to the IDIVSJPN VSAM KSDS data set.

Managing recovery fault recording data set access

If Fault Analyzer fails to record an application abend due to exception conditions, such as insufficient virtual storage or an abend in Fault Analyzer, it then attempts to capture the abend situation with an MVS SDUMP (SVC dump) or MVS IEATDUMP (transaction dump or TDUMP). This process creates a separate dump data set, which is linked with the recovery fault recording history file fault entry. With this recovery process, Fault Analyzer is normally able to provide interactive reanalysis of the initial application abend in spite of the exception condition Fault Analyzer encountered during the capture. The linked SDUMP or TDUMP provides the storage data that would normally have been recorded in the 'minidump' section of the fault entry if the capture had not had the exception.

In addition to normal data set access, Fault Analyzer optionally permits control of its RFR TDUMP data sets with an XFACILIT security profile (for details, see Using the XFACILIT resource class for TDUMP RFR data sets on page 311).

Fault Analyzer RFR SDUMP data sets are only created if access is granted via the associated XFACILIT profile (for details, see Using the XFACILIT resource class for SDUMP RFR data sets on page 310).

Fault Analyzer tries to use SDUMP as the preferred dump type, and only if the necessary SDUMP access authorization is not available for the abending user ID, is the TDUMP access authorization checked. The dump process used by SDUMP is faster than the TDUMP process.

If the Fault Analyzer XFACILIT process described here is used as the SDUMP or TDUMP control of its RFR dumps, then the actual SDUMP or TDUMP can not be read or deleted by a normal end user, except through analysis or deletion of the fault entry it is linked to. For example, in a case where the 'payroll' application might have its own history file that general users don't have read access to, then this XFACILIT process means that any RFR SDUMPs or TDUMPS for 'payroll' are restricted from general users because they can't access the fault entries.

The XFACILIT access requirements for RFR SDUMP or TDUMP differ as explained in the following.

SDUMP recovery fault recording data sets

When an SDUMP is requested, it is generated by the DUMPSRV address space with a naming convention that is determined by DUMPSRV. Normally most users on the system, except for the systems programmers, are restricted by UACC(NONE) to these SDUMPs. To be able to request an SDUMP, Fault Analyzer must use authorized state, which it obtains from its internal SVC process.

Fault Analyzer only uses SDUMP in the recovery fault recording process if the user ID under which the abend occurred was granted XFACILIT access using the following XFACILIT resource class setup.

Using the XFACILIT resource class for SDUMP RFR data sets

To have Fault Analyzer use SDUMP in its RFR process, set up an XFACILIT class profile with the name IDI_SDUMP_ACCESS and provide ALTER access to the user IDs or groups where SDUMPs are required for RFR exceptions. The following define would permit Fault Analyzer to create SDUMPS for all users in the CICS® group, if their fault entry create encounters an exception.

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

The ALTER access is to the XFACILIT IDI_SDUMP_ACCESS profile, it is not to the actual SDUMP data sets. Fault Analyzer uses authorized state to permit access to RFR SDUMPs. The IDI_SDUMP_ACCESS profile acts as a switch Fault Analyzer can check to see if SDUMPs should be created for that user ID.

If by chance a fault entry creation has an exception requiring an RFR dump, then Fault Analyzer only creates and links an SDUMP to the fault entry if the user has ALTER access to the XFACILIT IDI_SDUMP_ACCESS profile.

If you have ALTER access to the XFACILIT IDI_SDUMP_ACCESS profile, then Fault Analyzer provides the equivalent access to the SDUMP RFR data set, as to the fault entry that it is associated with, when you are doing problem analysis and have READ or DELETE access to the fault entry.

Implicit deletion of a fault entry during automatic space management, causes any linked SDUMP RFR data set to also be deleted.

Because capturing an SDUMP is usually much faster than capturing a TDUMP, it is recommended that at least performance critical systems, such as CICS®, are given authority to use RFR SDUMPs by granting the above access.

TDUMP recovery fault recording data sets

This section describes the required authorization to create, read, and delete exception condition RFR TDUMPs. Because a TDUMP requester can nominate the TDUMP data set name, this section also describes the naming convention process.

Note: The older RFR TDUMP naming convention of user ID high-level qualifier is no longer used because of the security and data set deletion problems that were frequently encountered with user ID TDUMP high-level qualifiers.

To overcome the security concerns of normal data set profiles with respect to TDUMP recovery fault recording data sets, Fault Analyzer supports the use of the XFACILIT resource class as described in the following. Together with the use of the XFACILIT resource class, it is recommended that UACC(NONE) is used as the general data set profile access level for TDUMP recovery fault recording data sets, to prevent the possibility of security exposures. The exposure would exist if ALTER access was granted to all users on the RFR TDUMP data set profile to permit creation, instead of UACC(NONE) and the XFACILIT set up.

If a system has a situation where all end users have similar access privileges, then the RFR TDUMPs are still taken if you choose to not set up the XFACILIT IDIRFR_TDUMP_HLQ, and instead give all users ALTER access to the TDUMP data set profile. This environment would probably have all users with equal access to the history files on that system. However, if some users do not have read access to all history files, then IDIRFR_TDUMP_HLQ and UACC(NONE) on the data set profile should be considered to extend the protection to any linked RFR TDUMPS.

Using the XFACILIT resource class for TDUMP RFR data sets

To set up the XFACILIT resource class for Fault Analyzer TDUMP recovery fault recording data sets, the high-level qualifier must initially be determined.

The default data set name pattern for the TDUMP recovery fault recording data sets is

```
IDIRFRHQ.IDIRFR.&SYSNAME..D&YYMMDD..T&HHMMSS..S&SEQ.
```

To change this data set name pattern, see "Changing the default recovery fault recording IEATDUMP data set name (RFRDSN) on page 339" .

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.

For example, if the TDUMP data set name pattern has been set to

```
IDIRFRHQ.TDUMP.&SYSNAME..D&YYMMDD..T&HHMMSS..S&SEQ.
```

using the IDIOPTLM RFRDSN option, then add the XFACILIT class profile

```
IDIRFR_TDUMP_HLQ.IDIRFRHQ.TDUMP.**
```

If the high-level qualifier includes a symbol name such as TDUMP&SYSCLONE., then it might be necessary to set up more than one profile, depending on the expected symbol substitution values.

Having defined the XFACILIT profile (or profiles, if more than one due to symbol substitution), then provide the appropriate level ALTER or NONE for the users concerned. If the user's access level to the XFACILIT class is ALTER, then through Fault Analyzer, the user implicitly has TDUMP create capability to the data set whose high-level qualifier, after any symbol substitution, matches the XFACILIT profile name *hlq* value.

General access of ALTER to an XFACILIT profile does not override any normal data set profile protecting a recovery fault recording data set. It only permits the necessary access authorization to the linked TDUMP data set when performing actions through Fault Analyzer, such as reading it during reanalysis, or deleting it when the associated fault entry is deleted.

Sometimes, by chance, a fault entry creation has an exception requiring an RFR dump. Then Fault Analyzer only creates and links a TDUMP to the fault entry under one of these conditions:

- You have ALTER access to the appropriate `XFACILIT IDIRFR_TDUMP_HLQ` profile.
- You have ALTER access to the TDUMP data set profile.

If you have ALTER access to the appropriate XFACILIT IDIRFR_TDUMP_HLQ profile, then Fault Analyzer provides the equivalent access to the TDUMP RFR data set, as to the fault entry that it is associated with, when you are doing problem analysis and have READ or DELETE access to the fault entry.

Implicit deletion of a fault entry during automatic space management, causes any linked TDUMP RFR data set to also be deleted.

RFR TDUMP XFACILIT example

The following is an example of how to setup of the XFACILIT class to manage the Fault Analyzer recovery fault recording data sets. This example can be modified or expanded on by an installation as required.

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)
```

Managing XDUMP data set access

In addition to normal data set access, Fault Analyzer optionally permits control of its XDUMP data sets with an XFACILIT security profile.

If the Fault Analyzer XFACILIT process described here is used as the method to control the XDUMP data sets, then the actual XDUMP data set cannot be read or deleted by a normal end user, except through analysis or deletion of the fault entry it is linked to. For example, where the *"payroll"* application has its own history file that general users do not have READ access to, this XFACILIT process means that any XDUMP data sets for *"payroll"* are restricted from general users because they cannot access the fault entries.

Use UACC(NONE) as the general data set profile access level for XDUMP data sets, to prevent the possibility of security exposures. An exposure would exist if ALTER access was granted to all users on the XDUMP data set profile to permit creation, instead of UACC(NONE) and the following XFACILIT set-up. If, on a given system, all end users have similar access privileges, then the XDUMP data sets are still created if you choose to not set up the XFACILIT access, and instead give all users ALTER access to the XDUMP 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 using the XFACILIT profile with UACC(NONE) on the data set profile should be considered to extend the protection to any linked XDUMP data sets.

Using the XFACILIT resource class for XDUMP data sets

Set up an XFACILIT class profile with the name `IDIXDUMP_HLQ.`*`hlq`*`.**`. Replace *hlq* with one or more qualifiers of the data set name pattern specified using the XDUMPDSN option in the IDIOPTLM configuration-options module. (For details, see Changing the default extended minidump (XDUMP) data set name pattern (XDUMPDSN) on page 340.)

If the high-level qualifier includes a symbol name such as `XDUMP&SYSCLONE.`, then it might be necessary to set up more than one profile, depending on the expected symbol substitution values.

Having defined the XFACILIT profile (or profiles, if more than one due to symbol substitution), then provide the appropriate level (ALTER or NONE) for the users concerned. If the user's access level to the XFACILIT class is ALTER, then through Fault Analyzer, the user implicitly has XDUMP 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 an XDUMP data set. It only permits the necessary access authorization to the linked XDUMP data set when performing actions through Fault Analyzer, such as reading it during reanalysis, or deleting it when the associated fault entry is deleted.

Fault Analyzer only creates and links an XDUMP to the fault entry under one of these conditions:

- You have ALTER access to the appropriate XFACILIT `IDIXDUMP_HLQ` profile.
- You have ALTER access to the XDUMP data set profile.

If you have ALTER access to the appropriate XFACILIT IDIXDUMP_HLQ profile, then Fault Analyzer provides the equivalent access to the XDUMP data set, as to the fault entry that it is associated with, when you are doing problem analysis and have READ or DELETE access to the fault entry.

Implicit deletion of a fault entry during automatic space management, causes any associated XDUMP data set to also be deleted.

XDUMP XFACILIT example

The following is an example of the recommended setup of the XFACILIT class for the purpose of managing the Fault Analyzer XDUMP data sets. You can modify or expand on this example as required.

This example assumes that the XDUMPDSN option in the IDIOPTLM configuration-options module has been specified with the following value:

```
'IDIHLQ.XDUMP.&&SYSNAME..D&&YYMMDD..T&&HHMMSS..S&&SEQ.'
```

1. Define an XFACILIT profile and grant universal access of ALTER to this profile:

   ```
   RDEFINE XFACILIT IDIXDUMP_HLQ.IDIHLQ.XDUMP.** UACC(ALTER)
   ```

2. Define a generic data set profile for the same data sets with universal access of NONE:

   ```
   ADDSD 'IDIHLQ.XDUMP.**' UACC(NONE)
   ```

Managing copied SDUMP data set access

When moving or copying fault entries with an associated tightly coupled SDUMP data set, Fault Analyzer creates a copy of the original SDUMP data set and links the copied fault entry and copied SDUMP data set together.

Fault Analyzer permits control of copied SDUMP data sets with an XFACILIT security profile.

If the process described in this topic is used to control copied SDUMP data sets, the actual copied SDUMP data set cannot be read or deleted by a normal end user, except through analysis or deletion of the fault entry it is linked to. For example, where the *"payroll"* application has its own history file that general users do not have READ access to, this XFACILIT process means that any copied SDUMP data sets for *"payroll"* are restricted from general users because they cannot access the fault entries.

To prevent the possibility of security exposures, use UACC(NONE) as the general data set profile access level for copied SDUMP data sets. An exposure would exist if ALTER access was granted to all users on the copied SDUMP data set profile to permit creation, instead of UACC(NONE) and the following XFACILIT setup. If, on a given system, all end users have similar access privileges, then the copied SDUMP data sets are still created if you choose to not set up the XFACILIT access, and instead give all users ALTER access to the copied SDUMP 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, consider using the XFACILIT profile with UACC(NONE) on the data set profile to extend the protection to any copied SDUMP data sets linked to fault entries.

Using the XFACILIT resource class for copied SDUMP data sets

Set up an XFACILIT class profile with the name `IDISDUMP_HLQ.hlq.**`. Replace *hlq* with one or more qualifiers of the data set name pattern specified using the SDUMPDSN option in the IDIOPTLM configuration-options module. (For details, see Specifying the copied SDUMP data set name pattern (SDUMPDSN) on page 341.)

If the high-level qualifier includes a symbol name such as SDUMP&SYSCLONE., 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 there is more than one due to symbol substitution), then provide the appropriate level (ALTER or NONE) for the users concerned. Users with ALTER access to the XFACILIT class implicitly have create capability through Fault Analyzer to the copied SDUMP data set whose high-level qualifier, after any symbol substitution, matches the XFACILIT profile name *hlq* value.

General ALTER access to an XFACILIT profile does not override any normal data set profile protecting a copied SDUMP data set. It only permits the necessary access authorization to the copied SDUMP data set linked with a fault entry when performing actions through Fault Analyzer such as:

- Reading the data set during reanalysis
- Deleting the data set when the associated fault entry is deleted

Fault Analyzer tries to create and link a copied SDUMP to a fault entry when:

- The fault entry is copied using either the C or M line command from the Fault Entry List display or the batch IDIUTIL IMPORT control statement
- The fault entry is already associated with an original SDUMP data set or a previously copied SDUMP data set

To create and link a copied SDUMP data set, the user must be granted ALTER access to either the appropriate XFACILIT `IDISDUMP_HLQ` profile or the copied SDUMP data set profile.

If you have ALTER access to the appropriate XFACILIT IDISDUMP_HLQ profile, then Fault Analyzer provides the equivalent access to the copied SDUMP data set, as to the fault entry that it is associated with, when you are doing problem analysis and have READ or DELETE access to the fault entry.

Implicit deletion of a fault entry during automatic space management, causes any associated copied SDUMP data set that is linked with the fault entry to also be deleted.

Example

XFACILIT example: copied SDUMP data sets

The following is an example of the recommended setup of the XFACILIT class for managing SDUMP data sets copied by Fault Analyzer. You can modify or expand on this example as required.

This example assumes that the SDUMPDSN option in the IDIOPTLM configuration-options module has been specified with the following value:

```
'IDIHLQ.SDUMP.&&SYSNAME..D&&YYMMDD..T&&HHMMSS..S&&SEQ.'
```

1. Define an XFACILIT profile and grant universal access of ALTER to this profile:

   ```
   RDEFINE XFACILIT IDISDUMP_HLQ.IDIHLQ.SDUMP.** UACC(ALTER)
   ```

2. Define a generic data set profile for the same data sets with universal access of NONE:

   ```
   ADDSD 'IDIHLQ.SDUMP.**' UACC(NONE)
   ```

Registering Fault Analyzer in the IFAPRDxx parmlib member

If you purchased Fault Analyzer as part of product code 5755-AD1 IBM Application Delivery Foundation for z/OS, include an entry in the IFAPRDxx parmlib member as follows:

```
PRODUCT OWNER('IBM CORP')
        NAME('IBM APP DLIV FND')
        ID(5755-AD1)
        VERSION(*) RELEASE(*) MOD(*)
        FEATURENAME('*')
        STATE(ENABLED)
```

If you purchased Fault Analyzer separately, an entry in IFAPRDxx parmlib is not required. However, if you would prefer to have an entry, the following can be added:

```
PRODUCT OWNER('IBM CORP')
        NAME('FAULT ANALYZER')
        ID(5755-AD2)
        VERSION(*) RELEASE(*) MOD(*)
        FEATURENAME('FAULT ANALYZER')
        STATE(ENABLED)
```

All parameters except VERSION, RELEASE, and MOD must be specified exactly as shown. Any syntactically valid values can be specified for VERSION, RELEASE, and MOD.

After the IFAPRDxx parmlib member is updated, it can be activated dynamically (until the next IPL) using the following console command:

```
SET PROD=xx
```

Refer to *MVS™ Initialization and Tuning Reference* for general information about the IFAPRDxx parmlib member.

Additional Fault Analyzer IFAPRDxx processing

If a product above is not defined in IFAPRDxx, when Fault Analyzer is first invoked it will register during initialization as product code 5755-AD2.

If a product above is defined with STATE(DISABLED) or STATE(NOTDEFINED), the product will not be selected for registration.

To prevent Fault Analyzer from running, use the following IFAPRDxx entry:

```
```

```
PRODUCT OWNER(’IBM CORP’)
NAME('FAULT ANALYZER')
ID(5755-AD2)
VERSION(*) RELEASE(*) MOD(*)
FEATURENAME('FAULT ANALYZER')
STATE(DISABLED)
```

If you change how a product is purchased, for example if you upgrade from using a stand-alone version to the IBM Application Delivery Foundation for z/OS product bundle, you must explicitly remove the existing product definition from the in-storage tables kept by z/OS® when activating the new definition. Follow this scenario to do this dynamically (without IPL):

1. In IFAPRDxx, define the new product as described above, and update the old product with STATE(DISABLED).
2. Activate the update using the following operator command:

   ```
   SET PROD=xx
   ```

3. You can now safely remove the old product definition from IFAPRDxx.

IBM® advises against defining IFAPRDxx entries that have NAME(*) or ID(*) fields, as this could result in unintended product registrations. If a match is found on an entry defined with NAME(*) and ID(*) with STATE(ENABLED), Fault Analyzer will register as product code 5755-AD2.

Chapter 14. Using the Fault Analyzer IDIS subsystem

Fault Analyzer uses a subsystem known as IDIS for services that can otherwise not be performed or which might cause incomplete analysis if not started.

A separate IDIS subsystem is required on each MVS™ image that runs Fault Analyzer.

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 following message to be issued in the abending address space.

  ```
  IDI0082E DB2 Call Level Interface error: SQL return code -1 for SQLAllocConnect
  to DB2 system system-id
  ```

- Register SVC dumps on behalf of the IDIXTSEL post-dump exit. This registration is primarily intended for CICS® system dumps and to facilitate the capturing of Java™ faults but also allows the capture of SLIP dumps.

- Perform history file access management if PDSESHARING(NORMAL) is used, in order to reduce abend S213-70 cross-system sharing conflicts.
- Optionally, manage history file $$INDEX members for improved performance. While this feature is the default for eligible history files, it can be disabled by specifying the NOUPDINDEX PARM field option for the IDIS subsystem, as described in Starting the IDIS subsystem on page 320. For details, see Caching of history file $$INDEX data on page 319.
- 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 320. For details of the NoDup(ImageFast(*minutes*,IMS(…))) option, see NoDup on page 616.
- 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 320. For details, see Fast Exclude options processing on page 372.
- Provide recovery fault recording support. (For general information about recovery fault recording, see Recovery fault recording on page 51.)

- Provide improved performance of message and abend code explanation retrieval for the Fault Analyzer ISPF interface LOOKUP command by caching information in storage. Additionally, perform automatic updates of the VSAM KSDS message and abend code explanation repository when required. Refer to Setting up the message and abend code explanation repository on page 308 for information about required access authorization.
- Provide Java™ analysis support by means of the Java-supplied Diagnostic Tooling Framework for Java™ (DTFJ). The DTFJ process runs from the BXPAS address space, which is spawned from the IDIS subsystem when the PARM='JAVA' option is used in the IDIS subsystem startup JCL.

- Perform reading of message and abend explanations from source data sets, as well as caching of selected explanations for improved performance.
- Delete migrated tightly coupled dump data sets when the associated fault entry is deleted. See IDIS subsystem requirements for deletion of migrated tightly-coupled associated dump data sets on page 325.

The 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 does not affect system performance adversely, since no resources are consumed by the IDIS subsystem when it is not in use.

OMVS home path

The IDIS subsystem must be defined to the security system with an OMVS UID value and home path. The home path must exist with appropriate access permissions.

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.

The IDIS subsystem uses `ISGQUERY GATHERFROM=SYSPLEX` as part of its integrity and sharing strategy. Customers who use non-IBM enqueue management products should contact their vendor to discuss possible setup requirements to support this function.

Other than that, no special action is needed to use this feature. If XCF becomes unavailable, IDIS processing continues to perform all functions, although reduced performance might result.

Caching of history file $$INDEX data

The default value for the PARM parameter in the Fault Analyzer IDIS subsystem startup JCL is *"UPDINDEX"* (see Starting the IDIS subsystem on page 320). When the default value is used, the IDIS subsystem manages the $$INDEX member access of all PDSE history files:

- That are used on the MVS™ image where the IDIS subsystem is running.
- To which the IDIS subsystem has UPDATE access.

For details about the $$INDEX member, see Special members in the history file data set on page 25. PDS-format history files are not managed in the IDIS subsystem because their enqueue and parallel update limitations prevent any effective caching of their $$INDEX members.

After the $$INDEX member from a history file is read, the information is kept in the IDIS subsystem address space during periods of high activity. This approach provides fast access for any requesters of this data, since no I/O is required. Real-time analysis, interactive and batch reanalysis, and the Fault Analyzer ISPF interface, use the cached IDIS subsystem data.

Each time the cache is accessed for read or write, the time limit for in-storage retention is reset. The reset ensures that a history file that is highly active continues to provide fast cache access. This approach is beneficial to environments, such as CICS®, where multiple abends might occur in rapid succession, and often all need to update the same history file.

The time limit for the IDIS subsystem in-storage retention is set to 5 minutes. The $$INDEX member is written back to the history file and the IDIS subsystem relinquishes control of it when:

- The time limit expires.
- A request for update of the same history file is pending from another MVS™ image in the same sysplex.

Tip: Make the setting of the IDIS subsystem option PARM='UPDINDEX' or PARM='NOUPDINDEX' 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 362.

The UPDINDEX option can be used to ensure that certain low-priority jobs do not cause excessive delays in the Fault Analyzer execution due to serialization of the history file $$INDEX member. These jobs are the jobs that share history files with performance sensitive jobs or execution environments, such as IMS™ and CICS®, in a CPU constrained environment.

If you use the UPDINDEX option, ensure that the IDIS subsystem has UPDATE access through normal security server data set profiles to all history files that it is expected to handle. XFACILIT access is not sufficient.

If the IDIS subsystem is called for a history file to which it does not have UPDATE access, or if it fails to update a history file for any other reason, then no further attempt is made to manage the $$INDEX member of that history file until one of the following steps is performed:

- The IDIS subsystem is stopped and restarted.
- The interactive IDIS subsystem interface is used to reset the excluded status of the history file. For details, see Using the interactive IDIS subsystem interface on page 142.

Note: For maximum Fault Analyzer performance, consider the DeferredReport option. For details, see DeferredReport on page 589.

Starting the IDIS subsystem

To start the Fault Analyzer IDIS subsystem, a simple job as shown below can be submitted:

```
//IDISS    JOB  …
//IDISSTST EXEC PGM=IDISAMAN,TIME=NOLIMIT,REGION=region-size,PARM='options'
//IDIDOC2  DD   DISP=SHR,DSN=IDI.SIDIDOC2
//* (Optional DD statements might follow, as described below)
```

where

REGION=*region-size*

Specifies the region size to be used for the IDIS subsystem.

In most cases, a region size of 100 megabytes should be adequate (REGION=100M). However, if a very large number of history files are being managed by the IDIS subsystem, or if the history file sizes are very large, then

it might be necessary to specify an even larger region size. For information about how to estimate the required region size, see IDIS subsystem storage requirements on page 322.

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 Options on page 571. 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 319.

This value is the default.

NOUPDINDEX

Specifies that the abending job performs all history file updates.

Note: Specifying NOUPDINDEX will disable the Fault Analyzer Recovery Fault Recording (RFR) feature.

IMAGEFAST

Enables IMS fast duplicate fault suppression specified using the NoDup(ImageFast(*minutes*,IMS(...))) option. For details, see NoDup on page 616.

This value is the default.

NOIMAGEFAST

Disables IMS fast duplicate fault suppression, regardless of NoDup(ImageFast(...)) settings.

FASTEXCLUDE

Enables fast Exclude options processing. For details, see Fast Exclude options processing on page 372.

This value is the default.

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 creation would normally only

be done if there are MVS™ images in a sysplex which do not share DASD and history files with the main IDISXCFM default group, and would be set for the images that do not share 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.

JAVA

Enables Java™ analysis. See IDIS subsystem requirements for Java on page 324 for more information.

NOJAVA

Disables Java™ analysis. This value is the default.

SLIP

Enables the capture of SLIP dumps.

NOSLIP

Disables the capture of SLIP dumps. This value is the default.

Multiple PARM field options must be delimited by one or more blank characters.

Alternatively, the IDIS subsystem can be established using a started task. The IDIS subsystem dynamically allocates data sets to SYSOUT=*, so it must be run under the job entry subsystem (JES).

Note: Ensure that the TIME=NOLIMIT parameter is specified as shown in the example above to prevent IDIS subsystem abend S522.

The subsystem name that is used by Fault Analyzer for this subsystem is IDIS. This name does not need to be defined in the IEFSSNxx parmlib member as it is dynamically defined by the IDISAMAN program.

The //IDIDOC2 DD statement is used to allocate the IDI.SIDIDOC2 SMP/E target data set, which might contain updates to the Fault Analyzer VSAM KSDS message and abend code explanation repository. If updates are available, and either the //IDIDOC2 DD statement has not been specified, or the high-level qualifier of the IDI.SIDIDOC2 data set is not the same as the high-level qualifier of the IDI.SIDIDOC1 data set identified via the DataSets(IDIDOC(...)) option, then message IDI0165A on page 733 is issued. If this situation occurs, then add the //IDIDOC2 DD statement as shown above.

Do not change the LE HEAPZONES option setting to anything other than HEAPZONES(0,...) when the IDIS subsystem is active.

IDIS subsystem storage requirements

If using PARM='NOUPDINDEX', then there is no requirement for any REGION size specification—a default 32 MB region is adequate.

If using PARM='UPDINDEX' (this value is the default), then a larger region size should be used. The region size in megabytes can be approximated as follows:

```
32 + (num_hist_files * (0.5 + (avg_num_entries * 0.0005)))
```

where:

num_hist_files

The total number of PDSE history files that are managed by the IDIS subsystem.

avg_num_entries

The average number of fault entries in the managed history files.

If a maximum number of fault entries has been set for a history file using the IDIUTIL batch utility SetMaxFaultEntries control statement (for details, see Managing history files (IDIUTIL utility) on page 441), then this number is the number of fault entries that should be included for that history file.

IDIS subsystem requirements for DB2

You must add DD statements for all DB2® subsystems that are not accessible via LINKLIST, and for which you want Fault Analyzer to perform analysis:

- If you have more than one version of DB2® installed.
- If the DB2® load module library is not in LINKLIST.

Here is the DD statement format:

```
//DB2subsystem-id DD DISP=SHR,DSN=data-set-name
```

where *subsystem-id* is the DB2® subsystem ID (usually 4 characters), and *data-set-name* is the associated load module library. For a data sharing group, the group attach name is used as the subsystem ID, regardless of whether the DB2® subsystem is running in data sharing mode or not.

If, for example, the DB2® subsystem with an ID of DSN1 requires the load library DSN1.SDSNLOAD, which is not in LINKLIST, then add this JCL DD statement to the Fault Analyzer IDIS subsystem job.

```
//DB2DSN1  DD DISP=SHR,DSN=DSN1.SDSNLOAD
```

Do not include a DSNAOINI DD statement in the IDIS subsystem JCL, as this DDname is allocated dynamically by Fault Analyzer as needed for the appropriate DB2® subsystem.

The IDIS subsystem needs EXECUTE access to the DSNACLI plan, and SELECT authority to the following SYSIBM catalog tables:

SYSIBM.SYSDBRM
SYSIBM.SYSPACKAGE
SYSIBM.SYSPACKSTMT
SYSIBM.SYSPACKLIST

SYSIBM.SYSPLAN
SYSIBM.SYSSTMT

IDIS subsystem requirements for Java

The IDIS subsystem must have UPDATE access to at least the default history file in order to create a Java™ fault entry.

Specifying an IDIJLIB DD statement

An optional IDIJLIB DD statement can be included in the IDIS subsystem JCL for Java™ as follows:

```
//IDIJLIB  DD   PATH='path'
```

The IDIJLIB JCL statement specifies a target directory for zFS executables. The IDIS subsystem writes a small number of program files to this directory as part of its execution, thus the IDIS subsystem must have authority to read, write, and execute files in the specified directory. Additionally, diagnostic information might also be written to this path. The path name is case sensitive and the path must exist. An example of a possible specification is `PATH='/u/user-id/idij'`, where *user-id* is the user ID under which the IDIS subsystem is running.

Note: If specified, the IDIJLIB path is also used for deletion of migrated tightly-coupled associated dump data sets. For details, see IDIS subsystem requirements for deletion of migrated tightly-coupled associated dump data sets on page 325.

If IDIJLIB is not provided, then the default path for the IDIS subsystem user ID is used. In this case, if the IDIS subsystem user ID does not have a valid default path, message IDI0155W on page 730 is issued.

For each image in a sysplex, Fault Analyzer creates a subdirectory of the IDIJLIB path, using the &SYSCLONE name. This naming ensures that each image writes to a separate directory.

Specifying a default JVM for Java dump analysis

Performing Java™ dump analysis requires a JVM that has a service level equal to, or greater than, the JVM that recorded the Java™ dump. Optional DD statements can specify default JVMs to use when the version of Java™ indicated by the dump `JAVA_HOME` path cannot be found on the system.

You can specify default JVMs by adding the IDIJVM, IDIJVM6, and IDIJVM11 DD statements to the IDIS subsystem JCL for Java™.

- The IDIJVM DD statement specifies the path to a default 31-bit JVM.
- The IDIJVM6 DD statement specifies the path to a default 64-bit JVM for Java 10 or earlier versions.
- The IDIJVM11 DD statement specifies the path to a default 64-bit JVM for Java 11 or later versions.

```
//IDIJVM   DD   PATH='path'
//IDIJVM6  DD   PATH='path'
//IDIJVM11 DD   PATH='path'
```

Note: The Diagnostic Tool Framework for Java (DTFJ) has internal changes after Java 8. IDIJVM11 is a required DDname if co-existing with Java 8 64-bit applications.

IDIS subsystem requirements for deletion of migrated tightly-coupled associated dump data sets

Deletion of migrated tightly-coupled associated dump data sets is performed via a spawned BPX batch region. This is done to avoid potential performance issues.

The IDIS subsystem creates, and subsequently deletes, a "dummy" file in the path determined in the following order:

1. Specified via the IDIJLIB DD statement (for details, see Specifying an IDIJLIB DD statement on page 325).
2. The OMVS segment home path for the IDIS subsystem user ID (see OMVS home path on page 319).
3. `/tmp`

If unsuccessful in using any of these paths, message IDI0192I on page 738 is issued.

Specifying an IDIJLIB DD statement

An optional IDIJLIB DD statement can be included in the IDIS subsystem JCL as follows:

```
//IDIJLIB  DD   PATH='path'
```

where path is where Fault Analyzer will temporarily create the necessary dummy file.

Note: Note: If specified, the IDIJLIB path is also used as target for Java analysis zFS executables. For details, see "IDIS subsystem requirements for Java" on page 315.

Stopping the IDIS subsystem

The IDIS subsystem can be stopped and restarted at any time.

When the IDIS subsystem is inactive (stopped), only basic Fault Analyzer processes continue to run, such as the creation of new fault IDs. SVC dump registration, Recovery Fault Recording (RFR), Java DTFJ processing, some DB2® catalog data access and deletion of migrated tightly-coupled dump data sets will not work. There is also an increase in CPU and I/O usage, compared to having the IDIS subsystem running with PARM='UPDINDEX'.

Attention: Never use the CANCEL command to stop the IDIS subsystem.

To permit normal termination, use a MODIFY command to stop the IDIS subsystem by, for example: `F name,STOP` or `P name`

where *name* is the appropriate identifier for the MODIFY command, depending on how the IDIS subsystem was started.

If either the 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 IDIS subsystem is already active when another attempt to start it is performed, the following message is 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 IDIS subsystem and *job-id* is the JES job or started task ID.

Chapter 15. Modifying your ISPF environment

To use Fault Analyzer with ISPF, you need to ensure that the appropriate data sets have been allocated and that one or more ways to invoke Fault Analyzer have been provided. This process is explained in the following topics.

Allocating ISPF data sets

The following data sets must be allocated to the respective ISPF DDnames (either in the TSO logon procedure or using any other installation-specific method):

DDname	Data set name
ISPPLIB	IDI.SIDIPLIBIPV.SIPVPENU (1)
ISPMLIB	IDI.SIDIMLIBIPV.SIPVMENU (1)
ISPSLIB	IDI.SIDISLIB
ISPTLIB	IDI.SIDITLIBIPV.SIPVTENU (1)
SYSEXEC	IDI.SIDIEXEC

Note: (1) The data set is created as part of the ADFz Common Components installation.

A sample REXX EXEC that can be used to invoke Fault Analyzer from within ISPF is provided as member IDISISPF in data set IDI.SIDISAM1.

The EXEC performs the necessary dynamic definition of the required data sets using the ISPF LIBDEF and TSO ALTLIB services.

Note:

To enable the editing or browsing of data sets using File Manager for z/OS®, all necessary ISPF libraries for File Manager also must be made available when Fault Analyzer is invoked. Refer to the File Manager for z/OS® 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 for z/OS® 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, the LOOKC command, and the LOOKUP command. See:

- Invoking Fault Analyzer from SDSF (SFA command) on page 330
- Invoking the LOOKUP command using cursor selection (LOOKC command) on page 330
- LOOKUP on page 106

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) PARM(HEAPZONES(0,ABEND)/)'
  X,EXIT
 ' ',' '
   *,'?' )
&ZTRAIL=.TRAIL
```

Notes:

1. The above IDI application ID is shown as an example only. If a different value has been used for an earlier version of Fault Analyzer, then changing it should be avoided since it results in the loss of all ISPF interface user-tailoring.
2. Ensure that the LE HEAPZONES parameter is set to HEAPZONES(0,ABEND) while Fault Analyzer is active.

As an alternative to calling program IDIPDDIR directly from the selection panel as shown above, you might instead consider invoking the IDISISPF sample REXX EXEC. This invocation would eliminate the need to have Fault Analyzer ISPF and TSO libraries allocated prior to entering ISPF.

Remember to also update the)BODY section as appropriate.

As an alternative to the direct invocation of the Fault Analyzer ISPF interface via the ISPF selection panel, a customized front-end can be used. A sample front-end is shown in Sample customized ISPF interface front-end on page 741, which can be modified to suit any specific requirements your installation might have.

Invoking Fault Analyzer using an ISPF 3.4 line command

There is an alternative to the typical invocation of the Fault Analyzer ISPF interface, which uses an option from an ISPF selection menu. Sometimes it is more convenient to invoke the Fault Analyzer ISPF interface directly as a line command against an MVS™ dump data set, a history file, or a CICS® auxiliary trace data set on the ISPF option 3.4 Data Set List Utility panel. This invocation has the advantage of being able to perform interactive dump analysis, or display fault entries, or CICS® trace information using the selected data set immediately. You do not have to initiate the appropriate action by using the File pull-down menu for a dump data set, or by typing the data set name for a history file.

To facilitate this, a REXX exec (named FA for *"fault analysis,"* for example) is required in one of the data sets that is allocated to the SYSEXEC concatenation of the ISPF user. This exec name can then be entered next to a dump data set, history file data set name, or a CICS® auxiliary trace data set name in an ISPF option 3.4 data set list. The exec invokes Fault Analyzer using the correct ISPF NEWAPPL ID, and passes the name of the data set as a parameter. The exec determines the data set type from its allocation attributes.

The FA exec:

```
Parse Arg dsn .

If dsn = '' Then
  Do
    Say 'Data set name required.'
    Exit(4)
  End

/* Determine if dsn is a history file or a MVS DUMP.                */
outl. =
x = Outtrap('OUTL.',,"NOCONCAT")
Address TSO "LISTDS " || dsn
x = Outtrap(OFF)

svcdump = 0
auxtrac = 0
If outl.0 > 2 Then
  Do
    lrecl = Word(outl.3,2)
    If lrecl = 4160 Then
      svcdump = 1
    Else
      If lrecl = 4096 Then
        auxtrac = 1
  End

/* Invoke the Fault Analyzer ISPF interface passing the appropriate  */
/* PARM string.                                                      */
/* NOTE: The HEAPZONES(0,ABEND) parameter is only required if this   */
/* is not the default Language Environment option in effect.         */
/* However, this parameter can be left as is regardless, without any */
/* adverse side effects.                                             */
Address ISPEXEC
If svcdump = 1 Then
  'SELECT PGM(IDIPDDIR) NEWAPPL(IDI) '!!,
  'PARM(HEAPZONES(0,ABEND)/DSN('dsn'))'
```

```
Else If auxtrac = 1 Then
  'SELECT PGM(IDIPDDIR) NEWAPPL(IDI) '!!,
  'PARM(HEAPZONES(0,ABEND)/AUXTRACEDSN('dsn'))'
Else
  'SELECT PGM(IDIPDDIR) NEWAPPL(IDI) '!!,
  'PARM(HEAPZONES(0,ABEND)/ISPFHISTDSN('dsn'))'
Exit
```

This example is included in data set IDI.SIDIEXEC as member IDISFA.

While IDISFA works as a line command against ISPF 3.4 data sets, as long as IDI.SIDIEXEC 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, such as FA.

Invoking Fault Analyzer from SDSF (SFA command)

An exec named IDISSDSF is provided in data set IDI.SIDIEXEC. This exec can be used to extract the history file data set name and the fault ID from message IDI0003I on page 696, 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 328.

Invoking the LOOKUP command using cursor selection (LOOKC command)

An exec named IDISMLKP is provided in data set IDI.SIDIEXEC. 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 328.

Providing ISPF interface defaults for new users

To provide installation-specific defaults for new users of the Fault Analyzer ISPF interface, do the following:

1. Invoke the Fault Analyzer ISPF interface.
2. Set options to the values that new users should see when first using this interface. These values might include:
    - The initial history file or view selected from the Fault Entry List display. Alternatively, use the FAISPFOPTS(INITHIST(...)) option to provide the desired defaults. For details, see InitHist on page 609.
    - Fault Analyzer preferences (from the "Options" pull-down menu).
    - Batch reanalysis options (from the "Options" pull-down menu). Alternatively, use the FAISPFOPTS(BATCHOPTS(...)) option to provide the desired defaults. For details, see BatchOpts on page 116.
    - 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 FAISPFopts(HistCols(...)) option in the IDICNFxx parmlib member, rather than defining them through the ISPF profile member defaults. This approach is necessary because users who might make other changes to these columns, are otherwise not able to reset the columns to the installation-wide defaults by pressing PF4.

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 that 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 is saved in their private profile data set identified by the ISPPROF DDname, and read from there on any subsequent use of the Fault Analyzer ISPF interface.

Providing installation-specific batch reanalysis JCL control statements

Whenever a user performs batch reanalysis of a fault entry, using the B line command from the Fault Entry List display, Fault Analyzer generates the appropriate JCL stream based on options that are specified on the user's Batch Reanalysis Options display. To permit installations to always add their own JCL control statements to such generated jobs, Fault Analyzer provides support for an optional user-provided skeleton member. If found, this member is inserted immediately following the JOB card of the generated JCL stream. The skeleton 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
```

If **Job card style** on the Batch Reanalysis Options display has been set to S, but no **Job Card Statements** have been specified, then the IDISJCTL member can be used to provide both a JOB card and any additional control cards. As an alternative to the IDISJCTL member, see BatchOpts on page 116.

Increasing the display area for Fault Analyzer reports

ISPF supports screen sizes with a width greater than 80. By allowing Fault Analyzer to use a wider screen, readability of online reports is greatly improved and the necessity of using left and right scrolling commands can be eliminated.

Here is the process:

1. Check that you have met the technical requirements that are set out in Technical details for screen size adjustments on page 753.
2. Alter the emulator settings to use, for example 27x132.

3. Logon to ISPF.
4. Go to ISPF Option 0. Entering `SETTINGS` on any command line displays the same screen.
5. Set the screen format to DATA or MAX. See the relevant ISPF publications for explanations of the difference in behavior of DATA and MAX.

Chapter 16. Customize Fault Analyzer by using USERMODs

Some Fault Analyzer customization tasks can only be performed using SMP/E USERMODs. Other tasks can be performed using a configuration-options module; see Customize Fault Analyzer by using an IDIOPTLM configuration-options module on page 337.

Enabling Fault Analyzer to be invoked

In order for Fault Analyzer to be given control when a program abends, a number of invocation exits must be installed.

The following provides information about the installation of the two invocation exits that are required for non-CICS environments. Refer to Customizing the CICS environment on page 396 for invocation exits for the CICS® environment.

Installing the MVS change options/suppress dump exit IDIXDCAP

IDIXDCAP is installed as a IEAVTABX_EXIT dynamic exit routine by including the following in a PARMLIB PROG*xx* member, which is automatically selected at IPL time:

```
EXIT ADD EXITNAME(IEAVTABX_EXIT) MODNAME(IDIXDCAP)
```

To activate the IDIXDCAP exit before the next IPL, issue the following operator command, where *xx* matches the suffix used on the PARMLIB PROG*xx* member:

```
SET PROG=xx
```

Installing the Language Environment abnormal termination exits IDIXCEE and IDIXCE64

Two separate Language Environment (LE) invocation exits are required: one for 31-bit applications, and one for 64-bit applications.

31-bit applications (IDIXCEE):

Add the Fault Analyzer IDIXCEE invocation exit to the LE CEEEXTAN abnormal termination exit CSECT list as follows.

In a copy of the CEE.SCEESAMP data set CEEWDEXT sample member, replace the comment

```
<<< REPLACE THESE 2 LINES WITH A COPY OF CEEEXTAN
AND OVERRIDE AS DESIRED >>>
```

with these lines:

```
         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.

64-bit applications (IDIXCE64):

Add the Fault Analyzer IDIXCE64 invocation exit to the LE CELQXTAN abnormal termination exit CSECT list as follows.

In a copy of the CEE.SCEESAMP data set CEEWQEXT sample member, replace the comment

```
<<< REPLACE THESE 2 LINES WITH A COPY OF CELQXTAN
AND OVERRIDE AS DESIRED >>>
```

with these lines:

```
         CEEXAHD AM=64  ,User exit header (AMODE 64)
         CEEXART  TERMXIT=IDIXCE64
         CEEXAST        ,Terminate the list
```

Activating the exits

These exits are activated by performing an IPL. If any of the updated load modules are in LPA, then the IPL must be performed with the CLPA option.

If an IPL of MVS™ is not planned as part of the Fault Analyzer installation, the exits 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 USERMODs from the APPLY job output and issue the command:

  ```
  SETPROG LPA,ADD,MOD=(module-list),DSN=data-set-name
  ```

  Note: Ensure ECSA size is sufficient to accommodate the additional storage required by the SETPROG LPA,ADD command.

  If one or more of the updated load modules are not in LPA, then continue with the instructions for "LE not in LPA" below.

  Note: Remember to reissue the SETPROG command after the next IPL, unless the IPL is performed with the CLPA option.

- **LE not in LPA**

  If LE is not in LPA, it is assumed to be in LINKLIST. To activate the exits, issue the command:

  ```
  F LLA,REFRESH
  ```

  If you prefer to selectively refresh only the affected load modules, then you need to check the SMP/E APPLY output for the updated data sets and load module names, add these to a CSVLLAxx parmlib member using the LIBRARIES and MEMBERS parameters, and then issue the command `F LLA,UPDATE=xx`. Alternatively, use Data Set Commander for z/OS® to perform the refresh.

For general information about implementing LE abnormal termination exits, refer to the *Language Environment® Customization* book.

CICS notes:

1. The CICS® version of the AMODE 31 LE exit, CEECXTAN, which is installed with sample job CEEWCEXT, is very similar to the IDIXCEE exit. If you are going to install both, make sure that you double-check that you have used the correct names and not installed the same exit twice.
2. There is currently no AMODE 64 version of this exit for CICS®.

Working with applications that use a non-Language Environment run time

The following USERMODs are applicable to non-LE applications only.

Enabling implicit Fault Analyzer invocation from PL/I V2R3 applications (++IDISPLI/++IDISPLIA)

To have Fault Analyzer invoked implicitly for abends occurring in applications using the PL/I version 2 release 3 non-LE runtime library, the IDISPLI or IDISPLIA USERMOD must be applied. Either USERMOD modifies the ONCODE processing in the PL/I runtime load module IBMBLIIA to call the Fault Analyzer program SNAP interface (IDISNAP). However, while the IDISPLI USERMOD returns to PL/I after the call to IDISNAP, the IDISPLIA USERMOD issues a deliberate abend U3001, which can make it more suitable for environments, such as IMS™, where ROLLBACK might be required.

This USERMOD is not required if using the LE run time.

To apply this USERMOD, edit and submit the sample job IDISPLI or IDISPLIA.

Always invoking Fault Analyzer from PL/I PLIDUMP (++IDISPDM)

If PL/I applications have been written to handle errors by calling PLIDUMP with the 'S' option, then Fault Analyzer cannot be invoked (see Application-handled error conditions on page 304). 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.

Obtaining load modules from CA-Panexec

Fault Analyzer provides the ability for users to install an exit to obtain CSECT information for load modules that are managed by CA-Panexec. 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 before Binder INCLUDE processing. The load module named in IDILMODX should be accessible via an MVS™ LOAD macro.

The IDILMODX-named exit is passed a parameter list in register 1, addressing two pointers to two 8-byte fields. The first field contains the name of the load module to be examined (the load module the Binder includes to extract CSECT data), while the second 8-byte field is a DD name, or blank. If the DD name is not blank it means that this DD is intended to be used for the Binder in searching for the load module name. Fault Analyzer normally only sets the DD name field to DFHRPL when running under CICS®. On most other occasions, the DD name is blank, indicating Fault Analyzer leaves it to the Binder to locate the data set from which the module should be read. The blank DD contains X'00'. The IDILMODX-named exit is permitted to update the second parameter to a new DD name from which the Binder should read the module, or to leave the DD name unchanged. If the DD name is updated, then it is used for the Binder INCLUDE call of the module.

A sample CA-Panexec exit is available as member IDIPANEX in data set IDI.SIDISAM1.

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

Some aspects of the Fault Analyzer customization can only be performed via SMP/E USERMODs (for details, see Customize Fault Analyzer by using USERMODs on page 333), while others must be performed by using an IDIOPTLM configuration-options module.

An IDIOPTLM configuration-options module must consist of the following data area in a load module format.

Table 8. IDIOPTLM data area

Offsets		Type	Len	Name	Description
Dec	Hex				
0	(0)	CHAR	44	LEDSN	Deprecated
44	(2C)	CHAR	44	CNFDSN	See Specifying an alternative parmlib data set for IDICNFxx (CNFDSN) on page 339.
88	(58)	CHAR	100	RFRDSN	See Changing the default recovery fault recording IEATDUMP data set name (RFRDSN) on page 339.
188	(BC)	BIN	4	CICSNOA	See Changing the action of Fault Analyzer when invoked via the CICS XDUREQ global user exit (CICSNOA) on page 340.
192	(C0)	CHAR	44	SSCHKDSN	See Specifying an alternative security server test data set name (SSCHKDSN) on page 340.

Table 8. IDIOPTLM data area (continued)

Offsets		Type	Len	Name	Description
Dec	Hex				
236	(EC)	CHAR	100	XDUMPDSN	See Changing the default extended minidump (XDUMP) data set name pattern (XDUMPDSN) on page 340.
336	(150)	CHAR	100	SDUMPDSN	See Specifying the copied SDUMP data set name pattern (SDUMPDSN) on page 341.
436	(1B4)	BIN	4	NOIPVOPT	See Ignoring IBM Application Delivery Foundation for z/OS Common Components options (NOIPVOPT) on page 341.
440	(1B8)	BIN	1	PREVEXIT	See Suppressing IDIXDCAP real-time analysis if prior exit RC=8 (PREVEXIT) on page 342.
441	(1B9)	BIN	15		Reserved. Must be all binary zeroes.

To specify any of these settings, edit and submit a job to create an IDIOPTLM configuration-options load module. A sample job is provided as member IDIOPTLM in data set IDI.SIDISAM1. Options in the IDIOPTLM load module are read whenever normal options processing is performed.

The configuration-options load module must be named IDIOPTLM and it must be placed in IDI.SIDIAUTH or another APF-authorized library. The library should also be in LNKLST so that the IDIOPTLM load module can be found by all jobs.

Specifying an alternative parmlib data set for IDICNFxx (CNFDSN)

To accommodate installations that do not provide general READ access to SYS1.PARMLIB (or any one of the data sets in the logical parmlib concatenation), Fault Analyzer provides the CNFDSN setting in the IDIOPTLM configuration-options module. This setting permits an alternative data set name to be specified for the IDICNFxx configuration member (for details, see Parmlib member IDICNFxx on page 369).

The specified data set name can contain MVS™ system symbols and must be allocated as a PDS(E) with LRECL=80 and RECFM=FB. The data set must be given universal READ access.

Fault Analyzer does not require the IBM ADFz Common Components IPVCNF00 parmlib member to exist, but it will attempt to process IPVCNF00 unless the NOIPVOPT value is set to 1 in the IDIOPTLM configuration-options module. For details, see Ignoring IBM Application Delivery Foundation for z/OS Common Components options (NOIPVOPT) on page 341.

To determine if the IPVCNF00 parmlib member exists, Fault Analyzer must have READ access to either:

- The logical parmlib concatenation
- A data set specified in the IBM ADFz Common Components IPVOPTLM configuration-options module

Otherwise, abend S913 and a message similar to ICH408I (depending on the security server) will occur.

For information about the IPVOPTLM configuration-options module and the IPVCNF00 parmlib member, see *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*.

Changing the default recovery fault recording IEATDUMP data set name (RFRDSN)

In situations where an SDUMP cannot be used, 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 51.)

The default IEATDUMP data set name pattern is:

```
IDIRFRHQ.IDIRFR.&SYSNAME..D&YYMMDD..T&HHMMSS..S&SEQ.
```

To change the default data set name pattern, the RFRDSN setting in the IDIOPTLM configuration-options module is used.

The data set name pattern should include symbolic values to ensure that each allocated data set is unique. For example:

```
CL100'IDIHLQ.TDUMP.&&SYSNAME..D&&YYMMDD..T&&HHMMSS..S&&SEQ.'
```

For additional information about the use of symbols, see IDIOPTLM data set name symbol substitution on page 342.

Note: It is important to ensure that all users are able to allocate data sets with the pattern specified. See Managing recovery fault recording data set access on page 309 for more information.

If it is desirable that the recovery fault recording IEATDUMP data set name pattern contains local date and time instead of UTC, then change &YYMMDD to &LYYMMDD and &HHMMSS to &LHHMMSS.

Use the special high-level qualifier NULLFILE to disable RFR TDUMP capture. If this name is used, message IDI0136W on page 726 is issued.

Changing the action of Fault Analyzer when invoked via the CICS XDUREQ global user exit (CICSNOA)

The CICSNOA setting in the IDIOPTLM configuration-options module can be used to change the action of Fault Analyzer when invoked via the CICS® XDUREQ global user exit, and should only be considered if an installation requires special handling of CICS® transaction abends as outlined in the following.

When a prior CICS® exit program has specified a return code UERCBYP (suppress transaction dump), then Fault Analyzer by default continues to perform analysis. This behavior is equivalent to the way in which non-CICS abends are handled by the MVS™ pre-dump exit, IDIXDCAP.

When using the CICSNOA setting, a prior XDUREQ exit program return code of UERCBYP stops Fault Analyzer from performing analysis.

Specifying an alternative security server test data set name (SSCHKDSN)

The SSCHKDSN setting in the IDIOPTLM configuration-options module is used to provide an alternative data set name to be used when Fault Analyzer is checking the security server for normal response. This is done before real-time reanalysis as part of general validation of the execution environment. In the absence of an alternative data set name specified in the SSCHKDSN option, Fault Analyzer will use the default data set name SILLY99.DATA88.SET77.NAME66.CHECK55. The specified data set name does not need to exist, but must be syntactically valid.

Changing the default extended minidump (XDUMP) data set name pattern (XDUMPDSN)

An XDUMP is a data set that is tightly coupled with a fault entry, containing pages of storage in addition to those in the minidump.

The default XDUMP data set name pattern is:

```
IDIXDPHQ.XDUMP.&SYSNAME..D&YYMMDD..T&HHMMSS..S&SEQ.
```

Use the XDUMPDSN setting in the IDIOPTLM configuration-options module to change the pattern for naming extended minidump data sets.

The data set name pattern should include symbolic values to ensure that each allocated data set is unique. For example:

```
CL100'IDIHLQ.XDUMP.&&SYSNAME..D&&YYMMDD..T&&HHMMSS..S&&SEQ.'
```

For additional information about the use of symbols, see IDIOPTLM data set name symbol substitution on page 342.

One or more data set qualifiers must be used with the IDIXDUMP_HLQ XFACILIT profile to grant access to users who need to create the data sets. For additional information, see Using the XFACILIT resource class for XDUMP data sets on page 313.

If the high-level qualifier specified in the XDUMPDSN option is NULLFILE, then Fault Analyzer will not allocate an extended dump data set. The result of this is that user storage areas not directly involved in the event-related source line or statement might be missing from reanalysis of the fault entry, the DeferredReport option might be overridden if the MaxMinidumpPages option value is exceeded (potentially causing severe performance issues, especially under CICS), or additional space might be required in the TSO region for interactive reanalysis.

Specifying the copied SDUMP data set name pattern (SDUMPDSN)

Use the SDUMPDSN setting in the IDIOPTLM configuration-options module to specify the pattern for naming copied SDUMP data sets.

The data set name pattern should include symbolic values to ensure that each allocated data set is unique. For example:

```
CL100'IDIHLQ.SDUMP.&&SYSNAME..D&&YYMMDD..T&&HHMMSS..S&&SEQ.'
```

For additional information about the use of symbols, see IDIOPTLM data set name symbol substitution on page 342.

One or more data set qualifiers must be used with the `IDISDUMP_HLQ` XFACILIT profile to grant access to users who need to create the data sets. For additional information, see Using the XFACILIT resource class for copied SDUMP data sets on page 315.

If no data set name is specified in the SDUMPDSN option, or if the high-level qualifier is NULLFILE, Fault Analyzer will not create a copied SDUMP data set.

Ignoring IBM Application Delivery Foundation for z/OS Common Components options (NOIPVOPT)

Security server access violations can occur when Fault Analyzer attempts to process the IPVCNF00 parmlib member, if an installation does not provide users with general READ access to the system parmlib concatenation, and either:

- IBM Application Delivery Foundation for z/OS® Common Components is not installed.
- The IBM Application Delivery Foundation for z/OS® Common Components parmlib member, IPVCNF00, is not configured.

Set the NOIPVOPT option to 1 in the IDIOPTLM configuration-options module to prevent Fault Analyzer from attempting to find the IPVOPTLM configuration-options module or process the IPVCNF00 parmlib member.

For information about the IPVOPTLM configuration-options module and the IPVCNF00 parmlib member, see *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*.

Suppressing IDIXDCAP real-time analysis if prior exit RC=8 (PREVEXIT)

To suppress Fault Analyzer real-time analysis via the IDIXDCAP pre-dump invocation exit, if a prior pre-dump exit has set RC=8 to suppress the dump, set the PREVEXIT options field to 1.

The PREVEXIT options field replaces the deprecated IDISOPT2 SMP/E USERMOD.

IDIOPTLM data set name symbol substitution

This information about symbol substitution applies to data set names specified in the RFRDSN, SDUMPDSN, and XDUMPDSN options.

These data set names should contain MVS system symbols to ensure that each allocated data set is unique. In particular, &SEQ. (the Fault Analyzer sequence number variable) should be included in the name. The &SEQ. variable will be substituted by a 5-digit decimal number.

For data sets that are allocated as the result of a fault entry move or copy (see Copying history file entries on page 138 and Moving history file entries on page 140), or IDIUTIL IMPORT (see IMPORT control statement on page 448) operation, the symbols &JOBNAME. and &SYSNAME. will be resolved in accordance with their respective values when the fault entry is created. For example, if a fault entry was created on system SYS1 during real-time processing, and then later copied on system SYS2, the &SYSNAME. symbol will continue to be substituted by the value SYS1.

All symbols specified, other than &SEQ., &JOBNAME. and &SYSNAME., will be substituted using the ASASYMBM service with current system values. For information about available symbol names, see *z/OS MVS Assembler Services Reference*.

Ensure the final data set name, with all symbolic variables substituted by the longest possible values, will not exceed 44 characters. If it does, then Fault Analyzer will truncate the final data set name to 44 characters (or 43 characters if the last character becomes a period). In this case, the data set name might no longer be unique and therefore cause allocation errors.

Chapter 18. Setting up history files

Fault Analyzer requires at least one PDS or PDSE data set to be allocated as a fault history file.

Although DUMMY specification of the history file is supported, by using an actual data set you can take advantage of the features provided with the ISPF interface.

Attention:

- With one exception, do not rename a history file unless it is empty. If you do, associated dump data sets will no longer be usable and will not be deleted automatically when the fault entry is deleted.

  The exception to this rule is when following the procedure outlined in Changing the size of a PDSE history file on page 347.
- If a history file is deleted without first deleting all fault entries, any associated tightly-coupled dump data sets will be orphaned. Use IDIUTIL DELETE(ALL) to delete all fault entries before renaming or deleting a history file. For details see DELETE control statement on page 444.

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:

```
kilobytes = 500 * number-of-entries
```

where:

number-of-entries

The maximum number of concurrent entries that you want to hold in the history file.

A more accurate determination of the average fault entry size can be made by dividing the actual history file space utilization by the total number of members after having recorded a representative number of faults.

A suggested size of the initial history file is 100 cylinders. Allocate the history file with secondary space to avoid potential out-of-space conditions.

Allocating a PDS or PDSE for a history file

To allocate one or more history files, edit and submit the sample job IDISHIST.

This job allocates two PDSE data sets named `IDI.HIST` and `IDI.HIST.TEST` respectively. All users whose abending jobs are recorded in these history files need write access to the data sets. Refer to the instructions in the sample job for information about changes you might need to make to this job.

Notes:

1. PDSE data sets are recommended to be used for history files because of their automatic space management, IDIS subsystem management (see "Caching of history file $$INDEX data on page 319"), and superior directory integrity for shared usage. (You can also use PDS-managed history files.)
2. Fault Analyzer does not support PDSE version 2 history file data sets with greater than 0 member generations defined.

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 87.

Important: Allocate the history file with enough space to accommodate all data that is expected to be written to it. Otherwise, out-of-space conditions might occur that cause Fault Analyzer to abend (for example, system abend SE37).

To assist with space management of history files, Fault Analyzer provides the IDIUTIL batch utility, which can be used to:

- Delete old history file entries (DELETE control statement).
- Set up an AUTO-managed (or self-maintaining) history file (SetMinFaultEntries control statement).

These functions can also be performed using the Fault Analyzer ISPF interface.

In a sysplex, history files can be shared between any number of systems.

Related information

Managing history files (IDIUTIL utility) on page 441

AUTO-managed PDSE history files on page 345

Change fault history file settings on page 88

Sharing of history files across a sysplex on page 362

Allocating secondary space for history files

For PDSE history files in particular, it is important to allocate the data set with secondary space. Failure to do so might result in out-of-space conditions with subsequent loss of data.

Fault Analyzer normally auto-manages a PDSE history file within the *logical history file size* on page 90: the allocated size that is established the first time the history file reaches the minimum number of fault entries. (See AUTO-managed PDSE history files on page 345.) However, primarily due to the concurrent update capability of a PDSE, it is not always possible to guarantee the availability of free space as new fault entries are written. Therefore, it is crucial for Fault Analyzer to have the ability to extend the history file with secondary space in situations where the expected space is not available.

Because of the logical history file size concept employed by Fault Analyzer, the used size of the history file is automatically reverted to the original size when the next update occurs, and the secondary extent that was allocated is left unused until perhaps a similar situation that requires extra space occurs again. The important thing to note is that the history file does not continue to grow indefinitely.

When the IDIS subsystem detects a PDSE history file without secondary space, it issues message IDI0191W on page 738.

Converting history files between PDS and PDSE

When converting history files from PDS to PDSE data sets, or from PDSE to PDS data sets, set the minimum or maximum fault entries values. Failure to do this might cause unexpected results, including out-of-space conditions.

If the converted history file is a PDSE data set, do one of the following:

- Run IDIUTIL with the SETMINFAULTENTRIES control statement.
- Use the **Change Fault History File Settings** dialog in the ISPF interface to set the **Minimum Fault Entries** value.

If the converted history file is a PDS data set, do one of the following:

- Run IDIUTIL with the SETMAXFAULTENTRIES control statement.
- Use the **Change Fault History File Settings** dialog in the ISPF interface to set the **Maximum Fault Entries** value.

Related information

SETMINFAULTENTRIES control statement on page 447

SETMAXFAULTENTRIES control statement on page 446

Change fault history file settings on page 88

AUTO-managed PDSE history files

By default, a newly created PDSE history file is AUTO-managed.

PDSE history files can also be managed by using the IDIUTIL SetMinFaultEntries control statement or through the Fault Analyzer ISPF interface.

A history file is not actively AUTO-managed until at least 25 fault entries have been created. (The minimum number of fault entries is by default 25, but this value can be changed by using the IDIUTIL batch utility SetMinFaultEntries control statement, or by using the Fault Analyzer ISPF interface.) Until 25 fault entries have been created, the history file data set is permitted to increase in size by allocation of secondary extents.

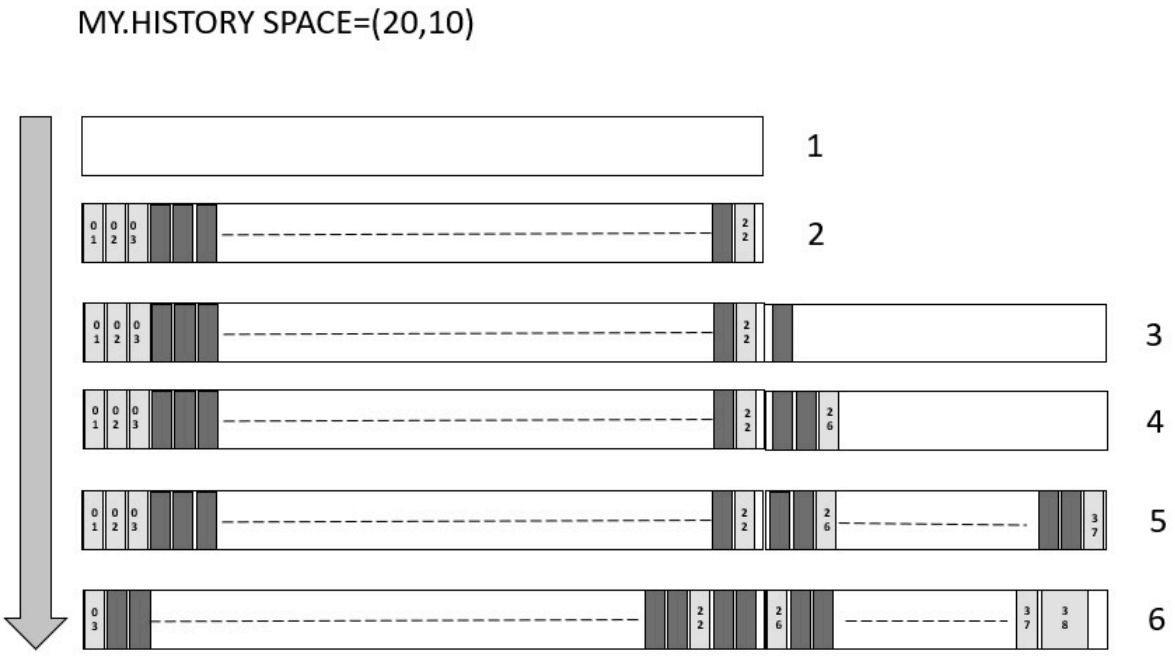
As soon as 25 fault entries have been exceeded, Fault Analyzer records the current size of the allocated history file. This size is the *logical history file size*. Additional fault entries are still permitted to be written to the history file to use up all allocated space.

When there is no longer sufficient unused space in the history file, then Fault Analyzer deletes the oldest eligible fault entries, in excess of 25, to provide the space required for writing new fault entries. This process generally prevents the allocation of additional data set extents and ensures that the history file does not continue to grow indefinitely with related out-of-space conditions.

For information about the assignment and reuse of fault IDs, see Fault history files on page 24.

Example

Figure 194. AUTO-managed PDSE history file example



1. Primary allocation: A new history file is allocated with primary space 20 cylinders and secondary space 10 cylinders.
2. Fault entries are written to the history file. By the time the initial 20 cylinders have been used, let us assume the history file contains 22 fault entries.
3. The next fault entry triggers a secondary allocation. When the 23rd entry is written, an additional 10-cylinder extent is allocated because the minimum of 25 fault entries has not yet been reached.
4. Once the 26th fault entry is written, the current size of the allocated history file is recorded as the logical history file size. In this example: 20 cylinders + 10 cylinders = 30 cylinders
5. Fault entries continue to be written until all 30 cylinders have been used. Let us assume that when this happens, the history file contains 37 fault entries.
6. Before the 38th fault entry is written, fault entries are deleted (oldest first) until the new fault entry fits within the allocated 30 cylinders.

Regardless of size, the deletion of old fault entries stops when there are 25 or fewer fault entries left.

Related information

SETMINFAULTENTRIES control statement on page 447

Change fault history file settings on page 88

Changing the size of a PDSE history file

You can change the allocated size of a PDSE history file.

1. Stop the IDIS subsystem.
2. Rename the original history file to a unique name to ensure that the file is not used as a history file by others. For example, append ".OLD" to the history file name.
3. Allocate a new, larger history file with secondary extents using a unique name that is different from the original history file (for example, by appending ".NEW").
4. Copy the complete contents of the ".OLD" history file to the ".NEW" history file using IEBCOPY or ISPF option 3.3.
5. Using **File > Change Fault History File Settings** from the action bar, set the Logical History File Size (Pages) value to 0. For additional information about this setting, see Change fault history file settings on page 88.
6. Rename the ".NEW" history file to the original history file name.
7. Start the IDIS subsystem.
8. Delete the ".OLD" history file using ISPF option 3.2 or similar.

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 59.

A view is essentially a list of history files, that are all displayed together using the Fault Analyzer ISPF interface.

If not using the XFACILIT class to provide access to individual history file fault entries (see Managing history file fault entry access on page 363), 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 350.

The list of history file data set names to make up a view are simply placed into a member in another PDS that was created specifically to hold the view definitions. This member is generally an installation-wide data set to which all users have read access. It is pointed to by the IDIVIEWS data sets in the DataSets option. The names of the members are used as the view names. The users are given a list of the view names when using the ISPF options pull down to change the history file and pressing F4 to list views. The first line of these members can be an optional comment starting with an asterisk in column one. If this comment line exists in a view member it is displayed next to the view name in the selection screen. The comment line can provide a description of the view to assist the users choice.

Figure 195: Sample view data set member on page 348 shows an example of a view data set member.

Figure 195. Sample view data set member

```
* Department P024 history files ❶
P024.&SYSNAME..CICS.HIST ❷
P024.IMS.HIST ❸
* Include the installation-wide dehistory file ❹
IDI.HIST ❺
```

In the above example:

❶

This line is a comment (asterisk in column 1). Because this comment is on the first line of the view member, it is also used as the view description when displaying the list of views using the Fault Analyzer ISPF interface.

❷, ❸ **and** ❺

These are the fully qualified history file names that are displayed simultaneously through the Fault Analyzer ISPF interface when selecting this view. Each data set name must be specified on a single line, starting in column 1. Surrounding the data set names in single quotes is optional. There is no limit on the total number of data set names that can be specified in a single view.

The symbol substitution rules that apply to the data set names that are specified in a view member are the same as those that apply to the data set names that are specified in the IDIVIEWS suboption of the DataSets option. See DataSets option data set name substitution symbols on page 585 for more information.

❹

This line is a comment that is ignored by the Fault Analyzer ISPF interface.

No specific view data set attributes are required, except that it must be a PDS or PDSE. 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 348) or -Match (see Specifying initial fault entry selection criteria on page 349) specification. As an example, use record format VB with a logical record length of 32,756 bytes. The view member must not contain sequence numbers.

For easier reference to fault IDs across multiple history files, the ability to set up to three alphabetic fault prefix characters for individual history files is available. Using different prefix characters between different groups helps users recognize the owning group when viewing faults in a composite view display. The fault prefix characters can be set or changed using the IDIUTIL batch utility SETFAULTPREFIX command.

Specifying a default column layout

A default column layout can be specified by using the -HistCols keyword in column 1 of any record in a view.

Tip: Avoid specifying this on the first record if a view description is also required.

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 FAISPFopts(HistCols(...)) option (see FAISPFopts on page 603). 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 65).

The complete -HistCols specification must fit on a single view member record. Continuation over multiple records is not supported.

No part of the -HistCols specification is case sensitive, including the keyword itself, and any number of blank characters, or none, can precede the open parenthesis following the keyword.

If multiple -HistCols specifications are found in a single view member, then only the last one takes effect.

Here is an example of a view data set member specifying a default column layout.

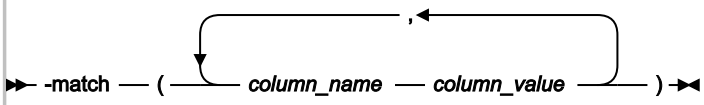
```
* All IMS history files
-HistCols(IMS_Pgm  Jobname  User_ID  System   Abend Date       Time    )
SYS1.IMS.HIST
SYS2.IMS.HIST
```

Specifying initial fault entry selection criteria

Initial fault entry selection criteria can be specified by using the -Match keyword in column 1 of any record in a view.

Tip: Avoid specifying this on the first record if a view description is also required.

Figure 196. Syntax



The *column_value* permits the use of wildcards. The supported wildcard characters are an asterisk (*) to indicate zero, one or more characters, and a percent sign (%) to indicate a single required character. Only table data rows that match the specified string are shown.

The complete -Match specification must fit on a single view member record. Continuation over multiple records is not supported.

No part of the -Match specification is case sensitive, including the keyword itself, and any number of blank characters, or none, can precede the open parenthesis following the keyword.

If multiple -Match specifications are found in a single view member, then only the last one takes effect.

Here is an example of a view data set member specifying initial fault entry list criteria.

```
* My CICS view
-Match(CICS_Trn *)
PROD.CICS.HIST
```

View considerations if not using XFACILIT resource class

To do the most effective setup of views for an installation which is not using the XFACILIT resource class to control history file fault entry access, an understanding of all of the different sets of users of Fault Analyzer is required. The main aspects of each user group that need to be determined are:

1. Their data set security profiles for data sets with write access.
2. The need to review or work on faults from jobs submitted by other user groups.
3. The need for read access only, or no access at all, to fault entries between groups.

These requirements are already understood to some degree by the installation security administrators, as they are the ones primarily dealing with data set security requirements. The extra element is any requirement for fault visibility between the groups.

Without using the XFACILIT resource class to control history file fault entry access, the security of history files is the data set security in your system. For users' jobs to record faults in the fault history file, they need write (update) access to the history file that the job uses. All users in the same group thus have read and delete access to the faults in a history file set up for the group.

It is generally better to maintain separate fault history files for different user groups, providing the write (update) access only to the history file their jobs use. However, this approach makes it harder for a user to look at the faults across the environment if they are in many different history files. To solve this problem, create a "view" to let a use see a composite view of many history files in the one Fault Analyzer ISPF display.

Managing history files across MVS systems without shared DASD

Views can be used for easy access to fault entries in more than one history file. However, this easy access is only possible when all history files in a view are accessible from the same MVS™ system.

If the systems share DASD, they can simply use a common history file. Two other solutions are provided for sending fault entries from the system on which the fault occurred to another system for analysis:

- The first solution uses the Notification user exit and requires no manual intervention. See Automated implementation on page 350.
- The second solution uses the Formatting user exit to transmit fault entries "on demand" during interactive reanalysis. See On-demand implementation on page 359.

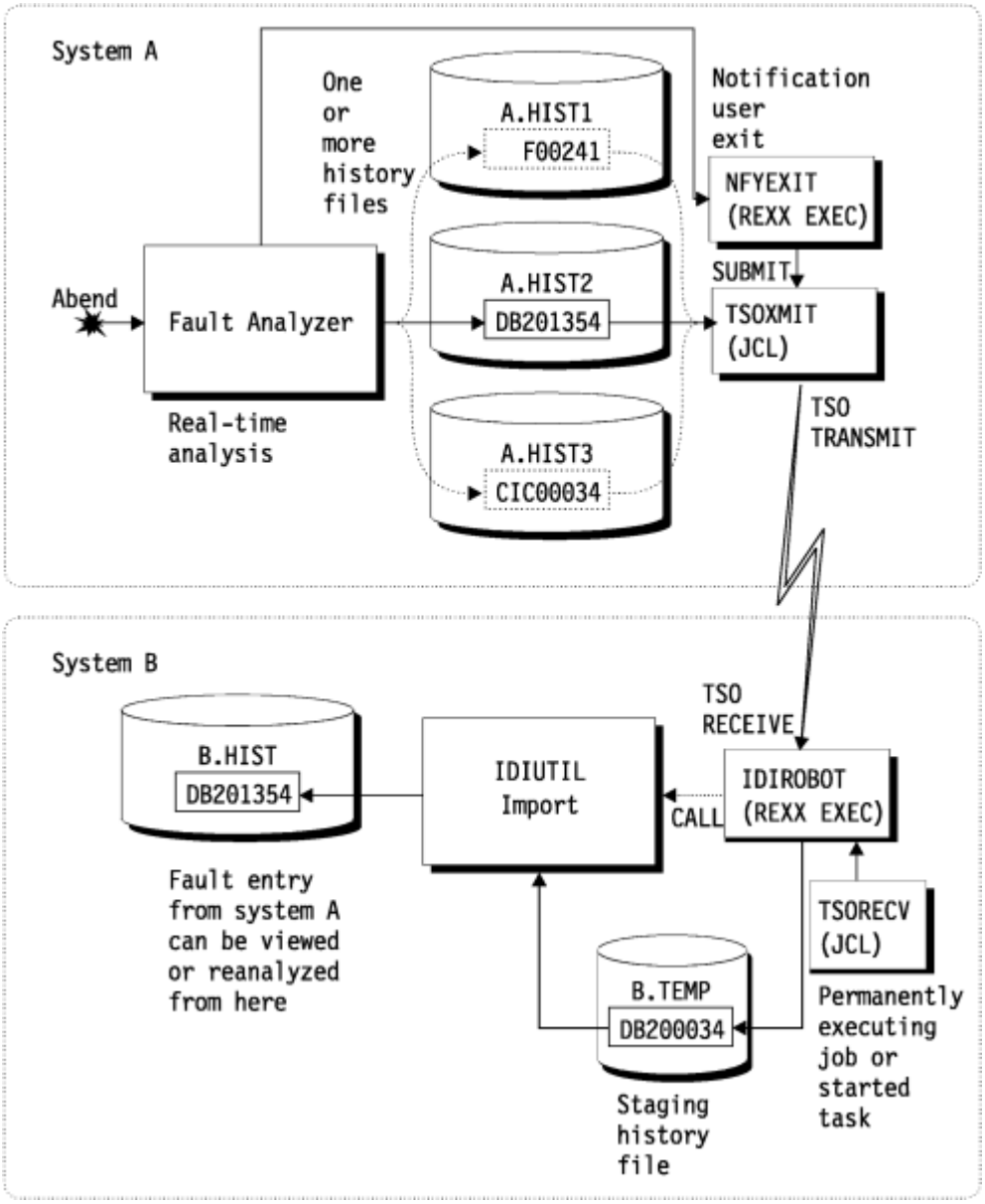
Automated implementation

Given two MVS™ systems, A and B, which do not share DASD, fault entries that are created on system A can be copied automatically to system B for viewing or reanalysis by using the Fault Analyzer ISPF interface.

The entries are copied by using a Notification user exit on system A, which submits a TSO batch job to transmit fault entries as PDS members to a dedicated user ID on system B. On system B, a continually running batch TSO job receives fault entries into a staging data set. It then calls the IDIUTIL batch utility to import the fault entry into a local history file.

Figure 197: Fault entry import by way of TSO XMIT/RECEIVE on page 351 shows this concept.

Figure 197. Fault entry import by way of TSO XMIT/RECEIVE



See Customizing Fault Analyzer by using user exits on page 464 for information about user exits in general. See Notification user exit on page 498 for information about the Notification user exit specifically. See Managing history files (IDIUTIL utility) on page 441 for information about the IDIUTIL batch utility.

NFYEXIT: sample Notification user exit

NFYEXIT is available in softcopy format as member IDISXNFY in data set IDI.SIDISAM1.

Figure 198. Sample Notification user exit (NFYEXIT) to submit batch TSO XMIT job

```
nodeid   = 'MVSB'                       /* <--- verify/change */        ❶
userid   = 'IDIROBOT'                   /* <--- verify/change */        ❷
jobcard  = '//NOTIFY   JOB  MSGCLASS=Z'  /* <--- verify/change */        ❸
/*********************************************************************/  ❹
/* #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_IDIHIST ¬= "MY.HISTFILE" the exit 0                   */
/*********************************************************************/
"MAKEBUF"
queue jobcard
queue '//*************************************************************'
queue '//* Export fault entry'
queue '//*************************************************************'
queue "//EXPORT   EXEC PGM=IDIUTIL"
queue "//DD1      DD DISP=(,PASS),"
queue "//            SPACE=(CYL,(10,100,5),RLSE),"
queue "//            DCB=(DSORG=PO,RECFM=VB,LRECL=10000)"
queue "//SYSPRINT DD SYSOUT=*"
queue "//SYSIN    DD *"
queue "  EXPORT("ENV.IDIHIST"("ENV.FAULT_ID"),DD1)"
queue "/*"
queue '//*************************************************************'
queue '//* Terse the export data set'
queue '//*************************************************************'
queue "//TERSE    EXEC PGM=AMATERSE,PARM='PACK'"
queue "//SYSPRINT DD SYSOUT=*"
queue "//SYSUT1   DD DISP=SHR,DSN=*.EXPORT.DD1"
queue "//SYSUT2   DD DISP=(,PASS),"
queue "//            SPACE=(CYL,(10,100),RLSE)"
queue "//SYSPRINT DD SYSOUT=*"
queue '//*************************************************************'
queue '//* Perform TSO XMIT of the exported and tersed fault entry'
queue '//*************************************************************'
queue "//XMIT     EXEC PGM=IKJEFT01"
queue "//DD1      DD DISP=SHR,DSN=*.TERSE.SYSUT2"
queue "//SYSTSPRT DD SYSOUT=*"
queue "//SYSTSIN  DD *"
q_rec("  XMIT" nodeid"."userid "DDNAME(DD1) -")
q_rec("  NONOTIFY")
queue '/*'
/* 'Submit' the stacked TSO batch job */
n = queued()
"IDIALLOC DD(DD1) SYSOUT PGM(INTRDR)"
if rc = 0 then do /* allocation worked so generate output */
  address mvs "EXECIO" n "DISKW DD1 (FINIS"
  "IDIFREE DD(DD1)"
  say 'Fault entry' ENV.FAULT_ID 'sent to' nodeid'.'userid
end
else do
  "IDIWTO Allocation of INTRDR failed"
  say 'Fault entry' ENV.FAULT_ID 'job submission failure'
end
exit 0

/* Pad record with blanks to 80 bytes.                               */
q_rec: procedure
parse arg rec
if (length(rec) < 80) then rec = rec||copies(' ',80-length(rec))
queue rec
return 0
```

Notes:

❶ 'nodeid' specifies the target system to which the fault entry is sent.

❷ 'userid' specifies the user ID for which fault entries are received on the target system. Use this user ID solely to receive fault entries.

❸ Ensure that the job card adheres to local standards.

❹ You can add checks here to see whether a fault is eligible to be sent to another system. The example shows how the user ID or history file name can be used, but any fields in the ENV or NFY data areas can be checked.

The exit in Figure 198: Sample Notification user exit (NFYEXIT) to submit batch TSO XMIT job on page 353 can be called for any faults that occur on system A. To facilitate this exit, add these options to the IDICNFxx config member, where *exec.lib* is your REXX EXEC PDS or PDSE data set:

```
DataSets(IDIEXEC(exec.lib))
Exits(NOTIFY(REXX(NFYEXIT)))
```

IDIROBOT: sample REXX exec to receive fault entries

IDIROBOT is available in softcopy format as member IDISROBT in data set IDI.SIDISAM1. This exec performs the following actions:

1. Receive files for the IDIROBOT user into a staging data set from where they are imported into a local history file by using the IDIUTIL batch utility.
2. Create the IDIUTIL IMPORT user exit, IDIROBEX (see Figure 244: Sample REXX IDIUTIL user exit on page 509).

Figure 199. Sample TSO receive REXX exec (IDIROBOT) part 1

```
histfile = 'B.HIST'              /* <--- verify/change */ ❺
temphist = 'B.TEMP'              /* <--- verify/change */ ❻
seconds  = '60'                  /* <--- verify/change */ ❼
use_exit = 'Y'                   /* <--- Y|N. verify/change */  ❽
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
```

Figure 200. Sample TSO receive REXX exec (IDIROBOT) part 2

```
/* 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
  if faultid <> "" then do
    /* Receiving a PDS/E.                                          */

    say 'Receiving' dsn'('faultid') from' node'.'fromid
    queue "DSN('"temphist"')"
    queue 'END'
    'RECEIVE'
  end
  else do
    /* Receiving a sequential data set - assume AMATERSE PACKed.   */
    say 'Receiving' dsn 'from' node'.'fromid
    queue "DSN('"temprecv"')"
    queue 'END'
    'RECEIVE'

    /* Perform AMATERSE UNPACK.                                    */
    "ALLOC DD(SYSPRINT) DUMMY"
    "ALLOC DD(SYSUT1) DA('"temprecv"') SHR"
    "ALLOC DD(SYSUT2) DSN('"temphist"'),
           NEW CATALOG UNIT(SYSALLDA) RECFM(V B) LRECL(10000),
           CYLINDERS SPACE(10,100) DIR(5)"
    address tso "CALL *(AMATERSE) 'UNPACK'"
    say 'UNPACK rc =' RC

    /* Get fault ID (member name).                                 */
    "LISTDS '"temphist"' MEMBERS"
    /* Sample output:                                             */
    /* FRED.$$TEMP$$.HIST                                         */
    /* --RECFM-LRECL-BLKSIZE-DSORG                                */
    /*   VB    10000 27998   PO                                   */
    /* --VOLUMES--                                                */
    /*   E$US21                                                   */
    /* --MEMBERS--                                                */
    /*   F01103                                                   */
    mbr_start = 0
    do i = 1 to var.0
      /*say "var."i"='"var.i"'"*/
      if mbr_start = 0 then do
        if strip(var.i) = "--MEMBERS--" then do
          mbr_start = i + 1
          leave
        end
      end
    end
    if mbr_start = var.0 then do
      /* One, and only one, member.                               */
      faultid = strip(var.mbr_start)
    end
    else do
      say 'ERROR: More than one member found in data set' temphist,
          '- terminating'
      exit 12
    end
    'FREE DD(SYSUT2)'
    'FREE DD(SYSUT1)'
    'FREE DD(SYSPRINT)'

    "DELETE '"temprecv"'"
  end
```

Figure 201. Sample TSO receive REXX exec (IDIROBOT) part 3

```
    /* The target history file in the 'histfile' variable could be   */
    /* determined here based on any of the initialized variables     */
    /* dsn, fromid, node or faultid.  This sample EXEC uses a single */
    /* history file only.                                            */ ❾

    /* Perform IDIUTIL IMPORT.                                       */
    'ALLOC DD(SYSIN) NEW REU UNIT(VIO) RECFM(F B) LRECL(80)'
    "ALLOC DD(SYSPRINT) SYSOUT"
    if use_exit = 'Y' then
      parms.1  = "EXITS(IMPORT(REXX(IDIROBEX)))"
    else
      parms.1  = "* Using IDIOPTLM for dump data set names"
    parms.2  = "IMPORT("histfile","
    parms.3  = "  "temphist"("faultid"),PACKAGE)"
    parms.0  = 3
    "EXECIO * DISKW SYSIN (STEM parms. FINIS"
    address tso "CALL *(IDIUTIL)"
    say 'IMPORT rc =' RC
  end
  else do
    /* Sleep for 60 seconds before attempting to receive again */
    address tso "call *(idisleep) '"seconds"'"
  end
end
```

Notes:

❺

This item is the name of the target history file in which the received fault entries are placed. To select the history file that is based on where the fault originally occurred, see ❾.

❻

This item is a staging data set that is used for the TSO receive command and from which fault entries are imported into the target history file.

Important: Do not use a preallocated data set. Let the exec allocate and delete this staging data set for each fault received, as shown in the sample provided.

For the IDIROBOT exec and IDIUTIL IMPORT processing to work, the staging data set must never be used as a regular history file and must never contain more than a single member. If it is used as a regular history file (for example, if it is displayed using the Fault Entry List display, or used as the target of an IDIUTIL FILES or LISTHF control statement), then a $$INDEX member will likely be created, which will cause the processing not to work. Also, it is possible that the data set becomes IDIS subsystem managed, which will subsequently result in serialization issues.

By ensuring that the staging data set only exists for the duration of the receive and IMPORT processing, the possibility that these issues will occur is eliminated.

❼

The IDIROBOT exec enters a WAIT state to preserve resources between checking for fault entries to be received. The time interval in number of seconds between receiving fault entries can be specified here. All fault entries on the JES output queue for the chosen user ID are received, then the IDIROBOT exec enters the WAIT.

❽

- If you set the RFRDSN, XDUMPDSN, and SDUMPDSN options to valid data set name patterns in the IDIOPTLM configuration-options load module, there is no need to use the IDIROBEX user exit. (See Customize Fault Analyzer by using an IDIOPTLM configuration-options module on page 337.) In this case, set "use_exit" to 'N'.
- If you use the IDIROBEX user exit, any dump data set names provided by the exit override the equivalent option setting in IDIOPTLM.

❾

The sample exec uses only a single target history file for all received fault entries. It is possible to assign a target history file that is based on one of these items:

- The original history file name (in variable 'dsn').
- The sending user ID (in variable 'fromid').
- The node ID from where it was sent (in variable 'node'),
- The fault ID itself (in variable 'faultid').

Figure 202: Sample TSO batch job to execute IDIROBOT exec (IDISTSOB) on page 359 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.

Figure 202. Sample TSO batch job to execute IDIROBOT exec (IDISTSOB)

```
//IDISTSOB JOB  <job card parameters>
// SET EXECDSN=exec.lib                          <--- verify/change
//TSOBATCH EXEC PGM=IKJEFT01
//SYSEXEC  DD   DISP=SHR,DSN=&EXECDSN.
//SYSPRINT DD   SYSOUT=*
//SYSTSPRT DD   SYSOUT=*
//SYSIN    DD   DUMMY
//SYSTSIN  DD   *
  IDIROBOT
/*
//IDIEXEC  DD DISP=SHR,DSN=*.TSOBATCH.SYSEXEC
//IDITRACE DD SYSOUT=*
```

Important: 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 data set and all history files used as targets for imports.

Because the IDIROBOT exec never exits, the IDISTSOB job executes indefinitely. However, the exec causes the job to enter a WAIT state between attempts to receive incoming data to prevent using unnecessary resources. To end the job, use the MVS™ CANCEL command during a period of inactivity. Alternatively, the exec could be made to recognize a special file that if sent to the selected user ID could trigger the exit to terminate.

A started task could be defined instead to execute this JCL in order to prevent tying up a JES initiator.

On-demand implementation

You can use a Formatting user exit to transmit fault entries that match certain criteria to another system for analysis. This process is initiated as required from within the interactive reanalysis report.

The following figure shows the sample Formatting REXX user exit, IDIXMIT. It is available in softcopy format as member IDIXMIT in data set IDI.SIDISAM1.

Figure 203. Sample Formatting user exit (IDIXMIT) to submit batch TSO XMIT job (Part 1 of 2)

```
nodeid   = 'MVSB'                       /* <--- verify/change */    ❶
userid   = 'IDIROBOT'                   /* <--- verify/change */    ❷
jobcard  = '//NOTIFY   JOB  MSGCLASS=Z'  /* <--- verify/change */    ❸
"MAKEBUF"
queue jobcard
queue '//*************************************************************'
queue '//* Export fault entry'
queue '//*************************************************************'
queue "//EXPORT   EXEC PGM=IDIUTIL"
queue "//DD1      DD DISP=(,PASS),"
queue "//            SPACE=(CYL,(10,100,5),RLSE),"
queue "//            DCB=(DSORG=PO,RECFM=VB,LRECL=10000)"
queue "//SYSPRINT DD SYSOUT=*"
queue "//SYSIN    DD *"
queue "  EXPORT("ENV.IDIHIST"("ENV.FAULT_ID"),DD1)"
queue "/*"
queue '//*************************************************************'
queue '//* Terse the export data set'
queue '//*************************************************************'
queue "//TERSE    EXEC PGM=AMATERSE,PARM='PACK'"
queue "//SYSPRINT DD SYSOUT=*"
queue "//SYSUT1   DD DISP=SHR,DSN=*.EXPORT.DD1"
queue "//SYSUT2   DD DISP=(,PASS),"
queue "//            SPACE=(CYL,(10,100),RLSE)"
queue "//SYSPRINT DD SYSOUT=*"
queue '//*************************************************************'
queue '//* Perform TSO XMIT of the tersed export data set'
queue '//*************************************************************'
queue "//XMIT     EXEC PGM=IKJEFT01"
queue "//DD1      DD DISP=SHR,DSN=*.TERSE.SYSUT2"
queue "//SYSTSPRT DD SYSOUT=*"
queue "//SYSTSIN  DD *"
q_rec("  XMIT" nodeid"."userid "DDNAME(DD1) -")
q_rec("  NONOTIFY")
queue '/*'
```

Figure 204. Sample Formatting user exit (IDIXMIT) to submit batch TSO XMIT job (Part 2 of 2)

```
/* 'Submit' the stacked TSO batch job.                              */
n = queued()
"IDIALLOC DD(DD1) SYSOUT PGM(INTRDR)"
if rc = 0 then do /* if alloc worked */
  address mvs "EXECIO" n "DISKW DD1 (FINIS"
  "IDIFREE DD(DD1)"
end
else ,
   "IDIWTO ALLOCATION OF INTRDR FAILED"
 "DROPBUF"
/*********************************************************************/ ❹
/* #Optionally, update the user title field to show that the fault   */
/* has been sent.                                                    */
/* For example:                                                      */
/* ENV.USER_TITLE = "Sent to" nodeid "on" DATE()"."                  */
/*********************************************************************/
/*********************************************************************/ ❺
/* #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.'"                                               */
/*********************************************************************/
exit 0

/* Pad record with blanks to 80 bytes.                              */
q_rec: procedure
parse arg rec
if (length(rec) < 80) then rec = rec||copies(' ',80-length(rec))
queue rec
return 0
```

Notes:

❶

'nodeid' specifies the target system to which the fault entry is sent.

❷

'userid' specifies the user ID for which fault entries are received on the target system.

❸

Ensure that the job card adheres to local standards.

❹

Uncommenting this code provides an optional method for identifying the fault entry as having been sent by providing the date on which this sending occurred, and the system to which it was sent, in the user title field.

❺

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 exec described in Automated implementation on page 350 or it might be any user ID and node for which the receive, and subsequent import, is done manually.

Manual import should be performed using IDIUTIL IMPORT with the PACKAGE option. See IMPORT control statement on page 448.

Using the on-demand implementation

While analyzing a fault entry, enter the following primary command:

```
EXEC IDIXMIT
```

The IDIXMIT name can be any other name you have called your copy of the IDIXMIT sample exec.

Sharing of history files across a sysplex

Fault Analyzer uses XCF messaging from IDIS subsystems to efficiently share PDSE history files across all MVS™ images in a sysplex.

Combining this method with the PDSESHARING(EXTENDED) option in the IGDSMSxx member of SYS1.PARMLIB allows concurrent writing and reading of individual fault entries from all MVS™ images in a sysplex without contention.

Sharing of PDS history files across a sysplex can cause unwanted contention if high activity to the same history file occurs concurrently from multiple MVS™ images, because the access method requires that the entire data set is enqueued upon for any updates. This problem does not apply to PDSE data sets, which only require serialization at the member level and are therefore a better choice.

Some environments have previously been set up with the unusual condition of not sharing master catalogs (or a user catalog) between all MVS™ images in a sysplex, and thus permitting individual MVS™ images to use the same data set name cataloged to a different DASD volume. Do not use this setup with Fault Analyzer history files, as the common data set name causes unnecessary contention through the IDIS subsystem and ENQ mechanisms between the MVS™ images. If different history files are desired between MVS™ images, then the use of the &SYSCLONE. substitution variable in the history file data set name should be considered as the best alternative, since the unique naming then removes the common name ENQ contention and provides a more logical and practical management.

An alternative to using shared history files is to use the Fault Analyzer ISPF interface "View" facility, which allows multiple history files to be viewed simultaneously. For details of this facility, see Setting up views on page 347.

If sharing history files between multiple MVS™ images, the following must be observed:

- Systems sharing a PDSE must be members of the same sysplex and it is recommended that all systems are running with PDSESHARING(EXTENDED) in the IGDSMSxx member of SYS1.PARMLIB. (Note that the first sysplex member to IPL determines the sharing mode used and forces all subsequent members that join the sysplex to run in that mode, whether EXTENDED or NORMAL.)
- XCF must be active.
- GRS (or a functionally equivalent subsystem) must be running. PDSE serialization is managed using resource major names SYSZIGW0 and SYSZIGW1. There is no need to make changes to GRS RNL lists, except if a serialization product other than GRS is used.

It is possible for OPEN errors to occur against PDSE data sets that are shared across a sysplex, irrespective of the above considerations. This situation might happen as a result of a cross-system sharing conflict, typically resulting in message IEC143I RC 213-70.

For more information about sharing PDSE data sets in a sysplex, see chapter *"PDSE Sharing and serialization"* in the Redbooks® publication *"Partitioned Data Set Extended Usage Guide"* (a copy can be downloaded from www.redbooks.ibm.com) and the *"Sharing PDSEs"* section in chapter 28 *"Processing a PDSE"* in the *"DFSMS: Using Data Sets"* manual.

If an installation is required to use PDSESHARING(NORMAL) then the IDIS subsystem is also required as this provides history file access management aimed at reducing cross-system sharing conflicts. However, if a history file is accessed outside of Fault Analyzer, then message IEC143I RC 213-70 might still occur.

No special Fault Analyzer options are required to use sysplex-shared fault history files.

Tip: Make the setting of the IDIS subsystem PARM='UPDINDEX' option, or the absence thereof, the same for all IDIS subsystems in the sysplex. For details about this option, see Caching of history file $$INDEX data on page 319.

Managing history file fault entry access

It is possible to manage access to history files using security server data set profiles only, for example if a given history file is only used within a single department where all users are equal with regards to access authority. In this case, all users typically have at least UPDATE access to the history file and are thus able to create new fault entries, as well as view, modify, or delete any existing fault entries, regardless of who created them.

If more control over access to individual fault entries is required, then Fault Analyzer supports the use of the XFACILIT resource class as described in the following.

Note: Because XFACILIT access is associated with some degree of processing overhead, you might want to use history files with normal data set profile access for performance critical applications, such as CICS® or IMS™.

Using the XFACILIT resource class for history file fault entries

If the required access to a history file fault entry is not already available via normal data set profiles, you can use the XFACILIT resource class to provide the access through Fault Analyzer.

To use the XFACILIT process for a history file, restrict the access available to the user via normal data set profiles, and then enable the access via the Fault Analyzer checked XFACILIT profiles. It is not possible to restrict Fault Analyzer access using the XFACILIT resource class beyond what the normal data set profiles allow, which is why the normal history file access should be set to UACC(NONE) for this setup.

The design is to provide permission via XFACILIT, not restriction. For example, if a user cannot even browse a history file using ISPF 3.4, they could still be permitted to use the history file fault entries via Fault Analyzer with the XFACILIT profiles.

There are two XFACILIT class profile names used by Fault Analyzer:

```
IDIHIST_GROUP_DSN.group.hist-dsn
IDIHIST_USERID_DSN.userid.hist-dsn
```

where:

group* and *userid

are the security server default GROUP and USER ID that are associated with the current user (in case of creating a new fault entry), or the security server GROUP and USER ID that are associated with a fault entry (in the case of viewing, updating, or deleting an existing fault entry).

The security server GROUP and USER ID that are associated with a fault entry are those that were current for the user who initially created the fault entry.

hist-dsn

The name of the history file in which the fault entry resides.

Examples of XFACILIT class profile names could be:

```
IDIHIST_GROUP_DSN.PAYROLL.IDI.HIST
IDIHIST_USERID_DSN.USER01.SYSA.PROD.HIST
```

If access is not available by normal data set profiles, then Fault Analyzer checks for access to a given history file fault entry using both of the above XFACILIT class profile names, that is, using group as well as user ID. If either is found to provide the access required, then the history file action is performed.

The XFACILIT class profile access levels used by Fault Analyzer translate to Fault Analyzer actions as follows:

Table 9. XFACILIT profile access levels required for Fault Analyzer actions

Action	Example	XFACILIT access level
Read	Viewing the saved report or performing reanalysis.	READ
Write or create	Updating user notes in an existing fault entry, or creating a new fault entry in a history file.	UPDATE or CONTROL
Delete	Explicit deletions using the D or DD line command only.	ALTER

XFACILIT implementation example 1

Given a history file that is named IDI.COMMON.HIST, prevent general access to the data set using, for example:

```
ADDSD 'IDI.COMMON.HIST' UACC(NONE)
```

Define the following XFACILIT profile and access (repeat for each instance of *group*):

```
RDEFINE XFACILIT IDIHIST_GROUP_DSN.group.IDI.COMMON.HIST XFACILIT UACC(NONE)
PERMIT IDIHIST_GROUP_DSN.group.IDI.COMMON.HIST XFACILIT CLASS(XFACILIT) ID(group) ACCESS(UPDATE)
```

Additional groups or users can be given access to the XFACILIT string as appropriate.

The resulting access to the IDI.COMMON.HIST history file is such that:

- Fault entries are accessible through Fault Analyzer only.
- Fault entries can be viewed or reanalyzed by:
  - The users who created the fault entries
  - Anyone else who is either a member of the same group, or who was granted explicit access to the XFACILIT profile containing the fault entry creator's default group ID

XFACILIT implementation example 2: Using global access table

To use a history file TEST.ZZ.HISTORY.DEFAULT and have fault entry access protected by group ID and user ID, first prevent general access to the data set:

```
ADDSD 'TEST.ZZ.HISTORY.**' UACC(NONE)
```

Then, to allow Fault Analyzer to grant access to the individual fault entry members in the PDS or PDSE 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.

Note:

1. For RACF®, the &RACUID and &RACGPID only works for profiles listed in the global access table.
2. It is a RACF® requirement, given how Fault Analyzer determines access authorization, that any XFACILIT profiles in the global access table are backed by a matching real XFACILIT profile. For example, add the following real XFACILIT profiles in order to enable the global access table profiles used here:

   ```
   PROFILE NOPREF
   RDEFINE XFACILIT IDIHIST_GROUP_DSN.*.TEST.ZZ.HISTORY.** UACC(NONE)
   ```

```
RDEFINE XFACILIT IDIHIST_USERID_DSN.*.TEST.ZZ.HISTORY.** UACC(NONE)
SETR REFR RACLIST(XFACILIT)
```

The above XFACILIT definitions cover all history file data set names starting with the TEST.ZZ.HISTORY qualifiers, for example:

TEST.ZZ.HISTORY.DEFAULT
TEST.ZZ.HISTORY.PAYROLL
TEST.ZZ.HISTORY.CICS.SYS01

XFACILIT implementation example 3: Using ACF2

OEM security servers need the commands converted to the specific product, such as the following ACF2 commands. The requirement is that the SAF RACROUTE requests give the equivalent return information for the OEM product, as would happen with RACF®. The following is an example for ACF2.

```
$KEY(TEST)
 ZZ.HISTORY.- UID(<string for MVS support>) READ(A) WRITE(A) ALLOC(A) EXEC(A)
 ZZ.HISTORY.- UID(*)
```

Here, the TEST.ZZ.HISTORY.* history files can only be directly accessed by the users in the <string for MVS™ support> list.

```
$KEY(IDIHIST_USERID_DSN) TYPE(XFC) or $KEY(IDIHIST_GROUP_DSN) TYPE(XFC)
 -.TEST.ZZ.HISTORY.- UID((<string for MVS support>) ALLOW
 -.TEST.ZZ.HISTORY.- UID(*) SERVICE(READ,UPDATE) ALLOW
```

TYPE(XFC) is the default for XFACILIT class. This setup only allows general users READ and UPDATE access, only MVS™ support can do explicit delete of fault entries. Automatic deletion is still done by Fault Analyzer according to the SetMaxFaultEntries/SetMinFaultEntries setting of each history file.

Using the IDIXFXIT user exit

If access was not granted via the IDIHIST_GROUP_DSN.*group.hist-dsn* or IDIHIST_USERID_DSN.*userid.hist-dsn* XFACILIT profiles, the optional IDIXFXIT user exit is called. If the IDIXFXIT exit is available, then it might ultimately grant the fault entry access. This approach might be useful if an installation has previously employed a different access authorization scheme and wants to continue using this instead of, or in conjunction with, the XFACILIT profiles.

The IDIXFXIT user exit must reside in an APF-authorized library and be available as a load module in LINKLIST with the name IDIXFXIT. If LE conforming, it should not contain a C or PL/I main function. The IDIXFXIT user exit must be link-edited with the NORENT option.

Entry specifications

On entry to IDIXFXIT, the contents of registers are:

Register

Contents

1

Address of input parameter list (see below).

13

Address of 72-byte register save area.

14

Return address.

15

Entry point address of IDIXFXIT.

Input parameter list

The address of the following parameter list is passed to the IDIXFXIT user exit in R1. All parameters are provided as C-style, null-terminated character strings: the value, followed by a byte containing X'00'.

Table 10. IDIXFXIT input parameters

Parameter	Number of bytes	Description
Parameter 1	Varying	The level of access required to the fault entry: Read, Update, or Delete.
Parameter 2	Varying	Security server user ID of IDIXFXIT user exit caller[1].
Parameter 3	Varying	Security server default group ID of IDIXFXIT user exit caller[1].
Parameter 4	Varying	The name of the job that created the fault entry.
Parameter 5	Varying	The history file data set name containing the fault entry.
Parameter 6	Varying	The fault entry HD segment data area. Mapping of the HD segment data area is provided in data set IDI.SIDISAM1 members: • IDISXPLA (Assembler) • IDISXPLC (C) • IDISXPLB (COBOL) • IDISXPLP (PL/I)

Note:

Table 10. IDIXFXIT input parameters

(continued)

Parameter	Number of bytes	Description
		1. Obtain the fault entry creator security server user and default group ID information from the HD segment data area pointed to by parameter 6.

Return specifications

On return from IDIXFXIT, the contents of registers must be as follows:

Register

Contents

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 that is written in C.

```
#include "IDISXPLC"
int IDIXFXIT(char *action, char *userid, char *group, char *jobnm,
             char *histDSN, DDIR_DATA_HD *pHD) {
  if (strcmp("MY.HIST", histDSN) == 0) return 2;  /* Update access ok */
  else return 0 ;                                 /* No access */
}
```

Chapter 19. Setting and changing default options for the site

The creation of a parmlib member, IDICNFxx, which contains Fault Analyzer options that override the product defaults and are available to all users of Fault Analyzer at your site, is explained in the following topics.

Parmlib member IDICNFxx

Default options for the site are held in the parmlib member IDICNFxx, where *xx* is either:

- 00
- A value matching the current MVS™ symbol, &SYSCLONE.

  &SYSCLONE is a standard MVS™ system symbol that represents a unique one- or two-character system ID. Refer to *MVS™ Initialization and Tuning Reference* for information about the MVS™ &SYSCLONE symbol.

  Using an IDICNF*&SYSCLONE* member name permits an installation to specify different options for each MVS™ image in a sysplex, while still using a common PARMLIB data set.

The initial search is for an IDICNF*&SYSCLONE* member name. If this member name is not found in any parmlib data set, then a search is made for an IDICNF00 member.

The IDICNFxx member can be created in SYS1.PARMLIB, or any other data set that is part of the logical parmlib concatenation.

All data sets in the logical parmlib concatenation must be given universal READ access.

Note: See Customize Fault Analyzer by using an IDIOPTLM configuration-options module on page 337 if you do not want to place the IDICNFxx member in the logical parmlib concatenation but use an alternative data set instead.

If the IDICNFxx member does not exist, then Fault Analyzer uses the product-supplied default options. In real time, message IDI0018W on page 699 is issued.

This is an example of an IDICNFxx parmlib member:

Figure 205. Sample 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     */
  IDILCOBO(APP2.LISTINGS)      /* OS/VS COBOL compiler listings      */
  IDILPLI (MY.LISTING.PLI    ) /* PL/I compiler listings             */
  IDILANGX(MY.IDILANGX)        /* LANGX files                        */
  IDIADATA(MY.SYSADATA)        /* SYSADATA files                     */
  IDIMAPS(IDI.SIDIMAPS)        /* Data area models                   */
)
```

A sample IDICNFxx parmlib member (which might not match the sample above) is available in the IDI.SIDISAM1 data set as member IDICNF00.

Note that only columns 1 - 71 of the IDICNFxx member are processed.

When installing Fault Analyzer, you must review the DataSets option in the IDICNF00 sample member:

- If you have installed Fault Analyzer with a high-level qualifier other than IDI, then you must include the DataSets option with specification of data sets for these DDnames:
  IDIMAPS
  IDIDOC
- If you have allocated your VSAM message and abend code explanation repository with a name other than IDI.IDIVSENU , then you must include the DataSets option with specification of the data set name for the IDIVSENU DDname.
- If you have allocated your default history file with a name other than IDI.HIST, then you must include the DataSets option with specification of the data set name for the IDIHIST DDname.

For details on the DataSets option, see DataSets on page 581.

You can change the parmlib member whenever you need to. For example, you can change the parmlib member to exclude other types of jobs. However, you should adjust an option in the user options file if you only want to change an option for one job.

Various techniques for changing options for individual jobs are described in Where to specify options on page 573, Batch reanalysis options on page 149, and Interactive reanalysis options on page 157.

The individual options are explained in Option descriptions on page 577.

Note: If an IDICNFUM user-options module is used, it might replace the IDICNFxx default options. For details, refer to User-options module IDICNFUM on page 574.

The next sections briefly describe the function of some of the other Fault Analyzer options that you might want to consider specifying in the IDICNFxx parmlib member. However, if you are installing Fault Analyzer and merely want to reach a point at which the supplied IVP jobs can be run, then neither of these options are required.

In the remainder of this document, whenever references are made to the IDICNFxx parmlib member, it is implied that the name can be either IDICNF00, or IDICNF*&SYSCLONE*.

Controlling which jobs are analyzed with Exclude processing

The term "work unit" is used in this section to refer to either a batch job, a started task, or a TSO user.

The Exclude and Include options (Exclude/Include on page 595) can be used to select which work unit abends you want analyzed.

If neither option is specified, the default is to include all work units.

It is possible to specify each of these options any number of times to achieve the desired inclusion or exclusion of specific work units. For example, all jobs can be excluded and then only certain jobs included, or if the opposite is desired, all jobs included (the default) and only some jobs excluded.

The selection process is as follows:

- The initial *"state"* of work unit selection is to include everything.
- Each specification of an Include or Exclude option is tested against the abending work unit:
  - If the criteria matches the abending work unit characteristics, then the current state is set to 'include' (if an Include option matched) or 'exclude' (if an Exclude option matched).
  - If the criteria does not match, then the state remains unchanged.
- If the final state after processing of all Include and Exclude options is 'exclude', then the abending work unit is excluded from analysis. In this case, you get no analysis report, and no fault entry is written to the history file.

Note:

1. If you have excluded the real-time analysis, then you cannot later reanalyze, since there is no fault entry.
2. To make exclusion of analysis 'transparent' to the abending work unit, you might want to consider using the Quiet option. Otherwise, a message is issued to indicate that exclusion has occurred.

It follows from the above that the order in which Include and Exclude options are specified is significant. For example, if the following were specified, all batch jobs executing under the user ID FRED would be included for analysis, regardless of job name or execution class:

```
Exclude(TYPE(JOB) NAME(TEST*))   /* Exclude all batch jobs with names
                                    starting with TEST */
Exclude(CLASS(Z))                /* Exclude all batch jobs in class Z */
Include(TYPE(JOB) USERID(FRED))  /* Include batch jobs belonging to FRED */
```

However, if the last two options were reversed as shown, FRED's batch jobs would be excluded if they were executing in class Z:

```
Exclude(TYPE(JOB) NAME(TEST*))   /* Exclude all batch jobs with names
                                    starting with TEST */
Include(TYPE(JOB) USERID(FRED))  /* Include batch jobs belonging to FRED */
Exclude(CLASS(Z))                /* Exclude all batch jobs in class Z */
```

Note: Final exclude/include status can be controlled by an Analysis Control user exit (including dump registration) by setting the ENV.EXCLUDE value to "Y" or "N".

Fast Exclude options processing

If the IDIS subsystem is started and the PARM='FASTEXCLUDE' option is in effect (this option is the default), then all Include and Exclude options specifications from the IDICNFxx parmlib member are cached in the IDIS subsystem. A Fault Analyzer invocation exit can obtain this information without the need for any I/O, and subsequently determine if a fault should be excluded from further processing prior to attaching the mainline IDIDA load module, where normal options processing is performed.

With the exception of CICS®, only jobs or started tasks that do not include an IDIOPTS DDname are eligible for fast Exclude options processing. With CICS® being a long-running system, Fault Analyzer periodically reads the Exclude/Include option specifications from the options data set, and caches these for subsequent use by the CICS® invocation exits when an application abend occurs.

Whenever a fault has been fast excluded, the IDI0034I on page 703 message is issued.

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 IDICNFxx parmlib member, then the IDIS subsystem should be cycled to cause these options to be reread.

To disable fast Exclude options processing, specify the IDIS subsystem PARM='NOFASTEXCLUDE' option.

Controlling report detail

The Detail option described at Detail on page 591 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 35 describes the analysis report, and the effect that different Detail values have on the report.

You can change this value for a reanalysis.

Limiting the size of minidumps

The MaxMinidumpPages option (MaxMinidumpPages on page 615) lets you limit the size of minidumps written to the history file. If the number of pages in the minidump exceed the limit in effect, then no minidump is written.

When choosing a minidump size limit, the space that is allocated to the fault history file should be considered. Each minidump page is 4 kilobytes.

Pointing to listings and REXX exec libraries

The DataSets option (DataSets on page 581) 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 480.

Apart from listing and side-file data sets, the DataSets option can also be used to select other data sets, such as the history file and REXX exec user exit libraries.

Suppressing noncritical SYSLOG messages

To prevent information and warning level messages from being written to the SYSLOG, use the Quiet option.

For information about the Quiet option, see Quiet on page 629.

Customizing the Fault Entry List display

If an installation-wide change to the information that is shown on the Fault Entry List display is required, this change can be achieved by using the FAISPFopts(HistCols(...)) option to specify the particular columns of information to be displayed in the order that you want. This provides users with a common base from which they can make further changes if they want.

For information about the FAISPFopts option, see FAISPFopts on page 603.

Specifying the cultural environment

The Locale option (Locale on page 614) lets you specify the locale to be used in order to enable cultural environment-dependent presentation.

Chapter 20. Providing compiler listings or Fault Analyzer side files

When analyzing an abend, Fault Analyzer attempts to provide source line information obtained from compiler listings or side files. (For detailed information about the types of compiler listings or side files searched for, and the order of precedence, see Locating compiler listings or side files on page 380.)

Enable the LangxCapture option during real-time analysis to write the source line information from a compiler listing or side file into the fault entry. This ensures the original source information remains available even if the abending program is subsequently recompiled. See LangxCapture on page 613.

If the relevant file cannot be located as a side file, but a compiler listing is found, then Fault Analyzer generates a side file from the listing and places it in a temporary data set, which is deleted once the analysis has completed.

If Fault Analyzer cannot find a listing either, then it is not able to provide source line detail, although it can still provide an analysis of the abend.

Even if a compiler listing or side file could not be found during real-time analysis, one can be provided later during reanalysis of the history file fault entry. If necessary, create the compiler listing or side file by recompiling the abending program after the abend has occurred. However, in order for the compiler listing or side file to match the load module as at the time of the abend, and thus facilitate Fault Analyzer source level analysis, it is important that the object deck from the recompilation remains unchanged.

Some compiler options, such as XREF, LIST, SOURCE and MAP, only change the information provided in the compiler listing. Other options, such as OPTIMIZE and SSRANGE, change the object code produced. When recompiling to add source support, all options should be left the same as the original compile, except for options that only affect the listing, such as LIST and XREF. Naturally, all input source, as well as the compiler version and maintenance level, should be the same.

For programs written in High Level Assembler, Fault Analyzer does not use the assembly listing but instead the SYSADATA file that is produced when specifying the assembler ADATA option.

At the time of assemble, this data set is referenced by the SYSADATA DDname, but during Fault Analyzer processing it is read back via the IDIADATA DDname.

You have the choice of storing either listings or side files. However, side files are preferable for two reasons:

1. They use less disk space.
2. Fault Analyzer has to convert listings. Using side files reduces the total analysis time.

If you have set up PDS or PDSE data sets for compiler listings or side files, then you should make sure that they are populated with the programs likely to need analysis. Alternatively, compiler listings can be stored in sequential data sets.

Fault Analyzer searches all data sets specified for a given DDname separately and chooses 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 378.

COBOL Report Writer Precompiler

If you are using the COBOL Report Writer Precompiler (program number 5798-DYR), it is important that you run it as a stand-alone precompiler as opposed to invoking it via the COBOL compiler EXIT option. Otherwise, information that is required by Fault Analyzer to identify the point of failure source code statement might be missing from the compiler listing.

Symptoms that you might experience if using the COBOL Report Writer Precompiler as a COBOL compiler exit are:

- Return code 3114 from IDILANGX if trying to convert the COBOL compiler listing file to a side file.
- The following messages are issued during fault analysis:

  IDISF8100S COBOL LISTING file contains NO recognized records

  IDISF8132S Input or Output file format invalid
- Failure to determine point of failure source line.

Required compiler options for IDILANGX

To create LANGX side files, the IPVLANGX utility is used. The IPVLANGX utility is provided with ADFz Common Components. IDILANGX is an alias of IPVLANGX. For details about running the IPVLANGX utility, see *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*, chapter *"IPVLANGX compiler listing to side file conversion utility"*.

The following are the compiler options needed to produce listings or side files suitable for Fault Analyzer. If compiler-generated TEST(SEPARATE) SYSDEBUG side files are used, then these options do not matter.

C:	AGGREGATE LIST NOOFFSET NOOPTIMIZE (Note 1) SOURCE XREF NOIPA
C++:	ATTRIBUTE (Note 4) LIST LONGNAME NOOFFSET NOOPTIMIZE (Note 1) SOURCE

	XREF NOIPA
OS/VS COBOL:	DMAP NOLST NOOPTIMIZE (Note 1) PMAP, NOCLIST (Note 2) SOURCE VERB XREF or SXREF
COBOL compilers other than OS/VS COBOL:	LIST,NOOFFSET (Note 2) NOOPTIMIZE (Note 1) MAP SOURCE XREF(SHORT) (Note 3)
Enterprise PL/I:	AGGREGATE ATTRIBUTES(FULL) NOBLKOFF (Note 6) LIST (Note 5) MAP NEST NONUMBER OFFSET NOOPTIMIZE (Note 1) OPTIONS SOURCE STMT XREF(FULL)
PL/I compilers other than Enterprise PL/I:	AGGREGATE ATTRIBUTES(FULL) ESD LIST (Note 5) MAP NEST NOOPTIMIZE (Note 1)

	OPTIONS SOURCE STMT XREF(FULL)
Assembler:	ADATA

Notes:

1. Although NOOPTIMIZE is recommended, the use of OPTIMIZE is allowed (including OPT(1) or OPT(2) for C), in which case the compiler merges and rearranges statement numbers in the compiled code. The Fault Analyzer analysis is limited to what can be determined from the optimized compiler listing, which can vary from having no effect on the Fault Analyzer report, to inaccurate identification of the source line that failed. The source line number is usually close, but not necessarily accurate with OPTIMIZE. It depends on the compiler's rearrangement or elimination of source statements during optimization processing. Data field values cannot be shown if storage has not been assigned, that is, if the value is in a register only. When OPTIMIZE is in effect, use LIST and NOOFFSET because OFFSET does not account for code movement.
2. Although LIST and NOOFFSET (or PMAP and NOCLIST for OS/VS COBOL) are recommended, the use of NOLIST and OFFSET (or NOMAP and CLIST for OS/VS COBOL) is allowed, in which case Fault Analyzer is not able to warn the user if the compiler listing is not a good match with what is in storage.
3. XREF(SHORT) is a minimum requirement; XREF(FULL) is permitted and has no detrimental effect.
4. ATTRIBUTE is a minimum requirement; ATTRIBUTE(FULL) is permitted and has no detrimental effect.
5. LIST is required to correctly report Enterprise PL/I STATIC EXTERNAL variables and parameters. However, the PL/I V6 complier does not make 64-bit parameters available, even when the LIST option is effect. 64-bit parameters are available with PL/I V5.
6. See section IDISBLKO on page 395 for a description of the IDISBLKO sample provided as a member in the IDI.SIDISAM1 data set.

TEST option considerations

With all compilers, the use of the TEST option provides program information in addition to what is available from the side files.

If TEST(NONE,SYM,SEPARATE) is used when compiling a COBOL program, or TEST(STMT,SYM,NOHOOK,SEPARATE) is used when compiling an Enterprise PL/I program, then a SYSDEBUG file that is suitable for use by Fault Analyzer is written. (See Locating compiler listings or side files on page 380 for information about how the SYSDEBUG file is searched for by

Fault Analyzer.) If the SYSDEBUG file is to be used instead of a compiler listing, or a LANGX side file created from a compiler listing, then it should be retained for use by z/OS® Debugger and Fault Analyzer.

The COBOL SYSDEBUG side file typically uses about 30% less DASD space than a Fault Analyzer IDILANGX side file.

The Enterprise PL/I SYSDEBUG side file is not a stand-alone debugging aid (unlike the COBOL equivalent, which is). The load module always contains the statement number table (which provides source line offsets), and also contains symbol information when the program has GET/PUT DATA statements.

DEBUG option considerations

Use the DEBUG option with the XL C/C++ compilers to write DWARF debugging information. This debugging information can be written to an MVS™ data set or zFS file. Fault Analyzer locates DWARF files from information in the load module.

DWARF files for all the compile units in a load module can be combined, together with source, into an MDBG side file (the CDADBGLD utility is used for this). Like DWARF, MDBG files can reside in an MVS™ data set or zFS file. MDBG files stored in MVS™ data sets are located via the IDISYSDB DD. MDBG side files in zFS are self-locating, and as such must adhere to the following:

- MDBG files should reside in the same zFS 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).

Limitations when using COBOL NOTEST(DWARF,SOURCE)

If PTF UI60153 has not been installed and COBOL programs are compiled with NOTEST(DWARF,SOURCE), the compiler does not include cross-reference information in the DWARF.

As a result, data field values and declarations will be missing from the Synopsis and Event Details sections of the report, as well as from COBOL Explorer.

Limitations when using PL/I TEST(NOSEPARATE NOSOURCE)

When NOSOURCE is specified and program source is not available, source line numbers and storage areas are still reported.

TEST(SEPARATE) and TEST(NOSEPARATE SOURCE) are equivalent, the former locates source in a separate SYSDEBUG side file, the latter in the program object.

Naming compiler listings or side files

Store compiler listings or side files in sequential data sets, or as members of PDS or PDSE data sets.

When you store a compiler listing or side file in a PDS or PDSE data set, the member name to use depends on the application programming language:

- 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.
- For COBOL programs, use the name of your application program.
- For PL/I programs, use the primary entry point name or CSECT name of your application program.
- For z/OS XL C and C++ programs, use the CSECT name of your application program.

If you store with any other name, then Fault Analyzer is unable to find the compiler listing or side file.

If compiler listings or side files are stored in sequential data sets, and the data set names follow a convention that permits the program name to be part of the data set name, then the specification of these data sets in the DataSets option can be done easily using variable substitution, as described in DataSets option data set name substitution symbols on page 585.

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. Failure to follow these requirements prevents source code information from being determined for these types of programs.

CSECT naming requirements for C programs

To enable automatic source support for C programs from the IDILC or IDILANGX concatenation, it is a requirement that CSECTs are named using the following #pragma statement:

```
#pragma csect(code,"csect_name")
```

Where, if a PDS or PDSE is used, *csect_name* matches the member name of the compiler listing or LANGX file. This enables the side file search to automatically locate compiler listings.

For more information about preparing z/OS® XL C and C++ programs for use with Fault Analyzer, see *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*, in the chapter *"Quick start guide for compiling and assembling programs for use with the ADFz family of products"*.

To handle cases where C programs were not compiled with the `#pragma csect` option, two sample EXECs are provided in data set IDI.SIDISAM1 to facilitate source provision:

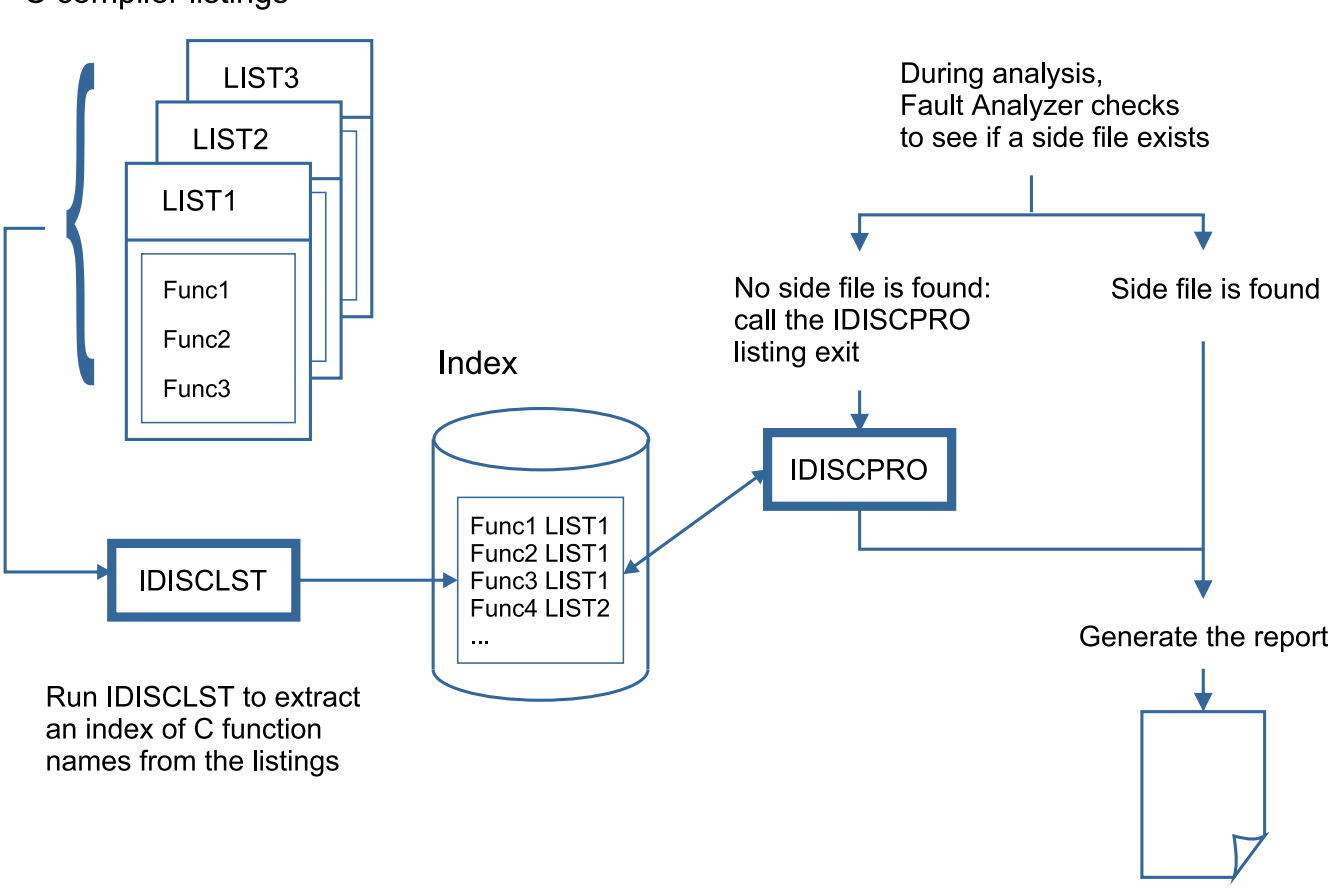
IDISCLST

This EXEC can be used to process C compiler listings to extract the C function names from the PPA1 and build an index of function names and their matching listing data set.

IDISCPRO

This EXEC is a Fault Analyzer listing exit. It scans the index file created by IDISCLST and returns the name of the matching listing data set for a given function.

Figure 206. Extracting function names from C programs that were not compiled with #pragma csect



Locating compiler listings or side files

Fault Analyzer chooses a matching compiler listing or side file from the first of the following possible sources:

1.

TEST option data:

- If the program was compiled with the TEST option, other than TEST(SEPARATE) for COBOL or Enterprise PL/I, a temporary Fault Analyzer side file is generated from debug information contained in the load module.
- If a COBOL V4 or Enterprise PL/I program is compiled with TEST(SEPARATE), 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.

  See TEST option considerations on page 377 for recommended language-dependent suboptions.
- If a COBOL V4 program compiled with TEST(SEPARATE), 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.
- If a COBOL V6 program is compiled with TEST(SEPARATE) and the DSNAME suboption is specified, the SYSDEBUG side-file data set name is stored in the program object. Otherwise, the SYSDEBUG side-file data set name is obtained from the IDISYSDB concatenation, the Compiler Listing Read user exit, the COBOL IGZIUXB exit, or EQAUEDAT.

  Note: Under CICS®, the batch form of EQAUEDAT is used as Fault Analyzer runs in an attached TCB without CICS® command access.

2. The z/OS® Debugger EQAUEDAT exit (optional).

   Refer to *IBM® z/OS® Debugger Customization Guide* for details about the EQAUEDAT exit.

   If a COBOL or Enterprise PL/I program compiled with TEST(SEPARATE), then the SYSDEBUG side file data set name from the load module is used as input to the EQAUEDAT exit.

   No input side file name is provided when searching for IDILANGX or compiler listings.

   This exit might return a different side file data set name.

   Note: Under CICS®, the batch form of EQAUEDAT is used as Fault Analyzer runs in an attached TCB without CICS® command access.

3. If a COBOL or Enterprise PL/I program, then any SYSDEBUG data sets provided via the IDISYSDB DDname are searched. For XL C/C++ programs, MDBG data sets provided via the IDISYSDB DDname are searched. The data sets are searched individually, that is, they are not treated as a logical concatenation.
4. Fault Analyzer side files:
   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 LANGX side file for a program, a number of tests are performed, subject to the programming language used, to verify that the file matches the load module:

  - **All languages**

    From the Assembler listing section of any compiler output, Fault Analyzer extracts the last few assembler instructions and compares them against the load module, and these assembler instructions must be at the correct offset in the load module according to the listing. If any of these instructions are not found at the correct offset in the load module, then this check fails, and the compiler listing or side file is not used to provide source-level information.

  - **COBOL-specific tests (excluding OS/VS COBOL)**

    For COBOL, four more length values are extracted from the listing. These are the TGT length, WORKING-STORAGE length, number of Data Division statements, and number of Procedure Division statements from the following four lines found in the compiler listing:

    ```
    TGT      WILL BE ALLOCATED FOR nnnnnnnn BYTES
    WRK-STOR WILL BE ALLOCATED FOR nnnnnnnn BYTES
    Data Division statements = nnnnnn
    Procedure Division statements = nnnnnn
    ```

    These four values are compared to those found in the load module, and if either do not match, then the compiler listing or side file is not used to provide source-level information.

- **If the TEST compiler option is used**

  For COBOL SYSDEBUG files, the date and time comparisons are replaced with a 'signature' check, allowing an unchanged program to be recompiled. The SYSDEBUG file can still be used during interactive reanalysis if the signature check fails by following the compiler listing mismatch pop-up displays.

  For COBOL programs, up to and including Enterprise COBOL V4, but excluding VS COBOL II, the compiler listing data set name is obtained from debug information in the load module, placed there by the compiler. In this case, the date and time in the listing are checked against the compile date and time in the load module. If the compiler listing file has been moved (or it is VS COBOL II), then a compiler listing can be specified by way of the IDILCOB DD, EQAUEDAT or the Compiler Listing Read user exit.

  For Enterprise COBOL V5 programs (or later) compiled with TEST(NOSOURCE) or NOTEST(DWARF), no source information is included in the program object and no compiler listing data set name is available in the DWARF.

However, compiler listings for these programs can be specified by way of the IDILCOB DD, EQAUEDAT or the Compiler Listing Read user exit.

For Enterprise PL/I SYSDEBUG files, the date and time comparisons are replaced with a set of 'integrity' checks, which determine whether the SYSDEBUG file can be used with the load module in storage. If any of the integrity checks fail, then the SYSDEBUG file cannot be used.

The reason why Fault Analyzer performs these checks before 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, Fault Analyzer prompts the user to optionally supply the location of a compiler listing, Fault Analyzer side file, or COBOL SYSDEBUG side file. For more details on this prompt, see Prompting for compiler listing or side file on page 235.

IDITRACE information

When you add the IDITRACE DDname to your job, 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=*
```

For an alternative method of activating this trace under CICS®, see IDITRACE under CICS on page 402. For information about the dynamic tracing via the **CTL.IDITRACE data area** field, see Analysis Control user exit on page 475.

If you do not use dynamic tracing via the **CTL.IDITRACE data area** field, trace information is written to the IDITRACE DDname destination. An example of compiler listing or side file search trace information follows:

Figure 207. Sample compiler listing/side file search trace

```
Listing/Side File search trace for IDISCBL1
  Execution program IDISCBL1 compile date 2013/02/18 time 12:17:24
  Rejected - DA.LISTING.COBOL(IDISCBL1)
    Failed -
      OPEN error, member not found.
  Rejected - NWILKES.$$TEMP$$.LISTINGS(IDISCBL1) Built 2013/02/18 12:17:41
    Failed -
      COBOL Working Storage length mismatch - load x'2A' listing x'32'
      COBOL Data Division Statements mismatch - load 6 listing 7
    Passed -
      COBOL TGT lengths match load and listing both 148
      COBOL Procedure Division Statements match, both 10
      Object code length check passed.
  Accepted - NWILKES.IVPCB.LISTINGS(IDISCBL1) Built 2013/02/18 12:17:24
    Passed -
      COBOL TGT lengths match load and listing both 148
      COBOL Working Storage lengths match load and listing both x'2A'
      COBOL Data Division Statements match, both 6
      COBOL Procedure Division Statements match, both 10
      Object code length check passed.
```

When performing interactive reanalysis, 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. See Prompting for compiler listing or side file on page 235.

However, if complete IDITRACE information similar to that produced in batch is required, follow these steps:

1. Enter the Fault Analyzer ISPF interface Fault Entry List display.
2. From the primary command line, issue the commands:
    - TSO FREE DD(IDITRACE)
    - TSO ALLOC DD(IDITRACE) DSN(*dsn*) DSORG(PS) SPACE(2,1) TRACKS LRECL(137) RECFM(V,B) NEW where *dsn* is the name of the trace data set.
3. Perform interactive reanalysis of the fault entry.
4. PF3 back to the Fault Entry List display.
5. Optionally, from the primary command line, issue the command:
    - TSO FREE DD(IDITRACE)

Compiler listings and side file access requirements

Compiler listings and side files are read by Fault Analyzer module IDIDA.

If the NoDeferredReport and Source options are in effect, the compiler listings or side files are read during real-time processing. This is done using the authority associated with the credentials under which the started task, batch job, or TSO user is executing.

For CICS, check and modify the CICSTranAnalysisUser option, if necessary.

If compiler listings or side files are read during reanalysis, it is done using the authority of the user ID performing the reanalysis.

Using the IDIXSFOR compiler listing/side file search and override exit

If the IDIXSFOR exit is available, Fault Analyzer invokes it to allow programmatic specification of the location of compiler listing or side file data.

The input parameter list is like that used by EQAUEDAT, but with an extra FORCE parameter at the end. The FORCE parameter can be used to tell Fault Analyzer to accept a side file, even if the last 12 instructions of the object code do not match. The FORCE parameter can also be used to request that no further side file processing should be performed for the current compile unit.

IDIXSFOR invocation

The following describes the entry and return specifications for IDIXSFOR.

Entry specifications

On entry to IDIXSFOR, the contents of registers are:

Register

Contents

1

Address of input/output parameter list (see below).

13

Address of 72-byte register save area.

14

Return address.

15

Entry point address of IDIXSFOR.

Input parameter list

Register 1 must contain the address of a parameter list, consisting of a list of addresses (OS linkage). Each address in the parameter list points to a parameter.

Table 11. IDIXSFOR input parameters

Parm #	Length	Input/Output	Description
1	See parm #2	Input/Output	Side file data set name **Input** Fault Analyzer might provide an input data set name if the compile option was TEST(,SEPARATE). **Output** Exit-provided new data set name/path to try. The length of this name must not exceed 1024 bytes. The member name must be included if the data set is a PDS or PDSE.
2	4	Input/Output	Side file data set name (parm #1) length **Input** Length of input side file name. **Output** Length of new data set name or path.
3	4	Input	Language code **Value** **Meaning**

Table 11. IDIXSFOR input parameters (continued)

Parm #	Length	Input/Output	Description
			3 C source **4** COBOL **7** Assembler, OS/VS COBOL, non-LE VS COBOL II **10** PL/I (other than Enterprise PL/I) **11** Enterprise PL/I **35** C side file compiled with FORMAT(DWARF) **37** C MDBG side file **41** VS COBOL II (under LE)
4	See parm #5	Input	Compile unit name
5	4	Input	Compile unit name (parm #4) length
6	See parm #7	Input	Load module name
7	4	Input	Load module name (parm #6) length
8	See parm #9	Input	Load library name
9	4	Input	Load library name (parm #8) length
10	4	Output	Force return option word **Value** **Meaning** **0** No forced override **1** Force accept mismatch of last 12 object instructions

Table 11. IDIXSFOR input parameters (continued)

Parm #	Length	Input/Output	Description
			-1 Ignore side file processing for this compile unit

Return specifications

On return from IDIXSFOR, the contents of registers are:

Register

Contents

0-1

Undefined.

2-14

Unchanged.

15

Undefined.

Example (Assembler)

A sample IDIXSFOR user exit is provided as member IDIXSFOR in data set IDI.SIDISAM1.

Compiler listings and side file attributes

Compiler listings and side files must be allocated using the following attributes:

DDname

Attributes

IDIADATA

Sequential data set or PDS or PDSE, RECFM=VB, LRECL≥8188

IDILC

Sequential data set or PDS or PDSE with either of the following attributes:

- RECFM=VB or VBA and LRECL≥137
- RECFM=FB or FBA and LRECL=133

IDILCOB

Sequential data set or PDS or PDSE, RECFM=FBA, LRECL=133

IDILCOBO

Sequential data set or PDS or PDSE, RECFM=FBA, LRECL=121

IDISYSDB

Sequential data set or PDS or PDSE, RECFM=F or FB, 80≤LRECL≤1024

IDILANGX

Sequential data set or PDS or PDSE, RECFM=VB, LRECL≥1562

IDILPLI

Sequential data set or PDS or PDSE, RECFM=VBA, LRECL≥125

IDILPLIE

Sequential data set or PDS or PDSE, 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 storage is done by using the PACK ON option from within ISPF edit of the file. The ISPF packed format is not permitted for IDILANGX or IDIADATA data sets.

Using the IDIRLOAD DDname for CSECT mapping

Since Fault Analyzer generally searches for compiler listings or side files using the uppercase eight-character CSECT name, it follows that resolving CSECT names is a prerequisite 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 **File > Analyze MVS Dump Data Set** from the **Fault Entry List** display, 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 preallocate 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 an 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. This approach can be useful if performing multiple simultaneous Fault Analyzer dump analyses in separate ISPF split screen sessions. Refer to the description of the IDIRLOAD_DD field in ENV - Common exit environment information on page 653 for information about how to use an alternative DDname during initial RFR fault entry reanalysis.

If during reanalysis, Fault Analyzer does not have the link-edit information for a load module due to any of the reasons mentioned above, then it uses the IDIRLOAD concatenation to search for a module of the same name and uses IEWBIND to extract the CSECT name information. This approach facilitates potential IDILANGX source mapping for those CSECTs.

It is important that any load libraries that are allocated to the IDIRLOAD DDname match the load modules being mapped, otherwise incorrect CSECT and source line determination might result.

To facilitate source level analysis of PL/X programs, the user must provide matching Fault Analyzer side files, created by calling IDILANGX with `PARM='mbr_name (PLX'` and specifying both SYSADATA and SYSLOGIC input data.

IDIDATST side file availability test utility

A utility program, IDIDATST, is provided to help determine the availability of matching source mapping side files for programs within load modules. Normally, IDIDATST would be used during an installation phase, to check if existing compile listings or side file rebuild processes had produced usable side files, should abends occur in the program suite. It is a way of determining before an abend occurs, if matching compile listings or side files are available for a program.

The basic operation of IDIDATST, with no input parameters, is to scan all programs in load modules of the `//STEPLIB DD` first concatenation data set. Each program or CSECT is tested for an available matching side file from the corresponding IDICNFxx DataSets option list of side file data sets, and a search report is written to `//IDITRACE DD`, if there is one in the job step. All of the side file DD statements for each language, such as `//IDILCOB` for COBOL, can be added to the IDIDATST job step, and they are searched first for the corresponding program language before searching any IDICNFxx options data sets.

A sample job for invoking the IDIDATST utility is as follows:

```
//IDIDATST JOB ...
//RUN      EXEC PGM=IDIDATST,PARM='parms'
//STEPLIB  DD   DISP=SHR,DSN=load_lib_dsn
//IDICSV   DD   SYSOUT=*
```

The optional PARM field can be added to the `EXEC PGM=IDIDATST` JCL statement in the form:

Figure 208. Syntax

where:

load_module_name

The name of a load module in the `//STEPLIB DD` first concatenation data set. If the load module name that is specified is shorter than 8 characters, then it is used as a filter to match against the same first characters of all load module names.

For example, specifying

```
PARM='PGM=KACBB092'
```

tests only the load module KACBB092, whereas specifying

```
PARM='PGM=KAC'
```

tests all load modules with names starting with KAC in the first STEPLIB data set.

A consolidated report is created from each run of IDIDATST, with a one-line result for each program search, which is written to the `//IDICSV DD`. The `IDICSV` output is in comma-separated-value (CSV) format, which might be useful if importing the data into a spreadsheet program, for example. For a copy of the same data in a column-aligned format (which is more suited to editor review), add `//IDIFLAT DD` to align the CSV data with blanks.

The IDIDATST process for a STEPLIB does not re-test programs or CSECTS that occur in a subsequent load module.

IDIDATST cannot exactly replicate a side file test that would occur under an abend condition.

An optional `//IDINOTST DD` can be added to provide a blank-delimited list of prefixes for program names that are to be excluded from side file checking. For example:

```
//IDINOTST DD *
XYZ ERR1*
ABCD
/*
```

Appending an asterisk to the program name prefix is optional and does not alter the IDIDATST behavior.

ISPF-packed compiler listings

ISPF provides a facility to PACK data sets. In this format, ISPF replaces any repeating characters with a sequence that shows how many times the character is repeated. With this facility, you can use direct access storage devices (DASD) more efficiently. When opening a compiler listing, Fault Analyzer detects if the listing is in PACKed format and unpacks it before use.

To pack a compiler listing under ISPF, you can issue the PACK primary command from within an EDIT session on the listing, and then save the listing.

Alternatively, you can automate the creation of packed compiler listings by inserting an extra step into your compile job. An example job is shown below. This example uses the ISPF LMCOPY command, from a REXX exec, to copy the listing from one data set to another, specifying the PACK option.

```
//PACKLIST EXEC  PGM=IKJEFT01
//SYSPROC   DD DSN=rexx-exec-library,DISP=SHR
//ISPPLIB   DD DISP=SHR,DSN=ISP.SISPPENU
//ISPSLIB   DD DISP=SHR,DSN=ISP.SISPSLIB
//ISPTLIB   DD DISP=SHR,DSN=ISP.SISPTENU
//ISPMLIB   DD DISP=SHR,DSN=ISP.SISPMENU
//ISPLOG    DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1210)
//ISPLIST   DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1210)
//SYSTSPRT  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)
/*
```

Both *compile-step-listing-data-set* and *packed-listing-data-set* are names of PDS or PDSE data sets without member specification, and *member-name* is the member name of the compiler listing (which usually matches the name of the program being compiled - see Naming compiler listings or side files on page 378).

The LMCOPY REXX exec must reside in the *rexx-exec-library* data set as member LMCOPY, for example:

```
/* 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, you can save more space by converting a compiler listing into a LANGX side file and discarding the listing. (See *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*, chapter *"IPVLANGX compiler listing to side file conversion utility"*, section *"Creating side files using IPVLANGX"* for information.)

You can reprint most of the original compiler listing information from the LANGX side file. See the description of the IPVLANGP utility in *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*.

Providing Java source information to Fault Analyzer

There are two options for providing Java™ application source information to Fault Analyzer.

1. The IDIJAVA option or DD concatenation
2. Specifying source jars on the Java™ classpath

Fault Analyzer searches for matching source files in each path specified in the IDIJAVA option or IDIJAVA DD concatenation.

If IDIJAVA paths were not specified or source files were not found, Fault Analyzer searches in each explicitly specified jar file and directory for source files that match the application events.

Optimizing Java source information

If Java source is never required or never available, specify the NOSOURCE option to tell Fault Analyzer not to look for matching Java™ source files. If source jar files or Java™ source files are not on the application class path, specify the `-dropcp-` option to tell Fault Analyzer to ignore the application classpath when locating source files. Instead, only paths specified using the IDIJAVA option will be considered when locating source information.

Examples: Using IDIJAVA DD and IDIJAVA IDIOPTS

This topic shows how to use IDIJAVA DD and IDIJAVA IDIOPTS to provide Java™ application source information to Fault Analyzer in different scenarios.

BPXBATCH/BPXBATSL

To use the IDIJAVA DD with BPXBATCH, the alternate entry point BPXBATSL must be used, as it allows spawned programs access to DD data set and path allocations.

```
//RUNJAVA EXEC PGM=BPXBATSL,REGION=500M
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
//STDPARM DD *
PGM /bin/sh java AnApp arg1 arg2
/*
//STDENV DD *
 CLASSPATH=<the application classpath>
 JAVA_HOME=<path to java>
 LIBPATH=<paths to application DLLs>
 PATH=${PATH}:${JAVA_HOME}/bin
 IBM_JAVA_OPTIONS=-Xmx400m
 JAVA_TOOL_OPTIONS=-agentlib-…
/*
//IDIJAVA DD  PATHOPTS=ORDONLY,
// PATH='/u/hunter2/src'
//        DD PATHOPTS=ORDONLY,
// PATH='/opt/apis/coffee-src.jar'
```

The IBM® JZOS batch launcher

Using the IDIJAVA DD with the JZOS batch launcher is the same as for BPXBATSL: specify theIDIJAVA DD on the JZOS batch launcher job step.

```
cd <JAVA_HOME>/mvstools
cp samples/jcl/JVMJCL80 "//'SYS1.SAMPLIB(JVMJCL80)'"
cp samples/jcl/JVMPRC80 "//'SYS1.PROCLIB(JVMPRC80)'"
cp -X JVMLDM80 "//'SYS1.SIEALNKE(JVMLDM80)'"
```

The batch launcher is included with Java™ for non-SMP/E installations of Java™. For more information, refer to Java™ Batch Launcher and Toolkit for z/OS (JZOS) in the documentation for IBM® SDK, Java™ Technology Edition.

Specifying long path values with IDIJAVA DD

To specify paths longer than 72 characters in JCL, one option is to express the path using JCL symbols. For example:

```
//* Define each part of the path as a symbol:
// SET QQ='''' * This sets symbol &QQ. to a single quote
// SET SRCPART1='/u/hunter2/org/very/large/financial/'
// SET SRCPART2='organisation/JavaSourceFiles/prod'
// SET SRCPART3=&QQ.&SRCPART1.&SRCPART2&QQ.
//*
...
//IDIJAVA PATHOPTS=ORDONLY,
// PATH=&QQ.&SRCPART3/Version1/.QQ.
```

Using the IDIJAVA option

The IDIJAVA option can be specified system-wide in parmlib member IDICNFxx, or it can be specified on a specific job step using the IDIOPTS DD:

```
//IDIOPTS DD *
 Datasets(
   IDIJAVA(
         -dropcp-
           /u/hunter2/srcFiles
          /u/hunter2/deps/dep2-src.jar
      )
   )
/*
```

Each path (represented here by *"xxxxx..."*) must be separated by a space. Long paths can be continued across lines by specifying a '+' between each part of the path. Paths can be up to 1023 characters in length.

```
//GO.IDIOPTS  DD DATA,DLM='##',SYMBOLS=JCLONLY
  datasets(idijava(
                  /01xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx +
                  /02xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx +
                  /03xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx +
                  /04xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx +
                  /05xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx +
                  /06xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx +
                  /07xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx +
                  /08xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx +
                  /09xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx +
                  /10xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx))
```

Compiling Java for optimal debugging

The type and amount of debug information included within a compiled `.class` file depends upon the defaults of the build tool and the explicitly specified debug options.

Debug information options in IBM® Java™

These are the relevant debug options for the IBM® javac compiler:

-g:source

Include the Java™ source file name in the compiled class file.

-g:lines

Include a LineNumberTable in the class file. This table maps each source file line number to the generated java bytecode instructions; the machine code statements used by the Java™ Virtual Machine.

-g:vars

Include a LocalVariableTable for each non-native and non-abstract method in the class file. This table includes the name and type of each local variable and the scope in which the variable is valid.

Important: If there is any likelihood that debugging of Java™ programs will be required, use the following option to include all debug information in all compiled `.class` files.

```
javac -g:source,lines,vars <class files>
```

The result of this option is that any stack traces that appear because of exceptions will include location information of each stack entry. For example:

```
Exception in thread "main" java.lang.NullPointerException
        at com.example.Book.getTitle(Book.java:16)
        at com.example.Author.getBookTitles(Author.java:25)
        at com.example.Bootstrap.main(Bootstrap.java:14)
```

If the debug options are not specified or the `-g:none` option is used, much less information is available for problem determination:

```
Exception in thread "main" java.lang.NullPointerException
        at com.example.Book.getTitle(Unknown location)
        at com.example.Author.getBookTitles(Unknown location)
        at com.example.Bootstrap.main(Unknown location)
```

Not only does this reduce the value of stack traces, it prevents the names of local variables from appearing when debugging programs. Instead, all variables simply appear as *"<local variable>"* or a similarly unhelpful name, depending on the development environment.

Generating debugging information with common Java build tools

There are many approaches for building large Java™ applications. Each of the common build tools including Apache Ant, Maven, and Gradle have different ways that debug flags must be specified for debug information to be included.

Apache Ant javac task

In the build.xml file, ensure that any instances of the javac task specify the debug options as in the following example:

```
<javac debug="true" debuglevel="source,lines,vars" <other attributes> >
<javac/>
```

Where *<other attributes>* refers to other site-specific or project-specific javac task compilation attributes. Refer to the Ant documentation for more information about the javac Ant task.

Maven maven-compiler-plugin

Maven is capable of compiling Java™ class files without specific instructions. However, to ensure that debug information is included, add a specific entry for maven-compiler-plugin. Ensure that the debug option appears in the configuration section for the plugin.

```
<build>
<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins<groupId/>
    <artifactId>maven-compiler-plugin<artifactId/>
    <version>${maven.compiler.version}<version/>
```

```
    <configuration>
      <compilerargs>
        <arg>-g:source,lines,vars<arg/>
      <compilerargs/>
    <configuration/>
  <plugin/>
<plugins/>
<build/>
```

Refer to the Maven documentation for more information about maven-compiler-plugin usage.

Gradle

In the Gradle build file, ensure that the following properties are specified:

```
compileJava.options.debugOptions.debugLevel = "source,lines,vars"
compileTestJava.options.debugOptions.debugLevel = "source,lines,vars"
```

IDISBLKO

The IDISBLKO utility converts compiler listings generated with the BLKOFF option into a format simulating the required NOBLKOFF option.

The NOBLKOFF option is required for Enterprise PL/I as described in Required compiler options for IDILANGX on page 375. If your compiler listings were created using BLKOFF, you can use the sample REXX utility IDISBLKO provided to attempt to convert the listing to appear as if it were created with NOBLKOFF. This sample is provided as a member IDISBLKO in the IDI.SIDISAM1 data set.

This sample EXEC can be invoked either as a Fault Analyzer compiler listing user exit or it can be used against a listing file data set or a specific member within that data set. Complete usage instructions are provided in the IDISBLKO sample.

Note: While this sample has been tested using different PL/I compiler versions and programs, it is not guaranteed to work in all scenarios. You must review the modified output to ensure accuracy.

Chapter 21. Customizing the CICS environment

There are three exit points at which Fault Analyzer can be invoked for transaction abends under CICS®.

XPCABND

Global user exit using program IDIXCX53. This exit is the main exit provided to invoke Fault Analyzer for CICS® transaction fault analysis.

XDUREQ

Global user exit using program IDIXCX53. This exit can be used to invoke Fault Analyzer for CICS® dumps generated from an EXEC CICS® DUMP command.

The analysis performed by Fault Analyzer at this exit point is the same as for the XPCABND exit point.

LE Exit

LE abnormal termination exit using program IDIXCCEE. This exit is only effective with Language Environment®-based application programs when the CEECXTAN exit has been set.

CICS® AKCS abends can be analyzed using this exit, if both of the following are true:

- The failing program is LE enabled.
- An entry exists in the CICS® dump table for AKCS, specifying that a transaction dump is to be taken.

The first two of these exits are CICS® global user exit points, and Fault Analyzer is enabled and disabled at these points using CICS® calls. By default, these exit points are not enabled in a CICS® region. They are enabled either by adding an entry to the CICS® PLT (see Adding the required programs to the startup PLT on page 398), or by using the supplied CFA transaction once CICS® has initialized (see Controlling CICS transaction abend analysis on page 400).

The LE abnormal termination exit, however, requires a modification to LE (see Configuring Language Environment for CICS to invoke Fault Analyzer on page 397), 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 for this mechanism to work, the LE exit must first be enabled system wide. Once enabled at a system wide level, then the initial setting in a CICS® region is enabled.

IDI.SIDIAUTH needs to be added to the DFHRPL concatenation of the CICS® JCL for any of the above exits to be successfully enabled.

To use Fault Analyzer with CICS®, you need to perform the following steps:

1. **Configure Language Environment® for CICS® to invoke Fault Analyzer**

   For details, see Configuring Language Environment for CICS to invoke Fault Analyzer on page 397.

2. **Define the required programs to your CICS® system**

   For details, see Defining required programs to CICS on page 397.

3. **Add the required programs to your startup PLT**

   For details, see Adding the required programs to the startup PLT on page 398.

4. **Add the required programs to your shutdown PLT**

   For details, see Adding the required programs to the shutdown PLT on page 398.

5. **Define a transaction for Fault Analyzer**

   For details, see Enabling dynamic control of analysis of CICS transaction abends on page 399.

Configuring Language Environment for CICS to invoke Fault Analyzer

Fault Analyzer provides a Language Environment® abnormal termination exit for CICS® as another method of invoking Fault Analyzer to the CICS XPCABND global exit. For more information, see CICS LE abnormal termination CEECXTAN CSECT exit IDIXCCEE on page 302. To enable this exit, you must add it to the CEECXTAN CSECT for Language Environment® for CICS®. To do this, make a copy of the CEEWCEXT softcopy sample member in the CEE.SCEESAMP data set. Make the changes suggested in the sample member and replace the lines:

```
         <<< REPLACE THESE 2 LINES WITH A COPY OF CEECXTAN
             AND OVERRIDE AS DESIRED  >>>
```

with

```
         CEEXAHD        ,User exit header
         CEEXART  TERMXIT=IDIXCCEE
         CEEXAST        ,Terminate the list
```

For general information about implementing a Language Environment® abnormal termination exit for CICS®, refer to the *Language Environment® Customization* book.

Note: The non-CICS version of this exit, CEEEXTAN, installed with sample job CEEWDEXT in data set CEE.SCEESAMP, is very similar to this exit. If you are going to install both, make sure that you have used the correct names and not installed the same exit twice.

Defining required programs to CICS

Unless the CICS® autoinstall program is active, the following assembler programs and BMS map must be defined in a group that is included in a group list used during CICS startup:

- IDIPLT
- IDIPLTD
- IDIPLTS
- IDIXCX53 (Applicable to all current versions and releases of CICS)
- IDIXMAP (BMS map)
- IDIXFA (Optional CICS transaction)

The sample job in member IDISCICS of data set IDI.SIDISAM1 defines the programs, BMS map, and optional control transaction in a group named FA. To customize the sample job for your own installation:

- Replace the *xxx* prefix in the data set names of the sample job with the correct names for your installation.
- Replace *list-name* with the appropriate CICS startup SIT GRPLIST name.
- You can rename the FA group name.
- The optional control transaction is named CFA in the sample job, and it is referred to as "CFA" elsewhere in this documentation. However, you can rename it. See Enabling dynamic control of analysis of CICS transaction abends on page 399 for information about this transaction.

To enable Fault Analyzer to be invoked under CICS, you must add IDI.SIDIAUTH to the DFHRPL concatenation.

CICS tracing must be active for Fault Analyzer to display CICS trace information.

You must use the ABCODE keyword on an EXEC CICS ABEND statement 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, Fault Analyzer performs analysis only if invoked through the IDIXCCEE exit.

Adding the required programs to the startup PLT

To have Fault Analyzer install at CICS® startup, add the IDIPLT program name to your startup PLT. IDIPLT enables IDIXCX53 as an XPCABND global user exit during CICS® startup.

Note: For CICS® open TCB users, adding the Fault Analyzer IDIPLT program name to the startup PLT is mandatory.

IDIPLT should be placed at the end of the PLT list, so that Fault Analyzer is not invoked before normal CICS® services are available, should an abend occur in another PLT program. You can also enable Fault Analyzer at the XDUREQ global user exit by adding program name IDIPLTD to your startup PLT.

IDIPLTS can be included in the startup PLT if you require SDUMP screening to be installed.

Adding the required programs to the shutdown PLT

To ensure that all Fault Analyzer activity is correctly terminated, it is necessary to add the following entry to your CICS® shutdown PLT:

```
DFHPLT TYPE=ENTRY,PROGRAM=IDIPLT
```

The entry should be added before the DFHDELIM entry (shutdown phase 1).

As well as ensuring correct Fault Analyzer termination, IDIPLT also disables Fault Analyzer in all the currently enabled CICS® exit points. This disablement prevents any subsequent abend analysis occurring during CICS® shutdown. If analysis is required during shutdown, then do not add IDIPLT to your shutdown PLT. Note that it is then possible for system 33E abends to occur during shutdown.

If the Fault Analyzer SDUMP screening feature has been installed you can optionally add a corresponding entry to the CICS SHUTDOWN PLT to disable this feature during CICS® shutdown:

```
DFHPLT TYPE=ENTRY,PROGRAM=IDIPLTS
```

Enabling dynamic control of analysis of CICS transaction abends

When Fault Analyzer is installed, you have the option to install a control transaction that dynamically controls the behavior of Fault Analyzer under CICS®.

To install the control transaction, define a CICS® transaction for program IDIXFA in the same group used for the required program definitions. (See Defining required programs to CICS on page 397 for details.) You can use the CEDA transaction to do this, with default values for the parameters.

The sample batch job in member IDISCICS of data set IDI.SIDISAM1 defines a control transaction named CFA, and it is referred to as "CFA" elsewhere in this documentation. However, you can rename it.

Ensure that the priority of the control transaction is high enough to enable it 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 that screen the SDUMP SVC (SVC 51), do not install Fault Analyzer dump screening - either via the CFA transaction or by the IDIPLTS CICS® PLT program. If another screening program is installed when you attempt to install Fault Analyzer dump screening, the Fault Analyzer code is not installed and a message is issued.

Sample definition job

Figure 209: Sample Fault Analyzer CICS program and transaction definition job on page 400 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 this example, but can be changed if you desire.

Figure 209. Sample Fault Analyzer CICS program and transaction definition job

```
//IDICICS JOB  ...
//IDICICS EXEC PGM=DFHCSDUP,REGION=1024K,
//            PARM='CSD(READWRITE),PAGESIZE(60),NOCOMPAT'
//STEPLIB  DD DISP=SHR,DSN=xxx.SDFHLOAD
//DFHCSD   DD DISP=SHR,DSN=xxx.DFHCSD
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DEF PROGRAM(IDIPLT) GROUP(FA) EXECKEY(CICS)
      CEDF(NO) DATALOCATION(ANY)
      CONCURRENCY(QUASIRENT)
  DEF PROGRAM(IDIPLTD) GROUP(FA) EXECKEY(CICS)
      CEDF(NO) DATALOCATION(ANY)
      CONCURRENCY(QUASIRENT)
  DEF PROGRAM(IDIPLTS) GROUP(FA) EXECKEY(CICS)
      CEDF(NO) DATALOCATION(ANY)
      CONCURRENCY(QUASIRENT)
  DEF PROGRAM(IDIXCX53) GROUP(FA)
      CEDF(NO) DATALOCATION(ANY)
      CONCURRENCY(QUASIRENT)
  DEF PROGRAM(IDIXCCEE) GROUP(FA)
      CEDF(NO) DATALOCATION(ANY)
      CONCURRENCY(QUASIRENT)
  DEF PROGRAM(IDIXFA) GROUP(FA) EXECKEY(CICS)
      CEDF(NO) DATALOCATION(ANY)
      CONCURRENCY(QUASIRENT)
  DEF PROGRAM(IDIVPCLE) GROUP(FA)
      CONCURRENCY(QUASIRENT)
  DEF TRANSACTION(CFA) GROUP(FA)
      PROGRAM(IDIXFA) TASKDATALOC(ANY)
      SHUTDOWN(ENABLED)
  DEF MAPSET(IDIXMAP) GROUP(FA)
  ADD G(FA) L(list-name)
/*
```

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.

The ABCODE keyword must be used on an EXEC CICS® ABEND statement in order for Fault Analyzer to be invoked. For example:

```
EXEC CICS ABEND ABCODE('abcd') END-EXEC
```

If the NODUMP keyword is used on an EXEC CICS® ABEND statement, then Fault Analyzer only performs analysis if invoked via the IDIXCCEE exit.

Controlling CICS transaction abend analysis

Once the CFA transaction is installed (you might have chosen to install it under a different name, as per the above), then it can be used to install or uninstall the following Fault Analyzer invocation exits:

- XPCABND CICS® global user exit
- XDUREQ CICS® global user exit
- LE abnormal termination exit
- XEIIN global user exit

In addition, the CFA transaction can be used to install or uninstall the Fault Analyzer SDUMP screening feature.

Prior to installing either the XPCABND or XDUREQ exits, the CFA transaction issues a CICS® NEWCOPY command for program IDIXCX53 if the exit is in the "Uninstalled" state. To load a new copy of IDIXCX53, for example after applying maintenance, use the CFA transaction to uninstall the XPCABND and XDUREQ exits, then reinstall the IDIXCX53 exit.

There are two ways to interact with the CFA transaction; either from a CICS® terminal, or from the MVS™ console. Each of these are described in the following sections.

Using CFA from a CICS® terminal

To use the CFA transaction from a CICS® terminal, simply enter CFA. You can optionally pass a command parameter as described in Using CFA from an MVS console on page 403. The initial display is similar to the following:

Figure 210. Sample CFA transaction display

```
                   Fault Analyzer Control Transaction

Options: I=Install U=Uninstall
                  Current Status/Error Message
_ XPCABND         Installed
_ XDUREQ          Installed
_ LE Exit         Installed
_ SDUMP Screening Installed
_ XEIIN           Uninstalled

        Current    HWM  Setting    MWS
Active     0000   0000     0002
Waiting    0000   0000     0020   0123









IDITRACE OFF DeferredReport ON

PF1=Help PF3=Exit PF4=Opts PF5=Clear FND Area PF9=IVP PF11=TRACE ENTER=Update
```

Initially, the display shows the current status of the CICS® Fault Analyzer exit points, plus details of any active and waiting Fault Analyzer tasks. By entering an I or U (for Install or Uninstall) next to a specific exit point, its status can be changed accordingly. If there is an active analysis task (as shown in the example above), then a CICS® TASK FORCEPURGE can be issued for that task by entering an F in the input field next to the active task details. This function is only possible if CICS® transaction isolation is INACTIVE, or if ACTIVE, that the IDIXFA program is defined to have an EXECKEY of CICS®.

PF9 can be used to display a list of IVP tests. For details, see Verifying Fault Analyzer customization under CICS on page 432.

The PF11 function is described in IDITRACE under CICS on page 402.

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 211: Sample CFA Exit Options display on page 402.

Figure 211. Sample CFA Exit Options display

```
                       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  OFF
  Check MAX and Current values. . . .  CheckMaxCurr          OFF
Retain CICS dumps . . . . . . . . . .  RETAINCICSDUMP . . .  AUTO
  Suppress dumps from EXEC CICS DUMP.  IncludeExecCicsDump   NO
Fast duplicate minutes  . . . . . . .  NODUP  . . . . . . .  00000
  Include EXEC CICS DUMP. . . . . . .  IECD . . . . . . . .  ON
Use signed-on user for analysis . . .  CICSTRANANALYSISUSER  NO






           PF3=Exit
```

These options are updated each time a new fault entry is created to reflect the values that are in effect via the IDIOPTS DD statement or IDICNFxx parmlib member.

Clearing the NoDup(CICSFAST(…)) recording area

Under CICS®, Fault Analyzer maintains a history of recent abends that it has processed, called the FND (fast-no-dup) area. This area is used in conjunction with the NoDup(CICSFAST(…)) option. From the CFA transaction display, press PF5 to access the history display. PF5 can then be used to clear this area, which means that no previous abends are used in subsequent FAST duplicate determination. You are prompted to press PF5 again to confirm that you want to clear the FND area.

IDITRACE under CICS

PF11 from the CFA transaction can be used to turn the Fault Analyzer debugging trace, IDITRACE, ON or OFF. The trace is written to the IDITRACE DDname, dynamically allocated to the JES spool. For example, entering '?' against a CICS® region under SDSF, might result in the following display with the IDITRACE output selectable with 'S' at ❶:

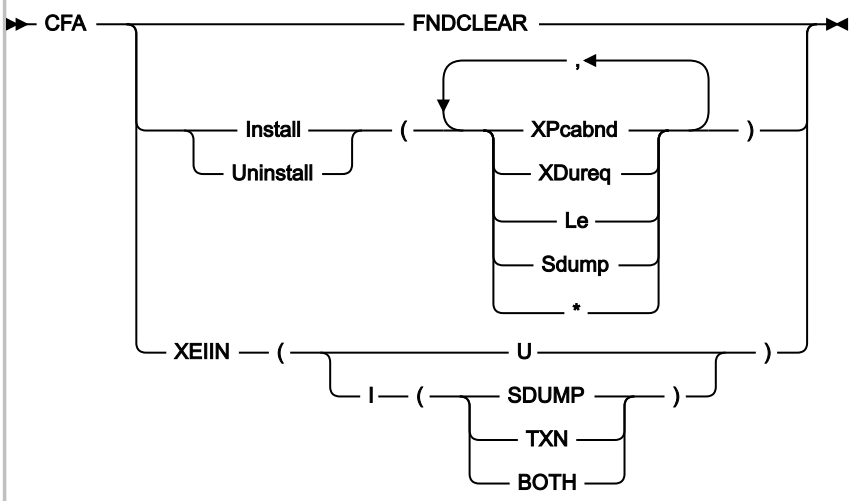
Figure 212. Sample CICS region SDSF Job Data Set display

```
  Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF JOB DATA SET DISPLAY - JOB AS650F1  (JOB31639)     DISPLAY RESET
COMMAND INPUT ===>                                          SCROLL ===> CSR
PREFIX=*  DEST=(ALL)  OWNER=SOMEONE  SYSNAME=
NP   DDNAME   StepName ProcStep DSID Owner    C Dest              Rec-Cnt Page
     JESMSGLG JES2                2 SOMEONE  X                         4
     JESJCL   JES2                3 SOMEONE  X                       259
     JESYSMSG JES2                4 SOMEONE  X                     2,173
     SYSPRINT AS650F1  AS650F1   103 SOMEONE  X                       126
     MSGUSR   AS650F1  AS650F1   105 SOMEONE  X                     2,328
     PLIMSG   AS650F1  AS650F1   106 SOMEONE  X                         0
     DFHCXRF  AS650F1  AS650F1   107 SOMEONE  X                       641
     COUT     AS650F1  AS650F1   108 SOMEONE  X                         0
     CEEMSG   AS650F1  AS650F1   109 SOMEONE  X                         0
     CEEOUT   AS650F1  AS650F1   110 SOMEONE  X                         0
     PRINTER  AS650F1  AS650F1   116 SOMEONE  X                         0
❶    IDITRACE AS650F1  AS650F1   117 SOMEONE  X                       172
     CRPO     AS650F1  AS650F1   119 SOMEONE  X                         0
```

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.

Figure 213. CFA command syntax



where:

FNDCLEAR

Clears the FND area (see description in Using CFA from a CICS terminal on page 401).

INSTALL | UNINSTALL

Installs or uninstalls specific Fault Analyzer CICS® exits (other than the XEIIN global user exit). Each of the options below can be specified in any sequence, enclosed in parenthesis:

XPCABND

Installs or uninstalls the CICS® XPCABND global user exit.

XDUREQ

Installs or uninstalls the CICS® XDUREQ global user exit.

LE

Installs or uninstalls the CICS® IDIXCCEE LE abnormal termination exit.

SDUMP

Installs or uninstalls CICS® SVC dump screening.

*****

Installs or uninstalls all of the above (XPCABND, XDUREQ, LE and SDUMP).

XEIIN

Installs or uninstalls the Fault Analyzer CICS® XEIIN global user exit.

U

Uninstalls the XEIIN exit.

I

Installs the XEIIN exit for the specified operation:

SDUMP

Installs the CICS® XEIIN global user exit for system dumps.

TXN

Installs the CICS® XEIIN global user exit for transaction dumps.

BOTH

Installs the CICS® XEIIN global user exit for both system and transaction dumps.

Examples:

- To uninstall all exits from CICS® region CICS01, enter the command:

```
F CICS01,CFA U(*)
```

- To install the LE exit for use by CICS® region TESTCICS, enter the command:

```
F TESTCICS,CFA I(L)
```

- To install the XPCABND and XDUREQ exits for use by CICS® region PRODCICS, enter the command:

```
F PRODCICS,CFA I(XP,XD)
```

- To clear the FND area in CICS® region MYCICS, enter the command:

```
/F MYCICS,CFA FNDCLEAR
```

- To uninstall the XEIIN exit from CICS® region CICSP1, enter the command:

```
/F CICSP1,CFA XEIIN(U)
```

- To install the XEIIN exit for system dumps by CICS® region CICSA, enter the command:

```
/F CICSA,CFA XEIIN(I(SDUMP))
```

CICS web services transaction

In addition to the dynamic control transaction, CFA (see Enabling dynamic control of analysis of CICS transaction abends on page 399), Fault Analyzer also provides the same capabilities using a CICS web services transaction.

To install this optional feature, the following steps are required:

1. Create a directory in z/OS® UNIX System Services to hold the CICS document templates.
   The CICS web services transaction uses a CICS document template which is read from a z/OS® UNIX System Services directory and that directory/path is referenced in the CICS document template resource definition. Create a z/OS® UNIX System Services directory, using local naming standards, for example, `/faultanalyzer/doctemplate`.
2. Copy the CICS document templates to the directory.
   Sample job IDISCHFZ, found in data set IDI.SIDISAM1, is provided to copy the document templates to the z/OS® UNIX System Services directory created in step 1. Review the instructions in the sample job. The main requirement is to update the z/OS® UNIX System Services path to the one created above, for example, `/faultanalyzer/doctemplate`.
3. Make the necessary CICS resource definitions.
   Sample job IDISCIC2, found in data set IDI.SIDISAM1, is provided to make the necessary CICS resource definitions. Review the instructions in the sample job. The directory name created in step 1, for example, `/faultanalyzer/doctemplate`, needs to be referenced in the two DOCTEMPLATE definitions.

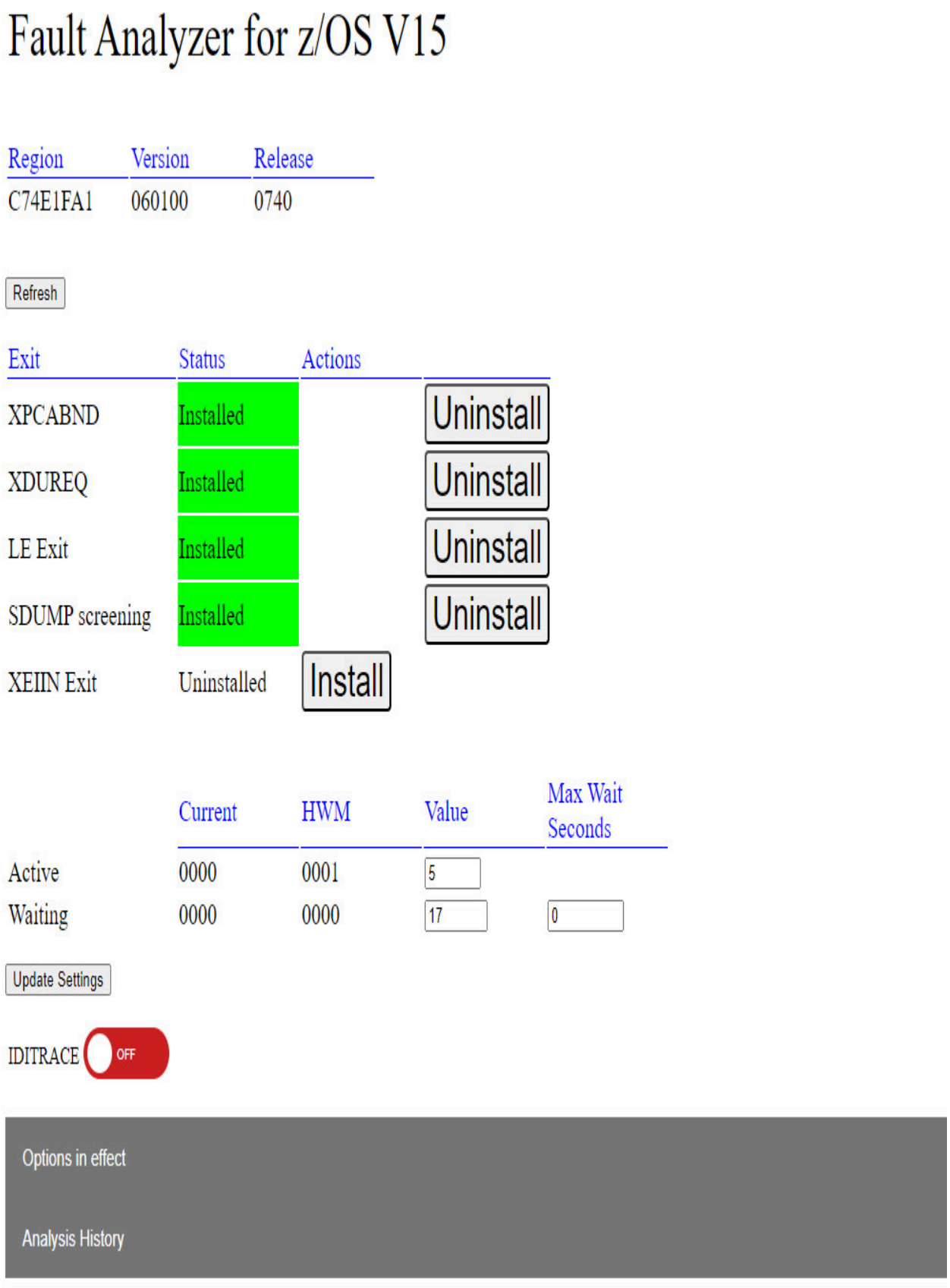
   The sample job defines a CICS TCPIPSERVICE. However, an existing TCPIPSERVICE could also be used. Once the definitions have been made, install the GROUP.
4. Test the CICS web transaction.
   To test the Fault Analyzer CICS web services, enter the following transaction in your browser:

   ```
   http://host:port/IDIXFAW
   ```

   Where:
   - `<host>` is the host name/IP address where CICS is running.
   - `<port>` is the port number as specified in the TCPIPSERVICE definition.

If successful, a screen similar to the following is displayed.

Figure 214. CICS Web transaction



Ensuring transaction abend analysis is not suppressed by DUMP(NO)

If the active transaction definition for an abending transaction has the DUMP(NO) option specified, then CICS® does not call the XPCABND global user exit, and Fault Analyzer is not invoked. To check the DUMP setting for a transaction, do one of the following:

- Use the CEDA transaction to view the transaction definition in question and check the DUMP(YES|NO) setting. Care should be taken when using this method. There might be multiple definitions of the same transaction, so the order in which the definitions are installed by CICS® is important.
- Check the active transaction definition in a dump.
- Use the CICS®-supplied transaction, CECI, to check the DUMP setting for the active transaction. This checking can be done by issuing the following command:

  ```
  CECI INQUIRE TRANSACTION(nnnn)
  ```

  where *nnnn* is the transaction ID in question.

  Having issued this command, the displayed DUMPING value has the following meaning:
  - A value of 00186 means DUMP(YES).
  - A value of 00187 means DUMP(NO).

CICS NoDup(CICSFAST) override assembler exit (IDINDFUE)

In certain circumstances where Fault Analyzer has detected an abend as a CICS® NoDup(CICSFAST) duplicate (see NoDup on page 616 for information about the NoDup option), it might be desirable to override this detection, and hence instigate normal abend analysis. These circumstances might, for example, be for specific abends or transactions, or combinations of these, for which a full analysis is required. To accommodate this situation, there exists a NoDup(CICSFAST) assembler exit, IDINDFUE. Fault Analyzer attempts to load this exit program before issuing the CICS® NoDup(CICSFAST) message, IDI0066I on page 710. If the load is successful, then program IDINDFUE will be invoked.

Invocation

IDINDFUE must be written in assembler language. It is called by using the OS linkage convention.

IDINDFUE might be called from IDIXCCEE or IDIXCX53, that is, from the CICS® XPCABND Global User Exit (GLUE). It must adhere to the CICS® rules for coding an XPCABND GLUE program. See the CICS® customization guide for details. When you are assembling IDINDFUE, do not use the CICS® command-level translator.

The exit program must be:

- Link-edited into a load module named IDINDFUE.
- Placed into the DFHRPL concatenation of your CICS® region if auto load is active.
- Included in the PPT if auto load is not active.

Entry specifications

Contents of registers 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 12. IDINDFUE input parameters

Parameter	Number of bytes	Description
Parameter 1	256	User work area
Parameter 2	4	Abend code
Parameter 3	4	Transaction ID
Parameter 4	8	Abending program name

Return specifications

Contents of register on return from IDINDFUE:

Register

Contents

0-14

Unchanged.

15

Return code:

0

Indicates that the abend should continue to be treated as a duplicate.

1

Indicates that abend analysis should proceed, that is, overriding the NoDup(CICSFAST) detection.

Note: If an IDINDFUE exit passes back a return code of 1, indicating that analysis should be performed, no NoDup(NORMAL) duplicate detection is performed.

Example

A softcopy of the sample user exit shown in the following is also provided as member IDINDFUE in data set IDI.SIDISAM1. This sample assembler exit sets R15 to 1 if the transaction ID is FTN1, and the abend code is FABN.

```
IDINDFUE CSECT
         PRINT ON,NOGEN
         STM   R14,R12,12(R13)
         LR    R12,R15
         USING IDINDFUE,R12
         L     R4,0(,R1)          Work Area
         L     R2,4(,R1)          Abend code
         L     R3,8(,R1)          Transaction ID

         SR    R15,R15            Assume no continuance.
         CLC   0(4,R2),=C'FABN'   Check Abend code
         BNE   RETURN
         CLC   0(4,R3),=C'FTN1'   Check Transaction ID
         BNE   RETURN
         LA    R15,1              Indicate analysis to be performed
*
RETURN   DS    0H
         L     R14,12(,R13)
         LM    R0,R12,20(R13)
         BR    R14                Return to Fault Analyzer
         END   IDINDFUE
```

CICS trace considerations

When Fault Analyzer gathers the trace entries for the abending task, a storage area is used to copy the trace entries into. The size of this storage area, and hence the number of trace entries copied, is determined as follows:

- For CICS® TS 5.1 and above, the CICS® system initialization parameter, TRTRANSZ is used.
- For earlier CICS® releases, minimum of 64K is used, unless the CICS® system initialization parameter, TRTRANSZ, specifies a value higher than 64K, in which case that value is used, up to a maximum of 256K.

Preventing LE from causing the CICS trace to wrap

When a CICS® transaction abends and Language Environment® is active in the abending enclave, Language Environment® by default writes diagnostic information to a transient data queue named CESE. This situation occurs if the IBM-supplied Language Environment® default runtime option TERMTHDACT(TRACE) is in effect. Because these diagnostics are recorded before Fault Analyzer receives control to process the abend, the CICS® trace table is liable to wrap around, and application trace data might therefore be lost. Depending on your level of MVS™, it is recommended that the TERMTHDACT option is set to one of the following:

TERMTHDACT(TRACE,CICSDDS,...)

This value causes Language Environment® to write its diagnostics to the CICS® transaction dump data set.

TERMTHDACT(QUIET)

This value suppresses most of the Language Environment® diagnostics.

Specifying CICS options through the IDIOPTS DDname

To avoid the need to recycle CICS® in case compiler listing or side file data sets change, specify these via the DataSets option in a user options file that is pointed to by the IDIOPTS DDname. For details, refer to DataSets on page 581 and User options file IDIOPTS on page 575.

It is also better to use the IDIOPTS DDname pointing to a data set and containing:

```
DataSets(IDIHIST(history-file-dsn))
```

than to use the following statement in the CICS® JCL:

```
//IDIHIST DD DISP=SHR,DSN=history-file-dsn
```

If you use

```
//IDIHIST DD DISP=SHR,DSN=history-file-dsn
```

then JCL allocation holds an ENQ on the history file data set name for the full time when CICS® is running. However, if you use

```
DataSets(IDIHIST(history-file-dsn))
```

then the allocation is dynamically obtained and released when required by Fault Analyzer. As a result, history file maintenance and renames can be done while a CICS® system that has written to it is still running. The

```
DataSets(IDIHIST(history-file-dsn))
```

option can even be changed to point to a new history file while CICS® is running and takes effect when the next fault entry is created.

The IDIOPTS data set used must be a PDS or PDSE to permit update using ISPF EDIT while CICS® is running.

Language Environment abend considerations

If an abend occurs in Language Environment® while trying to recover from a transaction abend under CICS®, then Fault Analyzer is not invoked. This lack of invocation is because CICS® normal behavior for these types of abends is to not drive the XPCABND exit. However, by enabling the CICS® LE abnormal termination CEECXTAN CSECT exit IDIXCCEE, which is provided with Fault Analyzer, these types of abends are still analyzed. See Configuring Language Environment for CICS to invoke Fault Analyzer on page 397 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 400 for details about enabling this exit.

Installing the MVS post-dump exit IDIXTSEL

The Fault Analyzer post-dump exit, IDIXTSEL, is installed in the MVS™ IEAVTSEL CSECT installation exit list. The exit, which is only invoked for SVC dumps, is installed by the USERMOD, IDIWTSEL. It is used to register CICS® system abend dumps, recovery fault recording SDUMPs, SLIP dumps, and to facilitate the capturing of Java™ faults.

To install this USERMOD, edit and submit the sample job IDIWTSEL. This sample job includes IDIXTSEL in the IEAVTSEL installation exit list. If you have other exits defined in this list, add the IDIXTSEL exit last.

For more information about adding exits to IEAVTSEL, see *MVS™ Installation Exits*.

To activate this change, IPL again or cancel the DUMPSRV address space so that it restarts with the new exit.

For dump registration via this exit to occur, it is necessary to also:

- Start the Fault Analyzer IDIS subsystem. For information on this, see Using the Fault Analyzer IDIS subsystem on page 318.
- Ensure that the default Fault Analyzer history file is a PDSE, or that an Analysis Control dump registration user exit is used to assign the registration fault entry to a PDSE history file. Dump registration fault entries can only be created in a PDSE history file.

For details about specifying a default history file name, see Parmlib member IDICNFxx on page 369.

For details about the Analysis Control dump registration user exit, see Analysis Control user exit (MVS SVC Dump registration) on page 479.

See Verifying dump registration (IEAVTSEL) on page 439 for information about how to test the dump registration setup.

Maximizing CICS transaction abend analysis performance

To maximize CICS® transaction abend analysis performance, the following should be considered:

- Using the Fault Analyzer IDIS subsystem to manage the access to history file $$INDEX members. For details, see Caching of history file $$INDEX data on page 319.
- Using the DeferredReport option. For details, see DeferredReport on page 589.

  You can specify this option via the IDIOPTS DDname, as explained in Specifying CICS options through the IDIOPTS DDname on page 411, if you want to make it applicable to CICS® only.

Implementing an XEIIN global user exit

The XEIIN global user exit can be useful if an overlay is not being detected until CICS® task termination is doing the final freeing of storage. CICS® can detect a Storage Accounting Area (SAA) overlay either during an explicit FREEMAIN or during task termination. If detected at task termination, the task is not abended by CICS® and hence Fault Analyzer is not invoked. CICS® attempts to take a system dump, with dump code SM0102, however if CICS® system dumps are being suppressed, further analysis is not possible.

To assist with SAA overlays Fault Analyzer provides a program that can optionally be installed in the XEIIN global user exit. If installed, this program, at each EXEC CICS® RETURN statement, validates all the CICS® storage accounting areas for the current task. If an overlay is found, then one of two actions can be performed:

1. A call can be made to take a CICS® transaction dump with dumpcode IDIS.

   If Fault Analyzer is installed in the XDUREQ global user exit, then this transaction dump is analyzed in the normal manner with a history file fault entry being created.

2. An SDUMP of the CICS® region can be taken.

   The request to take this SDUMP is made by the IDIS subsystem, and as such, it is not affected by the CICS® system dump suppression setting. If a SDUMP is captured, then it can then be analyzed interactively by way of the **File** menu option 5.

Enabling the Fault Analyzer XEIIN global user exit program, and setting which action should be performed, is done using the Fault Analyzer supplied CFA transaction. Unlike the other Fault Analyzer CICS® exits, there is no option to enable the XEIIN exit during CICS® PLT processing.

When invoked, the initial CFA display is 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
_ XEIIN            Uninstalled

        Current    HWM  Setting    MWS
Active     0000   0001     0001
Waiting    0000   0000     0017   0000




IDITRACE OFF DeferredReport ON

PF1=Help PF3=Exit PF4=Opts PF5=Clear FND Area PF9=IVP PF11=TRACE ENTER=Update
```

From this display, an "I" can be entered next to XEIIN to install the exit. A screen similar to the following is then displayed:

```
                    Fault Analyzer Control Transaction
Options: I=Install U=Uninstall
                  Current Status/Error Message
_ XPCABND         Installed
_ XDUREQ          Installed
_ LE Exit         Installed
_ SDUMP Screening Installed
_ XEIIN           Installed   N TXN dump N SDUMP Set at least one option to Y

        Current    HWM  Setting    MWS
Active     0000   0001     0001
Waiting    0000   0000     0017   0000









IDITRACE OFF DeferredReport ON

PF1=Help PF3=Exit PF4=Opts PF5=Clear FND Area PF9=IVP PF11=TRACE ENTER=Update
```

In this screen, enter "Y" next to either `TXN dump` or `SDUMP` to set the action to take when an SAA overlay is detected.

The I (Install) option must be used each time the TXN or SDUMP field is changed.

Note: Although there is no overhead in having any other Fault Analyzer exit installed/enabled, the XEIIN exit is invoked by CICS® on every EXEC CICS® command, and hence does incur a small overhead. As such, it is recommended to only use the XEIIN exit when needing to investigate SAA overlays.

Disabling 3270 screen buffer capture

To capture 3270 screen data, Fault Analyzer issues command RECEIVE BUFFER to the terminal. However, the following errors might occur:

- The terminal is not an actual 3270 device
- The terminal is not in a state to handle the command F2 READBUF

In such cases, CICS® abends the RECEIVE BUFFER with abend ATNI, which results in recursive calls to the XPCABND and XDUREC global user exits.

If it is an issue that the ATNI abends, add the following JCL statement to your CICS® start-up proc:

```
//IDINOCRB DD DUMMY
```

The statement tells Fault Analyzer to skip the 3270 screen capture (RECEIVE BUFFER) for all 3270 devices in the region. As a result, the section **Last CICS 3270 Screen Buffer** is not included in the Fault Analyzer analysis report.

Chapter 22. Customizing the DB2 environment

This section contains information specific to Fault Analyzer in DB2® environments.

Binding DB2

To run Fault Analyzer against abends occurring in applications that use DB2®, you must ensure that the DB2® Call Level Interface (CLI) is installed and the required setup has been performed to bind plan DSNACLI.

The DSNACLI plan can be created using the sample job in member DSNTIJCL of the DB2® SDSNSAMP data set. It is important that the DSNTIJCL job completes with return code zero. As stated in the job comments, it might be necessary to add the SQLERROR(CONTINUE) parm to achieve this. Refer to the *DB2® for OS/390® Call Level Interface Guide and Reference* for further information.

Ensure that the Fault Analyzer IDIS subsystem has SELECT authority to the required DB2® system catalog tables. See IDIS subsystem requirements for DB2 on page 323 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(DSNTEST1) QUALIFIER(DSN8510) MEMBER(DACBD001) -
      ACT(REP) ISOLATION(CS)
 RUN  PROGRAM(DSNTIAD) PLAN(DSNTIA51)  -
      LIB('DSN510.RUNLIB.LOAD')
 RUN  PROGRAM(DACBD001) PLAN(DSNTEST1) -
      LIB('CTEST.DB2.LOAD') -
      PARMS('/TERMTHDACT(UADUMP) ')
 END
//*
```

Improving Fault Analyzer DB2 performance

The following information is applicable to DB2® versions prior to DB2® V10. For DB2® V10 or later, this information can be ignored.

DB2® does not have an index defined on the SYSIBM.SYSDBRM catalog table. Fault Analyzer accesses the SYSIBM.SYSDBRM catalog table whenever it performs analysis for a DB2® fault. To avoid the possibility of poor performance when 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            ❶
          ON SYSIBM.SYSDBRM
         (
          PLNAME ASC,
          NAME ASC
         )
         USING STOGROUP mmmmmm        ❶
               PRIQTY pp              ❷
               SECQTY ss              ❷
CLOSE NO;
```

Notes:

❶

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.

❷

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
         CEEXOPT TERMTHDACT=(UADUMP)
         END
//*
//*        STEP  2: COMPILE COBOL PROGRAM
//*
//COBCOMP  EXEC IMSCOBOL
//COB.SYSIN    DD DSN=DA.IMSSAMP.COBOL(BATCHJ2),DISP=SHR
//COB.SYSPRINT DD DSN=DA.LISTING.COBOL(BATCHJ2),DISP=SHR
//LKED.FRED    DD DSN=*.HLASM.SYSLIN,DISP=OLD
//LKED.SYSIN   DD *
  Include FRED(CEEUOPT)
  NAME    BATCHJ2(R)
/*
//*
//*        STEP  3: RUN THE PROGRAM
//*
//PROGRUN  EXEC PROC=DLIBATCH,MBR=BATCHJ2,PSB=PSB1,COND=(4,LT),
//         DBRC=Y,MON=Y,FMTO=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 Java environment

Invoking Fault Analyzer from a Java try-catch block

To capture the current Java™ state from a Java™ program, Fault Analyzer can be invoked to create a history file fault entry, along with an associated MVS™ SVC dump.

The Fault Analyzer history file that is used is the default history file. To use a different history file, specify it with the _IDI_OPTS or _IDI_OPTSFILE environment variables described in Options on page 571. The fault entry that is created can subsequently be reanalyzed using the Fault Analyzer ISPF interface to review the Java™ and native code that was executing when Fault Analyzer was called.

To facilitate Java™ dump capture, the following is required:

- ALTER access must be granted to the Fault Analyzer IDI_SDUMP_ACCESS XFACILIT profile.

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

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

- The MVS™ post-dump exit IDIXTSEL must be installed. For details, see Installing the MVS post-dump exit IDIXTSEL on page 412.
- The IDIS subsystem must be started. For details, see Using the Fault Analyzer IDIS subsystem on page 318.

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

Figure 215. Syntax

You can specify an optional comment character string 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™ try-catch block:

```
public class JavaTest {
  public static void main() {
    ...
    try {
      ...
    }
    catch() {
      ...
      com.ibm.faultanalyzer.Snap.dump("Java error");  // Call Fault Analyzer
    }
  }
}
```

Alternative snap dump invocation methods

There are several overloaded versions of the Java™ "Snap.dump" method that you can use to invoke Fault Analyzer:

```
/**
  * Create a Fault Entry
  *
  * @param dumpTitle The user title to use for the Fault Entry. (non-null & non-empty).
  */
public static void dump(String dumpTitle)

/**
  * Create a Fault Entry
  *
  * @param dumpTitle The user title to use for the Fault Entry. (non-null & non-empty).
  *
  * @return The ID of the created Fault Entry, in the form <history file dataset>(<fault id>).
  * E.g. IDI.HIST(F00001)
  *
  */
public static String dump(String dumpTitle, Throwable t)

/**
  * Create a Fault Entry.
  * (This should only be used when gathering diagnostic information requested by IBM Support.)
  *
  * @param dumpTitle The user title to use for the Fault Entry. (non-null & non-empty).
  * @param debug Specifying 'true' will enable verbose debug information.
  * Provide the output to IBM Support.
  *
  * @return The ID of the created Fault Entry, in the form <history file dataset>(<fault id>).
  * E.g. IDI.HIST(F00001)
  */
public static String dump(String dumpTitle, Throwable e, boolean debug)
```

Adding the Snap class to the application class path

In the following, IDIXJAxx refers to:

- IDIXJAVA if using a version of Java earlier than version 11.
- IDIXJA11 if using Java version 11 or later.

To facilitate calling the com.ibm.faultanalyzer.Snap.dump method, the Fault Analyzer IDIXJAxx Java™ library must either exist in the application program build path, or be available via the current class path.

The IDIXJAxx Java™ library can be obtained through the following steps:

If using a version of Java JVM earlier than version 11:

1. Do one of the following:
   - Copy the jar file your application directory on z/OS by using JCL similar to the following:

     ```
     //* --- Copy IDIXJAVA to an zFS directory:
     //CPYXJAVA EXEC PGM=BPXBATCH
     //STDPARM DD *
     SH cp "//'IDI.SIDIDOC1(IDIXJAVA)'" /u/hunter2/idixjava.jar
     /*
     //STDOUT DD SYSOUT=*
     //STDERR DD SYSOUT=*
     ```

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

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

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

If using Java JVM version 11 or later:

1. Do one of the following:
   - Copy the jar file your application directory on z/OS by using JCL similar to the following:

     ```
     //* --- Copy IDIXJA11 to an zFS directory:
     //CPYXJAVA EXEC PGM=BPXBATCH
     //STDPARM DD *
     SH cp "//'IDI.SIDIDOC1(IDIXJA11)'" /u/hunter2/idixja11.jar
     /*
     //STDOUT DD SYSOUT=*
     //STDERR DD SYSOUT=*
     ```

   - Perform binary FTP transfer of IDI.SIDIDOC1(IDIXJA11) to your project development directory as file IDIXJA11.

     Note: The IDI.SIDIDOC1(IDIXJA11) data set and member are created as part of the Fault Analyzer SMP/E installation and might exist with a different high-level qualifier.

2. Do one of the following:
   a. Configure your project build path to include IDIXJA11 as an external JAR library dependency.
   b. Add the directory and file name to the current ClassPath.

Fault Analyzer Java SVC dump titles

The Fault Analyzer Java dump title depends on whether the dump is written using the IDISNAP interface or the BPXBATCH interface.

- **Java IDISNAP interface SVC dump title**

  history-file-name(*fault-id*)^^SVCDUMP(0x *asid*).JAVA IDISNAP^

  Note: ^ represents the non-printable character X'00'.

  The following is an example of a Java IDISNAP interface SVC dump title:

  ```
  TESTFA1.TEMP.DA5E1SP.DAJAVAZZ.JC83201.HIST(F00027).    .SVCDUMP(0x0138).JAVA IDISNAP.
  ```

- **Java BPXBATCH interface SVC dump title**

  history-file-name(*fault-id*)^^SVCDUMP(0x *asid*).JAVA IDI0162I^

  Note: ^ represents the non-printable character X'00'.

  The following is an example of a Java BPXBATCH interface SVC dump title:

  ```
  TESTFA1.TEMP.DA4E1SP.DAJAVAZZ.JZJ2CO2K.HIST(F00700).   .SVCDUMP(0x00DB).JAVA IDI0162I.
  ```

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 https://www.ibm.com/docs/en/sdk-java-technology/8?topic=options-xdump. This reference describes the Java™ internal events that can trigger a dump and the filters that can be applied.

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

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

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

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

The range suboption can be used to limit the number of dumps generated for a specific exception. In the above example, providing a range 1..4 means that dumps are only generated for four NullPointerException or SocketException throw events, and any more are ignored. This option is especially useful in cases where exceptions are caught and rethrown, but it does mean future Exceptions are not processed.

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

Using the Fault Analyzer Java wrapper utility

Fault Analyzer Java wrapper utility wraps a callable Java application in a try-catch block by using the FA Java class.

The try-catch block traps any unhandled exceptions from the Java™ application and calls Fault Analyzer using the com.ibm.faultanalyzer.Snap.dump method.

Usage

If using a version of Java JVM earlier than version 11:

Copy the IDIXJAVA jar file as binary to a zFS directory:

```
//* --- Copy IDIXJAVA to a zFS directory:
//CPYXJAVA EXEC PGM=BPXBATCH
//STDPARM DD *
SH cp "//'IDI.SIDIDOC1(IDIXJAVA)'" /u/hunter2/idixjava.jar
/*
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
```

If using a Java JVM version 11 or later:

Copy the IDIXJA11 jar file as binary to a zFS directory:

```
//* --- Copy IDIXJA11 to zFS directory:
//CPYXJAVA EXEC PGM=BPXBATCH
//STDPARM DD *
SH cp "//'IDI.SIDIDOC1(IDIXJA11)'" /u/hunter2/idixja11.jar
/*
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
```

Invoking the Fault Analyzer wrapper

You can call the wrapper utility through the command line or as part of a batch job (JCL).

Use either of the following commands to invoke the wrapper utility from a command line application:

If using a version of Java JVM earlier than version 11:

- `java -jar idixjava.jar <clsName> <mainArgs>`

or

- `java -cp idixjava.jar FA <clsName> <mainArgs>`

If using Java JVM version 11 or later:

- `java -jar idixja11.jar <clsName> <mainArgs>`

or

- `java -cp idixja11.jar FA <clsName> <mainArgs>`

For a Java™ batch application that is invoked using BPXBATCH:

If using a version of Java JVM earlier than version 11:

```
//* --- Launch an earlier than version 11 batch Java application with
//* --- Fault Analyzer wrapper
// EXPORT SYMLIST=*
// SET ZFSDIR=<target directory>
//CPYXJAVA EXEC PGM=BPXBATCH
//STDENV DD *
JAVA_HOME=/usr/lpp/java800/31bit/J8.0/ &&
PATH=${PATH}:${JAVA_HOME}/bin
CLASSPATH=/u/hunter2/classes
/*
//STDPARM DD *
SH java -jar /u/hunter2/idixjava JavaApp arg1 arg2
/*
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
```

If using Java JVM version 11 or later:

```
//* --- Launch a batch Java version 11 or later application with
//* --- Fault Analyzer wrapper
// EXPORT SYMLIST=*
// SET ZFSDIR=<target directory>
//CPYXJAVA EXEC PGM=BPXBATCH
//STDENV DD *
JAVA_HOME=/usr/lpp/java800/31bit/J8.0/ &&
PATH=${PATH}:${JAVA_HOME}/bin
CLASSPATH=/u/hunter2/classes
/*
//STDPARM DD *
SH java -jar /u/hunter2/idixja11 JavaApp arg1 arg2
/*
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
```

Fault Analyzer passes the specified arguments to the main method of the user application. Any specified Java™ properties and environment variables are available to the user application as normal.

When an unhandled exception occurs, the resulting fault entry contains information about the Java™ event.

Figure 216. Sample event summary showing use of the Java wrapper utility

```
<H1> E V E N T   S U M M A R Y


The following events are presented in chronological order.

Event           Fail  Module   Program  EP
#  Type         Point Name     Name     Name     Event Location (*) Description
-- ------------ ----- -------- -------- -------- ------------------ ----------------------------------------
 1 JavaExc            n/a      n/a      Driver.main
                                                 L#4                From file:/u/testfa1/Driver.class
```

Using the Fault Analyzer JVMTI Agent

Fault Analyzer provides a JVMTI (JVM Tool Interface) agent. The agent runs inside a Java™ Virtual Machine (JVM) and can inspect the state of the running Java™ application.

The Fault Analyzer agent is added to the JVM at startup with the following command line option:

```
java -agentpath:<path-to-agent> <java Class Name>
```

For example, you could specify this command to run "HelloWorld" using the agent:

```
java -agentpath:IDIJAGNT.so HelloWorld
```

Can I add the agent after startup using Java™ instrumentation APIs?

No, the Fault Analyzer agent requests specific JVMTI capabilities that are only available before JVM initialization.

What does the Fault Analyzer Agent do?

When the JVM starts the agent requests that information about local variables be made available by the JVM. This means that the JVM runs in Full Speed Debug (FSD) mode.

For every exception the agent checks if there is a known catch block. If not, the agent creates a fault entry containing the local variables of the current thread stack frames.

What information does the Agent extract?

The agent obtains all accessible local variables with the current thread. The specific information gathered depends upon the kind of stack frames currently that are executing:

- Java Instance frames:
  - The executing line and file name is extracted
  - All local variable types, name and values are extracted.
  - All fields of the local instance are extracted, including modifiers, types, names, and values.
- Java static frames:

    - The executing line and file name is extracted
    - All local variable types, names, and values are extracted.
    - The static fields of the declaring class of the method, including static field modifiers, types, names, and values.
- Native frames:
    - The fields of the "local instance" are extracted, including modifiers, types, names, and values.
- Static native frames:
    - The static fields of the declaring class of the method, including static field modifiers, types, names, and values.

My Java program called a COBOL program using JNI and that abended. Will the agent report that?

No, the JVMTI API only allows access to Java™ exceptions. When an abend occurs, the JVM is no longer active and the information about local variable names is not available.

What can Fault Analyzer report for a JNI abend?

Fault Analyzer performs a post-mortem analysis using DTFJ APIs.

The DTFJ API allows access to a subset of local variable information; specifically all the object references returned by the JavaStackFrame interface. This API does not provide variable names, only the object values.

For more information about JavaStackFrame and the DTFJ API refer to the documentation for IBM® SDK, Java™ Technology Edition.

JVM version 9 issues and limitations

Some considerations if you are using the J9 VM.

If you are using Java™ Virtual Machine (JVM) version 9 (J9) with Language Environment (LE) enabled applications, you might need to run the JVM with the **-Xrs:sync** option to invoke Fault Analyzer for abends. For general information about J9 issues and limitations, see https://www.ibm.com/docs/en/sdk-java-technology/8?topic=reference-j9-issues-limitations.

Chapter 25. 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 Modifying your ISPF environment on page 327:

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 uses the ISPF LIBDEF service to logically place the data sets that are allocated to the IDIIPJPN, IDIIMJPN, IDIISJPN, and IDIITJPN DDnames ahead of the data sets that are allocated to the ISPPLIB, ISPMLIB, ISPSLIB, and ISPTLIB DDnames. The stacking feature of the LIBDEF service is used to ensure that any data sets defined using LIBDEF prior to invoking Fault Analyzer are restored on exit.

If a LIBDEF for either ISPPLIB, ISPMLIB, ISPSLIB, or ISPTLIB is already active at the time of invoking Fault Analyzer, then the existing LIBDEF data sets are included in the new LIBDEF, after the IDIIPJPN, IDIIMJPN, IDIISJPN, and IDIITJPN data sets. Because the maximum number of data sets that can be specified with LIBDEF when using the DATASET option is limited to 15, any data sets in excess of 14 that is already specified using LIBDEF, are not available. (This arithmetic assumes that only one data set is specified for the IDIIPJPN, IDIIMJPN, IDIISJPN, and IDIITJPN DDname, as is normally the case.) Therefore, it is important that any Fault Analyzer base function ISPF data sets, that are specified using LIBDEF at the time of invoking Fault Analyzer are among those that are included in the Fault Analyzer established LIBDEF, otherwise ISPF failures might result due to untranslated members not being found.

Setting the national language

The Language option (see Language on page 612) specifies the national language ID, which is used to select appropriate language-dependent data sets.

To make Japanese the default language for all users, specify the following option in your IDICNFxx parmlib member:

```
LANGUAGE(JPN)
```

Chapter 26. Verifying Fault Analyzer customization

The following topics provide instructions for different program languages and execution environments to verify the successful installation and customization of Fault Analyzer.

Verifying the use of XDUMPs

Regardless of which IVP is used to test the Fault Analyzer installation, XDUMPs can be tested by temporarily providing the option MaxMinidumpPages(0).

This option can be provided either via the IDIOPTS DDname, or in the IDICNFxx parmlib member.

Using this option will force a "spill" of all minidump pages to the XDUMP data set. As a result, IDI0194W on page 739 will also be issued. This is correct and expected in this situation.

Verifying the use of Fault Analyzer with assembler

AMODE 31

To verify Fault Analyzer with AMODE 31 assembler, edit and submit the sample job IDIVPASM in data set IDI.SIDISAM1. Refer to the instructions in the sample job for more information.

The job assembles and executes a program that abends with a system abend code of 0C7.

Since this program is a non-LE program, Fault Analyzer is invoked via the MVS™ change options/suppress dump exit IDIXDCAP.

The synopsis section of the Fault Analyzer report that is written to IDIREPRT should contain the following:

For the Japanese feature of Fault Analyzer, this section should be in Japanese if the Language(JPN) option is in effect.

```
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 #
  ------
  000042           CVB    R5,WORKD
```

```
The Assembler source code for data fields involved in the failure:
```

```
  List
  Stmt #
  ------
  000107  WORKD           DS D

Data field values at time of abend:

  WORKD = X'0000000002222278' (Address 00007E50 = R12 + X'110')
```

A complete sample report from running this IVP is provided as member IDISRP03 in the IDI.SIDISAM1 data set.

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 432 for more information.

AMODE 64

To verify Fault Analyzer with AMODE 64 assembler, edit and submit the sample job IDIVPAS6 in data set IDI.SIDISAM1. Refer to the instructions in the sample job for more information.

Verifying the use of Fault Analyzer with COBOL

This section is applicable only if you have COBOL installed at your site.

To verify Fault Analyzer with COBOL, edit and submit the sample job IDIVPCOB in data set IDI.SIDISAM1. Refer to the instructions in the sample job for more information.

The job compiles and executes a COBOL program, which abends with a return code of 3000.

As a result of the TER(UADUMP) LE option, Fault Analyzer is invoked via the MVS™ change options/suppress dump exit IDIXDCAP.

The synopsis section of the Fault Analyzer report that is written to IDIREPRT should contain the following:

For the Japanese feature of Fault Analyzer, this section should be in Japanese if the Language(JPN) option is in effect.

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 reason code X'0' occurred in module IDISCBL1 program IDISCBL1
entry point IDISCBL1 at offset X'29A'.

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' *** Invalid numeric data ***
  ERROR-COUNT = X'C1C2C3C4' *** Invalid numeric data ***
  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 applicable only if you have PL/I installed at your site.

AMODE 31

Depending on your version of PL/I, there are two separate AMODE 31 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 more information.

Each job compiles and executes a PL/I program, which abends and terminates the job step with a return code of 3000.

As a result of the TER(UADUMP) LE option, Fault Analyzer is invoked via the MVS™ change options/suppress dump exit IDIXDCAP.

The synopsis section of the Fault Analyzer report that is written to IDIREPRT should contain the following:

For the Japanese feature of Fault Analyzer, this section should be in Japanese if the Language(JPN) option is in effect.

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 entry point SUB1
at offset X'148'.
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 #
  ------
  000012       Array1(1) = Array1(2) / Divisor;

Data field values at time of abend:

  List
  Stmt #
  ------
  000010  ARRAY1(1) FIXED BIN(31,0) AUTO = X'00000001'
  000010  ARRAY1(2) FIXED BIN(31,0) AUTO = X'00000003'
  000010  DIVISOR   FIXED BIN(31,0) AUTO = X'00000000'
```

A complete sample report from running this IVP is provided as member IDISRP02 in the IDI.SIDIDOC11 data set.

AMODE 64

To verify Fault Analyzer with AMODE 64 Enterprise PL/I, edit and submit the sample job IDIVPPL6 in data set IDI.SIDISAM1. Refer to the instructions in the sample job for more information.

Verifying the use of Fault Analyzer with C

This section is applicable only if you have C installed at your site.

To verify Fault Analyzer with C, edit and submit the sample job IDIVPC in data set IDI.SIDISAM1. Refer to the instructions in the sample job for more information.

The job compiles and executes a C program in IDI.SIDISAM1(IDISRC1), which abends and terminates the job step with a return code of 3000.

The synopsis section of the Fault Analyzer report that is written to should contain the following:

For the Japanese feature of Fault Analyzer, this section should be in Japanese if the Language(JPN) option is in effect.

```
A system abend 0C9 occurred in module GO program IDISRC1 entry point main at
offset X'3E4'.
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 GO.  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.SIDISAM1(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 Fault Analyzer customization under CICS

This section is applicable only if you have CICS® installed, and have completed the customization of CICS® as described in Customizing the CICS environment on page 396.

Having installed the CFA transaction, and the exits that you want, invoke the CFA transaction from a CICS® terminal, and press PF9 to see the following display:

Figure 217. Sample CFA IVP Testing 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
```

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 on page 696 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 entry point IDIXFA at
offset X'D3C'.

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.
```

For this IVP, Fault Analyzer is expected to be invoked through the XPCABND exit.

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.
```

For this IVP, Fault Analyzer is expected to be invoked through the XDUREQ exit.

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 entry point IDIXFA at
offset X'DB2'.

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.
```

For this IVP, Fault Analyzer is expected to be invoked through the XPCABND exit.

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 entry point
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.
```

For this IVP, Fault Analyzer is expected to be invoked through the CICS LE exit, if installed. Otherwise, it is expected to be invoked through the XPCABND exit.

Verifying the use of Fault Analyzer with DB2

This section is applicable only 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 on page 435.

- 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 436.

Using a C program

To verify Fault Analyzer with DB2® using a C program, edit and submit the sample job IDIVPDB2 in data set IDI.SIDISAM1. Refer to the instructions in the sample job for more information.

The job executes an already compiled and linked ODBC C program, which has been provided as load module IDIVPDB2 in data set IDI.SIDIAUTH. The program deliberately abends with a system abend code of S0C4.

Note: This IVP is based on the DB2® ODBC IVP that is usually shipped by DB2® in the DSN.SDSNSAMP data set as members DSNTEJ8 (JCL) and DSN8OIVP (C source code). This IVP has been modified to deliberately abend while in the connection with DB2® so that Fault Analyzer is invoked and includes a report section for DB2® information. The Fault Analyzer version of the source code is provided for your reference at the end of the IDIVPDB2 sample member.

As a result of the TER(UATRACE) LE option, Fault Analyzer is invoked via the MVS™ change options/suppress dump exit, IDIXDCAP.

The synopsis section of the Fault Analyzer report that is written to IDIREPRT should contain the following:

For the Japanese feature of Fault Analyzer, this section should be in Japanese if the Language(JPN) option is in effect.

```
A system abend 0C4 reason code X'4' occurred in module IDIVPDB2 program IDIVCDB2
entry point main at offset X'590'.

A program-interruption code 0004 (Protection Exception) is associated with this
abend and indicates that:

    An attempt to store into the range 0-511 or 4096-4607 with low-address
    protection enabled.

The abend was caused by machine instruction 50000000 (STORE).

NOTE: Source code information could not be presented because the source line for
      program IDIVCDB2 is not available (it could be in an include file and
      NOSHOWINC is in effect). The source line # from the GONUMBER option is 126
      for offset X'DA8'.
```

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 more information.

The job compiles and executes a COBOL program, which abends with a system abend code of S0C9.

Note: This IVP is based on the DB2® COBOL IVP that is usually shipped by DB2® in the DSN.SDSNSAMP data set as members DSNTEJ2C (JCL) and DSN8BC3 (COBOL source code). This IVP has been modified to deliberately abend after having performed DB2® data base access, so that Fault Analyzer is invoked and includes a report section for DB2® information.

Before running this IVP, ensure that the DB2® sample data base environment is set up correctly. Follow the instructions in the *DB2® for z/OS® Installation and Migration Guide* for running the DSNTEJ2C DB2® IVP. Once DSNTEJ2C is running correctly, then either make the changes listed in the Fault Analyzer IDIVPDBB sample to the DB2® DSNTEJ2C sample, or make the same changes that were made to the DB2® DSNTEJ2C sample in the Fault Analyzer IDIVPDBB sample.

As a result of the TER(TRACE) LE option that is specified on the DB2® RUN command for the IDISDB2B program, Fault Analyzer is invoked via the LE CEEEXTAN exit, IDIXCEE.

The synopsis section of the Fault Analyzer report that is written to IDIREPRT should contain the following:

For the Japanese feature of Fault Analyzer, this section should be in Japanese if the Language(JPN) option is in effect.

Note: Due to differences in version, release or maintenance level of the compiler used, program offset information might differ from the sample below.

```
A system abend 0C9 occurred in module IDISDB2B program IDISDB2B at offset
X'1EE2'.
```

```
A program-interruption code 0009 (Fixed-Point-Divide Exception) is associated
with this abend and indicates that:

  The divisor was zero in a signed binary division.

The cause of the failure was program IDISDB2B in module IDISDB2B.  The COBOL
source code that immediately preceded the failure was:

  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. . . . . . . . . : NWILKES
Database Request Module Name: DB2V810.DB42.DBRMLIB.DATA(IDISDB2B)
Consistency Token . . . . . : X'17E9C40018AE6A18'
Primary Authorization ID. . : NWILKES
Current SQL ID. . . . . . . : NWILKES

                            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.EMPLOYEENUMBER 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 Modifying your ISPF environment on page 327. This invocation should present the Fault Entry List display, containing entries for the abends that occurred as a result of submitting the verification programs.

Initially, the default history file IDI.HIST, or the history file that is specified in the IDICNFxx parmlib member DataSets option, is used for the display.

To inspect the Fault Analyzer report generated at the time of abend, enter the command 'V' next to the entry you want to view, and press Enter.

For more information about how to use the Fault Analyzer ISPF interface, refer to The Fault Analyzer ISPF interface on page 55.

Verifying the recovery fault recording setup

To verify the setup of recovery fault recording, insert a JCL statement like the following into the abending step of any of the other IVP jobs, for example, the IDIVPCOB IVP job (see Verifying the use of Fault Analyzer with COBOL on page 429):

```
//GO.IDIRFRON DD DUMMY
```

When the IDIRFRON ddname is allocated to the abending step being analyzed by Fault Analyzer, then a deliberate abend U0777 is issued. This abend activates the recovery fault recording feature to write a TDUMP or SDUMP depending on the setup described in Recovery fault recording on page 51, and a recovery fault recording fault entry.

Before submitting the job, ensure that the IDIS subsystem is started.

Messages similar to the following should be displayed:

```
+IDI0001I Fault Analyzer V17R1M0 (HADQH10 2024/12/27)invoked by IDIXCEE using SYS1.PARMLIB.F1.USER(IDICNF00)
+IDI0157I Fault Analyzer about to deliberately abend U0777 and take RFR dump due to IDIRFRON DDname
+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.TDUMP.F1.D181215.T013821.S00072)
        STORCLAS (SCIDIRFR) MGMTCLAS (PRIMARY) DATACLAS (DEFAULT)
        VOL SER NOS= E$RF01
IGD104I IDIRFRHQ.TDUMP.F1.D181215.T013821.S00072 RETAINED,  DDNAME=SYS00038
IEA822I COMPLETE TRANSACTION DUMP WRITTEN TO IDIRFRHQ.TDUMP.F1.D181215.T013821.S00072
```

Reanalysis of the recovery fault recording fault entry identified in message IDI0126I on page 723 should result in a report, which is almost identical to the one from the same IVP run without the IDIRFRON DD statement.

Verifying dump registration (IEAVTSEL)

After you install the MVS post-dump exit IDIXTSEL (see Installing the MVS post-dump exit IDIXTSEL on page 412), specify user exits with the DumpRegistrationExits option (see DumpRegistrationExits on page 593), and start the IDIS subsystem (see Starting the IDIS subsystem on page 320), you can test the dump registration by:

- Using a SLIP dump on page 440
- Using a console dump on page 440

If the DumpRegistrationExits option specifies an Analysis Control or Notification dump registration user exit, it might be helpful to add WTO messages (or similar, depending on programming language used) to load module user exits, or IDIWTO commands to REXX user exits. See IDIWTO command on page 535.

For example, if you use a common normal Analysis Control user exit and dump registration Analysis Control user exit, see Determining which exit type is invoking a common user exit on page 469 for information about how to distinguish between the two exit type invocations.

Using a SLIP dump

1. Ensure the IDIS subsystem has been started with the SLIP parameter.
2. Set a SLIP trap using the MVS operator command:

   ```
   SL SET,ID=xxxx,COMP=U0777,A=SVCD,END
   ```

   where *xxxx* is an identifier of your choice.
3. Add the following JCL statement to any abending job, for example the Fault Analyzer assembler IVP job, IDIVPASM:

   ```
   //G.IDIRFRON DD DUMMY
   ```

4. Submit the job.

Using a console dump

1. Issue the MVS operator command

   ```
   DUMP TITLE='xxx'
   ```

   where *xxx* is a title of your choice.
2. When MVS issues the WTOR `<n> IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND`, reply with:

   ```
   R <n>,ASID=1,END
   ```

You can substitute any valid ASID number for ASID=1, or change it to (for example) `JOBNAME=name` or `TSONAME=name`. See the MVS system command documentation for details about the DUMP command.

Chapter 27. Managing history files (IDIUTIL utility)

The IDIUTIL batch utility can be used to perform history file management functions, such as listing and deleting history file fault entries.

The ability to delete a set of entries based on their date or some other criteria enables you to keep the number of entries in the history file at a manageable level.

Note: Do not delete members from a history file PDS or PDSE outside of the Fault Analyzer ISPF interface or the IDIUTIL batch utility (for example, directly from an ISPF data set member list). If you do, the history file index is out of synch until the recording of the next fault during real-time analysis or a run of the IDIUTIL batch utility against the history file.

Being able to list the entries in history files can help you to keep track of problems.

The maintenance functions of IDIUTIL are driven by a series of control statements that it reads from the SYSIN DD JCL data set. The control statements start in column 1 of the SYSIN records with any continuation statements having a blank in column 1. Comments can be placed in this control statement stream with an asterisk in column 1 of the comment line.

The SYSIN stream is processed sequentially, one control statement at a time. The target history files for control statements can be implicit or explicit depending on the control statement. The control statements that set up target history file data set names overwrite the target history file names previously in effect. The FILES control statement's only purpose is to set the target history file data set names for following control statements. This purpose makes sense when you see that the LISTHF and DELETE control statement syntax does not include a target history file keyword. They operate on the current target history file set, which can be one or more data set names.

Other control statements such as IMPORT and SETFAULTPREFIX carry a single history file in their syntax which resets the current history file set to that data set name, before carrying out its action.

As an alternative to the SYSIN stream, control statements for the IDIUTIL batch utility can be passed via the EXEC JCL statement PARM field. Control statements passed in this way must not include any imbedded blank spaces, but must be separated from other control statements by one or more blanks.

The IDIUTIL batch utility provides basic capabilities including selection of fault entries to list or delete based on criteria such as date and job name. Three user exit points permit more advanced selection and recording to be coded by the user. These exits are for the DELETE, LISTHF, and IMPORT control statements. They follow the same structure as the user exits provided with the Fault Analyzer real time and reanalysis functions. They can be written in REXX, Assembler, or a high-level language.

IDIUTIL control statements

The control statements are presented here in the order that helps explain their use, but the only order requirement for execution is that the LISTHF and DELETE control statements need to be preceded by one of the other control statements that populate the history file data set name set. The FILES control statement is the only one that can put more than one data set name into the set of history file data set names. All of the other control statements (apart from LISTHF, DELETE, and Exits) reset the current data set names to a single target data set name.

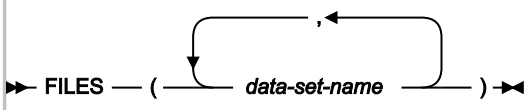
Values containing blanks, quotes (' or "), greater than signs (>), less than signs (<), equal signs (=), ampersands (&) or vertical bars (|) must be enclosed in single or double quotes. Any quote characters within a quoted value that are of the same type as those surrounding the value must be doubled up.

In the following syntax diagrams, either commas or blank characters are permitted as delimiters when repeating suboptions or values.

FILES control statement

The FILES control statement specifies a list of PDS or PDSE history file data set names.

Figure 218. Syntax



Description

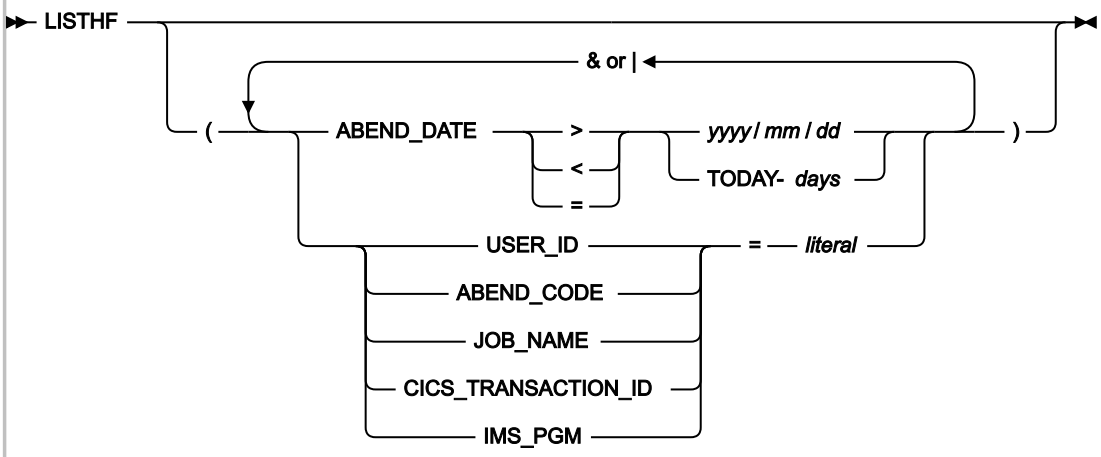
The LISTHF, LISTHFDUP, and DELETE control statements immediately following the FILES control statement operate on all of the history files in the FILES statement.

Examples showing the use of the FILES control statement are provided in Examples on page 450.

LISTHF control statement

The LISTHF (LIST History File) control statement specifies a set of optional qualifiers to select which fault entries should be listed.

Figure 219. Syntax



Description

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, CICS_TRANSACTION_ID, or IMS_PGM (all of which are field names in the ENV data area) to a

literal in the LISTHF control statement. Comparisons can be combined with and, or (& |) operators. The result of this simple syntax capability can be passed on to a user exit if more complex comparisons are desired.

Two special literal comparison qualifiers are recognized. An asterisk in the literal truncates the comparison for wild card capabilities, such as:

```
JOB_NAME = AB*
```

The other special literal is TODAY-*days*, which is converted to today's date, minus the number of numeric days specified at *days*, and then converted to a string of the 2001/02/23 format before comparison. Naturally, the TODAY-*days* literal is only meaningful when used with ABEND_DATE, such as:

```
ABEND_DATE < TODAY-30
```

The value specified for *days* must be in the range 0 - 2147483647.

When comparing ABEND_CODE, the format is four numeric digits for user abend codes and three hex digits preceded by S for system abends. For example, S0C4 for a system 0C4 and 4038 for a user 4038 abend. CICS® abend codes are four alphabetic characters, for example, ASRA.

The IDIUTIL ListHF user exit (see IDIUTIL ListHF user exit on page 511) can be used with the LISTHF control statement to apply extra selection criteria to the history file entries that should be listed.

An example showing the use of the LISTHF control statement is provided in Example 1. Listing history file entries on page 451.

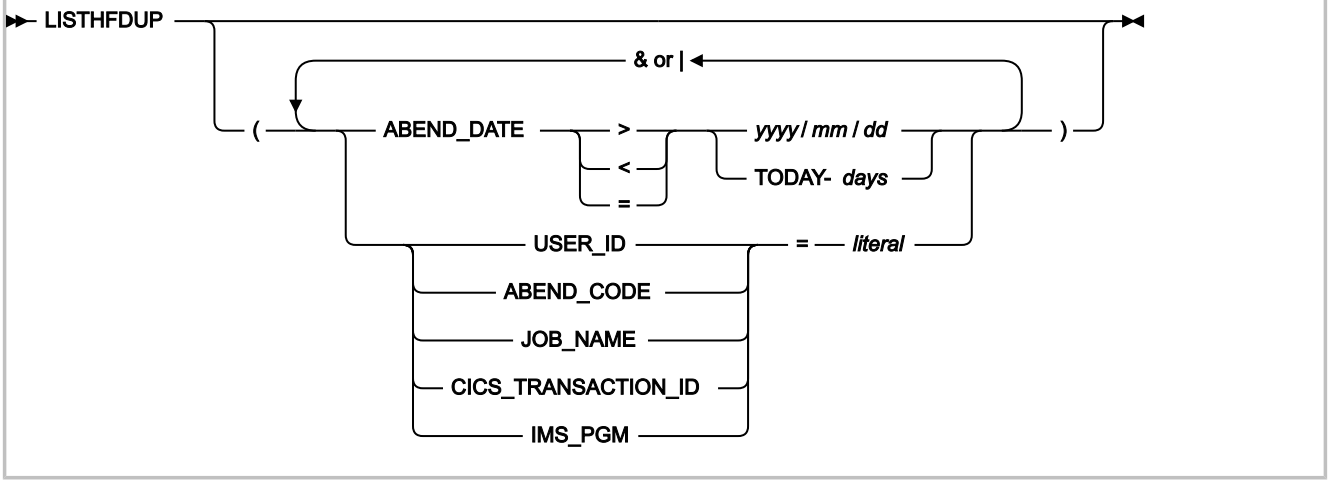
LISTHFDUP control statement

The LISTHFDUP (list history file with duplicates) can list all duplicate instances of a fault.

The LISTHFDUP control statement differs from the LISTHF control statement by being able to list all duplicate instances of a fault, rather than just fault entries. For example, if a given fault entry has had 10 duplicates recorded against it, LISTHFDUP will by default list all 11 instances of the fault separately with individual timestamps, whereas LISTHF will list only the original fault entry with a total duplicate count.

The LISTHFDUP control statement specifies a set of optional qualifiers to select which fault instances should be listed.

Figure 220. Syntax



Description

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, CICS_TRANSACTION_ID, or IMS_PGM (all of which are field names in the ENV data area) to a literal in the LISTHFDUP control statement. Comparisons can be combined with and, or (& |) operators. The result of this simple syntax capability can be passed on to a user exit if more complex comparisons are desired.

Two special literal comparison qualifiers are recognized. An asterisk in the literal truncates the comparison for wild card capabilities, such as:

```
JOB_NAME = AB*
```

The other special literal is TODAY-*days*, which is converted to today's date, minus the number of numeric days specified at *days*, and then converted to a string of the 2001/02/23 format before comparison. Naturally, the TODAY-*days* literal is only meaningful when used with ABEND_DATE, such as:

```
ABEND_DATE < TODAY-30
```

The value specified for *days* must be in the range 0 - 2147483647.

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.

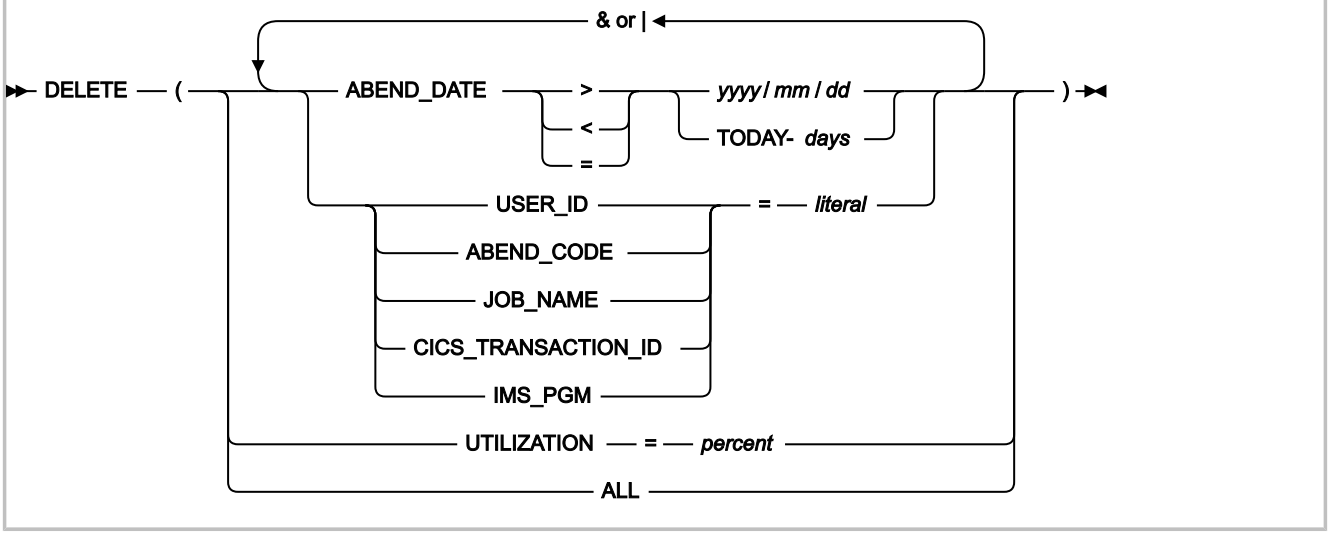
The IDIUTIL ListHFDUP user exit (see IDIUTIL ListHFDUP user exit on page 515) can be used with the LISTHFDUP control statement to apply extra selection criteria to the abend instances that should be listed.

An example showing the use of the LIsSTHFDUP control statement is provided in Example 2. Listing history file abend instances on page 451.

DELETE control statement

The DELETE control statement specifies qualifiers to select which fault entries should be deleted.

Figure 221. Syntax



Description: DELETE(ALL)

The DELETE(ALL) control statement is used to delete all fault entries in the history files that were specified unconditionally in a prior FILES control statement. The fault entries are deleted regardless of any lock flag setting and no user exit is called.

Run IDIUTIL with DELETE(ALL) against a history file before the history file itself is deleted to ensure that all associated dump data sets (such as RFR or XDUMP) are also deleted. If this is not done, orphaned dump data sets might be left allocated if the history file is deleted.

Important: When using this control statement, the user ID under which IDIUTIL is running must have ALTER data set security access to the history files specified. XFACILIT access is not sufficient.

Description: Other than DELETE(ALL)

With the exception of DELETE(ALL), the IDIUTIL DELETE function should only be used for PDS history files. PDSE history files (which we recommend are used) should use automatic space management. For information about automatic space management, see AUTO-managed PDSE history files on page 345.

Note: Deletion of a locked fault entry is only possible when overriding the default action using the IDIUTIL Delete user exit. For more information about IDIUTIL Delete user exit specification and usage, see EXITS control statement on page 450. 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 129.

The UTILIZATION qualifier can be used to delete entries, starting with the oldest entry, until the specified percentage of utilization is reached. This qualifier is only available for PDSE history files. UTILIZATION cannot be qualified with extra & or | operators, it must be the only operator in the DELETE statement when used. It can be preceded or followed by any other DELETE statements.

Note: When using the UTILIZATION qualifier, IDIUTIL attempts to delete usage back to the specified percentage full at the time of running IDIUTIL. It is *not* a way of telling Fault Analyzer to maintain the usage at the specified percentage as new fault entries are created. PDSE history files are by default AUTO-managed and use the existing data set extents as much as possible, without allocating more extents.

The remaining qualifiers follow the same rules as the LISTHF qualifiers above (see LISTHF control statement on page 442).

The IDIUTIL Delete user exit can be used with the DELETE control statement to further select the history file entries that should be deleted. See IDIUTIL Delete user exit on page 510.

Examples of using the DELETE control statement are provided in Example 3. Deleting history file entries by date on page 451 and Example 4. Deleting history file entries by utilization on page 451.

SETFAULTPREFIX control statement

The SETFAULTPREFIX control statement specifies the PDS or PDSE history file data set name followed by the new prefix.

Figure 222. Syntax

Description

data-set-name is the history file whose fault entry prefix characters are changed. *prefix* can be up to three characters. Only alphabetic prefix characters are permitted.

Prefixes for all existing fault IDs in the specified history file are changed to the specified prefix, and all fault IDs later created in this history file receive this prefix automatically.

Note: By default, this function is available to all users with UPDATE access to the history file. However, its usage can be restricted as explained in Restricting change of history file settings on page 307.

An example showing the use of the SETFAULTPREFIX control statement is provided in Example 5. Changing history file fault prefix characters on page 452.

SETMAXFAULTENTRIES control statement

The SetMaxFaultEntries control statement specifies a PDS history file data set name and the maximum number of fault entries to be set.

Figure 223. Syntax

Description

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

max-number must be 1 - 2147483647.

Note: If the maximum number of fault entries specified exceeds the current number of fault entries in the history file, then the oldest fault entries that are not locked are deleted until the history file only contains the number of fault entries specified.

This option is available for PDS history files only; use SetMinFaultEntries for PDSE history files.

Note: By default, this function is available to all users with UPDATE access to the history file. However, its usage can be restricted as explained in Restricting change of history file settings on page 307.

SETMINFAULTENTRIES control statement

The SetMinFaultEntries control statement specifies a PDSE history file data set name and the minimum number of fault entries to be set.

Figure 224. Syntax

Description

The history file is maintained automatically once a minimum of *min-number* fault entries exist in the history file, regardless of how many data set extents have been allocated to achieve this. Once the history file is maintained automatically, then the number of fault entries is limited only by the currently available data set space. No extra data set extents are generally allocated, and no out-of-space conditions are expected.

This option is available for PDSE history files only; use SetMaxFaultEntries for PDS history files.

See AUTO-managed PDSE history files on page 345 for more information about AUTO-managed PDSE history files.

min-number must be 25 - 2147483647.

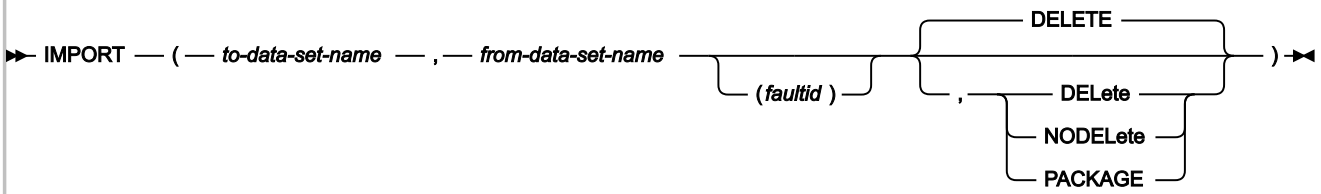
Note: By default, this function is available to all users with UPDATE access to the history file. However, its usage can be restricted as explained in Restricting change of history file settings on page 307.

An example showing the use of the SetMinFaultEntries control statement is provided in Example 6. Creating a self-maintaining history file on page 452.

IMPORT control statement

The IMPORT control statement specifies the "to" PDS or PDSE history file data set name followed by the "from" data set name.

Figure 225. Syntax



Description

The fault entries in the "from" data set are copied and their prefix characters set to the prefix of the "to" history file. When possible, the imported fault number is left the same, however, the fault number is altered where necessary to ensure no existing "to" history file fault entries are overwritten. If the DELETE option is in effect, the fault member is deleted in the "from" data set after a successful copy. DELETE is the default setting.

Any tightly coupled associated dump data sets are automatically attempted to be copied and linked with the new fault entries. This is necessary because Fault Analyzer maintains a unique link between a fault entry and its tightly coupled associated dump data set, both for access control and to enable the automatic deletion of the dump data set when the fault entry is deleted. For additional information about associated dump data sets, see Associated dump data sets on page 25.

The name of the copied dump data set is determined as follows:

- Depending on the dump data set type, the appropriate IDIOPTLM configuration-options load module option specification of RFRDSN, XDUMPDSN, or SDUMPDSN. This is the default. For details, see Customize Fault Analyzer by using an IDIOPTLM configuration-options module on page 337.
- Using an IDIUTIL Import user exit. For details, see IDIUTIL Import user exit on page 507.

If a single fault ID is included with the "from" data set specification, for example TEMP.HIST(F00234), then only that fault ID is imported and deleted.

This function allows fault members that have been sent using the TSO XMIT command from one system to be received into a staging data set and then imported to the required target history file. For an example of this, refer to Managing history files across MVS systems without shared DASD on page 350. The IMPORT user exit could be used with this function to notify users that the imported entries have arrived.

The PACKAGE option should only be used in these situations:

- By the IDIROBOT exec (see IDIROBOT: sample REXX exec to receive fault entries on page 354).
- When importing a fault entry package created using the P line command from the Fault Entry List display (see Packaging fault entries on page 141).
- When importing a fault entry package created using the IDIXMIT formatting user exit (see On-demand implementation on page 359).

Regardless of which method was used to create the fault entry package, the IMPORT input data set must be untersed before running IDIUTIL.

When the PACKAGE option is used, Fault Analyzer performs additional checks on the IMPORT input data set and parameters to ensure that *from-data-set-name* is either sequential or a PDS(E). If a PDS(E), then it is additionally checked that:

- *faultid* has been specified or the data set contains only one fault entry.
- The data set does not contain a $$INDEX member and is not managed by the IDIS subsystem.

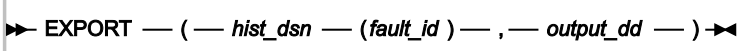
The IDIUTIL Import user exit (see IDIUTIL Import user exit on page 507) can be used with the IMPORT control statement to further select the history file entries that should be imported.

An example showing the use of the IMPORT control statement is provided in Example 7. Importing history file entries on page 452.

EXPORT control statement

The EXPORT control statement can be used to create a portable copy of a fault entry, which includes both the fault entry itself and any associated dump data set.

Figure 226. Syntax



Description

The EXPORT control statement is primarily intended for use by the IDIROBOT exec and the job generated by the Fault Analyzer ISPF interface P line command.

The target fault entry is specified using *hist_dsn* and *fault_id*. The portable copy of the fault entry is written to the DDname specified using *output_dd*. The data set allocated to *output_dd* is expected to be a temporary sequential, PDS or PDSE data set with RECFM VB and LRECL 10000 enforced. The data set should be empty as any existing data might be destroyed.

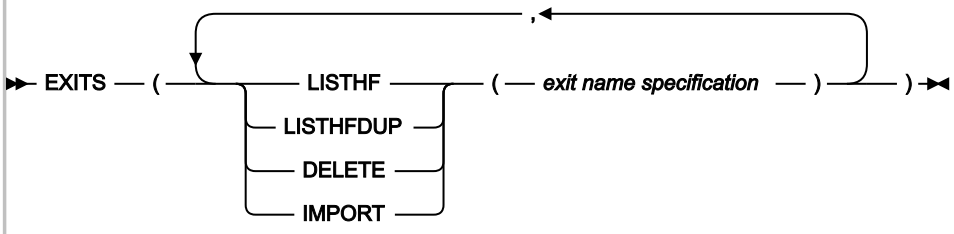
Although READ access to the fault entry can be provided through normal data set access or Fault Analyzer XFACILIT access, READ access to an associated tightly coupled dump data set must be provided through normal data set access.

The exported fault entry should be imported again into another history file using the IMPORT control statement.

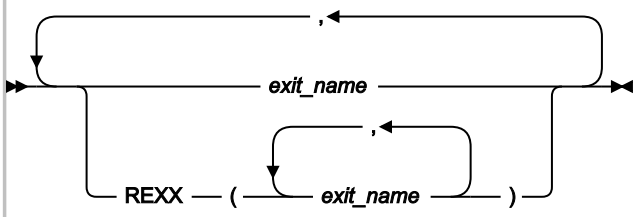
EXITS control statement

The EXITS control statement follows the same format as the Exits keyword for real-time analysis and reanalysis. The difference is that the exit points are for LISTHF, LISTHFDUP, DELETE, and IMPORT.

Figure 227. Syntax



exit name specification



Description

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 the case of LISTHFDUP, the exit is invoked for each instance of an abend (initial abend, normal duplicate, or a seperate CICSFast or ImageFast duplicate), or group of abends (CICSFast or ImageFast). The type of duplicate is provided in UTL.DUP_TYPE. 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, LISTHFDUP, 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, LISTHFDUP, DELETE, and IMPORT exit points are not recognized in or read from the configuration files used by Fault Analyzer real time analysis and reanalysis.

Deprecated options ACCOUNTING and NOACCOUNTING can still be specified, but are ignored.

A detailed description of each exit type is provided in Customizing Fault Analyzer by using user exits on page 464.

An example showing the use of the EXITS control statement is provided in Example 7. Importing history file entries on page 452.

Examples

The following are examples showing the use of the IDIUTIL batch utility.

Example 1. Listing history file entries

This example shows an IDIUTIL batch utility job to list all history file entries that are contained in the history file MY.HIST1 and all entries in history file MY.HIST2 that are for job names starting with TEMP, contain an abend code of S0C1, and are for user ID P0001.

```
//UTILJOB1 JOB  ...
//RUNUTIL  EXEC PGM=IDIUTIL
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
* In the first history file, list all entries
FILES(MY.HIST1)
LISTHF
* In the second history file, only list those entries that match a specific criteria
FILES(MY.HIST2)
LISTHF(USER_ID=P0001 & ABEND_CODE=S0C1 &
 JOB_NAME=TEMP*)
/*
```

Example 2. Listing history file abend instances

This example shows an IDIUTIL batch utility job to list all abend instances recorded in history file MY.HIST1 that occurred yesterday, whether duplicates or not.

```
//UTILJOB1 JOB  ...
//RUNUTIL  EXEC PGM=IDIUTIL
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
FILES(MY.HIST1)
LISTHFDUP(ABEND_DATE = TODAY-1)
/*
```

Example 3. 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 4. 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 5. 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 6. Creating a self-maintaining history file

This example shows a job to allocate a PDSE history file that is named MY.HIST, and using the IDIUTIL batch utility, set the fault wrap number for to 100 while permitting the future reuse of existing fault numbers.

```
//UTILJOB4 JOB  ...
//IDCAMS   EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN DD *
  ALLOC DSNAME('MY.HIST')          -
        NEW                        -
        SPACE(10 10)               -
        CYLINDERS                  -
        RECFM(V B)                 -
        LRECL(10000)               -
        DIR(10)                    -
        DSNTYPE(LIBRARY)
/*
//RUNUTIL  EXEC PGM=IDIUTIL
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
SetMinFaultEntries(MY.HIST,100)
/*
```

This code creates a history file with maximum automatic free space reclamation that should never run out of space provided that the data set allocation is sufficient to hold the minimum number of faults as per the SetMinFaultEntries specification.

You might also want to add a SETFAULTPREFIX control statement to your history file creation jobs (see SETFAULTPREFIX control statement on page 446) to make fault IDs in each new history file unique.

Example 7. 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 NWILKES  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 NWILKES  MVS2     U4038 2001/10/14 10:41:29
  F00093 BETTY    CICSUSER CICS04   ASRA  2001/10/12 21:16:37
  F00092 DACBB045 NWILKES  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   NWILKES  MVS1     S0C9  2001/09/12 12:39:27
  F00059 IMSLE3   NWILKES  MVS2     U4036 2001/09/12 12:38:31
```

then running the JCL:

```
//UTILJOB5 JOB  ...
//RUNUTIL  EXEC PGM=IDIUTIL
//IDIEXEC  DD  DISP=SHR,DSN=MY.REXX.EXECS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
Exits(IMPORT(REXX(IMPXMP)))
IMPORT(MY.HIST,MY.TEMP.HIST)
/*
```

with the IDIUTIL Import user exit in member IMPXMP of data set MY.REXX.Exits:

```
/* REXX */
If ENV.SYSTEM_NAME ¬= 'CICS04' then UTL.PERFORM_ACTION = 'N'
```

results in these fault entries in MY.TEMP.HIST:

```
Fault ID Job/Tran User     System   Abend Date       Time
  F00101 IDIVPCOB NWILKES  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 NWILKES  MVS2     U4038 2001/10/14 10:41:29
  F00092 DACBB045 NWILKES  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   NWILKES  MVS1     S0C9  2001/09/12 12:39:27
  F00059 IMSLE3   NWILKES  MVS2     U4036 2001/09/12 12:38:31
```

Note that the entries that were imported into MY.HIST have been deleted from MY.TEMP.HIST.

An extra example showing the use of the IDIUTIL batch utility import function is shown in Managing history files across MVS systems without shared DASD on page 350.

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 507.

In particular, the sample shown in Example 2: Creating a custom report and CSV file of fault entries with the IDIUTIL ListHF user exit on page 513 illustrates how a customized report might be written, and a comma-delimited file generated, which can be used as input to a spreadsheet application.

Chapter 28. Providing application-specific explanations and descriptions

The member IDIHUSRM of the IDI.SIDIDOC1 data set holds user-defined message and abend code explanations, as well as program name descriptions.

The structure of IDIHUSRM is straight-forward. Each item (message explanation, abend code explanation or program description) is identified by a header record, followed by the text to be displayed for that item. The text can be in more than one record. The item is finished by a new header record or the end of the file. Each record is a string. Use a text editor to maintain the items.

Note: If the Fault Analyzer IDIS subsystem is used, then it is necessary to stop and restart the IDIS subsystem in order to include new or changed explanations or descriptions in the in-storage cache. If the restart is not done, then new or changed explanations or descriptions are not found.

Fault Analyzer treats a line in IDIHUSRM that begins with `.*` as a comment and disregards it. It is not included in the analysis report.

User-defined message explanations

If one of your applications writes a message to the SYSLOG, and the application subsequently abends, then Fault Analyzer looks through IDIHUSRM, to see if it can find a message associated with the identifier of the issued message. If it finds a match, then Fault Analyzer is able to include the message explanation in the analysis report.

The message explanation header record in IDIHUSRM has the following format:

Figure 228. Syntax

where

message_identifier

A code identifying the message. This code can be any combination of numbers and letters. It is not case-sensitive. For example:

```
MYMESSAGE
m103a
dbg303
```

are all valid message identifiers.

For the purpose of providing message explanations, Fault Analyzer treats all characters from the start of any message, up until the first blank character, as constituting the message identifier being searched for.

Any plus signs (+) that MVS™ might add to the start of messages when these are being displayed are not considered part of the message identifier. If these are included in the IDIHUSRM member, then they are ignored.

Note that *":msg."* must begin in column 1.

The lines of the message following the header record are transferred directly as you type them in. So if you include leading spaces in the explanation, then Fault Analyzer includes the leading spaces when it prints the message.

In the following example, two messages are defined:

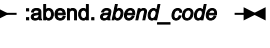
```
:msg.payrollmsg1
Processing of accumulated leave about to commence.
 - If processing fails after this point,
   check for negative accumulations.
:msg.payrollmsg2
Processing of accumulated leave completed.
```

When using the LOOKUP command, user-defined messages appear in the "Messages" main category, under the subcategory name consisting of the first 3 characters of the message ID.

User-defined abend code explanations

The abend code explanation header record in IDIHUSRM has the following format:

Figure 229. Syntax



where

abend_code

Is either a CICS or a non-CICS abend code specification.

A CICS abend code specification is the characters "CICS" followed by a 4-digit CICS abend code. It is not case sensitive. For example:

```
CICSFLT1
cicsa123
Cics0100
```

are all valid CICS abend code specifications.

A non-CICS user abend code specification is a three-character prefix followed by a four-digit user abend code. It is not case-sensitive. For example:

```
XYZ0001
prd1234
zzz0999
```

are all valid abend code specifications.

The three-character abend code prefix is used to identify the load module name in which the user abend is issued. That is, the first 3 characters of the load module name in which the abend occurred, must match the specified abend code prefix. This match permits the same user abend code to be issued from different load modules, each potentially with its own unique explanation.

Note that *":abend."* must begin in column 1.

In the following example, two non-CICS user abends are defined. The U1888 abend might be issued by a load module whose name begins with the 3 characters MD1, for example MD100P. The U0016 abend might be issued from a load module name whose name begins with the 3 characters EXT, for example EXTMAIN. The IDIHUSRM specification follows:

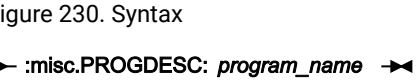
```
:abend.MD11888
An error occurred when attempting to write the invoice
to DDname SYSPRINT.
:abend.EXT0016
Incorrect EXTRACT option specified.
The problem might be one of the following:
  - Invalid range
  - Typo
Respecify and try again.
```

When using the LOOKUP command, user-defined abend code explanations appear in the "Abend Codes" main category, under the subcategory "Other Abend Codes", and further separated into categories according to the three character load module name prefix.

User-defined program descriptions

The program description header record in IDIHUSRM has the following format:

Figure 230. Syntax



where

program_name

The name of a user application entry point, program or load module. It is not case-sensitive. For example:

```
XYZMAIN
prd01
zzzpayrl
```

are all valid specifications.

A matching entry point name is searched for first. If a matching entry point name is not found, then the program name is searched for next. If a matching program name is not found either, then the load module name is searched for.

If a matching entry point, program or load module name is found, then its description is shown in the Fault Analyzer report Event Summary under the "Description" heading, as well as in the detail section for the event.

Note that *":misc."* must begin in column 1.

In the following example, two program descriptions are specified:

```
:misc.PROGDESC:accrevbl
Accounts receivable main module
:misc.PROGDESC:errrtn
Common error routine
```

When using the LOOKUP command, user-defined program descriptions appear in the "Miscellaneous Information" main category, under the subcategory "Program Descriptions".

Chapter 29. Maintaining Fault Analyzer

Fault Analyzer is maintained using the standard IBM® APAR/PTF service.

Note: For information about maintaining ADFzCC, see *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*.

For the latest Fault Analyzer for z/OS service information, refer to https://www.ibm.com/support/pages/latest-fault-analyzer-zos-service-information.

++APAR fixtests need never be restored, since they will either be superseded by a PTF, or a subsequent ++APAR fixtest will specify a later SMP/E REWORK date. However, it might be necessary to perform SMP/E restore and reapply of USERMODs if the maintenance causes a conflict.

Whenever maintenance has been applied to Fault Analyzer by means of SMP/E APPLY, or removed from Fault Analyzer by means of SMP/E RESTORE, perform the following steps:

Step 1: Remove libraries in LINKLIST from LLA and VLF control

If the following libraries are in LINKLIST, remove them from LLA and VLF control before performing the SMP/E APPLY:

- Fault Analyzer SMP/E target libraries
- Target libraries of any USERMODs provided by Fault Analyzer that update libraries in other products

Removing the libraries prevents errors that occur while loading the modules, when SMP/E has compressed or added extents to the libraries.

Step 2: IPL or update dynamically

IPL with CLPA. Alternatively, perform dynamic updates as follows:

1. If the Fault Analyzer module IDIDA has been placed in LPA using the SETPROG command, then do the following:
   a. Issue the command:

      ```
      SETPROG LPA,DELETE,MOD=(IDIDA),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 module IDIDA to LPA again to regain the region size space advantage, issue the command:

   ```
   SETPROG LPA,ADD,MOD=(IDIDA),DSN=LNKLST
   ```

Note: Ensure ECSA size is sufficient to accommodate the additional storage required by the SETPROG LPA,ADD command.

2. If using CICS®, refresh all installed CICS® exits. This refresh can be done by first uninstalling, and then reinstalling the exits, using the CFA transaction as described in Controlling CICS transaction abend analysis on page 400. There is no need to restart your CICS® region.
3. Stop and restart the Fault Analyzer IDIS subsystem as described in Using the Fault Analyzer IDIS subsystem on page 318.
4. Users of the Fault Analyzer ISPF interface should exit and re-enter ISPF for any updates to take effect.
5. Refresh the IEAVTABX_EXIT by issuing the operator command:

   ```
   SET PROG=xx
   ```

   where *xx* matches the suffix of a PARMLIB PROG*xx* member that contains:

   ```
   EXIT REPLACE EXITNAME(IEAVTABX_EXIT) MODNAME(IDIXDCAP) STATE(ACTIVE)
   ```

Step 3: Verify the service level (optional)

Information about Fault Analyzer installation status and applied maintenance is available through:

- The Fault Analyzer ISPF interface **Services->Service Information** pull-down menu.
- The IDICHKI utility, by submitting a job like this:

  ```
  //jobname JOB
  //EXEC PGM=IDICHKI
  //SYSPRINT DD SYSOUT=*
  ```

Chapter 30. Disabling Fault Analyzer

Different options are available for disabling real-time invocation of Fault Analyzer at the system or job level, as explained in the following sections.

Regardless of the method used, SMF type 89 records are also prevented from being written.

Note that Fault Analyzer CICS exits can alternatively be uninstalled from a given CICS region using the CFA transaction. This can be used to uninstall Fault Analyzer from only specific CICS regions, without affecting anything else. For details, see Controlling CICS transaction abend analysis on page 400.

Temporarily uninstalling Fault Analyzer

If for any reason you want to temporarily uninstall Fault Analyzer, then the easiest way is to do the following:

1. Rename the following invocation exit load modules in data set IDI.SIDIAUTH, for example, by changing the first character 'I' to an 'O':
   IDIXCCEE
   IDIXCEE
   IDIXCX53
   IDIXDCAP
   IDIXTSEL
2. Issue this MVS™ operator command:
   ```
   F LLA,REFRESH
   ```
3. Issue this MVS™ operator command:
   ```
   SETPROG EXIT,MODIFY,EXITNAME=IEAVTABX_EXIT,MODNAME=IDIXDCAP,STATE=INACTIVE
   ```
4. If using CICS®, then either bounce all CICS® systems, or use the Fault Analyzer-provided CFA transaction to uninstall all installed CICS® invocation exits.

Having done the above, then the exit processes are not able to find the expected load modules, and processing continues without them.

Note: Remember to rename the load modules back to their original names before applying any maintenance.

Reinstallation: To reinstall Fault Analyzer, do the following:

1. Rename the load modules in step 1 on page 461 of the above procedure back to their original names.
2. Issue this MVS™ operator command:
   ```
   F LLA,REFRESH
   ```
3. Issue this MVS™ operator command:

```
SETPROG EXIT,MODIFY,EXITNAME=IEAVTABX_EXIT,MODNAME=IDIXDCAP,STATE=ACTIVE
```

4. 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, this disablement can be achieved by adding an entry to your IFAPRDxx parmlib member.

If you purchased Fault Analyzer as part of product code 5755-AD1 IBM Application Delivery Foundation for z/OS, modify the entry for product code 5755-AD1 as follows:

```
PRODUCT OWNER('IBM CORP')
        NAME('IBM APP DLIV FND')
        ID(5755-AD1)
        VERSION(*) RELEASE(*) MOD(*)
        FEATURENAME('*')
        STATE(DISABLED)
```

If you purchased Fault Analyzer as product code 5755-AD2, modify the entry for product code 5755-AD2 as follows:

```
PRODUCT OWNER('IBM CORP')
        NAME('FAULT ANALYZER')
        ID(5755-AD2)
        VERSION(*) RELEASE(*) MOD(*)
        FEATURENAME('FAULT ANALYZER')
        STATE(DISABLED)
```

All parameters except VERSION, RELEASE, and MOD must be specified exactly as shown. Any syntactically valid values can be specified for VERSION, RELEASE, and MOD.

Refer to *MVS™ Initialization and Tuning Reference* for general information about the IFAPRDxx parmlib member.

If at a later stage Fault Analyzer needs to be re-enabled, either remove the above entry or change the STATE to ENABLED.

Turning off Fault Analyzer with a JCL switch (IDIOFF)

The Fault Analyzer invocation exits can be turned off at a job-step level by coding the following JCL statement in the job step:

```
//IDIOFF DD DUMMY
```

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 47).

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 298. 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

To provide greater flexibility and control of Fault Analyzer operation, a set of user exit points have been created where user exits can get control during Fault Analyzer operation. The user exits can be written in REXX, assembler, or high-level languages. They are normally passed two data structures. The first is a common environment structure passed to all user exits which provides the general information fields for the fault currently being processed. The second structure is normally fields to the particular exit being called. Some of the fields are used to pass information to the exits and others are for the user exit to pass data or required actions back to Fault Analyzer. For exits written in REXX, the data is passed in stem variables rather than in structures, since this approach is more manageable for REXX.

Note: REXX is the only supported programming language for Formatting user exits.

The exits can be used to perform functions such as dynamically selecting the history file data set or compile listing data sets. They can also be used to notify users by messages or email that a fault has occurred plus many other uses.

Note: If a condition occurs that causes Fault Analyzer Recovery Fault Recording (RFR) to be performed, no further user exits will be called during this fault analysis, except for the Notification user exit (for details, see Recovery fault recording on page 51).

Options settings and selections made in user exits affect the current analysis only.

The following user exits are available to users of Fault Analyzer:

- Analysis Control user exit on page 475
- Analysis Control user exit (MVS SVC Dump registration) on page 479
- Compiler Listing Read user exit on page 480
- Message and Abend Code Explanation user exit on page 485
- Formatting user exit on page 490
- End Processing user exit on page 494
- End Processing user exit (Fault entry refresh) on page 497
- Notification user exit on page 498
- Notification user exit (MVS SVC Dump registration) on page 506
- IDIUTIL Import user exit on page 507 (used only with the IDIUTIL batch utility)
- IDIUTIL Delete user exit on page 510 (used only with the IDIUTIL batch utility)
- IDIUTIL ListHF user exit on page 511 (used only with the IDIUTIL batch utility)

Figure 231: Fault Analyzer analysis exit points (IDIDA) on page 465 illustrates the exit points provided for real-time analysis, batch reanalysis, and interactive reanalysis, while Figure 232: IDIUTIL batch utility exit points on page 466 illustrates the exit points provided for the IDIUTIL batch utility.

Figure 231. Fault Analyzer analysis exit points (IDIDA)

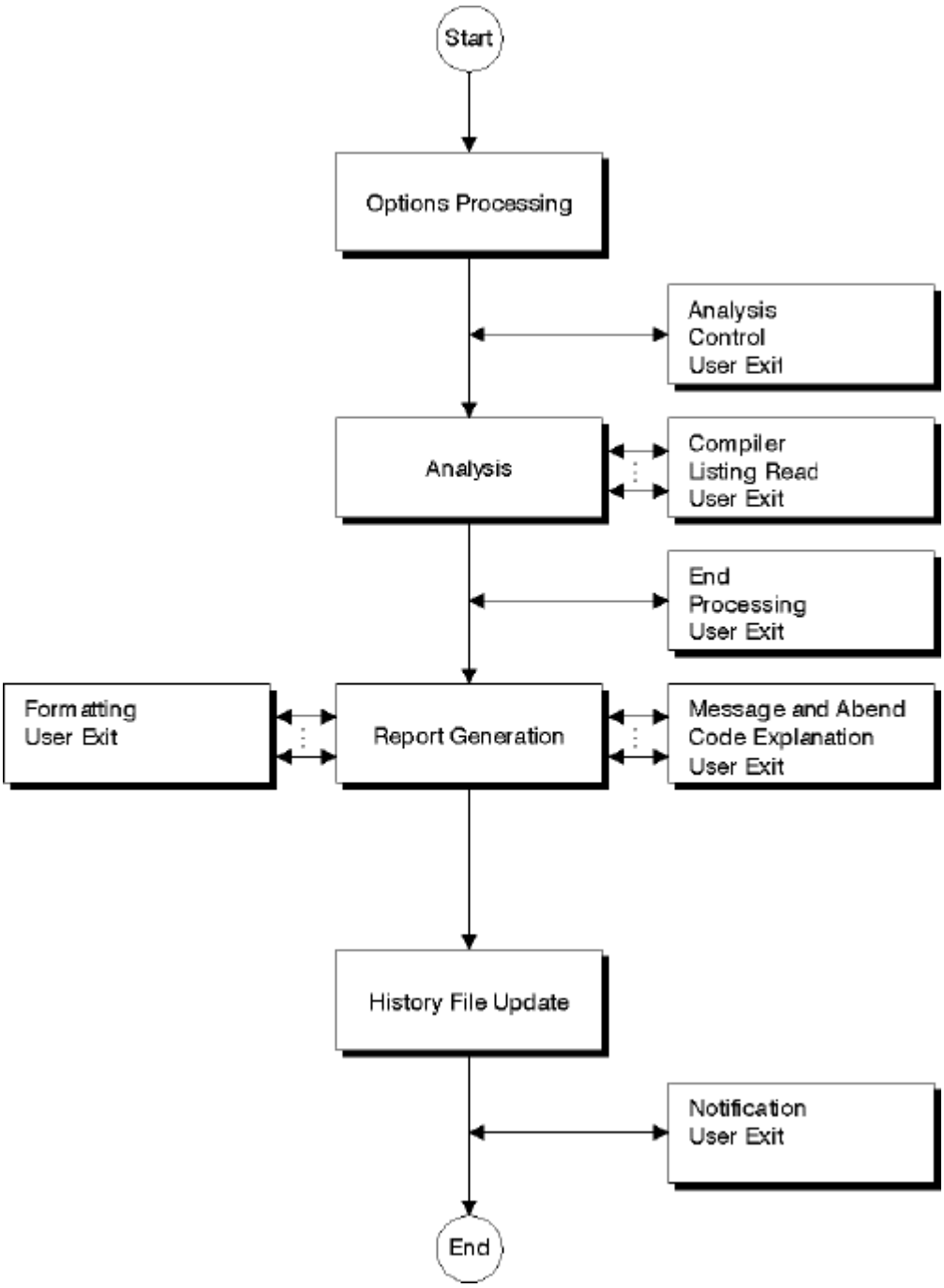
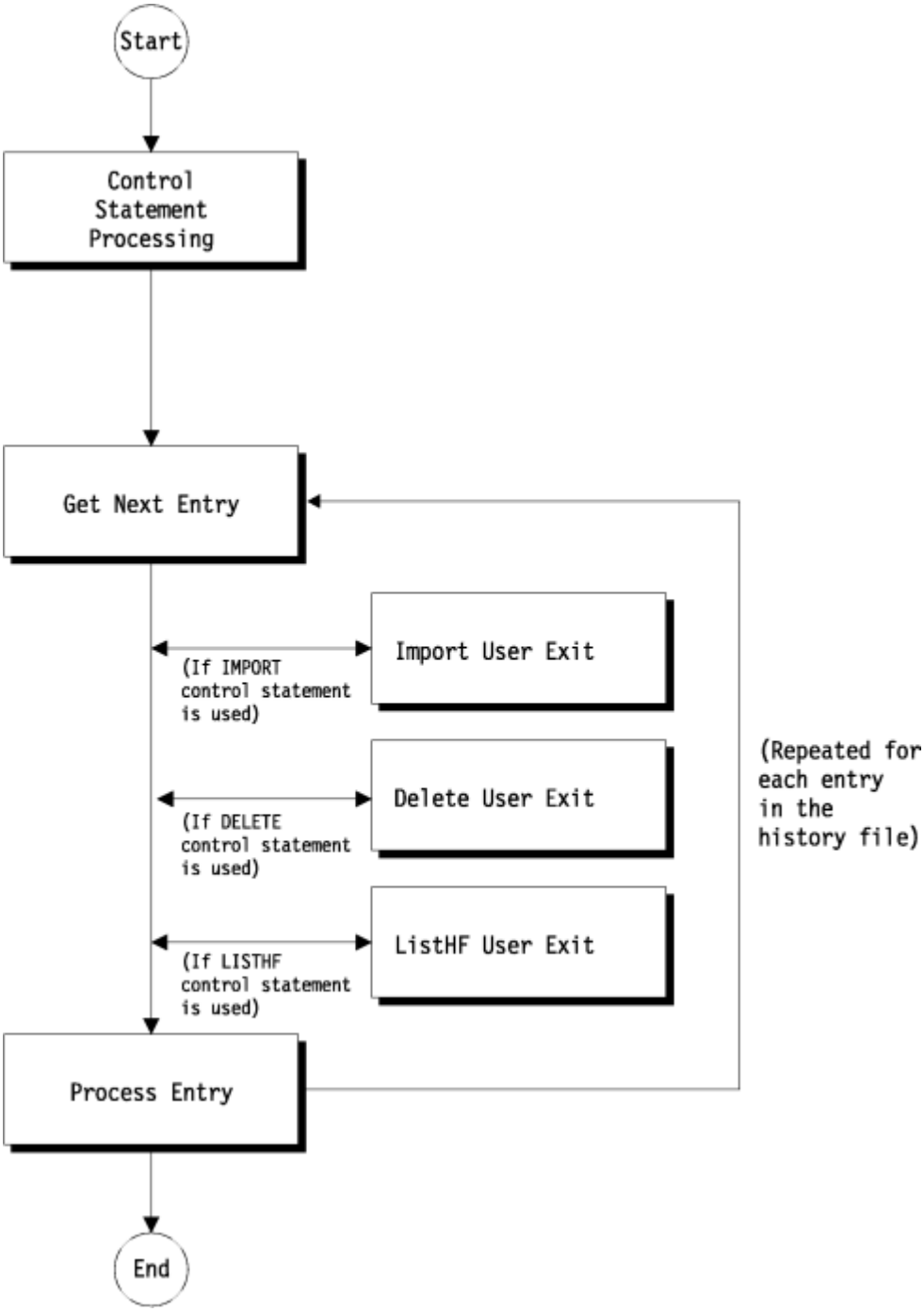


Figure 232. IDIUTIL batch utility exit points



User exits are specified to Fault Analyzer via the Exits option (see Exits on page 600), the DumpRegistrationExits option (see DumpRegistrationExits on page 593), or the RefreshExits option (see RefreshExits on page 630).

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 450.

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 653.

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 474 or Descriptions of IDIUTIL batch utility user exit types on page 507.

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 full words) to be received on entry to the load module:

REXX user exits

REXX user exits must be available via the IDIEXEC DDname. Data sets for this DDname can be provided in the DataSets option (see DataSets on page 581), or be specified through JCL DD statements in your job.

Modifiable parameter list values are always truncated to their defined field width on return from a user exit. Values that are shorter are delimited by adding a null character (X'00').

SYSPRINT type output from REXX EXECs, such as REXX messages and output from SAY or TRACE instructions, is suppressed by Fault Analyzer unless diagnostic tracing (see Diagnostic tracing on page 470 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 535 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".

Note: If a REXX user exit includes procedures with explicitly exposed variables, ensure that any Fault Analyzer parameters used within those procedures are also explicitly exposed.

Load module exits

Load module exits must be available via the standard MVS™ search path and cannot contain an LE main routine (C `main()` function or PL/I `PROC OPTIONS(MAIN)`) as they are executed under an existing LE enclave by the IDIDA subtask.

To assist with the writing of load module exits, Fault Analyzer provides the following parameter list data area mapping members in the samples data set:

Name

Language

IDISXPLA

Assembler

IDISXPLC

C

IDISXPLB

COBOL

IDISXPLP

PL/I

Load module exits can be either reentrant or non-reentrant and should be link-edited RMODE(ANY). They are invoked in AMODE(31).

Determining which exit type is invoking a common user exit

A user exit can be invoked by Fault Analyzer options (Exits or DumpRegistrationExits) or by the IDIUTIL Exits control statement. The Fault Analyzer options and the IDIUTIL control statement determine which user exit is invoked and under what conditions:

- Exits option (Control, Listing, Msgxpl, Format, End, Notify)
- DumpRegistrationExits option (Control, Notify)
- IDIUTIL Exits control statement (ListHF, Delete, Import)

You can use a different REXX user exit or load module user exit for each user exit type, or you can use the same REXX user exit or load module user exit across different user exit types.

If you use the same REXX user exit or load module user exit across different user exit types, the user exit can check which exit type it's being invoked for.

For example, a single REXX user exit is specified as a normal Analysis Control user exit, an End Processing user exit, and a dump registration Analysis Control user exit using the options:

```
Exits(Control(REXX(myexit)),End(REXX(myexit)))
DumpRegistrationExits(Control(REXX(myexit)))
```

In this case, the user exit can include code like the following to check which exit type is calling the user exit:

```
If ENV.EXIT_CALL_TYPE = 'C' then do
  "IDIWTO 'Normal Analysis Control user exit called'"
  /* Perform normal Analysis Control user exit processing... */
else if ENV.EXIT_CALL_TYPE = 'E' then do
  "IDIWTO 'End Processing user exit called'"
  /* Perform End Processing user exit processing... */
else if ENV.EXIT_CALL_TYPE = 'X' then do
  "IDIWTO 'Dump registration Analysis Control user exit called'"
  /* Perform dump registration Analysis Control user exit processing... */
else do
  "IDIWTO 'User exit unexpectedly called'"
end
```

Data area version checking

It is recommended that all user exits include a check for the expected version of any data areas used. This check is to ensure that incorrect processing does not result from changes to data areas that make them incompatible with earlier versions.

All user exit data areas include a VERSION field, whose current value at the time of writing the exit, can be obtained from the data area descriptions in Data areas on page 639.

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 402 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 before any user exit is invoked and again upon return from the exit.

Errors during validation of updated parameter list fields are identified by warning messages in the trace output.

Trace information is provided for all exit types for which exits are specified using the Exits option, and for which the execution mode of Fault Analyzer permits the exit type to be invoked. To facilitate trace information that shows the values of fields prior to exit invocation without first having to write the exit, use the special exit name 'NONE'.

The elapsed time in seconds for each exit invoked is provided.

An example of an exit trace containing a single Analysis Control user exit call to an exit named SAMPCTLX follows:

```
ANALYSIS CONTROL User Exit:
  Parameter values prior to exit invocation:
    ENV.VERSION . . . . . . . . . . . . : 0005
        EXIT_CALL_TYPE. . . . . . . . . : C
        FAULT_ID. . . . . . . . . . . . :
        ABEND_DATE. . . . . . . . . . . : 2001/03/23
        ABEND_TIME. . . . . . . . . . . : 10:02:03
        DUP_DATE. . . . . . . . . . . . :
        DUP_TIME. . . . . . . . . . . . :
        ORIGINAL_DATE . . . . . . . . . :
        ORIGINAL_TIME . . . . . . . . . :
        REALTIME. . . . . . . . . . . . : Y
        SYSTEM_NAME . . . . . . . . . . : MVSA
        JOB_NAME. . . . . . . . . . . . : CI03DA
        EXEC_PGM_NAME . . . . . . . . . : DFHSIP
        USER_ID . . . . . . . . . . . . : CICSUSER
        GROUP_ID. . . . . . . . . . . . : APC
        ABEND_CODE. . . . . . . . . . . : AEIL
        ABEND_REASON_CODE . . . . . . . :
        INVOCATION_ABEND_CODE . . . . . : AEIL
        ABEND_MODULE_NAME . . . . . . . : CICFRED
        CICS_TRANSACTION_ID . . . . . . : FRED
        CICS_TASK_NUMBER. . . . . . . . : 00026
        JOB_TYPE. . . . . . . . . . . . : C
        JOB_CLASS . . . . . . . . . . . : A
        ACCOUNTING_FIELDS . . . . . . . :
        ACCOUNTING_INFO . . . . . . . . :
        USER_1. . . . . . . . . . . . . :
        USER_2. . . . . . . . . . . . . :
        LOCK_FLAG . . . . . . . . . . . :
        LOCK_USERID . . . . . . . . . . :
        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 . . . . . :
      TERMID. . . . . : 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
      RETAINDUMP_OPT. . . . . : AUTO
      SOURCE_OPT. . . . . : Y
      IDIADATA_PRE. . . . . :
      IDIADATA_JOB. . . . . :
      IDIADATA_CFG. . . . . : FRED.SYSADATA
      IDILC_PRE . . . . . :
      IDILC_JOB . . . . . :
      IDILC_CFG . . . . . : FRED.LISTING.C
      IDILCOB_PRE . . . . . :
      IDILCOB_JOB . . . . . :
      IDILCOB_CFG . . . . . : FRED.LISTING.COBOL
      IDILCOBO_PRE. . . . . :
      IDILCOBO_JOB. . . . . :
      IDILCOBO_CFG. . . . . :
      IDILANGX_PRE. . . . . :
      IDILANGX_JOB. . . . . :
      IDILANGX_CFG. . . . . : FRED.WDBLANGX
      IDILPLI_PRE . . . . . :
      IDILPLI_JOB . . . . . : 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. . . . . :
      IDIJAVA_PRE . . . . . :
      IDIJAVA_JOB . . . . . :
      IDIJAVA_CFG . . . . . :
      STEPLIB . . . . . :
Parameter values after return from exit SAMPCTLX (elapsed 0.03 seconds):
```

```
    ENV.VERSION . . . . . . . . . . . . : 0005
        EXIT_CALL_TYPE. . . . . . . . . : C
        FAULT_ID. . . . . . . . . . . . :
        ABEND_DATE. . . . . . . . . . . : 2001/03/23
        ABEND_TIME. . . . . . . . . . . : 10:02:03
        DUP_DATE. . . . . . . . . . . . :
        DUP_TIME. . . . . . . . . . . . :
        ORIGINAL_DATE . . . . . . . . . :
        ORIGINAL_TIME . . . . . . . . . :
        REALTIME. . . . . . . . . . . . : Y
        SYSTEM_NAME . . . . . . . . . . : MVSA
        JOB_NAME. . . . . . . . . . . . : CI03DA
        EXEC_PGM_NAME . . . . . . . . . : DFHSIP
        USER_ID . . . . . . . . . . . . : CICSUSER
        GROUP_ID. . . . . . . . . . . . : APC
        ABEND_CODE. . . . . . . . . . . : AEIL
        ABEND_REASON_CODE . . . . . . . :
        INVOCATION_ABEND_CODE . . . . . : AEIL
        ABEND_MODULE_NAME . . . . . . . : CICFRED
        CICS_TRANSACTION_ID . . . . . . : FRED
        CICS_TASK_NUMBER. . . . . . . . : 00026
        JOB_TYPE. . . . . . . . . . . . : C
        JOB_CLASS . . . . . . . . . . . : A
        ACCOUNTING_FIELDS . . . . . . . :
        ACCOUNTING_INFO . . . . . . . . :
        USER_1. . . . . . . . . . . . . : ABCD
        USER_2. . . . . . . . . . . . . : 0123
        LOCK_FLAG . . . . . . . . . . . :
        LOCK_USERID . . . . . . . . . . :
        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 . . . . . . . . . . . . :
        TERMID. . . . . . . . . . . . . : SAMA
        TCB_ADDRESS . . . . . . . . . . :
        CSA_ADDRESS . . . . . . . . . . :
        TCA_ADDRESS . . . . . . . . . . :
        IDIHIST . . . . . . . . . . . . : USERA.DCAT
        CPU_HSECONDS. . . . . . . . . . :
        CICS_VRM. . . . . . . . . . . . :
        DB2_VRM . . . . . . . . . . . . :
        IMS_VRM . . . . . . . . . . . . :
        ZOS_VRM . . . . . . . . . . . . :
        DUPLICATE_COUNT . . . . . . . . : 00000
        POF_MODULE_NAME . . . . . . . . :
        POF_MODULE_LKED_DATE. . . . . . :
        POF_MODULE_LKED_TIME. . . . . . :
        POF_CSECT_NAME. . . . . . . . . :
        POF_CSECT_OFFSET. . . . . . . . : 0000000000
        POF_LOADED_FROM . . . . . . . . :
        EXEC_LOADED_FROM. . . . . . . . :
        MINIDUMP_PAGES. . . . . . . . . : 0000000000
    CTL.VERSION . . . . . . . . . . . . : 0002
        Exclude . . . . . . . . . . . . :
        DETAIL_OPT. . . . . . . . . . . : S
        DEFERREDREPORT_OPT. . . . . . . : N
        RETAINDUMP_OPT. . . . . . . . . : AUTO
        SOURCE_OPT. . . . . . . . . . . : Y
        IDIADATA_PRE. . . . . . . . . . :
        IDIADATA_JOB. . . . . . . . . . :
```

```
        IDIADATA_CFG. . . . . . . . . . : FRED.SYSADATA
        IDILC_PRE . . . . . . . . . . . :
        IDILC_JOB . . . . . . . . . . . :
        IDILC_CFG . . . . . . . . . . . : FRED.LISTING.C
        IDILCOB_PRE . . . . . . . . . . :
        IDILCOB_JOB . . . . . . . . . . :
        IDILCOB_CFG . . . . . . . . . . : FRED.LISTING.COBOL
        IDILCOBO_PRE. . . . . . . . . . :
        IDILCOBO_JOB. . . . . . . . . . :
        IDILCOBO_CFG. . . . . . . . . . :
        IDILANGX_PRE. . . . . . . . . . :
        IDILANGX_JOB. . . . . . . . . . :
        IDILANGX_CFG. . . . . . . . . . : FRED.WDBLANGX
        IDILPLI_PRE . . . . . . . . . . :
        IDILPLI_JOB . . . . . . . . . . :
        IDILPLI_CFG . . . . . . . . . . : FRED.LISTING.PLI
        IDILPLIE_PRE. . . . . . . . . . :
        IDILPLIE_JOB. . . . . . . . . . : *** Data from 9945 byte buffer at address 17FF08F8 follows:
                                          FRED.IDILPLIE.T001                          FRED.IDILPLIE.T002
                                          FRED.IDILPLIE.T003                          FRED.IDILPLIE.T004
                                          FRED.IDILPLIE.T005                          FRED.IDILPLIE.T006
                                          FRED.IDILPLIE.T007                          FRED.IDILPLIE.T008
                                          FRED.IDILPLIE.T009                          FRED.IDILPLIE.T010
                                          FRED.IDILPLIE.T011                          FRED.IDILPLIE.T012
                                          FRED.IDILPLIE.T013                          FRED.IDILPLIE.T014
                                          FRED.IDILPLIE.T015                          FRED.IDILPLIE.T016
                                          FRED.IDILPLIE.T017                          FRED.IDILPLIE.T018
                                          FRED.IDILPLIE.T019                          FRED.IDILPLIE.T020
                                          FRED.IDILPLIE.T021                          FRED.IDILPLIE.T022
                                          FRED.IDILPLIE.T023                          FRED.IDILPLIE.T024
                                          FRED.IDILPLIE.T025                          FRED.IDILPLIE.T026
                                          FRED.IDILPLIE.T027                          FRED.IDILPLIE.T028
                                          FRED.IDILPLIE.T029                          FRED.IDILPLIE.T030
                                          FRED.IDILPLIE.T031                          FRED.IDILPLIE.T032
                                          FRED.IDILPLIE.T033                          FRED.IDILPLIE.T034
                                          FRED.IDILPLIE.T035                          FRED.IDILPLIE.T036
                                          FRED.IDILPLIE.T037                          FRED.IDILPLIE.T038
                                          FRED.IDILPLIE.T039                          FRED.IDILPLIE.T040
                                          FRED.IDILPLIE.T041                          FRED.IDILPLIE.T042
                                          FRED.IDILPLIE.T043                          FRED.IDILPLIE.T044
                                          FRED.IDILPLIE.T045                          FRED.IDILPLIE.T046
                                          FRED.IDILPLIE.T047                          FRED.IDILPLIE.T048
                                          FRED.IDILPLIE.T049                          FRED.IDILPLIE.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.T112
                                  FRED.IDILPLIE.T113                        FRED.IDILPLIE.T114
                                  FRED.IDILPLIE.T115                        FRED.IDILPLIE.T116
                                  FRED.IDILPLIE.T117                        FRED.IDILPLIE.T118
                                  FRED.IDILPLIE.T119                        FRED.IDILPLIE.T120
                                  FRED.IDILPLIE.T121
      IDILPLIE_CFG. . . . . . . . . . :
      LOCALE. . . . . . . . . . . . . : S370
      FADATE. . . . . . . . . . . . . : N
      IDISYSDB_PRE. . . . . . . . . . :
      IDISYSDB_JOB. . . . . . . . . . :
      IDISYSDB_CFG. . . . . . . . . . :
      IDIJAVA_PRE . . . . . . . . . . :
      IDIJAVA_JOB . . . . . . . . . . : /u/temp/payroll/directory171/DEPT64directory/accountingDIR1/very/long/path/name/
                                          that/wraps/to/the/next/line
                                       '/u/temp/payroll/directory171/DEPT64directory/accountingDIR1''temp'
      IDIJAVA_CFG . . . . . . . . . . :
      STEPLIB . . . . . . . . . . . . :
```

Tracing REXX EXECs

If a user exit is written as a REXX EXEC, information written by the exit using SAY or TRACE instructions is merged with the Fault Analyzer exit parameter list trace records in the IDITRACE data set.

Descriptions of fault analysis user exit types

The following provides descriptions of each user exit type available for use with Fault Analyzer in real time or fault reanalysis mode of execution. Table 13: Fault analysis user exit types applicable to execution modes on page 474 shows which exit types are applicable to each mode of execution.

Table 13. 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

Table 13. Fault analysis user exit types applicable to execution modes

(continued)

User exit type	Real-time		Reanalysis		
	Normal	Dump registration	Batch		Interactive
			Normal	Refresh	
Notification	Yes (1)	Yes (2)	No	No	No

Notes:

(1)

Specified via the Exits option

(2)

Specified via the DumpRegistrationExits option

(3)

Specified via the RefreshExits option

Analysis Control user exit

The following describes the Analysis Control user exit.

Purpose

This exit can be used to examine and override current settings of the following options:

DataSets

Data sets for the following DDnames are provided:

- IDIADATA
- IDIHIST
- IDILANGX
- IDILC
- IDILCOB
- IDILCOBO
- IDILPLI
- IDILPLIE
- IDISYSDB

For the IDIADATA, IDILANGX, IDILC, IDILCOB, IDILCOBO, IDILPLI, IDILPLIE, and IDISYSDB DDnames, current data sets are provided as:

Pre-allocated data sets

These are data sets that were allocated prior to invoking Fault Analyzer, for example, via JCL DD statements. Data sets for each DDname is provided in the CTL.*ddname*_PRE field.

The pre-allocated data set name fields are read-only and any changes are ignored by Fault Analyzer.

Job-allocated data sets

These are DataSets option specifications from the user options file (IDIOPTS). Data sets for each DDname is provided in the CTL.*ddname*_JOB field.

Changes to the job-allocated data set name fields are honored by Fault Analyzer when it allocates this list of data sets.

Configuration data sets

These are DataSets option specifications from the IDICNFxx configuration member. Data sets for each DDname is provided in the CTL.*ddname*_CFG field.

Changes to the configuration data set name fields are honored by Fault Analyzer.

The final concatenation order of data sets is:

1. Pre-allocated data sets
2. Job-allocated data sets
3. Configuration data sets

For the IDIHIST DDname, the current history file is provided in the ENV.IDIHIST data area field. The user exit can choose to change this data set name, in which case the supplied data set name is used as the history file for the current fault.

The same rules for the use of substitution symbols in data set names which apply to the DataSets option (see DataSets option data set name substitution symbols on page 585), also apply to data set names returned by an Analysis Control user exit.

DeferredReport

The status of the DeferredReport option in effect is identified as either 'Y' (DeferredReport is in effect) or 'N' (DeferredReport is not in effect) in the CTL.DEFERREDREPORT_OPT data area field. Valid changes to this field override the current option setting.

Only applicable to real-time processing.

Detail

The abbreviated form of the Detail option in effect is provided in the CTL.DETAIL_OPT data area field. Valid changes to this field override the current option setting.

Not applicable to interactive reanalysis.

Exclude

The last matching Exclude criterion is provided in the CTL.EXCLUDE_CRITERION data area field. If the fault is excluded from analysis based on a matching Exclude criterion, then the CTL.EXCLUDE data area field is initialized to 'Y'. This field can be modified by the exit.

Only applicable to real-time processing.

Include

The last matching Include criterion is provided in the CTL.INCLUDE_CRITERION data area field. A blank Include criterion signifies the implicit product default, which is to include everything.

Only applicable to real-time processing.

Locale

The current locale name is provided in the CTL.LOCALE data area field. Valid changes to this field override the current option setting.

The LOCALE option suboption, FADATE, is provided in the CTL.FADATE data area field. Valid changes to this field override the current option setting.

RetainDump

The value of the RetainDump option in effect is provided in the CTL.RETAINDUMP_OPT data area field (the value provided in this field is the actual suboption of the RetainDump option, that is, "AUTO" or "ALL"). Valid changes to this field override the current option setting.

Note that an End Processing user exit (see End Processing user exit on page 494) 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 override the current option setting.

Only applicable to real-time processing.

In addition, the Analysis Control user exit can perform allocations of other data sets that might not be specified via options or provided in the passed data areas. In real time, one such allocation could be for IDIREPRT, which would allow an installation to control report destination attributes, such as the SYSOUT class. For more information on this type of IDIREPRT allocation, see Combining Fault Analyzer real-time reports on page 36, Controlling the SYSOUT class of real-time reports on page 36, and Suppressing real-time reports on page 37.

You can use the **CTL.IDITRACE data area** field to dynamically start or stop IDITRACE processing, or to direct the trace output to a data set.

When invoked

This exit is invoked after options processing has completed, and before the commencement of fault analysis.

Parameters

How parameters are passed to the exit depends on the exit type, REXX or load module.

Fault Analyzer initializes the parameter lists using current values for the particular fault and processing options in effect before invoking the Analysis Control user exit.

REXX

Two stems are available to the exit:

- ENV.

  Contains defined symbols for all fields in the ENV data area (see ENV - Common exit environment information on page 653).

- CTL.

  Contains defined symbols for all fields in the CTL data area (see CTL - Analysis Control user exit parameter list on page 640).

The defined variable names are identical to the field names. For example, to access the field VERSION in the ENV data area, use the REXX variable ENV.VERSION.

Load module

At entry to this exit, R1 contains the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.

  Address of an ENV data area (see ENV - Common exit environment information on page 653).

- 31-bit CTL address in word 2.

  Address of a CTL data area (see CTL - Analysis Control user exit parameter list on page 640).

  Note: The high-order bit is on to indicate that this parameter is the last parameter passed.

Example

The following is an example of an Analysis Control user exit that is written in REXX.

Figure 233. Sample REXX Analysis Control user exit

```
/* REXX */
/* Check data areas used */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if CTL.VERSION <> 2 then
  say 'Note: CTL data area version change - field usage review required!'
if ENV.REALTIME = 'Y' then do  /* Exclude all MVSA jobs from analysis */
  if ENV.SYSTEM_NAME = 'MVSA' then
    CTL.Exclude = 'Y'
  /* Select a separate history file for DB2, IMS, and other jobs
     based on jobname */
  if SUBSTR(ENV.JOB_NAME,1,3) = 'DB2' then
    ENV.IDIHIST = 'MY.DB2.HIST'
  else if SUBSTR(ENV.JOB_NAME,1,3) = 'IMS' then
    ENV.IDIHIST = 'MY.IMS.HIST'
  else
    ENV.IDIHIST = 'MY.OTHER.HIST'
end
exit 0
```

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

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

Analysis Control user exit (MVS SVC Dump registration)

The following describes the dump registration Analysis Control user exit.

Purpose

This exit can be used to examine and override current settings of the following options relevant to dump registration processing:

DataSets

Only the history file data set name is relevant to this exit. The current history file data set name is provided in the ENV.IDIHIST data area field. The user exit can choose to change this data set name, in which case the supplied data set name is used as the history file for the current fault. If the history file was pre-allocated, it is freed.

Exclude

The last matching Exclude criterion is provided in the CTL.EXCLUDE_CRITERION data area field. If the fault is excluded from analysis based on a matching Exclude criterion, then the CTL.EXCLUDE data area field is initialized to 'Y'. This field can be modified by the exit.

Include

The last matching Include criterion is provided in the CTL.INCLUDE_CRITERION data area field. A blank Include criterion signifies the implicit product default, which is to include everything.

When invoked

This exit is invoked after options processing has completed, and before the writing of the MVS™ SVC dump registration fault entry.

Parameters

See Parameters on page 478.

Example

The following is an example of an Analysis Control dump registration user exit that is written in REXX.

Figure 234. Sample REXX Analysis Control dump registration user exit

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if CTL.VERSION <> 2 then
  say 'Note: CTL data area version change - field usage review required!'
/* Exclude all MVSA jobs from analysis */
if ENV.SYSTEM_NAME = 'MVSA' then
  CTL.Exclude = 'Y'
/* Select a separate history file for DB2, IMS, and other jobs
   based on jobname */
if SUBSTR(ENV.JOB_NAME,1,3) = 'DB2' then
  ENV.IDIHIST = 'MY.DB2.HIST'
else if SUBSTR(ENV.JOB_NAME,1,3) = 'IMS' then
  ENV.IDIHIST = 'MY.IMS.HIST'
else
  ENV.IDIHIST = 'MY.OTHER.HIST'
exit 0
```

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

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

The DumpRegistrationExits option must be specified in the IDICNFxx parmlib member, or via an IDIOPTS DD statement in the IDIS subsystem JCL. The DumpRegistrationExits option is ignored if specified via an IDIOPTS DD statement anywhere else, such as in a CICS® region or batch job.

Compiler Listing Read user exit

The following describes the Compiler Listing Read user exit.

Purpose

This exit can be used to obtain source code information from sources other than compiler listings or Fault Analyzer side files stored as members in the usual IDI data sets (IDILANGX, IDILCOB, IDILPLI, and so on). For example, compiler listings that are stored in a compressed format, or that are accessible only via a proprietary access method.

Whenever source code information is required by Fault Analyzer, the steps shown in Locating compiler listings or side files on page 380 are performed. When the Compiler Listing Read user exit is shown invoked, this implies all specified and available exits. That is, all exits are first invoked to provide a side file and later, if necessary, all exits are invoked to provide a compiler listing. Only the last side file or compiler listing provided by any exit is used.

Fault Analyzer provides the following information to the Compiler Listing Read user exit about the required source code information:

- The load module name. This name is available in the LST.MODULE_NAME data area field. See Parameters on page 484 for references to the LST data area.
- The load module data set name in the LST.LOAD_MODULE_DSN data area field.
- The CSECT name. This name is available in the LST.CSECT_NAME data area field.
- The entry-point name. This name is available in the LST.EP_NAME data area field, truncated to a maximum of 256 characters. If truncated, the first 254 characters of the entry-point name is followed by a tilde ('~') and the last character of the untruncated entry-point name.
- The compile date. This date is available in the LST.COMPILE_DATE data area field, in the format YYYY/MM/DD.
- The compile time. This time is available in the LST.COMPILE_TIME data area field, in the format HH:MM:SS.
- The listing type. This type is available in the LST.LISTING_TYPE data area field, as one of the following:

  L

  Compiler listing or assembler SYSADATA file

  S

  Fault Analyzer side file

  The source code information passed back to Fault Analyzer must be of the type specified in this field.
- The language type. This type is available in the LST.LANGUAGE_TYPE data area field, as one of the following:
  - Assembler
  - C/C++
  - COBOL
  - OS/VS COBOL
  - PL/I
- The expected record format of listings or side files provided by the exit. This format is available in the LST.RECFM data area field, as V, VB, VBA, F, FB, FBA, and so on.
- The expected logical record length of listings or side files provided by the exit. This length is available in the LST.LRECL data area field, in decimal character format.

Based on the above information, a user exit can provide the requested listing or side file. This provision is done in one of the following ways:

- By passing one record at a time to Fault Analyzer via the LST data area:
  - Unless the name of a variable containing the listing record is passed on the IDIWRITE command in a REXX EXEC, the listing or side file data record must be provided in the DATA_BUFFER field.
  - For variable-length records, the length of the record in DATA_BUFFER must be provided in the DATA_LENGTH field in decimal character format. This length is not inclusive of any variable-length record descriptor word and must be less than or equal to the value in the LRECL field minus 4 bytes.
  - For fixed-length records, the length of the record in DATA_BUFFER is expected to match the LRECL field. Any value provided in the DATA_LENGTH field is ignored.

  or
- By placing the name of a sequential data set or a PDS or PDSE 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 533.

Load module

The address of a write routine is available in the ENV.WRITE_ROUTINE_EP data area field, as a hexadecimal 31-bit address.

This routine must be invoked with R1 pointing to a fullword containing the address of the ENV data area. For example, the following code fragments of user exits in different programming languages could be used to invoke the write routine:

Assembler:

```
ASMEXIT  CSECT
         …
         L     R2,0(,R1)
         USING ENV,R2
         L     R3,4(,R1)
```

```
         USING LST,R3
         …
         L     R15,ENV_WRITE_ROUTINE_EP
         LA    R1,ENV_VERSION
         ST    R1,*+8
         BAL   R1,*+8
         DC    F'0'
         BALR  R14,R15
         …
         COPY  IDISXPLA
         …
```

C:

```
#include "SAMPLES(IDISXPLC)"
typedef void WRTN(ENV *pENV);
#pragma linkage(WRTN,OS)
int cexit(ENV *pENV, LST *pLST) {
  …
  WRTN *write_rtn;
  write_rtn = (WRTN *)pENV->WRITE_ROUTINE_EP;
  write_rtn(pENV);
  …
}
```

COBOL:

```
      …
      PROGRAM-ID. COBEXIT
      …
      LINKAGE SECTION.
          COPY IDISXPLB IN LIB.
      PROCEDURE DIVISION USING ENV, LST.
      MAIN SECTION.
            …
            CALL WRITE-ROUTINE-EP USING ENV.
            …
      END PROGRAM COBEXIT.
```

PL/I:

```
PLIEXIT: PROC (ENVPTR,LSTPTR)  OPTIONS(BYVALUE,FETCHABLE) ;
Dcl (Envptr,Lstptr)     Pointer ;
%include syslib(IDISXPLP) ;
Dcl IDIWRITE         Entry Variable Options(Asm Byaddr) ;
  …
  Entryaddr(IDIWRITE) = Envptr->Write_Routine_EP ;
  Call IDIWRITE (Envptr->Env) ;
  …
End PLIEXIT ;
```

Return codes from the write routine are the same as the return codes from the IDIWRITE REXX command, except that RC=8 (syntax error) cannot be returned. See IDIWRITE command on page 533.

An indicator that can be used to cancel the usage of a listing or side file being provided by the user exit is available in the LST data area field DISREGARD_EXIT_LISTING. If this field is set to 'Y', any data records that might have been passed back to

Fault Analyzer from the user exit is discarded. By means of this indicator, the user exit can prevent Fault Analyzer from using a partial listing in case errors occur while providing data records.

When invoked

This exit is invoked whenever source code information is required in any Fault Analyzer execution mode.

Parameters

How parameters are passed to the exit depends on the exit type, REXX or load module.

Fault Analyzer initializes the parameter lists using current values for the particular fault and processing options in effect before invoking the Compiler Listing Read user exit.

REXX

Two stems are available to the exit:

- ENV.

  Contains defined symbols for all fields in the ENV data area (see ENV - Common exit environment information on page 653).
- LST.

  Contains defined symbols for all fields in the LST data area (see LST - Compiler Listing Read user exit parameter list on page 670).

The defined variable names are identical to the field names. For example, to access the field VERSION in the ENV data area, use the REXX variable ENV.VERSION.

Load module

At entry to this exit, R1 contains the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.

  Address of an ENV data area (see ENV - Common exit environment information on page 653).
- 31-bit LST address in word 2.

Address of an LST data area (see LST - Compiler Listing Read user exit parameter list on page 670).

Note: The high-order bit is on to indicate that this parameter is the last parameter passed.

Example

The following example is a Compiler Listing Read user exit that is written in REXX. The partitioned data set myid.LISTING.TERSE contains compressed compiler listings created by the IBM® AMATERSE utility.

When Fault Analyzer requests a compiler listing file, the exit searches the myid.LISTING.TERSE PDS for a member name that matches the module name. If a matching member is found, the exit decompresses it into a work data set and uses the IDIWRITE command to pass it to Fault Analyzer.

Figure 235. Sample REXX Compiler Listing Read user exit

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if LST.VERSION <> 1 then
  say 'Note: LST data area version change - field usage review required!'
if LST.LISTING_TYPE = 'L' then do
  modnm = strip(LST.MODULE_NAME)
  "IDIALLOC DD(INFILE) DSN(myid.LISTING.TERSE("||modnm||")) SHR"
  if rc = 0 then do
    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 "amaterse unpack"
    address mvs "execio * diskr outfile (finis"
    do while queued() <> 0
      parse pull rec
      LST.DATA_LENGTH = length(rec)
      LST.DATA_BUFFER = rec
      "IDIWRITE"
    end
    "IDIFREE DD(INFILE,OUTFILE)"
  end
end
exit 0
```

If the preceding sample exit exists as member ABC in data set X.Y.Z, you can invoke it by using the following options in either the IDICNFxx configuration member or the IDIOPTS user options file:

```
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 provide explanations of messages and abend codes for the analysis report. A Message and Abend Code Explanation user exit receives control after Fault Analyzer determines the availability of either a message explanation or an abend code explanation and before presenting the explanation in the analysis report. If Fault Analyzer finds an explanation using the softcopy books or override data sets, then the XPL.EXPLANATION_AVAILABLE data area field is set to 'Y'.

See XPL - Message and Abend Code Explanation user exit parameter list on page 690 for details about the data area.

The Message and Abend Code Explanation user exit can:

- 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.

  This text is available in the fields MESSAGE_TEXT1 through MESSAGE_TEXT10. MESSAGE_TEXT1 is initialized to contain the first or only line of the message, MESSAGE_TEXT2 through MESSAGE_TEXT10 are used if the message consists of multiple lines.

  Note: If the actual text of the message issued is not available, only the message ID is placed in the MESSAGE_TEXT1 field.

If an abend code explanation is being formatted by Fault Analyzer, the following information is provided in the XPL data area:

- The abend code.

  This code is available in the ABEND_CODE field. The contents of this field depends on the abend type. See below for details.

- The abend reason code.

  This code is available in the ABEND_REASON_CODE field as an 8-character hexadecimal value.

- The abend module name.

  This name is available in the ABEND_MODULE_NAME field.

- The abend type.

  This type is available in the ABEND_TYPE field which contains one of the following values:

  C

  Indicating a CICS® transaction abend.

  The field ABEND_CODE contains a 4-character CICS® abend code.

D

Indicating a CICS® dump code.

The field ABEND_CODE contains a 4-character CICS® dump code.

S

Indicating a system abend.

The field ABEND_CODE contains a 3-character hexadecimal left-justified system abend code.

U

Indicating a user abend.

The field ABEND_CODE contains a 4-character decimal user abend code.

Based on the above information, a user exit can provide a missing explanation or a replacement of the one found by Fault Analyzer. This provision is done by passing one record of the explanation at a time to Fault Analyzer via the XPL data area.

Unless the name of a variable containing the explanation record is passed on the IDIWRITE command in a REXX EXEC, the explanation data record must be provided in the DATA_BUFFER field.

Having updated the XPL data area with the data record information, two different methods are available for passing the data to Fault Analyzer depending on the exit type:

REXX

The Fault Analyzer environment IDIWRITE command is used. To use this command, code a REXX statement as follows:

```
ADDRESS FAULTA 'IDIWRITE [var-name]'
```

Successful completion of the IDIWRITE command is indicated by a zero return code.

For detailed information about the IDIWRITE command, see IDIWRITE command on page 533.

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 533.

The total number of characters that form the message or abend code explanation must not exceed *max-chars* in the following formula:

```
max-chars = 32752 - (num-recs * 2)
```

where *num-recs* is the total number of records.

An attempt to pass back a record that would cause the maximum number of characters to exceed *max-chars* results in RC=4 being returned by IDIWRITE (or the write routine used by load module exits) and the record being ignored.

No formatting of text is performed by Fault Analyzer. All records are presented in the analysis report exactly as they were provided by the Message and Abend Code Explanation user exit.

Records whose length exceed the formatting with at the time of presentation are wrapped at the current indentation, as determined by the leading number of blank characters in each record.

If more than one Message and Abend Code Explanation user exit provides a specific message or abend code explanation, then only the last explanation is used.

When invoked

This exit is invoked during formatting of the analysis report, regardless of the execution mode of Fault Analyzer.

Parameters

How parameters are passed to the exit depends on the exit type, REXX or load module.

Fault Analyzer initializes the parameter lists using current values for the particular fault and processing options in effect before invoking the Message and Abend Code Explanation user exit.

REXX

Two stems are available to the exit:

- ENV.

  Contains defined symbols for all fields in the ENV data area (see ENV - Common exit environment information on page 653).

- XPL.

  Contains defined symbols for all fields in the XPL data area (see XPL - Message and Abend Code Explanation user exit parameter list on page 690).

The defined variable names are identical to the field names. For example, to access the field VERSION in the ENV data area, use the REXX variable ENV.VERSION.

Load module

At entry to this exit, R1 contains the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.

  Address of an ENV data area (see ENV - Common exit environment information on page 653).

- 31-bit XPL address in word 2.

  Address of an XPL data area (see XPL - Message and Abend Code Explanation user exit parameter list on page 690).

  Note: The high-order bit is on to indicate that this parameter is the last parameter passed.

Example

The following is an example of a Message and Abend Code Explanation user exit that is written in REXX.

Figure 236. Sample REXX Message and Abend Code Explanation user exit

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if XPL.VERSION <> 1 then
  say 'Note: XPL data area version change - field usage review required!'
parse var XPL.MESSAGE_TEXT1 msgid msgtext
if msgid = 'MYMSG01' then do
  rec = 'This message indicates that:'
  'IDIWRITE rec'
  rec = '  - A serious problem has occurred'
  'IDIWRITE rec'
  rec = '  - Any data produced should be ignored'
  'IDIWRITE rec'
end
```

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

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

Formatting user exit

Note: The normal load module format user exit is not supported as a Formatting user exit. However, a special load module format user exit, IDIXUFMT, is available. For details, see The IDIXUFMT load module Formatting user exit on page 550.

Purpose

This exit can be used to create a user-specific section in the analysis report, which can, for example, be used to format data areas which are specific to the application environment being analyzed.

These are different ways to invoke exits of this type:

1. By using the Exits option.

   This option results in a User analysis report section being inserted between the System-Wide Information section and the Abend Job Information section. For details on specifying the Exits option, see Exits on page 600.

   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> A B E N D   J O B   I N F O R M A T I O N
   ⋮
   ```

   In the interactive reanalysis report, the user analysis section is selectable from the Interactive Reanalysis Report display as shown in the following:

   ```
   1.  Synopsis
   2.  Event Summary
   3.  System-Wide Information
   4.  User
   5.  Abend Job Information
   6.  Options in Effect
   ```

   The data from all invoked Formatting user exits is included in the report.

   The default "User" heading can be changed by setting UFM.USEROPTIONTITLE to a different value. The last value set by any Formatting user exit is used.

2. By using the EXEC command.

   This option is only available from the interactive reanalysis report. For details on using the EXEC command, see EXEC on page 101.

3. By making available a load module in an APF-authorized library named IDIXUFMT. For more information about this type of Formatting user exit, see The IDIXUFMT load module Formatting user exit on page 550.

The Formatting exit is initially provided with information about the point-of-failure event in the exit-specific UFM data area. (See Parameters on page 493 for references to the UFM data area.) However, information for any event can be obtained by using the IDIEventInfo command. This process is described in IDIEventInfo command on page 528.

Information about the existence of a load module, including its load address and length, can be obtained using the IDIModQry command. This process is described in IDIModQry command on page 530.

To obtain storage from the analyzed environment, or to write data to the report, extra commands are available:

Evaluate

This command is used to retrieve storage from the analyzed environment.

Details of this command are provided in Evaluate command on page 517.

List

This command is used to print storage in the report.

Details of this command are provided in List command on page 536.

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 538.

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 539.

The tagged text is passed back to Fault Analyzer using the IDIWRITE command, in one of three different ways:

1. Using a quoted string

   Example:

   ```
   IDIWRITE '<p>Paragraph text.'
   ```

   Either single quotes (') or double quotes (") can be used to enclose the string. However, both characters must be of the same type.

   If the string contains characters of the same type as those used to enclose the string, then these must be repeated twice. That is, to pass back the string

   ```
   'The TCB's address is not zero'
   ```

   specify

   ```
   'The TCB''s address is not zero'
   ```

2. Using a variable

   Example:

   ```
   data = '<p>Paragraph text.'
   IDIWRITE data
   ```

3. Using the UFM data area

   Example:

   ```
   UFM.DATA_BUFFER = '<p>Paragraph text.'
   UFM.DATA_LENGTH = length(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 233.

Parameters

Fault Analyzer initializes the parameter lists using current values for the particular fault and processing options in effect before invoking the Message and Abend Code Explanation user exit.

Two stems are available to the exit:

- ENV.

  Contains defined symbols for all fields in the ENV data area (see ENV - Common exit environment information on page 653).

- UFM.

  Contains defined symbols for all fields in the UFM data area (see UFM - Formatting user exit parameter list on page 674).

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.

Sample Formatting user exits

The following REXX samples are provided as data set members of IDI.SIDISAM1.

If a sample exit existed as member IDISUFM*n* in data set X.Y.Z, then providing the following options in either the IDICNFxx configuration member or the IDIOPTS user options file would cause it to be invoked during analysis:

```
DataSets(IDIEXEC(X.Y.Z))
Exits(FORMAT(REXX(IDISUFMn)))
```

Alternatively, the exit could be invoked from the interactive reanalysis report by entering the `EXEC IDISUFMn` command.

IDISUFM1

Sample Fault Analyzer Formatting user exit to display TCB information.

IDISUFM2

Sample Fault Analyzer Formatting user exit to display CICS CWA information.

IDISUFM3

Sample Fault Analyzer Formatting user exit to illustrate the use of formatting tags.

IDISUFM4

Sample Fault Analyzer Formatting user exit to use with Hogan applications.

IDISUFM5

Sample Fault Analyzer Formatting user exit to use with an MVS dump-analysis batch job to create a fault entry in a designated history file. Optionally, the exit can write a WTO message or send an email with information about the fault entry that was created.

IDISUFM6

Sample Fault Analyzer Formatting user exit to format LE CAA and CIB.

IDISUFM7

Sample Fault Analyzer Formatting user exit to display Name/Token values.

End Processing user exit

The following describes the End Processing user exit.

Purpose

This exit can be used to control post-analysis actions taken by Fault Analyzer:

- **History file selection.**

  Following analysis, detailed information about the fault is available which might not have been available at the time of calling any Analysis Control user exits. Based on this, an End Processing user exit can choose to change the history file to be used by modifying the ENV.IDIHIST data area field.

  Because information about duplicate faults depend on the history file that is selected, the End Processing user exit is invoked repeatedly until the history file name is no longer changed. On each reinvocation, duplicate fault information is updated for the history file specified on the previous invocation.

  If an invalid history file is provided in the ENV.IDIHIST, then no reinvocation of the End Processing user exit occurs, and the history file that was current when the End Processing user exit was last invoked is used.

- **Duplicate fault determination.**

  By default, Fault Analyzer deems a fault a duplicate of another fault in the same history file if the characteristics of the faults match and they occurred within the number of hours in effect for the NoDup(NORMAL(*hours*)) option of each other (see NoDup on page 616 for details). The End Processing user exit permits an installation to apply a different time interval for the determination of duplicate faults to the one in effect due to the NoDup option.

  If a duplicate fault was found based on the fault characteristics alone, then the following information is provided:

- The field EPC.MINUTES_SINCE_LAST_DUP is initialized to the number of minutes elapsed since recording of the last duplicate fault. The value is in the range 0 - 99999 (values greater than the limit of this field are all presented as the maximum value, 99999). If no value is provided, no duplicate fault was found.
- The field EPC.DUPLICATE_COUNT shows the total number of times that a fault was deemed a duplicate, not including the current fault.

  This total is determined by accumulating all instances of duplicate recorded faults, in the same history file, going back as far as the NoDup(Normal(...)) time period in effect. Any recorded faults encountered whose duplicate criteria match the current fault accounts for one instance, and if duplicates have been recorded against the fault, the duplicate count of that fault is also added.
- The field ENV.FAULT_ID identifies the recorded duplicate fault by its fault ID.
- The fields ENV.DUP_DATE and ENV.DUP_TIME identify the date and time of most recent duplicate fault.
- The fields ENV.ORIGINAL_DATE and ENV.ORIGINAL_TIME identify the date and time when the original fault was recorded. These fields provide the user with the ability to determine duplicates on the basis of time since the original fault, as opposed to the last duplicate fault, if so desired.

If both the fault characteristics matched and the elapsed time did not exceed the number of hours in effect for the NoDup(NORMAL(*hours*)) option, the field EPC.IS_DUPLICATE is initialized to 'Y'. However, the final determination of whether the fault is a duplicate or not resides with the End Processing user exit. If the returned value of this field is 'Y', the last recorded duplicate fault's (if any) duplicate count is incremented by one and message IDI0044I on page 705 is issued.

Note: CICS® or IMS™ fast duplicate fault suppression cannot be controlled using this exit. Only the NoDup(CICSFAST(...)) or the NoDup(ImageFast(...)) option (see NoDup on page 616) can be used to affect the determination of these types of duplicates. However, if a fault occurring under CICS® or IMS™ is not deemed a fast duplicate, then it is still considered for the normal type of duplicate suppression based on existing entries in the target history file and the NoDup(NORMAL(*hours*)) option in effect.

- **History file updates.**

By default, Fault Analyzer suppresses the entire fault entry, including the minidump, if the current fault is found to be a duplicate of a previously recorded fault in the same history file. However, the minidump might also be suppressed if its size exceeds the limit imposed by the MaxMinidumpPages option in effect.

The possible reasons for suppression in the following fields:

- The ENV.MINIDUMP_PAGES field, containing the size of the minidump as a number of 4K pages.
- The EPC.IS_DUPLICATE field (see *"Duplicate fault determination"* above).

The suppression intent by Fault Analyzer is indicated by the initialization of the following fields, both of which can be overridden by the End Processing user exit:

  - The EPC.SUPPRESS_MINIDUMP field. If set to 'Y', no minidump is written to the history file. If set to 'N', the minidump is written regardless of its size.
  - The EPC.SUPPRESS_FAULT_ENTRY field. If set to 'Y', no recording of the current fault is made in the history file (including the minidump). If set to 'N', fault recording is performed and the minidump can be written subject to the EPC.SUPPRESS_MINIDUMP field.
- **Dump suppression.**

  The End Processing user exit can set the EPC.SUPPRESS_DUMP field to 'Y' if dumps should be suppressed, or to 'N' if they should be permitted to be taken.

  Refer to Dump suppression on page 32 for general information about dump suppression.

When invoked

This exit is invoked on completion of real-time analysis, prior to updating the history file.

Parameters

How parameters are passed to the exit depends on the exit type, REXX or load module.

Fault Analyzer initializes the parameter lists using current values for the particular fault and processing options in effect before invoking the End Processing user exit.

REXX

Two stems are available to the exit:

- ENV.

  Contains defined symbols for all fields in the ENV data area (see ENV - Common exit environment information on page 653).
- EPC.

  Contains defined symbols for all fields in the EPC data area (see EPC - End Processing user exit parameter list on page 668).

The defined variable names are identical to the field names. For example, to access the field VERSION in the ENV data area, use the REXX variable ENV.VERSION.

Load module

At entry to this exit, R1 contains the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.

  Address of an ENV data area (see ENV - Common exit environment information on page 653).

- 31-bit EPC address in word 2.

  Address of an EPC data area (see EPC - End Processing user exit parameter list on page 668).

  Note: The high-order bit is on to indicate that this parameter is the last parameter passed.

Example

The following is an example of an End Processing user exit that is written in REXX.

Figure 237. Sample REXX End Processing user exit

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if EPC.VERSION <> 1 then
  say 'Note: EPC data area version change - field usage review required!'
if EPC.MINUTES_SINCE_LAST_DUP ¬= ' ' & EPC.MINUTES_SINCE_LAST_DUP < 48*60 then do
  /* Use 48 hours as the duplicate fault threshold */
  EPC.IS_DUPLICATE = 'Y'
  EPC.SUPPRESS_FAULT_ENTRY = 'Y'
end
EPC.SUPPRESS_DUMP = 'N'  /* Always permit dumps to be taken */
exit 0
```

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

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

End Processing user exit (Fault entry refresh)

The following describes the fault entry refresh End Processing user exit.

Purpose

This exit can be used to control history file updates.

By default, Fault Analyzer suppresses the minidump if its size exceeds the limit imposed by the MaxMinidumpPages option in effect.

The minidump size is available in ENV.MINIDUMP_PAGES field as a number of 4K pages.

The suppression intent by Fault Analyzer is indicated by the initialization of the EPC.SUPPRESS_MINIDUMP data area field, which can be overridden by the End Processing user exit. If set to 'Y', no minidump is written to the history file. If set to 'N', the minidump is written regardless of its size.

The End Processing user exit can choose to suppress the refresh of the entire fault entry using the EPC.SUPPRESS_FAULT_ENTRY data area field. If set to 'Y', no refresh of the current fault is performed (including the minidump). If set to 'N', the refresh is performed and the minidump can be written subject to the SUPPRESS_MINIDUMP field.

When invoked

This exit is invoked on completion of batch reanalysis, prior to updating the history file.

Parameters

See Parameters on page 496.

Example

The following is an example of a fault entry refresh End Processing user exit that is written in REXX.

Figure 238. Sample fault entry refresh REXX End Processing user exit

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if EPC.VERSION <> 1 then
  say 'Note: EPC data area version change - field usage review required!'
EPC.SUPPRESS_MINIDUMP = 'Y'  /* Always suppress the minidump */
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 IDICNFxx 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. For example, the exit might send an email to the person who is responsible for the failing application, or log the fault via the ADFzCC Event Processing user exit.

The reason for invoking the exit is identified in the NFYTYPE field of the NFY data area on page 673 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 on page 696 to indicate the assigned fault ID and history file.

A copy of the synopsis section of the real-time report is available in the NFY.SYNOPSIS data area field. Each line of the synopsis is delimited by a new-line character (X'15'). Refer to the NFY data area for extra details regarding this field.

R

Recovery fault recording

This value indicates a fault that was created as a result of the recovery fault recording feature of Fault Analyzer. (For more information about this feature, see Recovery fault recording on page 51.)

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 on page 723 to indicate the assigned fault ID and history file.

N

Normal duplicate.

This value indicates that the NoDup(NORMAL) option criteria matched for the current fault, and that no history file fault entry is therefore written. (For details about NoDup(NORMAL), see NoDup on page 616.)

The original history file name and fault ID are provided in the ENV.IDIHIST and ENV.FAULT_ID data area fields.

The DUPCOUNT field is set to 1.

F

Fast duplicate (CICS®).

This value indicates that the NoDup(CICSFAST) option criteria matched for the current fault, and that no analysis was therefore performed. (For details about NoDup(CICSFAST), see NoDup on page 616.)

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 is set to the number of duplicate occurrences for the 30-second recording period.

If you use the IDIWRITE command in a Notification user exit, Fault Analyzer invokes the ADFzCC Event Processing feature after the exit has completed, and uses the data that is written with the IDIWRITE command to pass to the ADFzCC Event Processing user exit. For more information about the ADFzCC Event Processing feature, see *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*.

The data that is written using the IDIWRITE command by multiple Notification user exits is sent to the ADFzCC Event Processing user exit in a single call. However, it is recommended to not use IDIWRITE in your own Notification user exits. Instead, specify the sample Notification user exit IDISXEPN in the EXITS option as follows:

```
EXITS(NOTIFY(<existing exit names,>REXX(IDISXEPN)))
```

and ensure that it can be located via the IDIEXEC DDname.

The `IDISXEPN` sample exit sends all available `ENV` and `NFY` data to the ADFzCC Event Processing user exit by using the IDIWRITE command.

When you issue the IDIWRITE command from a Notification user exit, the data is passed to the ADFzCC Event Processing user exit. Before you issue the IDIWRITE command, the data to be written is placed in the `NFY.DATA_BUFFER` field, and the data length is specified in the `NFY.DATA_LENGTH` field.

After the Notification user exit has completed, the buffer contents are sent to the ADFzCC Event Processing user exit. For the format of the buffer, see the section about load module IPVEPSND in *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*. Each segment in the buffer contains the contents of each individual call to the IDIWRITE command. When you use the sample exit IDISXEPN that is provided, each segment contains data in the following format:

```
data_area.field_name=value
```

For example:

```
ENV.JOB_NAME=BATCH1
```

By setting the `NFY.EPX_DEBUG_OPT` to `*` or any other nonblank character, the Notification user exit asks for any diagnostics from the ADFzCC Event Processing user exit to be written to `IDITRACE`.

Note: If the `NFY.EPX_DEBUG_OPT` is set to a non-blank character, the ADFzCC Event Processing user exit will run synchronously because the Notification user exit must wait until the ADFzCC Event Processing user exit has completed to receive debug information, which might impact the performance.

If you leave `NFY.EPX_DEBUG_OPT` blank (the default setting), Fault Analyzer continues without waiting for the Event Processing user exit to complete, but diagnostics are not written to `IDITRACE`.

If no Event Processing user exit is specified by using the ADFzCC `EVENTPROCESSINGEXIT` option, the data is ignored. For information about specifying an Event Processing user exit by using the `EVENTPROCESSINGEXIT` option, see *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*.

If you use the IDIWRITE command in a Notification user exit, Fault Analyzer invokes the ADFzCC Event Processing feature after the exit has completed, and uses the data that is written with the IDIWRITE command to pass to the ADFzCC Event Processing user exit. For more information about the ADFzCC Event Processing feature, see *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*.

The data that is written using the IDIWRITE command by multiple Notification user exits is sent to the ADFzCC Event Processing user exit in a single call. However, it is recommended to not use IDIWRITE in your own Notification user exits. Instead, specify the sample Notification user exit IDISXEPN in the EXITS option as follows:

```
EXITS(NOTIFY(<existing exit names,>REXX(IDISXEPN)))
```

and ensure that it can be located via the IDIEXEC DDname.

The `IDISXEPN` sample exit sends all available `ENV` and `NFY` data to the ADFzCC Event Processing user exit by using the IDIWRITE command.

When you issue the IDIWRITE command from a Notification user exit, the data is passed to the ADFzCC Event Processing user exit. Before you issue the IDIWRITE command, the data to be written is placed in the `NFY.DATA_BUFFER` field, and the data length is specified in the `NFY.DATA_LENGTH` field.

After the Notification user exit has completed, the buffer contents are sent to the ADFzCC Event Processing user exit. For the format of the buffer, see the section about load module IPVEPSND in *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*. Each segment in the buffer contains the contents of each individual call to the IDIWRITE command. When you use the sample exit IDISXEPN that is provided, each segment contains data in the following format:

```
data_area.field_name=value
```

For example:

```
ENV.JOB_NAME=BATCH1
```

By setting the `NFY.EPX_DEBUG_OPT` to `*` or any other nonblank character, the Notification user exit asks for any diagnostics from the ADFzCC Event Processing user exit to be written to `IDITRACE`.

Note: If the `NFY.EPX_DEBUG_OPT` is set to a non-blank character, the ADFzCC Event Processing user exit will run synchronously because the Notification user exit must wait until the ADFzCC Event Processing user exit has completed to receive debug information, which might impact the performance.

If you leave `NFY.EPX_DEBUG_OPT` blank (the default setting), Fault Analyzer continues without waiting for the Event Processing user exit to complete, but diagnostics are not written to `IDITRACE`.

If no Event Processing user exit is specified by using the ADFzCC `EVENTPROCESSINGEXIT` option, the data is ignored. For information about specifying an Event Processing user exit by using the `EVENTPROCESSINGEXIT` option, see *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*.

When invoked

This exit is invoked after Fault Analyzer has finished the recording of a fault in the history file.

Parameters

How parameters are passed to the exit depends on the exit type, REXX or load module.

Fault Analyzer initializes the parameter lists using current values for the particular fault and processing options in effect before invoking the Notification user exit.

REXX

Two stems are available to the exit:

- ENV.

  Contains defined symbols for all fields in the ENV data area (see ENV - Common exit environment information on page 653).

- NFY.

  Contains defined symbols for all fields in the NFY data area (see NFY - Notification user exit parameter list on page 673).

The defined variable names are identical to the field names. For example, to access the field VERSION in the ENV data area, use the REXX variable ENV.VERSION.

Load module

At entry to this exit, R1 contains the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.

  Address of an ENV data area (see ENV - Common exit environment information on page 653).

- 31-bit NFY address in word 2.

  Address of an NFY data area (see NFY - Notification user exit parameter list on page 673).

  Note: The high-order bit is on to indicate that this parameter is the last parameter passed.

Calling a non-REXX logging routine from REXX

For a Notification user exit that is written in REXX that, for example, calls an external logging routine that is not written in REXX, two extra stem variables are provided. The stem variables can be used to pass all of the available data area values to the external routine. These stem variables are:

ENV.RECORD
NFY.RECORD

Each of these contain the entire ENV or NFY data area respectively, in a single REXX variable. Neither exist as fields in the respective data areas for non-REXX exits, as they each represent the entire data area, which is already provided.

By using these variables in an argument list for an external non-REXX routine, the REXX exit need not be concerned with changes that might occur to these data areas in the future.

The external routine should use the language-dependent data area mappings provided for access to any data values (see Load module exits on page 468). All values in the ENV.RECORD and NFY.RECORD variables are considered read only and cannot be updated by the external routine. If updates are required, these must be made in the appropriate data field stem variables by the REXX exit itself.

Example 1

The following is an example of a Notification user exit that is written in REXX. It issues a TSO SEND message.

Figure 239. Sample REXX Notification user exit 1

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if NFY.VERSION <> 2 then
  say 'Note: NFY data area version change - field usage review required!'
/* Issue SEND command via TSO batch job */
queue "//NOTIFY   JOB  MSGCLASS=Z"
queue "//TSOBATCH EXEC PGM=IKJEFT01"
queue "//SYSTSPRT DD   SYSOUT=*"
queue "//SYSTSIN  DD   *"
/* Split the TSO SEND command over three data records that must
   each be padded with blanks to 80 bytes */
rec = "SEND 'Fault ID" 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)"
if rc = 0 then do
   address mvs "EXECIO" n "DISKW DD1 (FINIS"
   "IDIFREE DD(DD1)"
end
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 IDICNFxx 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 that is written in REXX. It sends an e-mail message via SMTP.

Figure 240. Sample SMTP REXX Notification user exit 2

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if NFY.VERSION <> 2 then
  say 'Note: NFY data area version change - field usage review required!'
/* Issue SMTP message via SMTP batch interface */
user = strip(ENV.USER_ID)
queue "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
```

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 IDICNFxx configuration member or the IDIOPTS user options file would cause it to be invoked:

```
DataSets(IDIEXEC(TEST.EXEC.PDS))
Exits(NOTIFY(REXX(NOTIFY1)))
```

Example 3

The following is an example of a Notification user exit that is written in REXX. It extracts and shows all lines of the captured real-time synopsis.

Figure 241. Sample SMTP REXX Notification user exit 3

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if NFY.VERSION <> 2 then
  say 'Note: NFY data area version change - field usage review required!'
/* Show synopsis */
rest = NFY.SYNOPSIS
do while rest<>''
  parse var rest nextline '15'x rest
  say nextline
end
exit 0
```

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 IDICNFxx configuration member or the IDIOPTS user options file would cause it to be invoked:

```
DataSets(IDIEXEC(TEST.EXEC.PDS))
Exits(NOTIFY(REXX(FRED)))
```

Example 4

The following is an example of a Notification user exit that is written in REXX and can be used to submit a batch reanalysis job to generate a saved report for a DeferredReport fault entry, and optionally save any side files used by means of the LangxCapture option.

Figure 242. Sample REXX Notification user exit 4

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if NFY.VERSION <> 2 then
  say 'Note: NFY data area version change - field usage review required!'
queue "//GENREP   JOB  MSGCLASS=X"
queue "//FA       EXEC PGM=IDIDA,"
queue "// PARM=('/FAULTID("ENV.FAULT_ID")',"
queue "//       'GenerateSavedReport',"
queue "//      )"
queue "//IDIHIST  DD   DISP=SHR,DSN="ENV.IDIHIST
queue "//SYSPRINT DD   SYSOUT=*"
/* 'Submit' the stacked batch reanalysis job */
n = queued()
"IDIALLOC DD(DD1) SYSOUT PGM(INTRDR)"
if rc = 0 then do
   address mvs "EXECIO" n "DISKW DD1 (FINIS"
   "IDIFREE DD(DD1)"
end
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 IDICNFxx configuration member or the IDIOPTS user options file would cause it to be invoked:

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

Notification user exit (MVS SVC Dump registration)

The following describes the dump registration Notification user exit.

Purpose

This exit can be used to provide installation-specific notification about the recording of an SVC dump fault entry in a history file.

The name of the history file in which the fault was recorded is provided in the ENV.IDIHIST data area field, and the fault ID assigned to the recorded fault is available in the ENV.FAULT_ID data area field.

Fault Analyzer issues message IDI0003I on page 696 to indicate the assigned fault ID and history file.

When invoked

This exit is invoked after Fault Analyzer has finished the registration of an MVS™ SVC dump fault entry in the history file.

Parameters

See Parameters on page 501.

Example

The following is an example of a dump registration Notification user exit that is written in REXX.

Figure 243. Sample REXX dump registration Notification user exit

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if NFY.VERSION <> 2 then
  say 'Note: NFY data area version change - field usage review required!'
/* Issue SEND command via TSO batch job */
queue "//NOTIFY   JOB  MSGCLASS=Z"
queue "//TSOBATCH EXEC PGM=IKJEFT01"
queue "//SYSTSPRT DD   SYSOUT=*"
queue "//SYSTSIN  DD   *"
/* Split the TSO SEND command over three data records that must
   each be padded with blanks to 80 bytes */
rec = "SEND 'Fault ID" 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)"
if rc = 0 then do
  address mvs "EXECIO" n "DISKW DD1 (FINIS"
  "IDIFREE DD(DD1)"
end
exit 0
```

Note that, unlike the normal Notification user exit, no user ID is available in the ENV data area.

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

```
DataSets(IDIEXEC(X.Y.Z))
DumpRegistrationExits(NOTIFY(REXX(ABC)))
```

The DumpRegistrationExits option must be specified in the IDICNFxx parmlib member or in an IDIOPTS DD statement in the IDIS subsystem JCL. The DumpRegistrationExits option is ignored if specified in an IDIOPTS DD statement anywhere else, such as in a CICS® region or batch job.

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

You can use the IDIUTIL Import user exit to:

- Control the import of a fault entry during history file management by using the IDIUTIL batch utility with the IMPORT control statement. This control is provided by setting the UTL.PERFORM_ACTION data area field to 'Y' to import the entry or to 'N' not to import it. By default, the field UTL.PERFORM_ACTION is always set to 'Y' before invoking the exit. See Parameters on page 508 for references to the UTL data area.

  When an entry has been successfully imported into the target history file, it is deleted from the source history file.
- Provide the name of the associated dump data set to be created from an exported fault entry in UTL.IMPORT_DUMP_DSN. This name overrides any default name obtained from the RFRDSN, XDUMPDSN, or SDUMPDSN option in the IDIOPTLM configuration-options module. For details, see Customize Fault Analyzer by using an IDIOPTLM configuration-options module on page 337.

When invoked

This exit is invoked once for each fault entry in a history file whenever the IDIUTIL batch utility is executed using the IMPORT control statement.

Parameters

How parameters are passed to the exit depends on the exit type, REXX or load module.

Fault Analyzer initializes the parameter lists using current values for the particular fault and processing options in effect before invoking the IDIUTIL Import user exit.

REXX

Two stems are available to the exit:

- ENV.

  Contains defined symbols for all fields in the ENV data area (see ENV - Common exit environment information on page 653).
- UTL.

  Contains defined symbols for all fields in the UTL data area (see UTL - IDIUTIL Batch Utility user exit parameter list on page 688).

The defined variable names are identical to the field names. For example, to access the field VERSION in the ENV data area, use the REXX variable ENV.VERSION.

Load module

At entry to this exit, R1 contains the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.

  Address of an ENV data area (see ENV - Common exit environment information on page 653).

- 31-bit UTL address in word 2.

  Address of a UTL data area (see UTL - IDIUTIL Batch Utility user exit parameter list on page 688).

Note: The high-order bit is on to indicate that this parameter is the last parameter passed.

Example

IDIROBEX is an example of an IDIUTIL user exit that is written in REXX.

This exit is included in the IDI.SIDISAM1 data set member IDISROBT. See Figure 199: Sample TSO receive REXX exec (IDIROBOT) part 1 on page 355 for details.

Figure 244. Sample REXX IDIUTIL user exit

```
/* REXX */
/* Dump data set names provided by this exit override the equivalent */
/* option setting in IDIOPTLM.                                       */
/* The relationship is as follows:                                   */
/* ENV.ASSOCIATED_DUMP_TYPE     IDIOPTLM option                      */
/* ---------------------------- ---------------                      */
/* 'S' SDUMP (SVC dump)         SDUMPDSN                             */
/* 'T' TDUMP (transaction dump) RFRDSN                               */
/* 'X' XDUMP (extended dump)    XDUMPDSN                             */
/* To disable the allocation of an associated dump data set, set     */
/* UTL.IMPORT_DUMP_DSN to 'NULLFILE'.                                */
if ENV.VERSION <> 5 then
say 'Note: ENV data area version change - field usage review required!'
if UTL.VERSION <> 2 then
say 'Note: UTL data area version change - field usage review required!'
ddsnhlq = 'IDIDUMP'                           /* <--- verify/change */
UTL.PERFORM_ACTION = 'Y'           /* Import current entry (default) */
if ENV.ASSOCIATED_DUMP_DSN <> "" then do
  t_parm = ENV.ASSOCIATED_DUMP_TYPE                        /* S/T/X */
  UTL.IMPORT_DUMP_DSN =,
    ddsnhlq"."t_parm"DUMP.&SYSNAME..D&YYMMDD..T&HHMMSS..S&SEQ."
end
exit 0
```

If the above sample exit existed as member ABC in data set X.Y.Z, then providing the following statements in your IDIUTIL batch utility history file management job would cause the exit to be invoked.

DD statement:

```
//IDIEXEC DD DISP=SHR,DSN=X.Y.Z
```

IDIUTIL batch utility control statement:

```
Exits(IMPORT(REXX(ABC)))
```

IDIUTIL Delete user exit

The following describes the IDIUTIL Delete user exit.

Purpose

This exit can 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 444). This control is provided by setting the data area field UTL.PERFORM_ACTION 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'. See UTL - IDIUTIL Batch Utility user exit parameter list on page 688 for details about the UTL data area.

The fault entries for which the user exit is invoked are those that match the specified DELETE control statement criteria.

When invoked

This exit is invoked once for each fault entry in a history file whenever the IDIUTIL batch utility is executed using the DELETE control statement.

Parameters

How parameters are passed to the exit depends on the exit type, REXX or load module.

Fault Analyzer initializes the parameter lists using current values for the particular fault and processing options in effect before invoking the IDIUTIL Delete user exit.

REXX

Two stems are available to the exit:

- ENV.

  Contains defined symbols for all fields in the ENV data area (see ENV - Common exit environment information on page 653).

- UTL.

  Contains defined symbols for all fields in the UTL data area (see UTL - IDIUTIL Batch Utility user exit parameter list on page 688).

The defined variable names are identical to the field names. For example, to access the field VERSION in the ENV data area, use the REXX variable ENV.VERSION.

Load module

At entry to this exit, R1 contains the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.

  Address of an ENV data area (see ENV - Common exit environment information on page 653).

- 31-bit UTL address in word 2.

  Address of a UTL data area (see UTL - IDIUTIL Batch Utility user exit parameter list on page 688).

Note: The high-order bit is on to indicate that this parameter is the last parameter passed.

Example

The following is an example of an IDIUTIL Delete user exit that is written in REXX.

Figure 245. Sample REXX IDIUTIL Delete user exit

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if UTL.VERSION <> 2 then
  say 'Note: UTL data area version change - field usage review required!'
UTL.PERFORM_ACTION = 'Y'  /* Delete current entry */
```

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 442). This control is provided by setting the data area field UTL.PERFORM_ACTION 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. See UTL - IDIUTIL Batch Utility user exit parameter list on page 688 for details about the UTL data area.

The fault entries for which the user exit is invoked are those that match the specified LISTHF control statement criteria.

When invoked

This exit is invoked once for each fault entry in a history file whenever the IDIUTIL batch utility is executed using the LISTHF control statement.

Parameters

How parameters are passed to the exit depends on the exit type, REXX or load module.

Fault Analyzer initializes the parameter lists using current values for the particular fault and processing options in effect before invoking the IDIUTIL ListHF user exit.

REXX

Two stems are available to the exit:

- ENV.

  Contains defined symbols for all fields in the ENV data area (see ENV - Common exit environment information on page 653).

- UTL.

  Contains defined symbols for all fields in the UTL data area (see UTL - IDIUTIL Batch Utility user exit parameter list on page 688).

The defined variable names are identical to the field names. For example, to access the field VERSION in the ENV data area, use the REXX variable ENV.VERSION.

Load module

At entry to this exit, R1 contains the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.

  Address of an ENV data area (see ENV - Common exit environment information on page 653).

- 31-bit UTL address in word 2.

  Address of a UTL data area (see UTL - IDIUTIL Batch Utility user exit parameter list on page 688).

Note: The high-order bit is on to indicate that this parameter is the last parameter passed.

Example 1: Invoking the IDIUTIL ListHF user exit

The following is an example of an IDIUTIL ListHF user exit that is written in REXX.

Figure 246. Sample REXX IDIUTIL ListHF user exit 1

```
/* REXX */
if ENV.VERSION <> 5 then
  say 'Note: ENV data area version change - field usage review required!'
if UTL.VERSION <> 2 then
  say 'Note: UTL data area version change - field usage review required!'
UTL.PERFORM_ACTION = 'Y'  /* List current entry */
```

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: Creating a custom report and CSV file of fault entries with the IDIUTIL ListHF user exit

The following is an example of an IDIUTIL ListHF user exit that is written in REXX.

This sample exit shows how one might write a customized report, as well as a comma-delimited file that can be used as input to a spreadsheet application.

In addition to the fields used, any other fields that are available from the ENV or UTL data areas can be included.

The report that is written by the provided sample includes the columns:

- Fault ID
- Date
- Time
- Lock
- Username
- User Title
- CPU Sec

See Available columns on page 67 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 CPU_HSECONDS on page 661 in ENV - Common exit environment information on page 653 for more information about this value.

The customized report is written to DDname MYREP. A MYREP DD statement must be included in the IDIUTIL job that invokes this exit, for example:

```
//MYREP DD SYSOUT=*
```

The comma-delimited file is written to DDname COMMA. A COMMA DD statement must be included in the IDIUTIL job that invokes this exit, for example:

```
//COMMA DD SYSOUT=*
```

The persistent user field, ENV.USER_1, is used to record the fact that the report header has been written.

Figure 247. Sample REXX IDIUTIL ListHF user exit 2

```
/* First ensure that the current data area versions match the         */
/* versions as at the time of coding the exit.                        */
If ENV.VERSION <> 5 Then
  Say 'Note: ENV data area version change - field usage review',
      'required!'
If UTL.VERSION <> 2 then
  Say 'Note: UTL data area version change - field usage review',
      'required!'
If ENV.USER_1='' Then Do
  /* Write report header */
  out.1="Fault ID Date       Time     Lock Username",
        "User Title                              CPU Sec"
  out.2="-------- ---------- -------- ---- --------",
        "---------------------------------------- -------"
  ADDRESS MVS "EXECIO 2 DISKW MYREP (STEM out."
  /* Write comma-delimited file header */
  out.1="Fault ID,Date,Time,Lock,Username,User Title,CPU Sec"
  ADDRESS MVS "EXECIO 1 DISKW COMMA (STEM out."
  ENV.USER_1='done'     /* Flag header done.                        */
End
/* The fault ID value is placed right-aligned in a work field.        */
fault_id=COPIES(' ',8-length(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 with 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
```

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
/*
```

Example 3: Collecting abend data for later statistical analysis with the IDIUTIL user exit

History files are dynamic. A fault entry can be deleted from a history file explicitly or by automatic space management. If you want to use fault entry data as input to an analytics application, you might need to collect and preserve the fault entry data outside of the history file. The IDISUTL2 member of the IDI.SIDISAM1 data set is a sample REXX program that you can run regularly for this purpose.

The sample JCL at the end of the IDISUTL2 member runs the RUNUTIL and MERGE steps:

- The RUNUTIL step runs IDISUTL2 as an IDIUTIL LISTHF user exit to generate a CSV-format file of the fault entries from one or more history file data sets. (This step is described in Example 2 on page 513.)
- The MERGE step appends the CSV data collected in the RUNUTIL step to the cumulative file specified by the ALLDATA DD statement.

IDIUTIL ListHFDUP user exit

The following describes the IDIUTIL ListHFDUP user exit.

Purpose

This exit can be used to control the listing of an abend instance during history file management using the IDIUTIL batch utility with the LISTHFDUP control statement (for details, see LISTHFDUP control statement on page 443). This control is provided by setting the data area field UTL.PERFORM_ACTION 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. Additionally, UTL.DUP_TYPE is provided, which provides information about the current abend instance type of duplicate. See UTL - IDIUTIL Batch Utility user exit parameter list on page 688 for details about the UTL data area.

The abend instances for which the user exit is invoked are those that match the specified LISTHFDUP control statement criteria.

When invoked

This exit is invoked once for each abend instance in a history file whenever the IDIUTIL batch utility is executed using the LISTHFDUP control statement.

Parameters

How parameters are passed to the exit depends on the exit type, REXX or load module.

Fault Analyzer initializes the parameter lists using current values for the particular abend instance and processing options in effect before invoking the IDIUTIL ListHFDup user exit.

REXX

Two stems are available to the exit:

- ENV.

  Contains defined symbols for all fields in the ENV data area (see ENV - Common exit environment information on page 653).

- UTL.

  Contains defined symbols for all fields in the UTL data area (see UTL - IDIUTIL Batch Utility user exit parameter list on page 688).

The defined variable names are identical to the field names. For example, to access the field VERSION in the ENV data area, use the REXX variable ENV.VERSION.

Load module

At entry to this exit, R1 contains the 31-bit address of a parameter list comprising two fullwords:

- 31-bit ENV address in word 1.

  Address of an ENV data area (see ENV - Common exit environment information on page 653).

- 31-bit UTL address in word 2.

  Address of a UTL data area (see UTL - IDIUTIL Batch Utility user exit parameter list on page 688).

  Note: The high-order bit is on to indicate that this parameter is the last parameter passed.

Example

The following is an example of an IDIUTIL ListHFDUP user exit that is written in REXX.

Figure 248. Sample REXX IDIUTIL ListHFDup user exit

```
/* REXX */
if ENV.VERSION <> 5 then
say 'Note: ENV data area version change - field usage review required!'
if UTL.VERSION <> 2 then
say 'Note: UTL data area version change - field usage review required!'
if UTL.DUP_TYPE = 'F'      /* If it is a CICSFast duplicate */
  UTL.PERFORM_ACTION = 'Y' /* list current entry */
else
  UTL.PERFORM_ACTION = 'N' /* otherwise don't */
```

If the above sample exit existed as member ABC in data set X.Y.Z, then providing the following JCL DD statement

```
//IDIEXEC DD DISP=SHR,DSN=X.Y.Z
```

and the following IDIUTIL batch utility control statement

```
Exits(LISTHFDUP(REXX(ABC)))
```

in your IDIUTIL batch utility history file management job would cause the exit to be invoked.

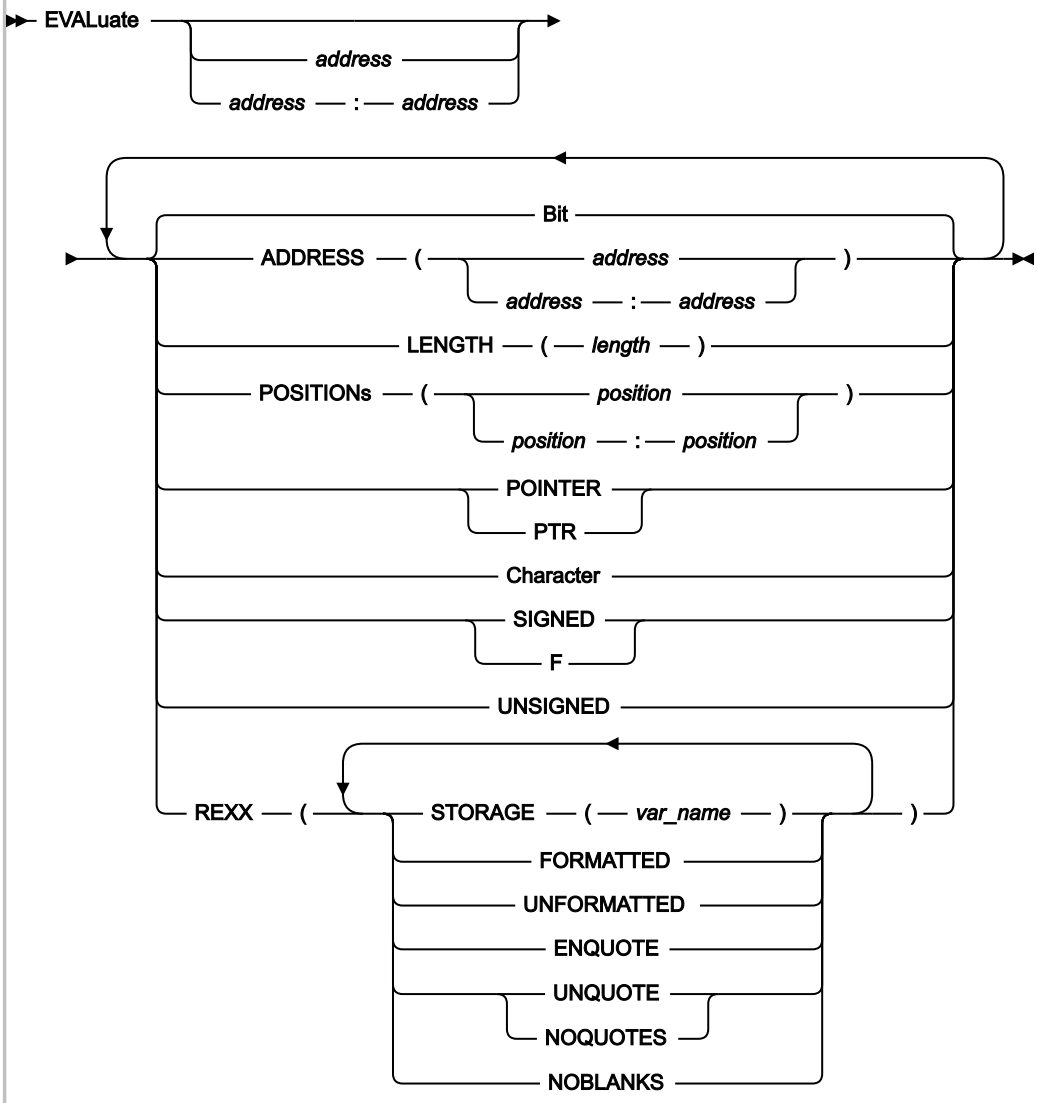
User exit REXX commands

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.

Figure 249. Syntax



Parameters

address start:end

Specifies either the start address, or an address range, as a positional parameter. This address is an alternative to using the ADDRESS keyword. See *ADDRESS* below for details about valid syntax.

ADDRESS(address) ADDRESS(address:address)

Specifies the address of the storage to be returned. Specify as either a single address or as an address range.

The address parameter is 64-bit enabled. For details, see Specifying 64-bit addresses on page 142. The address may optionally followed by a period (for example, "000176C0.").

LENGTH(length)

Specifies the number of bytes to be returned. Specify as a hexadecimal value (indicated by X'…') or as a decimal value (no indication required). If an address range is specified, then specification of LENGTH is ignored.

POSITIONs(position) POSITIONs(position:position)

Specifies the offset from the start address to the first byte of storage to be returned. Specify as either a single offset or as an offset range. Both POSITION and POSITIONS are accepted.

All offsets must be either hexadecimal (indicated by X'…') or signed decimal (optionally indicated by F'…').

If not specified, the default is 0.

Bit

Specifies that data is to be returned in hexadecimal format.

This parameter is only valid if FORMATTED is in effect.

POINTER PTR

Specifies that data is to be returned in hexadecimal format.

The valid LENGTH is 1 - 4 bytes. If LENGTH is greater than 4, then it is changed to 4.

This parameter is only valid if FORMATTED is in effect.

Character

Specifies that data is to be returned in character string format with non-printable characters replaced by periods. Further editing is determined by the specification of ENQUOTE, UNQUOTE, NOQUOTES, or NOBLANKS parameters.

This parameter is only valid if FORMATTED is in effect.

SIGNED F

Specifies that data is to be returned in signed decimal format. Leading zeroes are removed and a minus sign is supplied for negative integers.

The valid LENGTH is either 2 or 4 bytes. If LENGTH is 1 or 3, then it is changed to 2. If LENGTH is greater than 4, then it is changed to 4.

This parameter is only valid if FORMATTED is in effect.

UNSIGNED

Specifies that data is to be returned in unsigned decimal format. Leading zeroes are removed.

The valid LENGTH is 1 - 4 bytes. If LENGTH is greater than 4, then it is changed to 4.

This parameter is only valid if FORMATTED is in effect.

REXX(…)

Specifies that the storage is to be returned in a REXX variable. This parameter is required.

STORAGE(var_name)

Specifies the name of the REXX variable which is to receive the storage. This parameter is required.

FORMATTED

Specifies that storage is to be returned formatted. This value is the default.

UNFORMATTED

Specifies that storage is to be returned in raw hex format (one byte returned for each byte of storage).

ENQUOTE

Specifies that one leading and one trailing quote are to be added to the returned character string, and that any apostrophes found in the string are to be paired.

This parameter is only valid if character data is being returned.

UNQUOTE NOQUOTES

Specifies that all apostrophes (X'7D') in the returned character string are to be replaced by periods.

This parameter is only valid if character data is being returned.

NOBLANKS

Specifies that all blanks in the returned character string are to be replaced by periods.

This parameter is only valid if character data is being returned.

If a parameter is specified multiple times, then only the last specification takes effect.

Return codes

The Evaluate command provides the following return codes:

0

Storage was obtained successfully.

4

The requested data length was modified. An explanation is written to the IDITRACE DDname.

12

Command syntax error or storage not available. An explanation of the error is written to the IDITRACE DDname.

Example

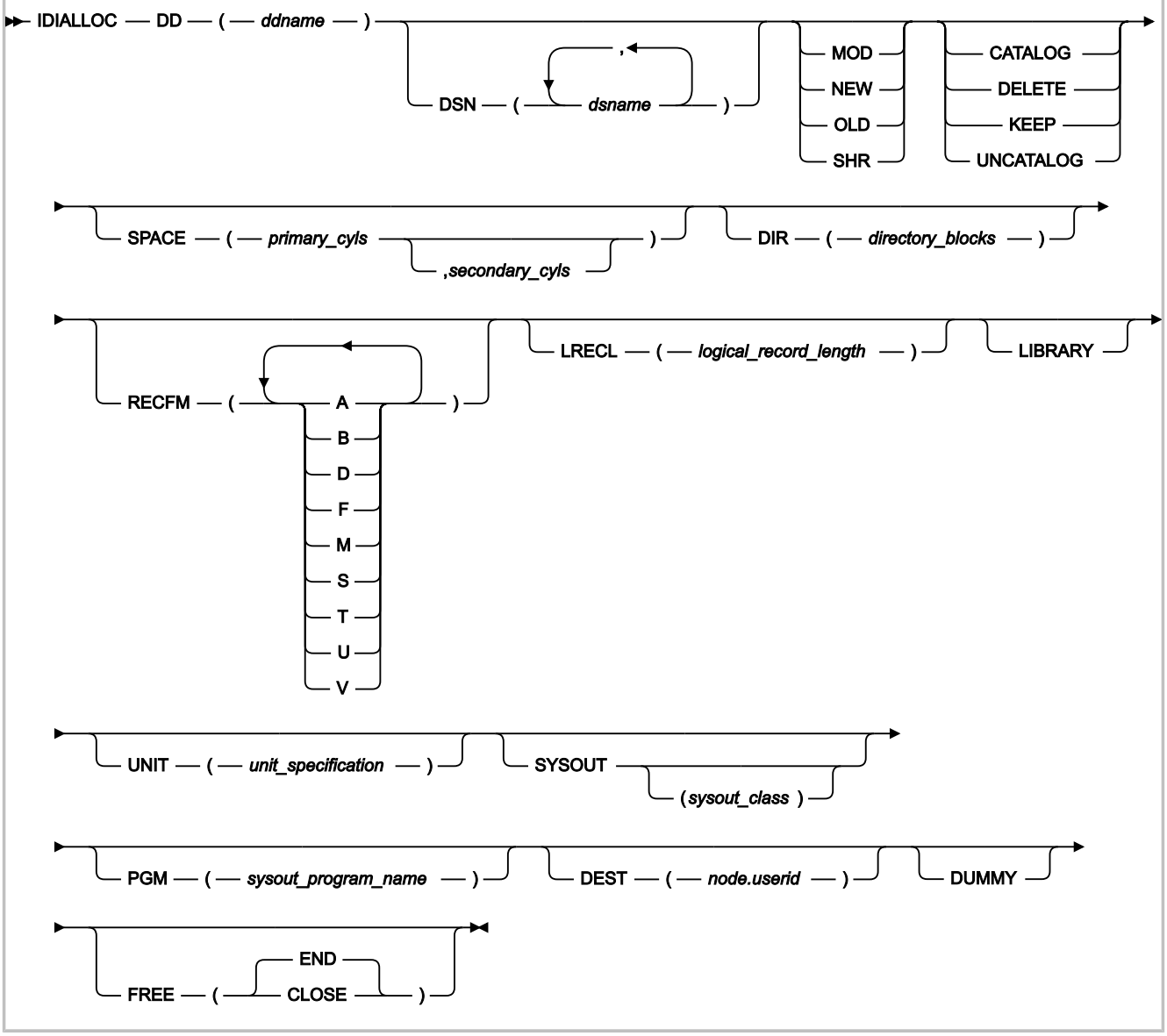
Figure 250. Evaluate command example

```
/* REXX */

"EVALUATE 0 LENGTH(128) REXX(STORAGE(x))"
if RC = 0 then say 'Storage at address 0 =' x
```

IDIALLOC command

The IDIALLOC command can be used to perform dynamic allocation and concatenation of data sets to DDnames.

Figure 251. Syntax



Note: No commas or blank characters can 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 DDname is always required.

DSN

Names of data sets to be allocated. If more than one data set name is specified, all data sets following the first are expected to already exist and will be concatenated to the specified DDname. zFS path names are not permitted.

MOD

Additions are to be made to data set.

NEW

Data set is to be created.

OLD

Data set exists and exclusive control is required.

SHR

Data set exists but exclusive control is not required.

CATALOG

Data set is to be cataloged.

DELETE

Data set is to be deleted when freed.

KEEP

Data set is to be kept when freed.

UNCATALOG

Data set is to be uncataloged.

SPACE

Primary space and increment as number of cylinders.

DIR

Number of directory blocks required.

RECFM

Record format:

A

ASA printer characters

B

Blocked

D

Variable length ASCII records

F

Fixed

M

Machine control character

S

Standard blocks or spanned

T

Track overflow

U

Undefined

V

Variable

LRECL

Logical record length (0 to 32760 value).

LIBRARY

Allocate a PDSE (partitioned data set extended) data set.

UNIT

Device type to which a file or data set is to be allocated.

SYSOUT

Data set is to be system output data set. The class can optionally be specified as a single character.

PGM

SYSOUT program name.

DEST

DEST node and user ID.

Just the user ID can be used for local destinations.

DUMMY

Allocate dummy data set.

FREE

Deallocation specification:

CLOSE

Requests that the system deallocate the data set when it is closed.

END

Requests that the system deallocate the data set at the end of the last step that references the data set. This value is the default.

The following syntax rules apply:

- DSN is mutually exclusive with the following parameters:
  - PGM
  - DEST
- DEST is mutually exclusive with the PGM parameter.
- 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 or PDSE was allocated, the member exists and can be opened for read.

1

The allocation was successful. However, a non-existing member of a PDS or PDSE was allocated which cannot be opened for read. If the member is opened for read, a system abend S013 occurs.

4

Allocation failed. An explanation of the error is written to the IDITRACE DDname. Allocation or concatenation errors relating to data sets, other than the first data set name specified, will not terminate the IDIALLOC command; processing continues.

8

Command syntax error. An explanation of the error is written to the IDITRACE DDname.

Example

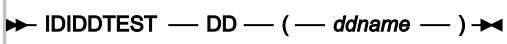
Figure 252. IDIALLOC command example

```
/* 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)"
if rc = 0 then do /* Check rdr allocation */
----processing----
end
else "IDIWTO INTRDR FAILED ALLOCATION"

/* Allocate a list of load libraries to the IDIRLOAD DDname */
"IDIALLOC DD(IDIRLOAD) DSN(MY.LOADLIB1 MY.LOADLIB2 MY.LOADLIB3) SHR"

/* Allocate a new PDSE history file data set named MY.HIST */
"IDIALLOC DD(DD6) DSN(MY.HIST) NEW CATALOG LIBRARY,
          RECFM(VB) LRECL(10000) SPACE(20,20)"
```

IDIDDTEST command

The IDIDDTEST command can be used to test if a DDname is allocated.

Figure 253. Syntax



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

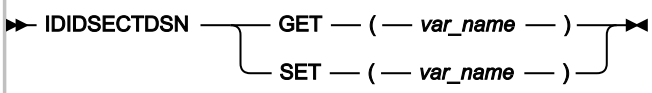
Figure 254. IDIDDTEST command example

```
/* REXX */

/* Test if the DDname DD1 is allocated */
"IDIDDTEST DD(DD1)"
if RC = 0 then say 'DD1 is allocated'
```

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 581) to ensure that the correct DSECT mapping is used for a given version of a product, for example CICS®.

Figure 255. Syntax



where:

GET(*var_name*)

Specifies that the current IDIDSECT data set concatenation is returned in the REXX variable *var_name*. The data set names in the returned list are blank delimited.

SET(*var_name*)

Specifies the name of a REXX variable containing the list of data set names that replace the current IDIDSECT concatenation. Multiple data set names must be separated by one or more blanks.

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

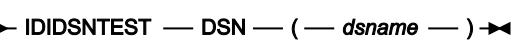
Figure 256. IDIDSECTdsn command example

```
/* 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
```

IDIDSNTEST command

The IDIDSNTEST command can be used to test if a data set exists.

Figure 257. Syntax



Parameters

dsname

Specifies the fully qualified name of the data set to be tested for existence. Do not enclose the data set name in single quotes.

Return codes

The IDIDSNTEST command provides the following return codes:

0

The specified data set exists.

4

The specified data set does not exist.

8

Command syntax error. An explanation of the error is written to the IDITRACE DDname.

Example

Example

Figure 258. IDIDSNTEST command example

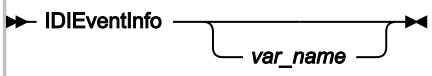
```
/* REXX */

/* Test if the data set MY.HIST is allocated */
"IDIDSNTEST DSN(MY.HIST)"
if RC = 0 then say 'MY.HIST exists'
```

IDIEventInfo command

The IDIEventInfo command can be used to by a Formatting user exit to obtain information about any event in the current fault.

Since the UFM data area only contains one set of fields for PSW, registers, and so on, it is necessary to use the IDIEventInfo command to change the values in the UFM data area from the values representing one event to those of another.

Figure 259. Syntax



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

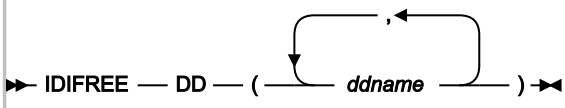
Figure 260. IDIEventInfo command example

```
/* REXX */

/* Process all events */
do event=1 to UFM.NUM_EVENTS
  "IDIEventInfo event"
  if RC<>0 then iterate
  ⋮
end
```

IDIFREE command

The IDIFREE command can be used to unallocate (free) DDnames that might have been allocated using IDIALLOC.

Figure 261. Syntax



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

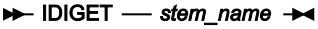
Figure 262. IDIFREE command example

```
/* REXX */

/* Free the DDnames DD1 and DD2 */
"IDIFREE DD(DD1,DD2)"
if RC = 0 then say 'Success!'
```

IDIGET command

You can use the IDIGET command to retrieve the content of a REXX stem variable from the current fault entry during reanalysis processing. This might be useful if, for example, a Formatting user exit saved information using the IDIPUT command during real-time processing.

Figure 263. IDIGET command syntax



Parameters

stem_name

Specifies the name of a REXX stem variable. The name of the stem variable must be the same as the name used with IDIPUT.

Return codes

The IDIGET command provides the following return codes:

0

Successful completion.

4

An error occurred.

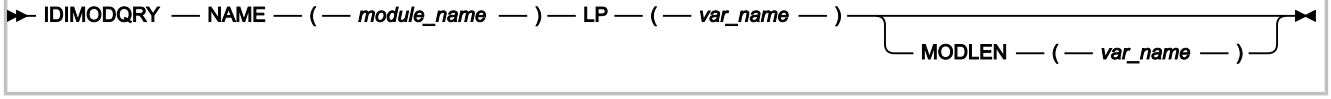
Example

```
/* REXX */

IDIGET xyz
if rc = 0 then do i = 1 to xyz.0
    say xyz.i
end
exit 0
```

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.

Figure 264. Syntax



where:

NAME(*module_name*)

The name of the module for which the load point, and optionally module length, are being requested.

LP(*var_name*)

The name of a REXX variable to hold the character format of the module load point.

MODLEN(*var_name*)

The name of a REXX variable to hold the character format of the hexadecimal module length. This value is optional.

Return codes

The IDIModQry command provides the following return codes:

0

Requested module found.

4

Requested module not found.

12

Syntax error.

Example

Figure 265. IDIModQry command example

```
/* REXX */

"IDIMODQRY NAME("IDISCBL1") LP(addr) MODLEN(len)"
if rc = 0 then "note 'Module IDISCBL1 at address" addr "length" len"'"
```

IDIPUT command

You can use the IDIPUT command to save the content of a REXX stem variable in the current fault entry. This might be useful if, for example, a Formatting user exit obtains information related to an abend that is not otherwise captured by Fault Analyzer, such as the on duty systems operator.

Figure 266. IDIPUT command syntax

Parameters

stem_name

Specifies the name of a REXX stem variable.

Return codes

The IDIGET command provides the following return codes:

0

Successful completion.

4

An error occurred.

Example

```
/* REXX */
mydata.0 = 1
mydata.1 = "Duty operator: Fred"
IDIPUT mydata
exit 0
```

IDIRegisterFaultEntry command

The IDIRegisterFaultEntry command can be used to register a fault entry at any time during analysis of an MVS™ dump data set, for example a CICS® system dump. This command allows the early creation of a fault entry in a history file, without the need to first exit interactive reanalysis. An installation can choose to automate the registration fault entry creation by issuing the IDIRegisterFaultEntry command from, for example, an Analysis Control user exit, or, alternatively, users can invoke a Formatting user exit on demand to issue this command during interactive reanalysis.

Note:

1. This command should not be used to register fault entries in a dump registration Analysis Control or Notification user exit, since doing so could result in extra unwanted fault entries being created. The IDIXTSEL dump registration processing automatically creates a fault entry, even if no user exit is used.
2. As an alternative to using the 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 611.

Figure 267. Syntax

where:

IDIHIST(*history_file_dsn*)

Specifies the history file in which the registration fault entry should be created. For interactive reanalysis, if this parameter is not specified, or if the user does not have UPDATE access to the specified history file, then a prompt is issued to allow the specification of the history file to be used.

For batch reanalysis, this parameter must be specified, otherwise, RC=4 is issued.

Return codes

The 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

Figure 268. IDIRegisterFaultEntry command example

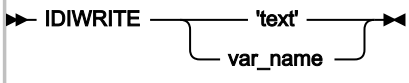
```
/* REXX */

/* 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
```

See Sample Formatting user exits on page 493 for another 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, Formatting, or Notification user exit. The type of data that can be provided via the IDIWRITE command is subject to the type of the user exit from which it is used. For more information, see the general section about each of the user exit types.

Figure 269. Syntax



where:

'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'`.

Using a quoted string is only supported in a Notification user exit or a Formatting user exit.

var_name

The name of a variable containing the data record.

If the IDIWRITE command is used without *var_name*, data records must be passed using the exit-specific data area as described for the relevant exit types.

Return codes

The IDIWRITE command provides the following return codes:

0

The record was written successfully.

2

Writing of records has been disabled due to previous errors.

4

The record was not written due to errors. An explanation of the error is written to the IDITRACE DDname.

8

Command syntax error. An explanation of the error is written to the IDITRACE DDname.

Example

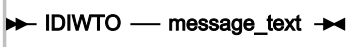
Figure 270. IDIWRITE command example

```
/* REXX */
/* Pass records to Fault Analyzer */

/* Method 1 - plain text */
"IDIWRITE 'This is record 1'"
if RC = 0 then say 'Method 1 success!'

/* Method 2 - using LST data area (Compiler Listing Read user exit) */
rec = 'This is record 2'
LST.DATA_LENGTH = length(rec)
LST.DATA_BUFFER = rec
"IDIWRITE"
if RC = 0 then say 'Method 2 success!'

/* Method 3 - letting REXX resolve data record variable */
rec = 'This is record 3'
"IDIWRITE '"rec"'"
if RC = 0 then say 'Method 3 success!'

/* Method 4 - letting Fault Analyzer resolve data record variable */
rec = 'This is record 4'
"IDIWRITE rec"
if RC = 0 then say 'Method 4 success!'
```

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).

Figure 271. Syntax



New-line characters (X'15') in the message text can be used to split long messages into multiple WTOs. Any other non-printable characters are changed to periods.

Return codes

The IDIWTO command always completes with RC=0.

Example

Figure 272. IDIWTO command example

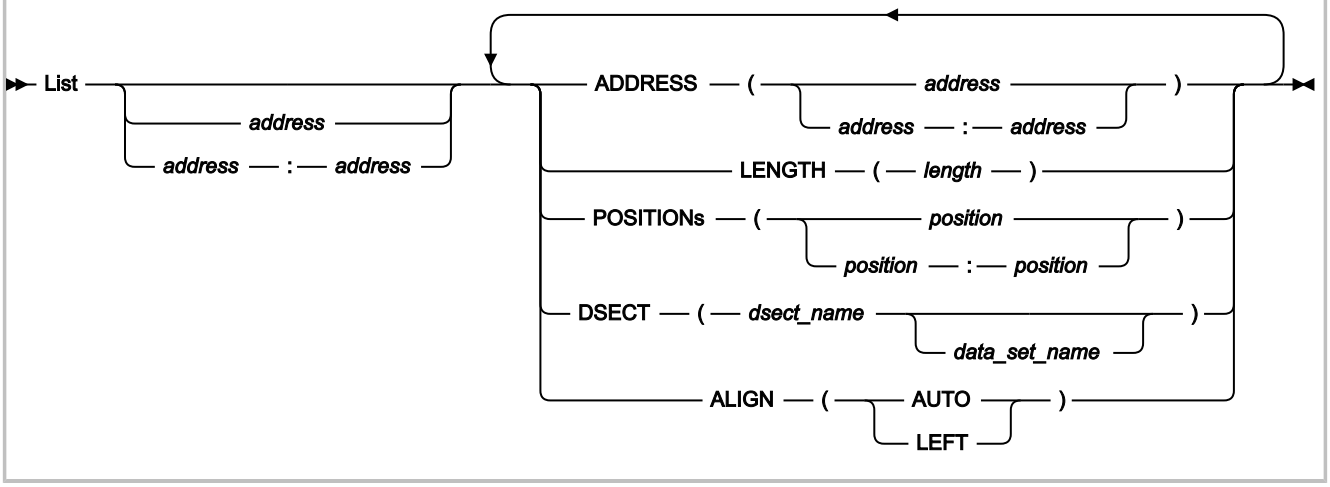
```
/* REXX */

/* Write a message to the MVS console */
"IDIWTO Minutes since last duplicate fault =" EPC.MINUTES_SINCE_LAST_DUP
```

To avoid possible confusion, it is recommended that any WTO messages that are issued do not include a message ID that could be mistaken for one of the formal messages issued by Fault Analyzer. If you want to prefix message IDs with *"IDI"*, then you might add an extra character to eliminate the chance of matching a Fault Analyzer message, for example, *"IDIX"* followed by a number.

List command

The List command can be used to by a Formatting user exit to print storage areas from the analyzed fault environment.

Figure 273. Syntax



Parameters

address start:end

Specifies either the start address, or an address range, as a positional parameter. This address is an alternative to using the ADDRESS keyword. See *ADDRESS* below for details about valid syntax.

ADDRESS(address) ADDRESS(address:address)

Specifies the address of the storage to be printed. Specify either a single address or as an address range.

The address parameter is 64-bit enabled. For details, see Specifying 64-bit addresses on page 142. The address may optionally followed by a period (for example, "000176C0.").

LENGTH(length)

Specifies the number of bytes to be printed. Specify as a hexadecimal value (indicated by X'...') or as a decimal value (no indication required). If an address range is specified, then specification of LENGTH is ignored.

POSITIONs(position) POSITIONs(position:position)

Specifies the offset from the start address to the first byte of storage to be printed. Specify as either a single offset or as an offset range. Both POSITION and POSITIONS are accepted.

All offsets must be either hexadecimal (indicated by X'...') or signed decimal (optionally indicated by F'...').

DSECT(dsect_name) DSECT(dsect_name data_set_name)

Specifies the name of a DSECT mapping member to be used when formatting storage at the requested address.

If a data set name is not specified, then the DSECT must be available through the IDIDSECT DDname (for details, see DataSets on page 581). Otherwise, the DSECT must exist in the specified data set name.

ALIGN(AUTO | LEFT)

Specifies the hexadecimal data alignment:

AUTO

The alignment is determined by the listed address and the available report width. This is the default.

LEFT

The hexadecimal data is left-aligned unconditionally.

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 223.

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

Figure 274. List command example

```
/* REXX */

"LIST 0 LENGTH(128)"
```

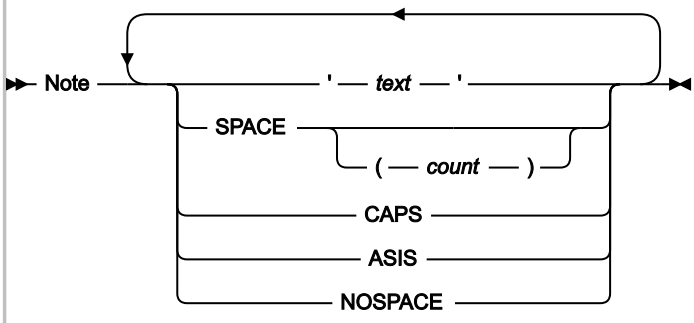
The above example might produce the following output:

```
Address  Offset    Hex                                EBCDIC
00000000           040C0000 810692C8 00000000 00000000 *....a.kH........*
00000010       +10 00FC7F08 00000000 070C1000 85532492 *.."........e..k*
00000020       +20 078D0000 00FC7F5A 078D1000 8775BB42 *......"!....g...*
00000030       +30 00000000 00000000 070C0000 85532496 *............e..o*
00000040       +40 00000000 00000000 00000000 00FC7F08 *............."."*
00000050       +50 00000000 00000000 040C0000 810676D0 *............a..}*
00000060       +60 040C0000 80FFB080 00080000 BF286880 *................*
00000070       +70 00080000 BF287940 040C0000 81068A00 *......` ....a...*
```

Note command

The Note command can be used to by a Formatting user exit to print a line of text.

Figure 275. Syntax



Parameters

'text'

Specifies the text to be printed, enclosed in apostrophes. Either single quotes (') or double quotes (") are permitted, as long as the same type is used as both the first and the last character.

Any quotes within the text, of the same type as used to enclose the string, must be specified twice. For example, to print

```
It isn't there
```

specify the text as either

```
"It isn't there"
```

or

```
'It isn''t there'
```

If no text is specified, then no data is printed. However, any blank lines specified using the SPACE parameter are still written.

SPACE SPACE(count)

Specifies the number of blank lines to be written ahead of the next text line.

If the SPACE parameter is specified without *count*, then it defaults to 1.

CAPS

Specifies that text is written in all uppercase characters.

ASIS

Specifies that text is written "as is" without any uppercasing being performed.

NOSPACE

Specifies that no blank lines are written ahead of the next text line. This value is the default.

If a parameter is specified multiple times, then only the last specification takes effect.

Return codes

The Note command provides the following return codes:

0

Command completed successfully.

12

Command syntax error. An explanation of the error is written to the IDITRACE DDname.

Example

Figure 276. Note command 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)"
```

The above example produces the following output:

```
This is a simple note.
This note follows the previous without any blank lines inserted.


This note has 2 blank lines ahead of it.
```

Formatting tags

The following describes the tags that are available to a Formatting user exit when formatting data for the report. The tags provide a way to create headings, lists, and so on for the displayed data using HTML-like syntax. The tag stream is passed back to Fault Analyzer from the Formatting user exit using the IDIWRITE command.

The following example showing the use of the formatting tags is also provided in softcopy format as member IDISUFM3 in data set IDI.SIDISAM1:

Figure 277. Sample REXX Formatting user exit 3 source

```
"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
```

Formatted, the above might appear as follows (point-and-shoot fields and highlighted text shown in bold style):

Figure 278. Sample REXX Formatting user exit 3 output

```
  File  View  Services  Help
──────────────────────────────────────────────────────────────────────────────
Interactive Reanalysis Options                                 Line 1 Col 1 80
Command ===> ____________________________________________ Scroll ===> CSR
JOBNAME: P35777    SYSTEM ABEND: 0C7            FAE1      2019/10/31  22:51:13

First paragraph.

     Second paragraph, indented 5 characters from the first. This <P> tag is
     treated as text only.
     Third paragraph. Note that this paragraph is not preceded by a blank
     line.

Fourth paragraph - now we are back at the left margin.
************************************ This line will not wrap at the prefer

Previous area and 00625F22 are both point-and-shoot fields to the Dump Storage
display for address 00625F22 in the interactive reanalysis report.

This is a long definition
term. . . . . . . . . . . . . : This is the matching definition description
                               which might wrap depending on the preferred
                               formatting width.

A shorter definition term . : The definition description of the second term.

Address 0 storage for a length of 32 bytes:
Address  Offset   Hex                                EBCDIC / ASCII
00000000          040C0000 810692C8 00000000 00000000 *....a.kH........*
00000010       +10  00FC7F08 00000000 070E0000 00000000 *..".............*

o   In an unordered list, each item is preceded by a bullet. If necessary,
    the item description will wrap at the preferred formatting width.

o   Another item in the same list.

Notes:

1.  In a note list, each note is numbered and the list is preceded by a
    "Notes:" heading.  If necessary, the note description will wrap at the
    preferred formatting width.

2.  Another note in the same list.

 Column Column
 1      2
    123     17
```

General rules for the formatting tags:

- All blanks are significant, except at the beginning and end of lines in a paragraph, and at the beginning and end of definition descriptions (text preceded by the <DD> tag).
- Text, including blank characters, that is not preceded by any tag implicitly causes a <P> tag to be inserted ahead of the text.
- All tags and attributes are non-case-sensitive.
- The maximum line width of any output is 132 characters. Beyond this, the text wraps.

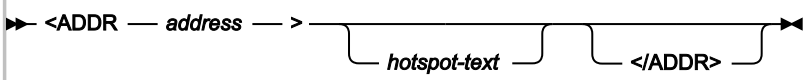
The following explains each tag in detail.

ADDR (address)

The ADDR tag defines an address point-and-shoot field.

When the address field is displayed in an interactive reanalysis report, the user can place the cursor on the field and press Enter to invoke the Dump Storage display at the specified address.

Figure 279. Syntax



address

The hexadecimal address to be displayed when selecting the point-and-shoot field.

The address parameter is 64-bit enabled. For details, see Specifying 64-bit addresses on page 142.

hotspot-text

The point-and-shoot field text to be displayed. If not specified, then the address is used.

Description

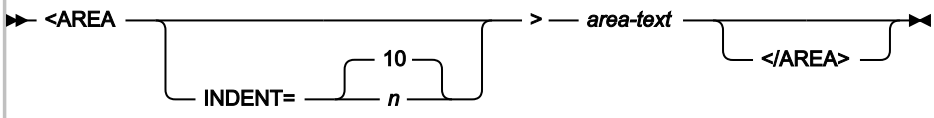
The ADDR tag does not cause a line break.

In the interactive reanalysis report, the point-and-shoot field text is presented in yellow.

AREA (area)

The AREA tag defines a section of a display. It is used to control indentation of text beyond what the default indentation provided for the higher level tags provide.

Figure 280. Syntax



INDENT

This attribute specifies the number of characters by which the current indentation is incremented.

area-text

These are tags and text for the display section.

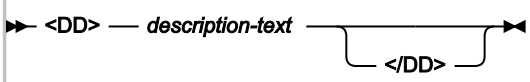
Description

The AREA tag can be nested within another AREA tag. Any other tags are implicitly ended.

DD (definition description)

The DD tag defines the description of a term in a definition list.

Figure 281. Syntax



description-text

This value is the term description.

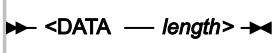
Description

See DL (definition list) on page 543.

DATA (data)

The DATA tag defines the number of characters that follow the tag for which no tag processing is to be performed. This tag is generally only used if the text to be written contains characters that might otherwise be misinterpreted as formatting tags.

Figure 282. Syntax



length

The number of characters in the input stream immediately following the DATA tag that should be treated as textual data regardless of any possible tag contents.

Description

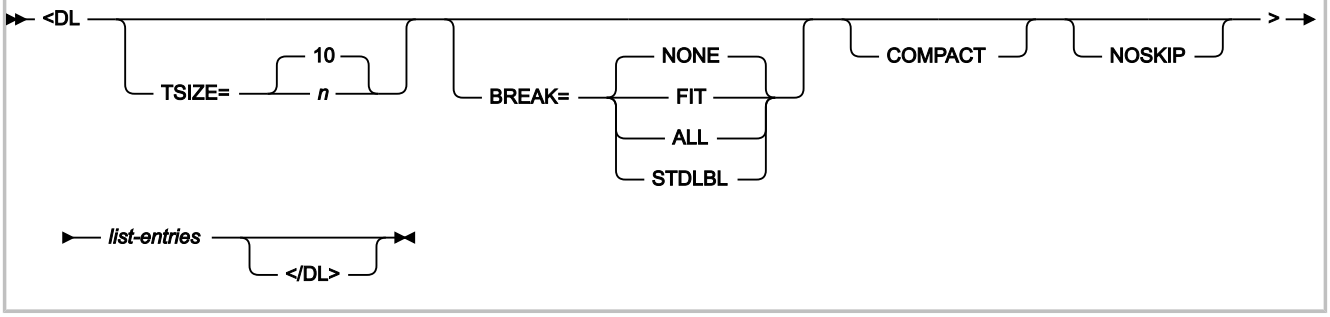
Text that might contain valid tags can be protected from causing formatting problems by using the DATA tag to specify the number of characters that should be treated as text only.

Note: The use of tag delimiters ('<' or '>') in text that is not preceded by a DATA tag is permitted as long as no valid tag syntax occurs. Invalid and incomplete tags are returned to the input stream as textual data.

DL (definition list)

The DL tag defines a list of terms and their corresponding definitions.

Figure 283. Syntax



TSIZE

This attribute specifies the indentation of the definition description. The minimum value is 3 characters and the default value is 10 characters.

BREAK

This attribute controls the formatting of the definition terms and descriptions:

- If BREAK=NONE, then the term is on the same line as the description, spilling into the description area if the length exceeds TSIZE. This value is the default.
- If BREAK=FIT, then the description is on the line below the term if the term exceeds the TSIZE value.
- If BREAK=ALL, then every definition is on the line below the term.
- If BREAK=STDLBL, then a standard Fault Analyzer label is used. If the term length exceeds the TSIZE value in effect, then the text is wrapped within the TSIZE width. Trailing periods and a colon are added to the last term line.

COMPACT

This attribute causes the list to format without a blank line between the items in the list.

NOSKIP

This attribute causes the list to format without creating a blank line before the first line of the list.

list-entries

These are the DT and DD tags that make up the list entries.

Description

In the interactive reanalysis report, the list entries are presented in white. However, if the STDLBL attribute is used, then the definition terms are in green.

DT (definition term)

The DT tag defines a term in a definition list.

Figure 284. Syntax

term-description

This value is specified via the DD tag.

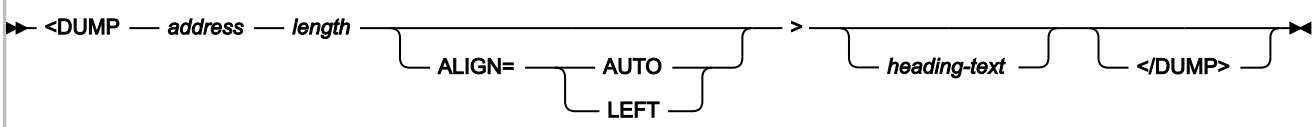
Description

See DL (definition list) on page 543.

DUMP (EBCDIC dump)

The DUMP tag inserts a hex-dump display in-line in the formatted output, as opposed to the ADDR tag which only shows the address itself. This insertion is particularly useful when formatting for the batch report, since the user is not otherwise able to view the storage.

Figure 285. Syntax



address

Hexadecimal address of storage area to be displayed.

The address parameter is 64-bit enabled. For details, see Specifying 64-bit addresses on page 142.

length

Hexadecimal length (in bytes) of storage area to be displayed.

ALIGN = AUTO | LEFT

Specifies the hexadecimal data alignment:

AUTO

The alignment is determined by the listed address and the available report width. This is the default.

LEFT

The hexadecimal data is left-aligned unconditionally.

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.

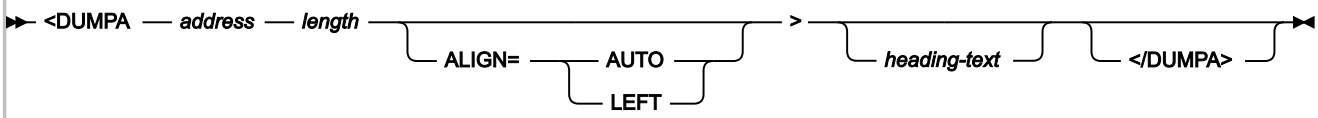
The character-interpreted section on the right-hand side of the display is based on EBCDIC-encoded hexadecimal values. If the data is known to contain ASCII-encoded values, then use the DUMPA tag instead (see DUMPA (ASCII dump) on page 546).

In the interactive reanalysis report, the heading text is presented in white.

DUMPA (ASCII dump)

The DUMPA tag inserts a hex-dump display in-line in the formatted output, as opposed to the ADDR tag which only shows the address itself. This insertion is particularly useful when formatting for the batch report, since the user is not otherwise able to view the storage.

Figure 286. Syntax



address

Hexadecimal address of storage area to be displayed.

The address parameter is 64-bit enabled. For details, see Specifying 64-bit addresses on page 142.

length

Hexadecimal length (in bytes) of storage area to be displayed.

ALIGN = AUTO | LEFT

Specifies the hexadecimal data alignment:

AUTO

The alignment is determined by the listed address and the available report width. This is the default.

LEFT

The hexadecimal data is left-aligned unconditionally.

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 545).

In the interactive reanalysis report, the heading text is presented in white.

HP (highlighted phrase)

The HP tag identifies text to be displayed with highlighted emphasis.

Note: Highlighting is only available in the interactive reanalysis report.

Figure 287. Syntax

text-to-be-highlighted

This text displays with highlighted emphasis.

Description

This tag does not cause a line break.

In the interactive reanalysis report, the text is presented in turquoise.

L (line)

The L tag defines a line of text that is not subject to the user preferred display width.

Figure 288. Syntax

line-text

The line text.

Description

Each line formats as an unindented non-flowing line of text. No blank lines are added before or after the line text.

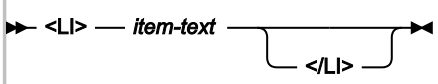
The width of the line is determined by the maximum line width of the display. No line should exceed this limit as it then flows onto the following line.

In the interactive reanalysis report, the line text is presented in white.

LI (list item)

The LI tag defines a list item within a note list or an unordered list.

Figure 289. Syntax



item-text

The item text.

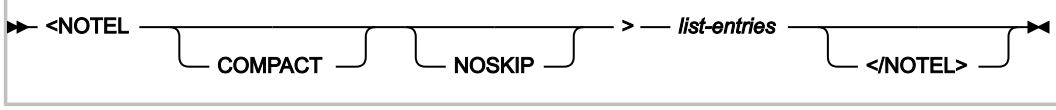
Description

In the interactive reanalysis report, the line text is presented in white.

NOTEL (note list)

The NOTEL tag defines a list of notes.

Figure 290. Syntax



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.

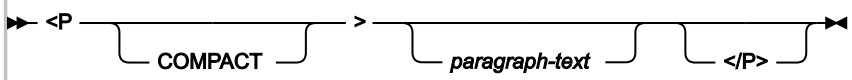
Description

This tag causes a line break.

P (paragraph)

The P tag defines a paragraph of text.

Figure 291. Syntax



COMPACT

This attribute causes the paragraph to format without a blank line before the paragraph.

paragraph-text

The text of the paragraph.

Description

Each paragraph formats as an unindented flowing block of text. A blank line is added before the paragraph unless the COMPACT attribute is specified.

The width of the displayed paragraph is determined by the preferred formatting width specified by the user via the View pull-down menu.

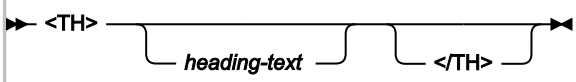
Paragraphs within a list align with the text of the list item.

In the interactive reanalysis report, the paragraph text is presented in white.

TH (table heading)

The TH tag defines a table heading line.

Figure 292. Syntax



heading-text

The table heading line.

Description

Each table heading line formats as an unindented non-flowing line of text. No blank lines are added before or after the line text.

The width of the line is determined by the maximum line width of the display. No line should exceed this limit as it then flows onto the following line.

In the interactive reanalysis report, the table heading line text is presented in blue.

U (underline)

The U tag identifies text to be displayed underlined.

Note: Underlining is only available in the interactive reanalysis report.

Figure 293. Syntax

text-to-be-underlined

This text displays underlined.

Description

This tag does not cause a line break.

In the interactive reanalysis report, the underlined text is presented in blue.

UL (unordered list)

The UL tag defines an unordered list.

Figure 294. Syntax

COMPACT

This attribute causes the list to be formatted without a blank line between the list items.

NOSKIP

This attribute causes the list to format without creating a blank line before the first line of the list.

list-entries

These are specified using the LI tag.

Description

This tag causes a line break.

The IDIXUFMT load module Formatting user exit

The IDIXUFMT load module Formatting user exit (in the following topics referred to as the IDIXUFMT exit) is a special type of user exit. It does not follow the normal rules that apply to other user exits described in this chapter, whether in REXX or in load module format.

The IDIXUFMT exit cannot be specified though the Exits option, but is located by load module name. If a load module with the name IDIXUFMT is found in an APF-authorized library during Fault Analyzer execution, then it is invoked during the formatting of the analysis report. The IDIXUFMT exit is called immediately before any other Formatting user exits specified through the Exits option.

The IDIXUFMT exit:

- Must be LE-compliant.
- Must not contain a "main" function.
- Must be link-edited with the NORENT option.

Fault Analyzer invokes the IDIXUFMT exit by way of the IDIXMFMT entry point, which is contained within IDIXLFMT, the non-executable load module that is provided with Fault Analyzer. The exit user code is invoked by way of entry point IDIXUFMT.

Entry specifications

The user code IDIXUFMT entry point is invoked with:

- R1 pointing to two fullwords:
  - The first fullword is the address of the ENV data area (see ENV - Common exit environment information on page 653).
  - The second fullword is the address of the UFM data area (see UFM - Formatting user exit parameter list on page 674).
- R13 pointing to save area.
- R14 containing the return address.
- R15 containing the IDIXUFMT entry point address.

Return specifications

On return from the IDIXUFMT entry point:

- R0 and R1 are undefined.
- R2 through R14 must be unchanged.
- R15 is undefined.

Sample IDIXUFMT exits

Two sample IDIXUFMT exits are provided, along with JCL to compile (or Assemble) and link:

- A sample C IDIXUFMT exit is provided in IDI.SIDISAM1(IDIXUFMC)
- A sample Asssembler IDIXUFMT exit is provided in IDI.SIDISAM1(IDIXUFMA)

IDIXUFMT functions

There are many Fault Analyzer functions available for use from within the IDIXUFMT exit.

All functions have uppercase names, consisting of 8 characters, or less. These basic names permit an exit that is written in, for example High Level Assembler, to call these functions.

All functions use C linkage.

The following topics describe these functions.

IDIXDLOC – Locate dump storage using system buffering

Format

```
#include "idixufmh.h"

int IDIXDLOC(int addr, int len);
```

General description

The IDIXDLOC() function is used to access storage in the analyzed address space by virtual address and length.

Note: To prevent out-of-storage conditions when accessing large amounts of virtual storage, use the IDIXXLOC function instead. For details, see IDIXXLOC – Locate dump storage using own buffering on page 558.

For real-time processing, the storage that is accessed is generally that of the actual address space which is being analyzed. For reanalysis, the storage is obtained from the minidump.

Regardless of real time or reanalysis mode, do not dereference storage areas that are not obtained by calling the IDIXDLOC() function, since protection exceptions might otherwise occur.

IDIXDLOC() is functionally equivalent to the Fault Analyzer REXX command "Evaluate".

Returned value

If the storage is available for the requested length, then IDIXDLOC() returns the address of the storage area.

IDIXDLOC() returns a negative value if the requested address is available, but for a length that is less than the requested length. The available length can be determined by subtracting the returned value from 0. To obtain the address of the partial storage area, call IDIXDLOC() again with the reduced length.

A zero value is returned if the requested address is not available, regardless of length.

Example

```
#include "idixufmh.h"

int maddr, cvt;
maddr = IDIXDLOC(16,4);  /* Get address of CVT pointer /
if (maddr > 0)
  cvt = (int )maddr;     /* Get CVT pointer */
```

IDIXEINF – Obtain event information

Format

```
#include "idixhfmt.h"

int IDIXEINF(UFM *p_ufm, int event_no);
```

General description

The specified UFM data area is populated with information applicable to the specified event number.

IDIXEINF() is functionally equivalent to the Fault Analyzer REXX command "IDIEventInfo."

Returned value

IDIXEINF() returns zero if the information was retrieved successfully.

IDIXEINF() returns non-zero if no information is available for the specified event number. An explanation of the error is written to the IDITRACE DDname.

Example

```
#include "idixhfmt.h"

UFM ufm;
int rc;
rc = IDIXEINF(&ufm, 1);
If (!rc) {  // Successful completion
  …
}
```

IDIXGETN – Get data area decimal character field value

Format

```
#include "dixhfmt.h"

#define IDIXGETN(pSrc) \
        (IDIXGETN)(pSrc, sizeof(pSrc))
int     (IDIXGETN)(char *pSrc, int src_len);
```

General description

IDIXGETN() is used to convert a data area character field that contains decimal characters to an "int" value.

IDIXGETN() is a macro that calls a function by the same name. The advantage of using the macro is that only a single argument is required.

Returned value

If successful, IDIXGETN() returns the converted signed "int" value, represented in the string. If unsuccessful, it returns an undefined value.

Example

```
#include "idixhfmt.h"

int num_events;
UFM ufm;
…
num_events = IDIXGETN(ufm.NUM_EVENTS);
```

IDIXGETS - Get data area character field as C string

Format

```
#include “idixhfmt.h"

#define IDIXGETS(pSrc) \
        (IDIXGETS)(pSrc, sizeof(pSrc))
char *  (IDIXGETS)(char *pSrc, int src_len);
```

General description

IDIXGETS() is used to create a NULL-terminated string from a data area character field. For data area fields not in buffered data format, the string is created in the LE HEAP storage and is automatically free'd when the IDIXUFMT exit processing finishes. The caller of IDIXGETS() must not free the storage area returned. For data area fields in buffered data format, the existing buffer address is returned.

It is not possible to update a data area field by modifying the string that is returned by IDIXGETS().

IDIXGETS() is a macro, which calls a function by the same name. The advantage of using the macro is that only a single argument is required.

Returned value

Returns the address of the requested data area field as a NULL-terminated string.

Example

```
#include “idixhfmt.h"

UFM ufm;
char *psz;
…
psz = IDIXGETS(ufm.EVENT_TYPE);
If (strlen(psz) >= 6 && memcmp(psz,"Abend ") == 0) {  // Abend event
  …
}
```

IDIXGETX – Get data area hexadecimal character field value

Format

```
#include "idixhfmt.h"

#define IDIXGETX(pSrc) \
        (IDIXGETX)(pSrc, sizeof(pSrc))
int     (IDIXGETX)(char *pSrc, int src_len);
```

General description

IDIXGETX() is used to convert a data area character field that holds hexadecimal characters (0-9 or A-F) to an "int" value.

IDIXGETX() is a macro, which calls a function by the same name. The advantage of using the macro is that only a single argument is required.

Returned value

If successful, IDIXGETX() returns the converted signed "int" value, represented in the string. If unsuccessful, it returns an undefined value.

Example

```
#include "idixhfmt.h"

int pgm_len;
UFM ufm;
…
pgm_len = IDIXGETX(ufm.PROGRAM_LENGTH);
```

IDIXLIST – Print storage area in report

Format

```
#include "idixhfmt.h"

int IDIXLIST(int addr, int len);
```

General description

IDIXLIST() can be used to print storage areas from the analyzed fault environment. The addr argument must be a virtual dump address.

The format of the printed storage area includes a heading, and both hexadecimal and EBCDIC/ASCII representation of the storage, as shown in the following example:

```
Address  Offset    Hex                                 EBCDIC / ASCII
00000000            040C0000 810692C8 00000000 00000000 *....a.kH........*
00000010       +10  00FC7F08 00000000 070C1000 85532492 *..".........e..k*
00000020       +20  078D0000 00FC7F5A 078D1000 8775BB42 *......"!....g...*
00000030       +30  00000000 00000000 070C0000 85532496 *............e..o*
00000040       +40  00000000 00000000 00000000 00FC7F08 *..............".*
00000050       +50  00000000 00000000 040C0000 810676D0 *............a..}*
00000060       +60  040C0000 80FFB080 00080000 BF286880 *................*
00000070       +70  00080000 BF287940 040C0000 81068A00 *......` ....a...*
```

IDIXLIST() is functionally equivalent to the Fault Analyzer REXX command "List".

Returned value

IDIXLIST() always returns zero.

Example

```
#include "idixhfmt.h"

IDIXLIST(0,128);  // Show address 0 for a length of 128 bytes
```

IDIXNOTE – Write simple line of text to report

Format

```
#include "idixhfmt.h"

int IDIXNOTE(char *psz, ...);
```

General description

IDIXNOTE() writes a line of unformatted text in the report. The psz argument must point to a NULL-terminated string. (To write formatted text, use the IDIXWRIT() function instead.)

If the psz argument is a format string suitable for use by the C sprintf() function, then more required arguments can follow.

This function is equivalent to the Fault Analyzer REXX command "Note".

Returned value

IDIXNOTE() always returns zero.

Example

```
#include "idixhfmt.h"

int i;

IDIXNOTE("Important data follows");
…
IDIXNOTE("A total of %d entries listed.",i);
```

IDIXTRCE – Write simple line of text to IDITRACE

Format

```
#include "idixhfmt.h"
```

```
int IDIXTRCE(char *psz, ...);
```

General description

IDIXTRCE() can be used to perform the following functions:

- Control the tracing of IDI* function calls by writing information to the IDITRACE DDname about passed parameters and final return code.
  - If the psz arg is specified as "(char)-1", then IDI* function call tracing is enabled.
  - If the psz arg is specified as "NULL", then IDI* function call tracing is disabled
- Write a line of unformatted text to the IDITRACE DDname. The psz argument must point to a NULL-terminated string. This mode of the function is equivalent to using the REXX command "SAY" from within a Fault Analyzer REXX user exit.

If the psz argument is a format string suitable for use by the C sprintf() function, then extra required arguments can follow.

Returned value

IDIXTRCE() always returns zero.

Example

```
#include "idixhfmt.h"

int i = 5;
IDITRACE((char *)-1);  /* start IDI* function tracing */
IDIXTRCE("Couldn't format data area ABC.");
IDIXTRCE("A total of %d control blocks formatted.", i);
```

IDIXWRIT – Write formatted text to report

Format

```
#include "idixhfmt.h"

int IDIXWRIT(char *psz, ...);
```

General description

The IDIXNOTE() function can be used only to write a simple line of text to the report. However, the string that is passed to the IDIXWRIT() function in the psz argument can contain formatting tags (see Formatting tags on page 539).

If the psz argument is a format string suitable for use by the C sprintf() function, then more required arguments can follow.

This function is equivalent to the Fault Analyzer REXX command "IDIWRITE".

Returned value

IDIXWRIT() always returns zero.

Example

```
#include "idixhfmt.h"

int i;
IDIXWRIT("<DL>");
for (i = 0; i < 10; ++i) {
  IDIXWRIT("<DT>Item %d</DT>",i);
  IDIXWRIT("<DD>Item description</DD>");
}
IDIXWRIT("</DL>");
```

IDIXWTO – Write message to MVS console

Format

```
#include "idixhfmt.h"

int IDIXWTO(char *psz, ...);
```

General description

IDIXWTO() is used to write a message to the MVS™ console. The psz argument must point to a NULL-terminated string.

If the psz argument is a format string suitable for use by the C sprintf() function, then more required arguments can follow.

This function is equivalent to the Fault Analyzer REXX command "IDIWTO".

Returned value

IDIXWTO() always returns zero.

Example

```
#include "idixhfmt.h"

IDIXWTO("Unable to complete IDIXUFMT exit processing!");
```

IDIXXLOC – Locate dump storage using own buffering

Format

```
#include "idixufmh.h"
int IDIXXLOC(void *bufptr, unsigned long long addr, int len);
```

General description

The IDIXXLOC() function is used to access storage in the analyzed address space by virtual address and length.

The address of a buffer must be specified in *bufptr*. The length of the buffer must be greater, or equal to, *len*.

For real-time processing, the storage that is accessed is generally that of the actual address space that is being analyzed. For reanalysis, the storage is obtained from the minidump or the extended dump data set.

Regardless of real time or reanalysis mode, do not dereference storage areas that are not obtained by calling the IDIXXLOC() function, because protection exceptions might otherwise occur.

IDIXXLOC() is functionally equivalent to the Fault Analyzer REXX command "Evaluate".

Returned value

If the storage is available for the requested length, then IDIXXLOC() returns the address of the provided buffer (*bufptr*). IDIXXLOC() returns a negative value if the requested address is available, but for a length that is less than the requested length. The available length can be determined by subtracting the returned value from 0. To obtain the address of the partial storage area, call IDIXXLOC() again with the reduced length.

A zero value is returned if the requested address is not available, regardless of length.

Example

```
#include "idixufmh.h"

int maddr, i;
char buffer[500];

maddr = IDIXXLOC(buffer, 0x001F0000, 500); /* Get 500 bytes of storage at addr 1F0000 */
if (maddr > 0)
  i = *(int *)maddr;                       /* Get first 4 bytes */
```

Interfacing with the ADFzCC Event Processing user exit

The ADFz Common Components product provides an Event Processing user exit feature, which permits the processing of data by an asynchronous installation-written back-end. To use this feature, install ADFz Common Components V1.8. For more information, see *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*.

You can use this feature via the Notification user exit. For more information, see Notification user exit on page 498.

Samples provided by FA

To use the event processing feature, FA provides the following samples in `IDI.SIDISAM1`:

Sample Notification exit IDISXEPN

A generic Notification user exit that calls IDIWRITE for all available `ENV` and `NFY` data area variables. For more information about this sample exit, see Notification user exit on page 498.

Sample Event Processing user exit IDISRTCC

The sample exit written in C. The sample writes the data provided by IDISXEPN to a data set. Then, the sample dynamically allocates a DD named `IDIINTR` to the internal reader that submits a job to run a REXX exec (`IDISRTCR`) that uses the created data set as input. Additional samples in COBOL and PL/I are also provided as `IDISRTCB` and `IDISRTCP` respectively.

To use the sample IDISRTCC exit, it must be specified via the ADFzCC EventProcessingExit option. For details about this option, see *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*.

Sample REXX exec IDISRTCR

The sample REXX exec IDISRTCR is executed by the job that is created by `IDISRTCC`. This exec reads the provided input data set and uses its contents (which are the ENV and NFY data area variables provided by IDISXEPN) to build an RTC work item request with relevant information about the fault entry. Then, the request is sent to the RTC server that is specified within the script. A work item under the appropriate RTC project is created, which contains information about the fault that occurred.

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

The following topics describe installation requirements for optional interfaces that give access to Fault Analyzer history files from platforms other than TSO/ISPF.

Installing the IBM Fault Analyzer plug-in for Eclipse

The IBM Fault Analyzer for z/OS® plug-in for Eclipse is an optional feature.

To install the plug-in:

Host requirements

- IBM Application Delivery Foundation for z/OS® Common Components 1.11.0 or later
- IBM Fault Analyzer for z/OS® V17.1.0 or later

Client requirements

IBM Explorer for z/OS V3.4.0 or later or IBM Developer for z/OS V17 or later (Aqua 3.4)

Note: To get the most up-to-date information on supported operating systems, detailed system requirements, and hardware requirements, you can generate a dynamic software product compatibility report.

1. Go to the Software Product Compatibility Reports page.
2. Type the product name in the Search by product name field and click the product name in the list.
3. In the search results, select a release.

See Fault Analyzer plug-in for Eclipse on page 283 for information about the plug-in.

Customizing the IBM Application Delivery Foundation for z/OS® Common Components server

See *IBM Application Delivery Foundation for z/OS® Common Components Customization Guide and User Guide* for information about installing the ADFzCC server.

A sample Fault Analyzer configuration file for the ADFzCC server is provided as member IDIGSVRJ in data set IDI.SIDISAM1:

```
CONFIG=FA
SPAWN_PROGRAM=IDIGMAIN
SPAWN_REGIONSZ=500 ❶
SPAWN_JOBNAME=IDISVRF
SPAWN_PARMS_SECTION
ISPF_PROF_DSN=&USERID..ISPF.ISPPROF ❷
ISPF_APPL=IDI ❸
```

The parameters you might need to change are as follows:

❶

Increase SPAWN_REGIONSZ if downloading extremely large fault entry minidumps with more than 80,000 pages.

❷

Change &USERID..ISPF.ISPPROF if a different data set is used for ISPF profile members.

&USERID is replaced by the user ID of the connected user. For example, if the user ID is FRED, and the ISPF application ID in ❸ is IDI, then the ISPF profile is retrieved from FRED.ISPF.ISPPROF(IDIPROF).

To use only a part of the user ID as the substitution value, substring specification can be used (for details, see Symbol substring specification on page 588). For example, to use only the first 3 characters of the user ID as the high-level qualifier, specify &USERID(1:3)..ISPF.ISPPROF.

❸

Change IDI if a different ISPF application ID is used for Fault Analyzer. The default value is IDI.

Installing the IBM Fault Analyzer plug-in for Eclipse

The IBM® Fault Analyzer plug-in for Eclipse is a plug-in to IBM Explorer for z/OS® or IBM Developer for z/OS.

- Downloads are available from https://ibm.github.io/mainframe-downloads/index.html.
- Documentation for IBM Explorer for z/OS and IBM Developer for z/OS is available at https://www.ibm.com/docs/en/adfz.

Installing IBM Fault Analyzer for z/OS® on VS code

IBM Fault Analyzer for z/OS® on VS Code is an optional feature.

Host requirements

- IBM Application Delivery Foundation for z/OS® Common Components 1.11.0 or later
- IBM Fault Analyzer for z/OS® V17.1.0 or later

Client requirements

- IBM Application Delivery Foundation for z/OS® Common Components Microsoft Visual Studio Code extension (always use the same version as the IBM Fault Analyzer for z/OS® Microsoft Visual Studio Code extension)
- Zowe™ Explorer V3.2.1

Note: To get the most up-to-date information on supported operating systems, detailed system requirements, and hardware requirements, you can generate a dynamic software product compatibility report.

1. Go to the Software Product Compatibility Reports page.
2. Type the product name in the Search by product name field and click the product name in the list.
3. In the search results, select a release.

See Fault Analyzer on VS Code on page 283 for information about this feature.

Enabling interactive reanalysis under CICS

This feature is an optional feature. It allows the use of the Fault Analyzer ISPF interface to view history files and fault entries from a CICS® logon, without the need for a TSO logon.

Refer to Performing interactive reanalysis under CICS on page 284 for information about using this interface.

Note: This feature requires the IBM Application Delivery Foundation for z/OS Common Components (which includes the required interactive panel viewer) to be installed.

It is recommended that a special CICS® region be set up for running Fault Analyzer in this mode to control the CPU and storage usage, without affecting other normal CICS® transactions. The reason is that in this mode, the Fault Analyzer analysis programs are running in attached subtasks in the CICS® region, and the JCL REGION= size needs to be large enough to accommodate the number of expected users concurrently logged on to run Fault Analyzer. The amount of storage needed varies depending on the size and complexity of the fault being analyzed, but a starting point of 32 megabytes per concurrent user is recommended. This storage is not CICS® DSA storage, and needs to be available to satisfy the MVS™ GETMAIN requests in the attached subtasks. That is, the estimated storage requirement (for example, 10x32M=320M) needs to be at least the difference between the REGION= size and the CICS® EDSALIM value.

The other setup requirements for a CICS® region to run Fault Analyzer as an interactive CICS® transaction are:

1. Make the required CICS® resource definitions.
2. Make the required CICS® JCL changes.

These requirements are described in the following sections. It is assumed that IDI is the data set name high-level qualifier that was used during Fault Analyzer installation.

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.

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 IPVTLIB DD names below, for example, IDI.IDIPPROF. This data set should be defined as a PDS or PDSE with LRECL=80 and RECFM=FB. Only a small data set is required, for example 5 tracks primary and 5 tracks secondary space allocation.

   Note: Each user has their ISPF-like settings written to the IPVPROF data set, and as such each user needs UPDATE access to this data set.

3. Add the following DD names to the CICS® JCL:

```
//IPVPLIB  DD DISP=SHR,DSN=IPV.SIPVPENU
//         DD DISP=SHR,DSN=IPV.SIPVMENU
//         DD DISP=SHR,DSN=IDI.SIDIPLIB
//         DD DISP=SHR,DSN=IDI.SIDIMLIB
//         DD DISP=SHR,DSN=IDI.SIDISLIB
//IPVTLIB  DD DISP=SHR,DSN=IDI.IDIPPROF
//         DD DISP=SHR,DSN=IPV.SIPVTENU
//         DD DISP=SHR,DSN=IDI.SIDITLIB
//IPVPROF  DD DISP=SHR,DSN=IDI.IDIPPROF
```

   Note: The data sets IPV.SIPV* are created as part of the installation of ADFz Common Components.

Configuring the Model Context Protocol (MCP) Server

Set up the workstation-based MCP server on Windows® to connect external AI assistants to Fault Analyzer via z/OSMF REST APIs.

Before you begin

Before configuring the server, ensure the following requirements are met:

- Install Java 21 on your Windows® system path.
- Ensure your z/OSMF REST API endpoint is active over HTTPS with a valid DNS hostname.
- Store your RACF user password securely in Windows Credential Manager. See Configuring local authentication using Windows Credential Manager on page 565 for details.

About this task

The MCP server exposes a set of functional APIs that are implemented based on Representational State Transfer (REST) services for scripting or testing purposes if the server is run in HTTP mode. You can evaluate and test the server before connecting it to an AI client. Access the exposed REST API services and an OpenAPI spec (/v3/api-docs) through the Swagger UI sending HTTP requests directly to the REST API from you browser by connecting to the following URL: http://localhost:8080/swagger-ui.html.

To configure the MCP server:

1. Obtain the executable JAR file (`FA-MCP-<version>-SNAPSHOT-exec.jar`) found as a member in the IDI.SIDIDOC1 data set .
2. Start the server in HTTP mode from a Windows® command prompt. HTTP serves a network audience, a shared/ remote server.

   ```
   java -jar FA-MCP-<version>-SNAPSHOT-exec.jar --transport.mode=http \
                           --zosmf.url=https://zosmf.example.com \
                           --zosmf.user=JDOE \
                           --zosmf.ssl.certificate-path=C:/certs/zosmf.crt
   ```

   Alternatively, the following is an example of an STDIO client JSON configuration file. STDIO serves a local AI client that launches the server as a child process.

   ```
   "Fault-Analyzer-MCP-Server" : {
     "type":"stdio",
     "command": "java",
     "args": [
       "-jar",
       "D:\\MCPServers\\FA-1.1.0-exec.jar",
       "--transport.mode=stdio",
       "--zosmf.url=https://pthfae1.prod.hclabc.com",
       "--zosmf.ssl.certificate-path=D:\\Certs\\zosmf_e1.crt",
       "--zosmf.user=JDOE"
     ]
   }
   ```

3. To connect desktop AI clients over standard I/O, specify `--transport.mode=stdio` in the client JSON configuration file.

What to do next

To verify if the server is running successfully, open the Swagger UI from your browser: http://localhost:8080/swagger-ui.html.

Configuring local authentication using Windows® Credential Manager

To run the Model Context Protocol (MCP) server in local or STDIO mode without exposing RACF passwords in configuration files, store your z/OSMF credentials in Windows® Credential Manager.

Before you begin

Ensure that you have a valid RACF user ID and password with access to z/OSMF REST interfaces on the target mainframe LPAR.

About this task

When running in local single-user mode (`server.is-remote=false`), the MCP server queries Windows Credential Manager at startup to retrieve your RACF credentials and request an authentication token from z/OSMF. Passwords are never written to disk or configuration files.

1. Use the following format for the credential target name:
   - `FA:zosmf:your_racf_userid`, for example `FA:zosmf:JSMITH`

2. Store the credential using either the Windows command line or the Control Panel GUI:
   Choose from:
   - **Command line:** Open a command prompt and run cmdkey:

     ```
     cmdkey /generic:FA:zosmf:your_racf_userid /user:your_racf_userid /pass:your_racf_password
     ```

   - **Control Panel GUI:** Open **Credential Manager**, select **Windows Credentials**, click **Add a generic credential**, and enter the target name, user ID, and password.
3. Verify that the credential target name matches the `zosmf.user` property in your server configuration file (`application.yml`).

Results

The MCP server can now authenticate securely with z/OSMF during local startup without prompting for passwords or storing plain-text credentials.

Chapter 33. Installing and configuring IBM Fault Analyzer for z/OS Abend Intelligence

Deploy the on-host IBM Fault Analyzer for z/OS Abend Intelligence pipeline to z/OS UNIX System Services (USS) to enable real-time AI analysis of abend dumps.

About this task

To install IBM Fault Analyzer for z/OS Abend Intelligence:

1. Download Python 3.10 from https://www.python.org/downloads/ and deploy it and the required packages to your z/OS UNIX System Services (USS) environment. The required packages are found in data set IDI.SIDIDOC1 member IDIZREQ. The Python engine imports components from the `schemas` and `custom_exceptions` packages. You must recreate this mandatory package subdirectory structure on your USS environment. If copied to a flat directory, the startup fails.

```
src/
├── IDIZMAIN.py            # entry point
├── IDIZUTIL.py
├── IDIZCNST.py
├── IDIZCONF.py
├── IDIZREQ.txt            # dependency pins
├── schemas/
│   ├── IDIZFDIN.py
│   ├── IDIZFDEX.py
│   └── IDIZFILE.py
└── custom_exceptions/
    └── IDIZEXCP.py
```

   Important: The `pyzfile` required package only works on Python 3.10. If using Python 3.11, IBM Z Open Automation Utilities (ZOAU) is mandatory.

   a. Verify the log directory path in `IDIZCNST.py`.

      Open `IDIZCNST.py` in an editor and check the `LOG_DIRECTORY` setting:

```
LOG_DIRECTORY = "/tmp"
```

      Important:

      Ensure that `LOG_DIRECTORY` specifies a z/OS UNIX directory where background batch job user IDs have write access (for example, `/tmp` or a dedicated directory such as `/var/fai/logs`). If the user ID lacks write permission to this directory, log file creation fails and no diagnostic execution logs are generated.

2. Create an FB 80 data set containing your LLM credentials and environment definitions. For details, see The STDENV DD statement environment variables for Abend Intelligence on page 569.
3. Register the IBM Fault Analyzer for z/OS Abend Intelligence user exits via the `IDIOPTS` DDname.

```
DATASETS(IDIEXEC(IDI.SIDISAM1))
EXITS(
    NOTIFY(REXX(IDISUEAN))
    FORMAT(REXX(IDISSHOW))
)
```

The IDISUEAN and IDISSHOW user exits are provided in the IDI.SIDISAM1 data set. Ensure these exits are available to Fault Analyzer via the IDIEXEC DDname.

IDISUEAN

This is a Fault Analyzer Notification user exit written in REXX that captures environment variables from the active fault recording session, invokes the orchestration launcher (IDISBFCR) to build a JSON payload in /tmp/, and submits an asynchronous two-step JCL job to the internal reader to analyze the failure using a large language model (LLM).

Verify or change the AI Service Configuration definitions in this exit as appropriate for your installation:

Table 14. IDISUEAN REXX Exit Configuration Variables

Variable Name	Example Value	Description
`LLM_PROTOCOL_TO_USE`	`'watsonx'` or `'ollama'`	Specifies the target large language model backend protocol: `LLAMA` or `WATSONX`.
`LLM_SERVICE_END-POINT`	`'http://13.82.3.70:11434'`	Specifies the base URL of the LLM endpoint (required for Ollama).
`LLM_MODEL_TO_USE`	`'llama3:latest'` or `'ibm/gran-ite-3-8b-instruct'`	Specifies the model identifier to invoke on the AI backend.
`USE_SOURCE_CODE`	`YES`	Attempts DWARF/LANGX/LISTING source extraction.
`PROGRAM_MODE`	BF	Only BF is supported.
`PYTHON_EXEC_TO_USE`	`'/usr/lpp/IBM/cyp/v3r10/pyz/bin/python3'`	Specifies the fully qualified USS path to the Python 3.10 executable.
`PYTHON_MAIN_TO_USE`	`'/usr/local/bin/fai/src/IDIZ-MAIN.py'`	Specifies the fully qualified USS path to the `IDIZMAIN.py` entry point.
`STDENV_DSN_TO_USE`	`'SYS1.FAI.ENVS(ENVWTSNX)'`	Specifies the RACF-protected `FB 80` data set containing API credentials shown in step 1 on page 567.
`IDIOPTS_TO_USE`	<*userid*>().FAI.OPTIONS(FAIOP-TS)	FAI options data set used in the generated JCL.
`USE_ZOAU_MODE`	NO	Setting this variable to `YES` passes `USE_-ZOAU=YES` into the JSON payload.

IDISSHOW

This is a Fault Analyzer Formatting user exit written in REXX that reads the Markdown report data set <*userid*>.FAI.REPORT(<*fault_id*>) and converts Markdown elements to Fault Analyzer Formatting user exit tags (see Formatting tags on page 539).

4. Submit a test job with a sample JSON payload in `/tmp/` to verify outbound TCP/IP connectivity to your LLM endpoint.
5. Verify the installation. You can verify the installation in the following ways:

    a. **JES Spool Check:** Open `STDOUT` in SDSF and verify that the step finishes with `RC=0` and displays the `### FAI Job Summary ###` block with `Status: Success`.

    Note: A return code of RC=253 indicates that the pipeline executed successfully, but the LLM endpoint timed out or was unreachable over TCP/IP. Verify TCP/IP firewall rules and the `LLM_SERVICE_ENDPOINT` setting in `IDISUEAN`.

    b. **AI Report Data Set Member Check:** View data set `<userid>.FAI.REPORT(<fault_id>)`, where `<fault_id>` is the fault ID created by the test job in step 4 on page 569, using for example ISPF 3.4 to verify that the Markdown report content was written.

    c. **Fault Analyzer Report Check:** Perform reanalysis of the fault entry written in step 4 on page 569 and confirm that the AI report is included in the User section of the report.
6. If the testing in step 5 on page 569 was successful, propagate the IDIOPTS DDname options to the IDICNFxx parmlib member to make them available to all jobs and users.

The STDENV DD statement environment variables for Abend Intelligence

When IBM Fault Analyzer for z/OS Abend Intelligence real-time host-based AI analysis, a BPXBATCH job is automatically submitted via the IDISUEAN Notification user exit to process fault data through a Python engine. The `STDENV` DD statement in this job points to a fixed-block 80 (`FB 80`) data set containing environmental settings and large language model (LLM) credentials. The data set name (and member name, if a PDS/E) is specified via the STDENV_TO_USE option in the IDISUEAN exec.

Sample STDENV data set environment variable specifications:

```
_BPX_SHAREAS=YES
WATSONX_API_KEY=your_api_key
WATSONX_PROJECT_ID=your_project_id
```

where, `_BPX_SHAREAS=YES` is required to read the report, and the `WATSONX_API_KEY` and `WATSONX_PROJECT_ID` are used to connect your watsonx.ai user. For details refer to the IBM watsonx documentation.

Important: The data set specified in `STDENV` must be protected with RACF `UACC(NONE)` to prevent unauthorized access to API credentials. For long values exceeding 80 characters, enable Language Environment continuation by placing a backslash (\) at column 80.

Part III. Fault Analyzer reference information

Chapter 34. Options

You can use options to exert some control over the way that Fault Analyzer produces output. For example, there are options to:

- Change the fault analysis report contents.
- Change the action that Fault Analyzer takes at the time of the abend.

You can provide options at most stages of processing. If the option is not relevant to the current mode of processing (for example, you are trying to set the Exclude option when doing a batch reanalysis), it is disregarded. Fault Analyzer does not produce unnecessary warning messages in this situation.

Options can be set or changed in the following ways, listed in the order of their processing:

1. Product defaults provided by Fault Analyzer.
2. SMP/E USERMODs.

   For more information, see Customize Fault Analyzer by using USERMODs on page 333.
3. IBM Application Delivery Foundation for z/OS (ADFz) Common Components IPVOPTLM configuration-options module.

   Fault Analyzer does not require the IPVOPTLM configuration-options module to exist. If it exists, Fault Analyzer will process it unless the NOIPVOPT value is set to 1 in the IDIOPTLM configuration-options module. See Ignoring IBM Application Delivery Foundation for z/OS Common Components options (NOIPVOPT) on page 341.

   For information about the IPVOPTLM configuration-options module, see *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*.
4. Installation-wide defaults specified in the ADFz Common Components IPVCNF00 parmlib member.

   Fault Analyzer does not require the IPVCNF00 parmlib member to exist. If it exists, Fault Analyzer will process it unless the NOIPVOPT value is set to 1 in the IDIOPTLM configuration-options module. For details, see Ignoring IBM Application Delivery Foundation for z/OS Common Components options (NOIPVOPT) on page 341.

   For information about the IPVCNF00 parmlib member, see *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*.
5. Configuration-options module IDIOPTLM.

   For more information, see Customize Fault Analyzer by using an IDIOPTLM configuration-options module on page 337.
6. Options located via an IDICNFUM user-options module.

   This module is only applicable to real-time analysis, and, if found, replaces step 7 on page 572.

For more information, see User-options module IDICNFUM on page 574.

7. Installation-wide defaults specified in the IDICNFxx parmlib member.

   The parmlib member is only read if a user-options module was not found in step 6 on page 571.

   For more information, see Parmlib member IDICNFxx on page 573.

8. Options that are specified in a user-options file through the _IDI_OPTSFILE environment variable.

   If found, replaces step 9 on page 572.

   For more information, see The _IDI_OPTSFILE environment variable on page 575.

9. Options that are specified in a user-options file through the IDIOPTS DDname.

   Only read if a user-options file was not found in step 8 on page 572.

10. Options that are specified in the JCL EXEC statement PARM field when performing batch reanalysis.

    For more information, see The JCL EXEC statement PARM field on page 576.

11. Options provided through the _IDI_OPTS environment variable.

    For more information, see The _IDI_OPTS environment variable on page 576.

12. Options set via the Analysis Control user exit.

    For more information, see Analysis Control user exit on page 475.

13. 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 494.

If you do not specify an option, it takes either the product default (as indicated on the syntax diagram for each option), or has no value at all.

Some options can retain only one value. If more than one instance of such an option is specified, only the last occurrence has an effect. For example, if

```
PARM='Detail(LONG) Detail(SHORT)'
```

is specified, then the active option is Detail(SHORT).

Some options can have more than one value. These are, for example, the DataSets, Exits, Include, and Exclude options. The way in which they accumulate information is described for each option.

Wherever you specify an option, it is subject to these syntax rules:

- Only columns 1 - 71 are processed.
- Options can be specified anywhere in a line. They do not have to start in column 1.
- You can use a blank or a comma as a delimiter.
- Options can be continued across any number of lines, except when specified in the JCL EXEC statement PARM field, where the z/OS®-imposed limit is 100 characters.
- When continuing option values across multiple lines (for example, long zFS path names specified via the DataSets(IDIJAVA(...)) option), then either of the following is supported:
  - Specify the value up until, and including, column 71 and start in column 1 on the following line.
  - Specify the option value continuation character '+' at the end of the lines to be continued. One or more blank characters must precede the plus sign. Quoted values that are continued using the + sign must be specified with each part of the value surrounded by quotes.

  Example:

  ```
  DataSets(IDIJAVA('/this/might/be/a/really/long' +
                           '/path/name'))
  ```

- Option names and keyword parameters are not case-sensitive. Option values are also not case-sensitive, unless explicitly stated for a given option.
- Comments are permitted anywhere and can be nested. The characters "`/*`" identify the beginning of a comment, and "`*/`" identify the end.

Where to specify options

You can specify options in two files (IDICNFxx 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 IDICNFxx 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 149 and Interactive reanalysis options on page 157 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 475.

Parmlib member IDICNFxx

You can create a member IDICNFxx in SYS1.PARMLIB (where xx is either 00 or the current MVS™ &SYSCLONE symbol value), or any other data set that is part of the logical parmlib concatenation. Optionally, an IDIOPTLM configuration-options module can be used to specify an alternative data set. For details, see Specifying an alternative parmlib data set for IDICNFxx (CNFDSN) on page 339.

The IDICNFxx 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 205: Sample IDICNFxx parmlib member on page 370.

User-options module IDICNFUM

To replace the installation-wide default options during real-time analysis, you can create a user-options module containing the names of one or more partitioned data sets and members, each containing Fault Analyzer options.

Fault Analyzer tries to open the data set members in the order of their specification. The first data set and member found to be available is used instead of the IDICNFxx parmlib member in the logical parmlib concatenation or in the alternative parmlib data set specified in the CNFDSN option of the IDIOPTLM configuration-options module.

The user-options module must be named IDICNFUM and must be a load module available via the standard MVS™ search path. If placed in a load library that is allocated to the JOBLIB DDname, this placement effectively provides job-level control of default options.

The load module can contain partitioned data set and member names only in standard MVS™ JCL syntax format:

```
data-set-name(member-name)
```

Each data set specification must be terminated by an X'00' byte. A second X'00' byte must follow the last data set specification to indicate the end of the list. If no data sets are specified, at least one X'00' byte is required.

For added flexibility in the use of user-options modules, the following symbolic names can be used in the specification of data set or member names:

&SYSUID.

The user ID that is associated with the abending job or CICS® transaction.

&JOBNM.

The job name of the abending job.

&PGMNM.

The program name on the EXEC statement in the abending job.

A job to create a sample user-options module is provided as member IDISCNFU in the softcopy samples data set.

The data set and member selected by Fault Analyzer is identified in message IDI0001I on page 695. If you want to see the data set and member names that were not selected (after substitution of variables), include the IDITRACE DDname in your job step. For example:

```
//IDITRACE DD SYSOUT=*
```

(See IDITRACE under CICS on page 402 for an alternative method of activating this trace under CICS®.)

The name of the selected user-options module data set and member is saved by Fault Analyzer in the history file and automatically reused as the default options file during reanalysis of the fault. For batch reanalysis, the data set and member is included in the generated JCL with the IDIBOPT DDname.

The _IDI_OPTSFILE environment variable

Your application program can initialize an environment variable, _IDI_OPTSFILE, with an MVS™ data set (and optionally, member) name, or a zFS path and file name, containing options for Fault Analyzer, prior to abending or calling IDISNAP.

- **MVS™ data set name specification**

  If specifying an MVS™ data set name, then the name must be immediately preceded by two forward slashes (//). The specified data set name is not case sensitive.

  The specified user options file must be fixed 80-byte record length format and all options must be specified within columns 1 - 71.

- **zFS path and file name specification**

  If specifying an zFS path and file name, then the name must not be preceded by two forward slashes.

  The specified user options file is not subject to any particular attributes and options can be specified in any columns.

Example:

```
//* Sample BPXBATCH job
//RUN EXEC PGM=BPXBATCH, …
⋮
//STDENV   DD * (select one of the following)
_IDI_OPTSFILE=//my.SEQ.OPTIONS
_IDI_OPTSFILE=//my.PDS.OPTIONS(faopts)
_IDI_OPTSFILE=/u/fred/options_file
⋮
/*
```

Note: No attempt is made to read options through the IDIOPTS DDname (see User options file IDIOPTS on page 575) 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 411).

Here is an example of a job that uses an in-stream user-options file to override any dump suppression if program MYAPPL abends:

Figure 295. Sample job specifying user-options file

```
//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 */
/*
```

The user options file must be fixed 80-byte record length format and all options must be specified within columns 1 - 71.

Note: If a user options file is found through the _IDI_OPTSFILE environment variable (see The _IDI_OPTSFILE environment variable on page 575), then no attempt is made to read options through the IDIOPTS DDname.

The JCL EXEC statement PARM field

You can specify options in the JCL EXEC statement PARM field when executing Fault Analyzer in batch reanalysis mode. For example:

```
//MYJOB2   JOB
//STEP1    EXEC PGM=IDIDA,
// PARM='/Detail(LONG),PreferredFormattingWidth(132)'
⋮
```

The _IDI_OPTS environment variable

Your application program can initialize an environment variable, _IDI_OPTS, with options for Fault Analyzer prior to abending or calling IDISNAP.

Example:

```
//* Sample BPXBATCH job
//RUN EXEC PGM=BPXBATCH, …
⋮
//STDENV   DD *
_IDI_OPTS=DataSets(IDIHIST(MY.HIST)),Detail(L)
⋮
/*
```

When changes to options take effect

In general, changes to options take effect immediately, regardless of how they are specified. The only exception is the IDIS subsystem, which must be stopped and restarted if changes are made to any of these options:

- DataSets (IDIVS*xxx*(...),IDIEXEC(...),IDIDOC(...),IDIDO*xxx*(...),IDIVIEWS(...))
- DumpRegistrationExits
- Exclude/Include
- Language
- Locale
- NoDup(ImageFast(...))
- Quiet

where *xxx* is the language ID of the Language option that is in effect. (IDIDO*xxx* does not support ENU.)

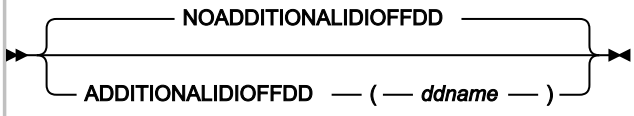
It is not necessary to stop and restart the IDIS subsystem for implementation of, or changes to, the IDIOPTLM configuration-options module, unless the CNFDSN setting is used to point to an IDICNFxx member that changes any of the above options.

Option descriptions

The following explains each option in detail.

AdditionalIDIOffDD

Figure 296. Syntax



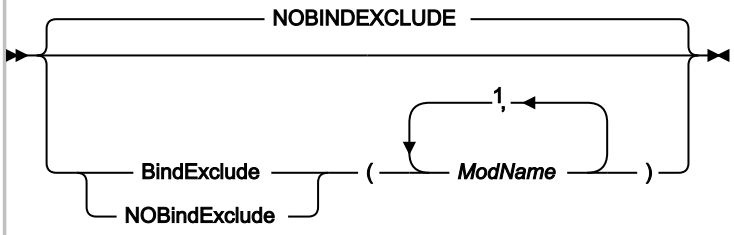
The AdditionalIDIOffDD option can be used to provide an extra DDname, which is equivalent to the normal Fault Analyzer IDIOFF DDname (for details, see Turning off Fault Analyzer with a JCL switch (IDIOFF) on page 462). If specified, any job with *ddname* allocated does not invoke Fault Analyzer for real-time abend processing.

This option is read by the Fault Analyzer IDIS subsystem, and the setting made available to abending regions via cross memory access. Changes to the AdditionalIDIOffDD option take effect only after stopping and restarting the Fault Analyzer IDIS subsystem (for details, see Using the Fault Analyzer IDIS subsystem on page 318).

This option is not included in the section of the fault analysis report that shows options in effect.

BindExclude

Figure 297. Syntax



Notes:

[1] Either comma or blank character is permitted as delimiter.

You can use the BindExclude option to specify one or more module names to exclude from binder calls.

The NOBindExclude option is used to reset the list of excluded modules or to remove specific modules from the exclusion list. If it is specified without any module names, all module names are removed from the exclude list. If the option is not defined, the default is NOBindExclude, no modules are excluded.

If BindExclude is specified multiple times, only the module names from the last BindExclude option is used; previous specifications are ignored.

This option does not apply to real-time analysis.

ModName

The name of the module to be excluded or removed from the exclusion list. Multiple module names can be specified separated by either a comma character, a space, or a combination of both. The name can be up to 8 characters long. A wildcard character (*) can be used.

Examples

Example 1: Exclude all modules starting with "IDI"

```
BindExclude(IDI*)
```

Example 2: Exclude specific modules: DFHA, DFHB, DFHC, and TRSA

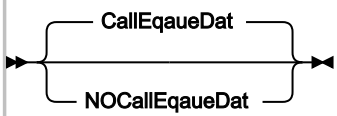
```
BindExclude(DFHA,DFHB DFHC,TRSA)
```

Example 3: Remove all modules beginning with "DFH" from the exclusion list. In this example, DFA, DFHB, DFHC, and TRSA are on the exclusion list, and only DFA, DFHB, and DFHC are removed, whileTRSA remains:

```
NOBindExclude(DFH*)
```

CallEqaueDat

Figure 298. Syntax



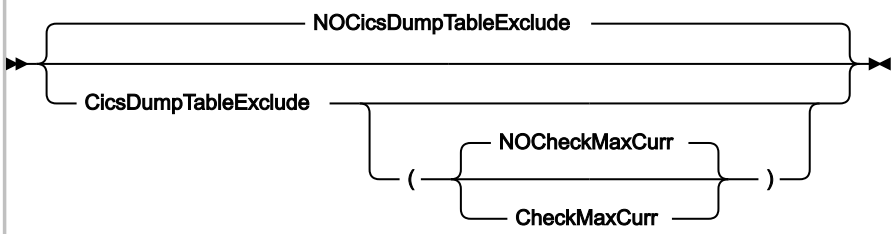
CallEqaueDat (the default) instructs Fault Analyzer to call the IBM® z/OS® Debugger listing exit to locate source listings to be used during analysis. If not found then processing continues.

NOCallEqaueDat prevents the exit from being called.

This option is not included in the section of the fault analysis report that shows options in effect.

CICSDumpTableExclude

Figure 299. Syntax



The CICSDumpTableExclude option excludes a CICS® transaction abend from Fault Analyzer real-time processing if the CICS® transaction dump table action for the same abend code specifies NOTRANDUMP.

If excluded, then message IDI0101I on page 717 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 47).

If the CheckMaxCurr suboption is specified, then Fault Analyzer compares the current number of CICS® dump requests for a specific dump code to the maximum setting for that dumpcode. If the current value exceeds the maximum value, then Fault Analyzer analysis is skipped and message IDI0180I on page 736 is issued.

Note: CICS creates the transaction dump table (TDT) entry only after the first abend occurs and Fault Analyzer is invoked (when using the XPCABND or LE abnormal termination exit). Therefore, when the CICS parameter TRDUMAX is set to 0, Fault Analyzer cannot exclude processing of the first abend instance because there is no TDT entry for CheckMaxCurr to check. Fault Analyzer can exclude the analysis of subsequent abends after CICS creates the TDT, because CheckMaxCurr can detect that the current instance count exceeds 0.

The default setting is NOCheckMaxCurr, in which case analysis continues regardless of the maximum setting for the dumpcode.

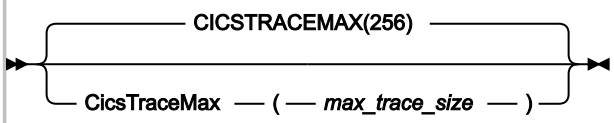
When NOCICSDumpTableExclude is in effect (default), then Fault Analyzer does not use the CICS® transaction dump table in its exclude checking.

This option is not included in the section of the fault analysis report that shows options in effect.

For changes to this option to take effect, uninstall and reinstall all CICS invocation exits using the CFA transaction. For details, see Controlling CICS transaction abend analysis on page 400.

CICSTraceMax

Figure 300. Syntax

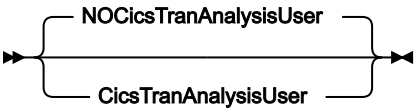


The CICSTraceMax option is applicable to CICS® system dump analysis only, and is used to specify the maximum CICS® trace table size in kilobytes which can be included in a fault entry minidump.

The value of *max_trace_size* must be in the range 0 - 2097151 kilobytes. If not specified, the CICSTraceMax option value defaults to 256 kilobytes.

If the actual CICS® trace size exceeds the CICSTraceMax option value in effect, then the trace table is read from the associated CICS® system dump during fault entry reanalysis.

CICSTranAnalysisUser

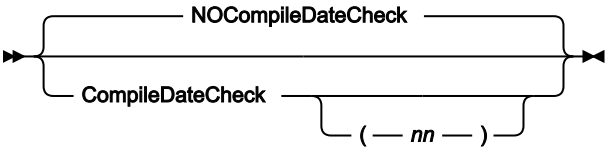


Specify the CICSTranAnalysisUser option to enable fault analysis to run under the credentials of the signed-on CICS user. Otherwise, fault analysis runs under the user ID of the CICS region.

This option is not included in the section of the fault analysis report that shows options in effect.

CompileDateCheck

Figure 301. Syntax



Use the CompileDateCheck option to specify whether you want Fault Analyzer to scan user programs in the event list to check if any were compiled within a specified number of days before the abend . If Fault Analyzer detects recently compiled user programs, the synopsis section of the fault analysis report includes a note.

CompileDateCheck

Enables compile date checking by using the default threshold of 10 days.

CompileDateCheck(*nn*)

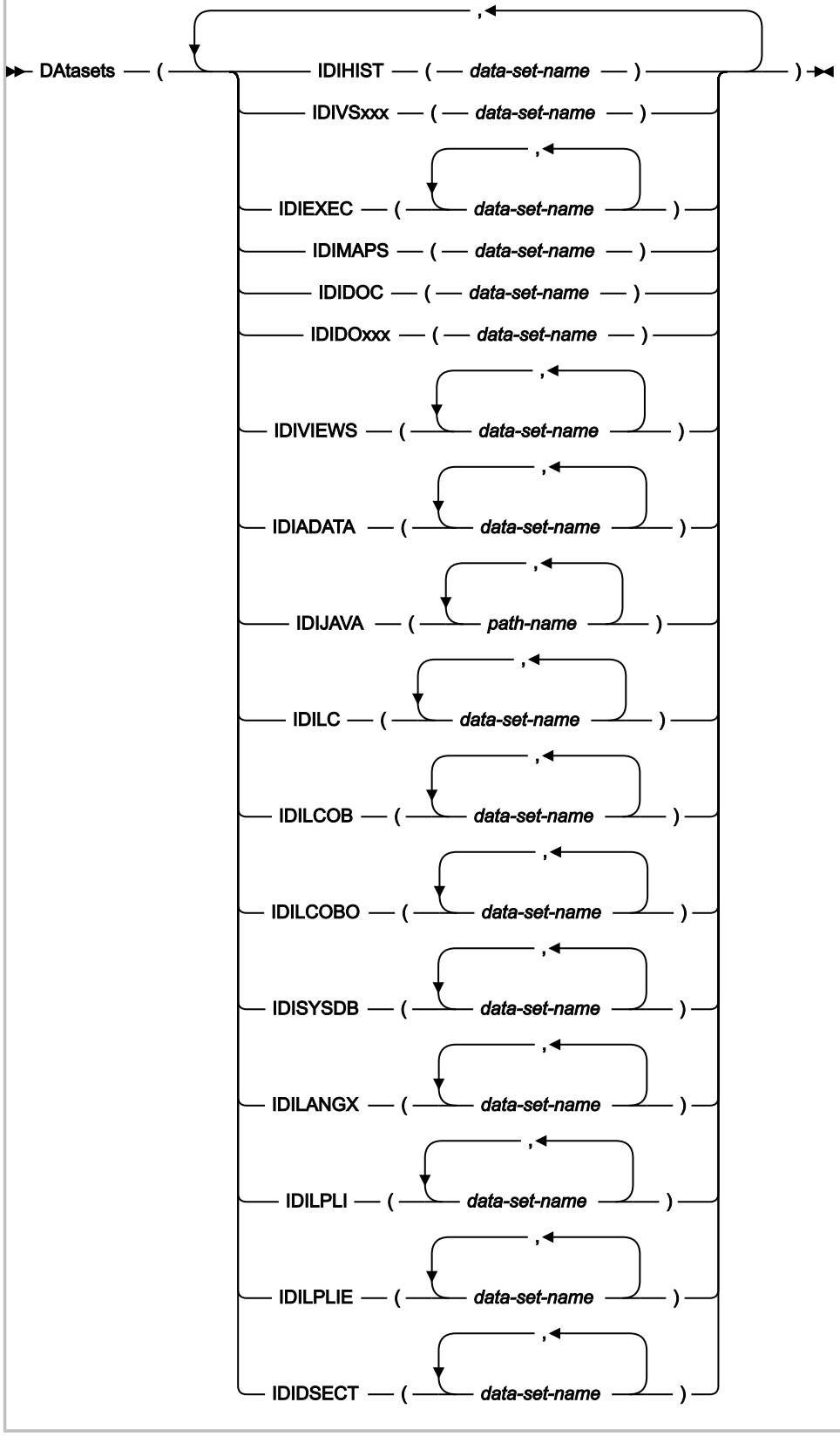
Enables compile date checking by using the specified variable *nn* as the threshold number of days before the abend date. The value of *nn* must be an integer from 1 to 9999. If no value is specified, a default threshold of 10 days is used.

NOCompileDateCheck

Disables compile date checking. Fault Analyzer does not include compile date notes in the synopsis section of the fault analysis report. This is the default setting.

DataSets

Figure 302. Syntax



The DataSets option specifies as suboptions the DDnames and their associated data set names that are to be dynamically allocated byFault Analyzer.

IDIHIST

The name of the PDS or PDSE history file where the fault entry is to be or was written. Default value is IDI.HIST.

IDIEXEC

The name of one or more PDS or PDSE data sets containing REXX user exits.

IDIMAPS

The name of the PDS or PDSE data set containing data area mappings provided with Fault Analyzer for z/OS®. Default value is IDI.SIDIMAPS.

IDIDOC

The name of the distributed book index and override PDS or PDSE data set. Default value is IDI.SIDIDOC1.

IDIDOxxx

The name of an extra distributed book index and override PDS or PDSE data set for a multicultural support feature, where xxx is a valid language ID for the Language option, other than ENU, for example, IDIDOJPN. The data set specified is only used when the equivalent Language option is in effect. No default value is provided.

IDIVSxxx

The name of the VSAM KSDS message and abend code explanation repository, where xxx is a valid language ID for the Language option, for example, IDIVSENU. For values of xxx other than ENU, the data set specified is only used when the equivalent Language option is in effect. Default value is IDI.IDIVSENU.

IDIVIEWS

The name of one or more PDS or PDSE data sets containing members defining the fault history files to be viewed in a single ISPF display.

IDIADATA

The name of one or more sequential or PDS or PDSE data sets holding assembler SYSADATA files.

For details about required data set attributes for this DDname, see Compiler listings and side file attributes on page 387.

IDIJAVA

One or more zFS path names to be used when searching for Java source code.

By default, path names specified through this option are concatenated ahead of the Java class paths. However, if the special value -DROPCP- is encountered anywhere within an IDIJAVA specification, then only the path names specified in the IDIJAVA DDname are searched. For example, the following specification will result in only the path names `/a/b/c` and `/d/e/f` being searched, ignoring the Java class paths:

```
DataSets(IDIJAVA(/a/b/c,-dropcp-,/d/e/f))
```

If the following IDIJAVA statement is specified, Java source code is not searched for.

```
DataSets(IDIJAVA(-dropcp-))
```

Path names must start with a slash (/) and may optionally be surrounded by single quotation marks.

If the path name includes single quotation marks, commas, or blanks, it must also be surrounded by single quotation marks. Any single quotation marks within the path name must be specified doubled up. For example, the path name `/a/b', c` must be specified as follows:

```
'/a/b'', c'
```

The maximum length of each path name, excluding any surrounding single quotation marks, is 1023 characters.

Path names are case sensitive. The -DROPCP- keyword is not case sensitive.

IDILC

The name of one or more sequential or PDS or PDSE data sets holding C compiler listings.

For details about required data set attributes for this DDname, see Compiler listings and side file attributes on page 387.

IDILCOB

The name of one or more sequential or PDS or PDSE data sets holding COBOL compiler listings (other than OS/VS COBOL).

For details about required data set attributes for this DDname, see Compiler listings and side file attributes on page 387.

IDILCOBO

The name of one or more sequential or PDS or PDSE data sets holding OS/VS COBOL compiler listings.

For details about required data set attributes for this DDname, see Compiler listings and side file attributes on page 387.

IDISYSDB

The name of one or more sequential or PDS or PDSE 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(,SEPARATE) option. MDBG side files are created using the CDADBGLD utility.)

For details about required data set attributes for this DDname, see Compiler listings and side file attributes on page 387.

IDILANGX

The name of one or more sequential or PDS or PDSE data sets holding LANGX side files.

For details about required data set attributes for this DDname, see Compiler listings and side file attributes on page 387.

IDILPLI

The name of one or more sequential or PDS or PDSE data sets holding PL/I compiler listings (other than Enterprise PL/I).

For details about required data set attributes for this DDname, see Compiler listings and side file attributes on page 387.

IDILPLIE

The name of one or more sequential or PDS or PDSE data sets holding Enterprise PL/I compiler listings.

For details about required data set attributes for this DDname, see Compiler listings and side file attributes on page 387.

IDIDSECT

The name of one or more PDS or PDSE 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 223.

IDIHIST, IDIEXEC, IDIMAPS, IDIDOC, IDIVSxxx, and IDIDOxxx have only one data set name value, and a second specification replaces, rather than accumulates.

Multiple specifications of the other DataSets suboptions are cumulative and all the data sets, wherever they have been specified, are included in the final logical concatenation of the respective DDname.

To specify a DUMMY data set for any DDname, a data set name of NULLFILE can be used.

The names of compiler listing or side file data sets used during real-time analysis are saved with the fault entry in the history file and are automatically used if reanalysis is performed. Therefore, there is generally no need to specify data sets using the DataSets option for reanalysis, unless a compiler listing or side file that was not available during real-time analysis is to be made available.

Data set logical replacement or concatenation order

The logical replacement or concatenation order of Fault Analyzer data sets is shown in the following:

1. Any data sets provided by an Analysis Control user exit (see Analysis Control user exit on page 475 for details).
2. Any explicitly coded JCL statements for the DDname.
3. Data sets from all DataSets options that are specified in the JCL EXEC statement PARM field (reanalysis only).
4. Data sets from all DataSets options that are specified in the user options file.

5. Data sets from all DataSets options that are specified in the parmlib configuration member.

   These options include the use of a system-wide alternate parmlib data set (for details, refer to Parmlib member IDICNFxx on page 369.) and a data set specified via a user-options module (for details, refer to User-options module IDICNFUM on page 574.).

Dropping IDICNFxx parmlib member data set specifications

It is possible to drop all data set specifications for a given DDname specified in the IDICNFxx parmlib member by using the special data set name, -DROPCNF-.

If, for example, the IDICNFxx 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.

DataSets option data set name substitution symbols

All data set names that are specified with the DataSets option can contain standard MVS™ symbols. These are resolved before any DDname-specific symbol substitution is performed by Fault Analyzer.

In Table 15: Permitted data set name substitution symbols by DDname on page 585, an X indicates permitted symbols that can be used as part of the data set names for each DDname:

Table 15. Permitted data set name substitution symbols by DDname

DDname	&PGM.	&SYSUID.	&TSOPFX.	&USERID.	&SYSNAME.
IDIHIST					X
IDIVSxxx					X
IDIEXEC				X (Note 1)	X
IDIMAPS					X
IDIDOC					X
IDIDOxxx					X
IDIVIEWS (Note 2)		X	X		X
IDIADATA	X	X (Note 3)	X (Note 3)	X	X

Table 15. Permitted data set name substitution symbols by DDname (continued)

DDname	&PGM.	&SYSUID.	&TSOPFX.	&USERID.	&SYSNAME.
IDIJAVA	X	X (Note 3)	X (Note 3)	X	X
IDILC	X	X (Note 3)	X (Note 3)	X	X
IDILCOB	X	X (Note 3)	X (Note 3)	X	X
IDILCOBO	X	X (Note 3)	X (Note 3)	X	X
IDISYSDB	X	X (Note 3)	X (Note 3)	X	X
IDILANGX	X	X (Note 3)	X (Note 3)	X	X
IDILPLI	X	X (Note 3)	X (Note 3)	X	X
IDILPLIE	X	X (Note 3)	X (Note 3)	X	X
IDIDSECT				X	X

Notes:

1

No value is available for the &USERID. variable when performing MVS™ dump analysis. (See File->Analyze MVS Dump Data Set on page 94.)

2

These symbols can be used in the data set names that are specified in the IDIVIEWS suboption of the DataSets option, as well as in the data set names that are specified in the individual view members contained within the IDIVIEWS data sets. For details, see Setting up views on page 347.

3

Substitution of these symbols is only performed during interactive reanalysis.

The available symbols on a given MVS™ system can be displayed with the MVS™ operator command D SYMBOLS.

If, at the time of performing symbol value substitution, a value for a symbol is unavailable, any data set names that include that symbol are ignored. An example of this ignoring is the &USERID. variable mentioned in the preceding note.

All symbol names specified must include the ending period, unless the symbol name is at the very end of a data set name. The symbol name, including the leading ampersand (&) and the ending period is replaced by the symbol value. Hence, if a symbol name is specified immediately ahead of a data set qualifier delimiter (period), then two consecutive periods must be specified, for example "&SYSUID..LISTINGS".

Data set names containing symbols do not cause errors during Fault Analyzer processing if, after their substitution, the data sets cannot be found, unless the data set is critical to the analysis.

Substitution symbols:

&PGM.

All instances of this symbol are substituted by the current program name when Fault Analyzer is searching for compiler listings or side files. To use this functionality, compiler listings or side files in sequential data sets can be stored with a naming convention that includes the program name.

For example, if the compiler listings for programs P1 and P2 are stored in the data sets FRED.LISTING.P1 and FRED.LISTING.P2 respectively, then these can both be located by Fault Analyzer by specifying "FRED.LISTING.&PGM." for the appropriate DDname in the DataSets option.

&SYSUID.

All instances of this symbol are substituted by the TSO user ID under which the Fault Analyzer ISPF interface is being used. By using this symbol in data set names that are specified in the IDICNFxx parmlib member, automatic user-specific access to private data sets can be provided, given that an appropriate data set naming convention is adhered to by the installation.

For example, specify the option

```
DataSets(IDIVIEWS(&SYSUID..VIEWS,PROD.VIEWS))
```

in the IDICNFxx parmlib member. Then all users of the Fault Analyzer ISPF interface for whom a data set named *user-id*.VIEWS exist, where *user-id* is the user's TSO user ID, can place private Fault Analyzer View definitions in this data set. These definitions either complement any installation-wide definitions in the PROD.VIEWS data set, or override them.

&TSOPFX.

All instances of this symbol are substituted by the TSO profile prefix for the user under which the Fault Analyzer ISPF interface is being used. By using this symbol in data set names that are specified in the IDICNFxx parmlib member, automatic user-specific access to private data sets can be provided, given that an appropriate data set naming convention is adhered to by the installation.

For example, specify the option

```
DataSets(IDIVIEWS(&TSOPFX..VIEWS,PROD.VIEWS))
```

in the IDICNFxx parmlib member. Then all users of the Fault Analyzer ISPF interface for whom a data set named *tso-prefix*.VIEWS exist, where *tso-prefix* is the user's TSO profile prefix, can place private Fault Analyzer View definitions in this data set. These definitions either complement any installation-wide definitions in the PROD.VIEWS data set, or override them.

&USERID.

All instances of this symbol are substituted by the user ID under which the abend occurred.

&SYSNAME.

All instances of this symbol are substituted by the system name from CVTSNAME, as at the time of substitution.

Symbol substring specification

Substring specification of all symbols (MVS™ system symbols, as well as Fault Analyzer specific symbols) is permitted.

Figure 303. Syntax

where:

symbol_name

The name of the symbol, for example USERID.

start

The character position where the substring is to start. The first character in the original symbol is position 1, the second is position 2, and so on. If *start* is positive, the system counts from the starting position to the end of the string. If *start* is negative (in other words, a minus sign appears before it), the system counts backwards from the ending position of the string.

number

The number of characters, from the starting position to the ending position of the string, that the substring is to contain. If *number* is not specified, the substring length defaults to 1. Do not specify a negative number for *number*.

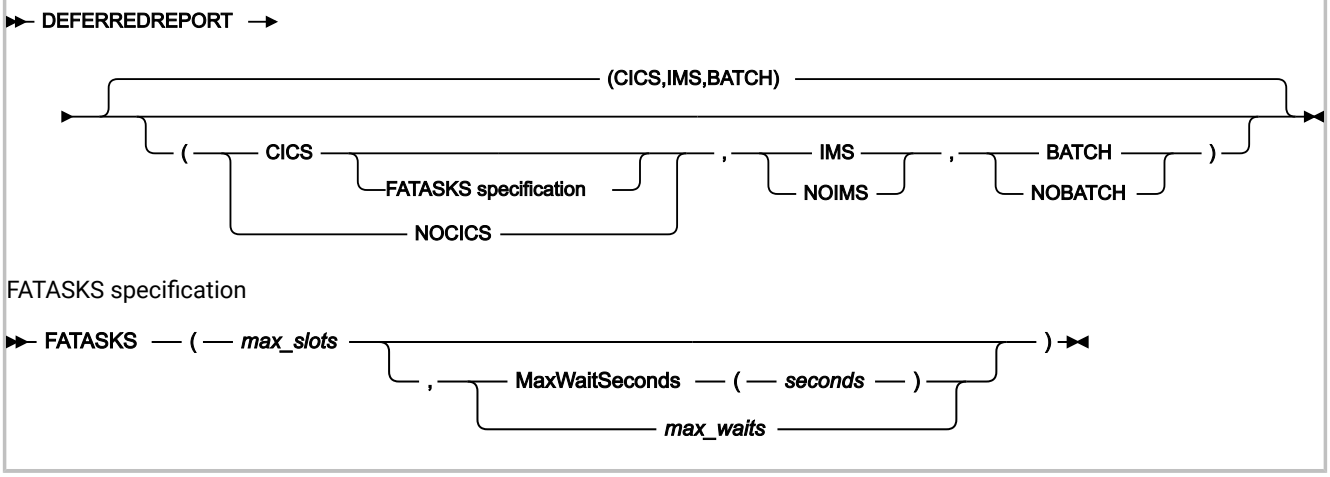
For example, if the substitution value for symbol &PGM. contains MYPROG1, and you want to limit the qualifier to the first six characters, you can specify

```
&PGM(1:6).
```

Refer to *MVS™ Initialization and Tuning Reference* section "Using substrings of system symbols" for complete information about symbol substring specification. Note that the use of double ampersand notation is not supported.

DeferredReport

Figure 304. Syntax



This option applies to real-time analysis only, and can be used when real-time performance is critical. When used, no real-time analysis report is produced, but a fault entry is written that can later be reanalyzed.

For compatibility with earlier versions of Fault Analyzer, specification of

```
DeferredReport(CICSFATasks(max_slots<,max_waits>))
```

is permitted until further notice. This specification is equivalent to

```
DeferredReport(CICS(FATasks(max_slots<,max_waits>)))
```

The execution environment under which the DeferredReport option is enabled is specified using the CICS® | NoCICS, IMS™ | NoIMS, or Batch | NoBatch suboptions.

- When the DeferredReport option is not specified, the product default is CICS® only:

  ```
  DEFERREDREPORT(CICS(FATASKS(1,20)),NOIMS,NOBATCH)
  ```

- When the DeferredReport option is specified without any suboptions, the default is all execution environments.

As far as the DeferredReport option is concerned, a CICS® transaction that also interfaces with IMS™, is controlled using the CICS® suboption. Hence, specification of

```
DeferredReport(NoCICS,IMS)
```

disables the DeferredReport option for such an application, whereas

```
DeferredReport(CICS,NoIMS)
```

enables it.

When used with CICS®, the optional FATASKS suboption can be specified:

max_slots

Specifies the maximum number of execution slots that are made available. Each execution slot permits one instance of Fault Analyzer real-time analysis to run, effectively enabling parallel execution or multi-tasking.

The valid range is 1 - 6.

The default is 1.

If NODeferredReport is specified, then the maximum number of execution slots is set to 1.

MaxWaitSeconds(*seconds*)

Specifies the maximum number of seconds that a fault is allowed to be queued waiting for analysis. If a fault has been waiting for longer than the current limit in effect, then message IDI0132W on page 725 is issued and the analysis of that fault is skipped, thus requiring normal CICS® transaction dump analysis to be performed (that is, not using Fault Analyzer).

The waiting queue length is always 20 when this suboption is specified. The behavior described for the *max_waits* suboption is still applicable to faults that occur when 20 queued up faults are already waiting.

The valid range is 0 - 3600. If 0 is specified, then no time limitation is imposed.

The default is 0.

Using MaxWaitSeconds(*seconds*) instead of *max_waits* is recommended.

max_waits

Specifies the maximum number of faults allowed to be queued for analysis. If the maximum has already been reached when another fault occurs, then message IDI0118W on page 721 is issued and the analysis of that fault is skipped, thus requiring normal CICS® transaction dump analysis to be performed (that is, not using Fault Analyzer).

The valid range is 1 - 20.

The default is 20.

Note: It might be desirable to reduce *max_waits* to a lesser value if analysis is slow in a CPU constrained environment. This reduction could have the effect of abends being rejected from analysis with the IDI0118W on page 721 message, but that might be preferred during high abend activity to having the task wait for abend processing in an over-committed environment.

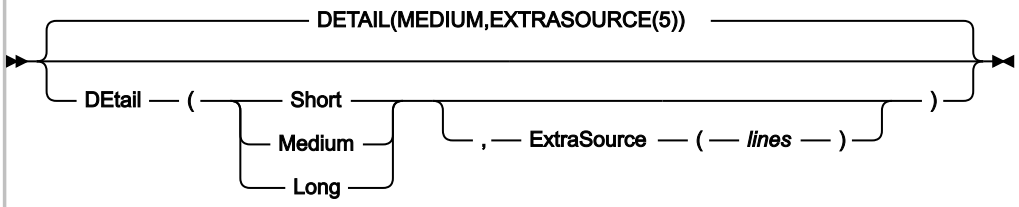
Note:

1. If XDUMPs are not enabled, DeferredReport is in effect, and the MaxMinidumpPages limit is exceeded, then the DeferredReport option is overridden and a report is written. If this situation occurs, then message IDI0133W on page 725 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 XDUMPs are enabled and the MaxMinidumpPages limit is adequate.

2. Although fault entries written with the DeferredReport option in effect do not initially contain a saved report, one is added the first time the 'V' or 'S' 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 475.
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 319.

Detail

Figure 305. Syntax



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:

Detail(SHORT)

- All sections of the report are included, except for the system-wide information. Detailed information is only included for the point of failure event.
- Summarized CICS® trace is included (when applicable).
- CICS Transaction Storage is not included.

Detail(MEDIUM)

- All sections of the report are included. However, detailed information is only included for events up until (and including) the first abend event that is also the point-of-failure event, or which follows the point-of-failure event. This option is the default.
- Summarized CICS® trace is included (when applicable).
- CICS Transaction Storage is not included.

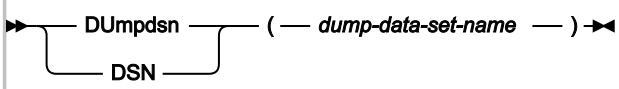
Detail(LONG)

- All sections of the report are included and detailed information is provided for all events.
- If available, general purpose registers, access registers, floating-point registers, and vector registers are shown in the event details section, regardless of the execution mode or type of machine instruction executed.
- Full CICS® trace is included (when applicable).
- 4K of storage around each user-code event general purpose register is included in the minidump.
- CICS Transaction Storage is included.

This option does not apply to interactive reanalysis.

DumpDSN

Figure 306. Syntax



Use the DumpDSN option when an abend caused a SYSMDUMP to be written to a dump data set and you want to reanalyze the dump using your own JCL. The option specifies the dump data set against which fault analysis is to be performed.

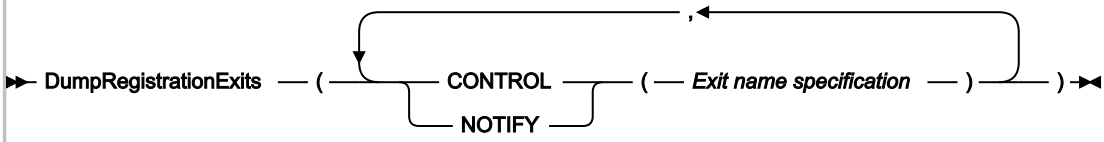
This option applies only to batch reanalysis and is ignored if specified in an IDICNFxx parmlib member.

If you initiate the reanalysis from the fault history file, then Fault Analyzer provides the dump data set name automatically.

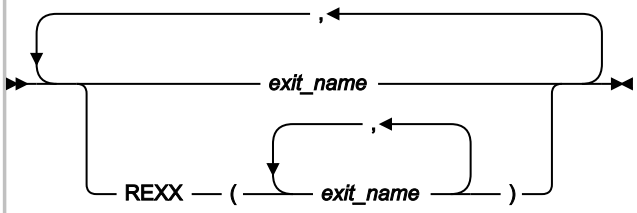
The logical record length of the SYSMDUMP data set must be 4160 if it does not contain ASA printer control characters, or 4161 if it does. Typically, only SYSMDUMP data sets that have been extracted from the JES SPOOL via SDSF contain ASA printer control characters.

DumpRegistrationExits

Figure 307. Syntax



Exit name specification



The DumpRegistrationExits option specifies the types and names of user exits to be invoked during the IDIXTSEL MVS™ post-dump exit processing.

The dump registration Analysis Control and Notification user exits are driven as the SVC dump data set name is recorded in a history file fault entry for later analysis by the end user. The SVC dumps might have been triggered by abends or Fault Analyzer recovery fault recording (RFR) processing; however, the dump registration exits are not driven for RFR processing. The IDIXTSEL process requires the IDI.SIDISAM1(IDIWTSEL) sample ++USERMOD to be installed (for details, see Installing the MVS post-dump exit IDIXTSEL on page 412).

The MVS™ SVC dump registration Analysis Control and Notification user exits run from the Fault Analyzer IDIS subsystem.

Therefore, the DumpRegistrationExits option must be specified in the IDICNFxx parmlib member, or via an IDIOPTS DD statement in the IDIS subsystem JCL. The DumpRegistrationExits option is ignored if specified via an IDIOPTS DD statement anywhere else, such as in a CICS® region or batch job.

Multiple specifications of the DumpRegistrationExits option are cumulative.

Exits can be either REXX EXECs or load modules:

- REXX EXECs must be specified as

  ```
  REXX(exit_name_1, exit_name_2, …)
  ```

  and be available via the IDIEXEC DDname.
- Load module exits must be available via the standard system search path (LPA, LINKLIST, or JOBLIB/STEPLIB JCL statement).

The possible exit types are:

CONTROL

Analysis Control user exit. This exit can be used to modify options in effect. For details, see Analysis Control user exit (MVS SVC Dump registration) on page 479.

NOTIFY

Notification user exit. This exit can, for example, be used to provide installation-specific notification of recorded faults. For details, see Notification user exit (MVS SVC Dump registration) on page 506.

The exit name that is specified as *exit_name* can be any valid TSO/E REXX EXEC or load module name. However, certain names are reserved for special purposes:

NONE

The special name 'NONE' represents a 'null' exit that is not invoked and causes further attempts to invoke exits of the specified type to be terminated.

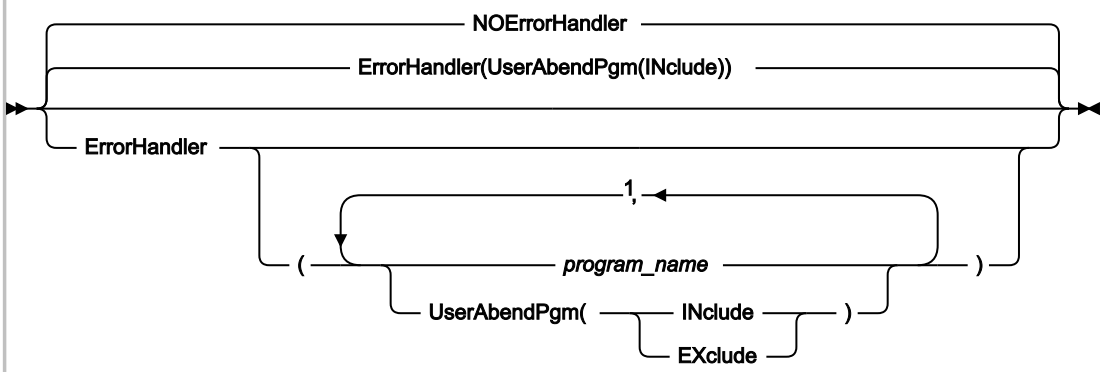
-DROPCNF-

The special name '-DROPCNF-' is used to drop exit specifications from the IDICNFxx parmlib member. For details, see Dropping IDICNFxx parmlib member user exit specifications on page 602.

Any number of exit names can be specified for a given exit type, and all exits are attempted invoked.

ErrorHandler

Figure 308. Syntax



Notes:

[1] Either comma or blank character is permitted as delimiter.

The ErrorHandler option can be used to specify one or more common error handler program or CSECT names, which should not be selected as the point-of-failure event in the Fault Analyzer report. Instead, the next earlier user-code event becomes the point-of-failure event, and this will then be used in duplicate fault determination, and so on.

The following suboptions can be specified:

program_name

One or more program or CSECT names of common error handlers, which are not selected as the point-of-failure event.

UserAbendPgm(INclude|EXclude)

By default, an event for a user abend is not selected as the point-of-failure. This is the equivalent to specifying UserAbendPgm(Include). To allow user abend events to be selected as the point-of-failure event, subject to the name not appearing in the program_name list, specify UserAbendPgm(Exclude).

Note: If no earlier user-code event can be determined, then it is possible that the error handler program event is still designated as the point of failure.

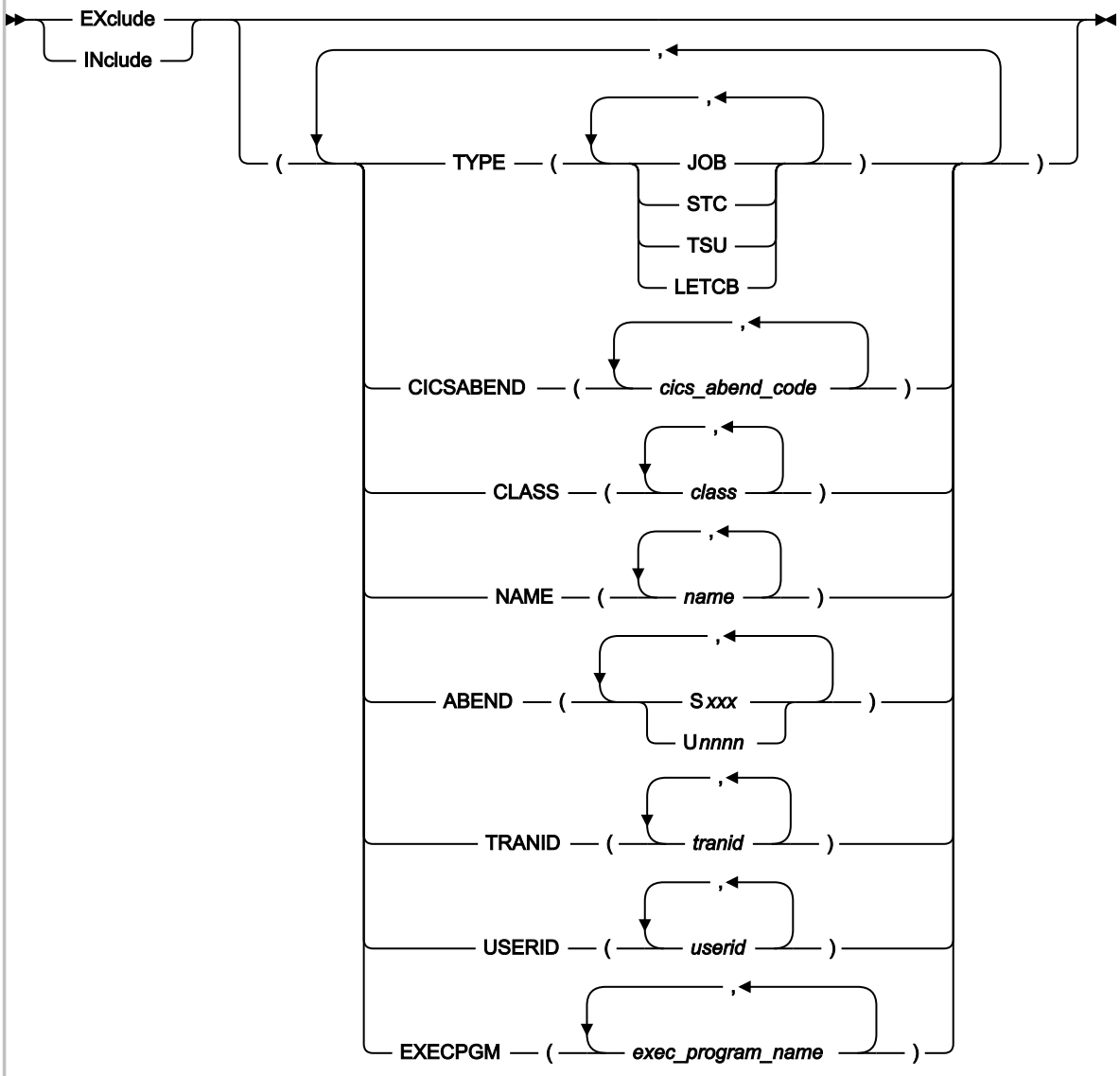
The last specification of the ErrorHandler or NOErrorHandler option is in effect. Subsequent specifications override any previous specification completely, that is, the program or CSECT names are not cumulative.

Exclude/Include

The Exclude and Include options are complimentary processes, sharing common parameters to control which job exceptions should be processed by Fault Analyzer. The Exclude/Include process, as described in Controlling which jobs are analyzed with Exclude processing on page 371, 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.

Figure 309. Syntax



where:

TYPE

Specifies the type of work unit as either JOB (batch jobs), STC (started tasks), or TSU (TSO users).

An extra type, LETCB, specifies that LE must be active for the abend TCB. This type can be used to distinguish between abends that occur in application tasks, as opposed to system tasks, provided that the application is written in a language that uses Language Environment®.

CICSABEND

Specifies one or more abend codes for CICS® transactions as *cics_abend_code*. Each abend code must be four alphanumerical characters.

The abend code tested is the final abend code for the transaction.

Note that CICS® system dumps captured via the IDIXTSEL MVS™ post-dump exit are not affected by the Exclude/Include option.

CLASS

Specifies one or more execution classes for batch jobs as *class*.

NAME

Specifies the names of one or more jobs, tasks, or TSO users as *name*.

ABEND

Specifies one or more system or user abend codes as one of the following:

- **S***xxx*

  where *xxx* is a three-character hexadecimal system abend code (for example, S0C4)

- **U***nnnn*

  where *nnnn* is a four-character decimal digit user abend code (for example, U4039)

If the specification of an abend code value does not start with 'S' or 'U', and it is less than or equal to three characters (including any wildcard characters), then it is assumed to be a system abend code. The abend code tested is the final abend code for the abending job step.

TRANID

Specifies the names of one or more CICS® transactions as *tranid*.

USERID

Specifies the TSO or CICS® user ID, or the user ID under which a batch job, a CICS® transaction, or a started task is executing, as *userid*.

EXECPGM

Specifies the program name from the JCL EXEC statement PGM keyword, as *exec_program_name*.

Note: The SYSABEND suboption, which has been replaced by the ABEND suboption, is supported for backwards compatibility only.

When an abending task meets the Exclude criteria, and no subsequent Include criteria also matches the task, the abend is not logged in the history file and no further Fault Analyzer processing is performed.

Specification rules:

- Individual suboptions, and values within suboptions, must be delimited by either one or more blank characters, or a comma.
- Wildcards are permitted in the specification of criterion values. The supported wildcard characters are an asterisk (*) to indicate zero, one or more characters, and a percent sign (%) to indicate a single required character.

  For examples of using wildcards with criterion values, see Exclude/Include wildcard examples on page 599.
- If no Exclude criteria are specified, the default is to include everything.
- All suboptions of an Exclude or Include criterion (that is, TYPE, CICSABEND, CLASS, NAME, ABEND, TRANID, USERID, and EXECPGM) must be satisfied for the criterion to be met (logical AND). It is possible to create Exclude or Include criteria that are never met, for example

  ```
  Exclude(TYPE(STC) CLASS(A))
  ```

  The reason why these criteria are never met is that a started task does not run in a JES initiator address space, and therefore is not associated with a particular class.

  In this case, and if no other Exclude criteria are specified (anywhere), then no job would be excluded, which means that Fault Analyzer analyzes any abends that occur.
- If more than one *type* (JOB, STC, TSU, or LETCB), *cics-abend-code*, *class*, *name*, *abend-code* (S*xxx* or U*nnnn*), *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.

If the IDIS subsystem is started and the default PARM='FASTEXCLUDE' option is in effect, changes to the Include/Exclude options only take effect on fast exclude processing after stopping and restarting the Fault Analyzer IDIS subsystem. For details, see Fast Exclude options processing on page 372.

Note: Every Include and Exclude criteria is checked against the abending task without regard to any previous Include or Exclude criteria. Hence, it is important to ensure that the order of these criteria in the parmlib config member, and, if available, in the user options file, result in the desired installation-specific rule set. For example, to exclude all batch jobs, except those executing in class A, you could specify the following sequence of criteria:

```
Exclude           /* This excludes everything */
Include(CLASS(A))  /* This includes batch jobs in class A only */
```

For more information, see Controlling which jobs are analyzed with Exclude processing on page 371.

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 462.

Using Exclude/Include options with dump registration processing

While Exclude/Include options are also applicable to dump registration processing, they are not very useful since not much is known about the conditions which resulted in the dump. This lack of knowledge is due to the dump being written by the dump services address space (for details, see Dump registration processing on page 47), and Fault Analyzer therefore invoked here, instead of in the address space which issued the dump.

It is therefore recommended that the dump registration Analysis Control user exit (see Analysis Control user exit (MVS SVC Dump registration) on page 479) 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 653).

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 402 for an alternative method of activating this trace under CICS®.)

The trace information is written to the IDITRACE DDname destination. An example of an Exclude/Include option-processing trace follows:

Figure 310. Sample Exclude/Include option-processing trace

```
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
```

EXCLUDE/INCLUDE trace information is not available if the fault is excluded via IDIS subsystem FastExclude processing.

Exclude/Include wildcard examples

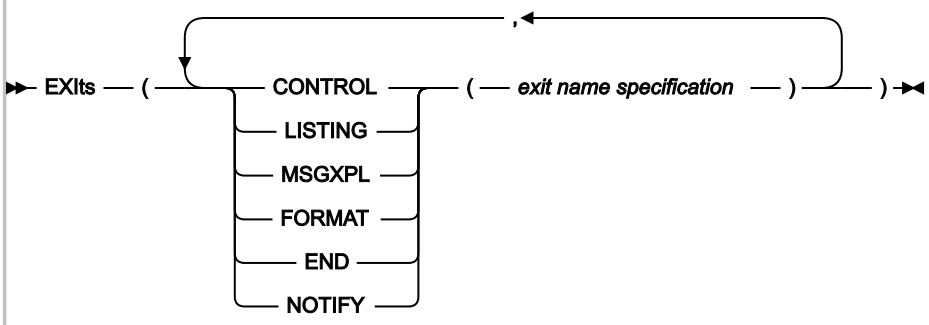
The following are examples of Exclude/Include criterion values with wildcards:

Table 16. Wildcard examples

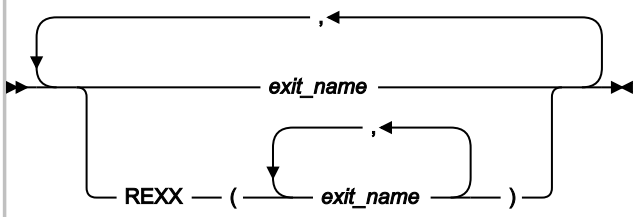
Criterion Value	Compare Value	Match
ABCD*	ABC	No
ABCD*	ABCD	Yes
ABCD*	ABCDE	Yes
A*CD	ABC	No
A*CD	ABCD	Yes
A*CD	ABCDE	No
A*DE	ABCDE	Yes
A**DE	ABCDE	Yes
A%DE	ABCDE	No
AB%DE	ABCDE	Yes
%B*	ABCDE	Yes

Exits

Figure 311. Syntax



exit name specification



The Exits option specifies the types and names of user exits to be invoked during Fault Analyzer execution. Any number of exit names can be specified for a given exit type, and all exits are attempted invoked.

Note: For information about specifying user exits for the IDIUTIL batch utility, see EXITS control statement on page 450 instead.

Multiple specifications of the Exits option are cumulative.

Exits can be either REXX EXECs or load modules (note that REXX is the only supported programming language for the Formatting user exit):

- REXX EXECs must be specified as

  ```
  REXX(exit_name_1, exit_name_2, …)
  ```

  and be available via the IDIEXEC DDname.
- Load module exits must be available via the standard system search path (LPA, LINKLIST, or JOBLIB/STEPLIB JCL statement).

The possible exit types are:

CONTROL

Analysis Control user exit. This exit can be used to modify options in effect or exclude a fault from analysis. For details, see Analysis Control user exit on page 475.

LISTING

Compiler Listing Read user exit. This exit can be used to obtain source code information from sources other than compiler listings or Fault Analyzer side files that are contained in available MVS™ PDS (or PDSE) data sets. For details, see Compiler Listing Read user exit on page 480.

MSGXPL

Message and Abend Code Explanation user exit. This exit can be used to provide message and abend code explanations to complement or substitute those provided by Fault Analyzer. For details, see Message and Abend Code Explanation user exit on page 485.

FORMAT

Formatting user exit. This exit can be used to add user-specific information to the analysis report. For details, see Formatting user exit on page 490.

END

End Processing user exit. This exit can be used to request suppression of the MVS™ system dump, the Fault Analyzer minidump, or the entire fault entry. For details, see End Processing user exit on page 494.

NOTIFY

Notification user exit. This exit can, for example, be used to provide installation-specific notification of recorded faults. For details, see Notification user exit on page 498.

The exit name that is specified as *exit_name* can be any valid TSO/E REXX EXEC or load module name. However, certain names are reserved for special purposes:

NONE

The special name 'NONE' represents a 'null' exit that is not invoked and causes further attempts to invoke exits of the specified type to be terminated.

-DROPCNF-

The special name '-DROPCNF-' is used to drop exit specifications from the IDICNFxx parmlib member. For details, see Dropping IDICNFxx parmlib member user exit specifications on page 602.

If one or more exits have been specified via the Exits option, information about the exits is written to the *"Options in Effect"* section of the analysis report. In this section, all specified exits are listed, and those of each type that were invoked (if any) are identified.

Note: The invocation status of the Notification user exit is not available as this exit is invoked after the completion of the analysis report.

An example of the information written to the *"Options in Effect"* section of the analysis report follows:

```
Exits:

  The following user exits were specified via Exits options.

  Type     Name     Type Invoked
  -------- -------- ---- ----------------------
  CONTROL  CTLEXITA LMOD No - module not found
           CTLEXITB REXX Yes
           CTLEXITC REXX Yes
           CTLEXITD LMOD No - module not found
  END      ABC1     LMOD Yes
  NOTIFY   NONE     n/a  (Not attempted)
```

This example indicates:

- Four Analysis Control user exits had been specified. The first of these (CTLEXITA) was a load module that could not be invoked. The second user exit specified (CTLEXITB) was a REXX EXEC that was invoked. No attempt was made to invoke the third (CTLEXITC) or fourth (CTLEXITD) user exits because of the successful invocation of the second.
- A single End Processing user exit (ABC1) had been specified as a load module. This user exit was invoked.
- A 'null' Notification user exit had been specified, which is never invoked.

Dropping IDICNFxx 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 IDICNFxx parmlib member by using the special exit name, -DROPCNF-.

If, for example, the IDICNFxx parmlib member contained:

```
Exits(Control(FRED1,FRED2))
```

and an IDIOPTS user-options file contained:

```
Exits(Control(FRED3,-DROPCNF-,FRED4))
```

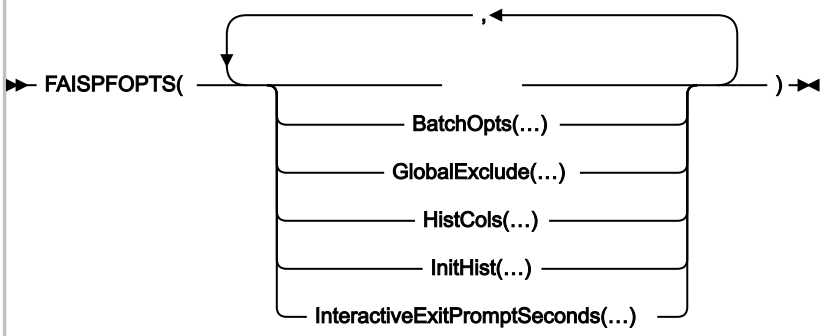
then the final logical concatenation of Analysis Control user exits would be:

```
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 IDICNFxx parmlib member.

FAISPFopts

Figure 312. Syntax



The FAISPFopts option is used to specify options for the Fault Analyzer ISPF interface. See:

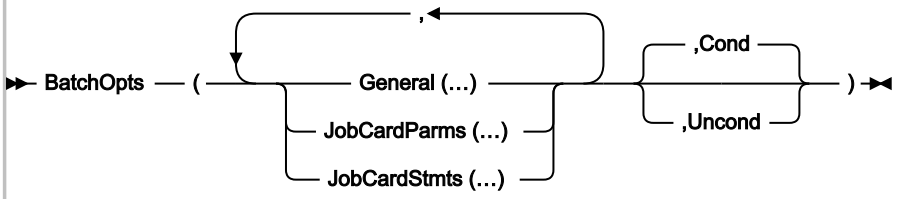
- BatchOpts on page 116
- GlobalExclude on page 608
- HistCols on page 609
- InitHist on page 609
- InteractiveExitPromptSeconds on page 610

This option is not included in the section of the fault analysis report that shows options in effect.

BatchOpts

Use the BATCHOPTS suboption to specify alternatives to the default settings in the Batch Reanalysis Options display. See Batch reanalysis options on page 149.

Figure 313. Syntax



BatchOpts COND and UNCOND suboptions

You can specify the COND and UNCOND suboptions at each level of the BATCHOPTS suboption:

- At the highest level, on the BATCHOPTS suboption
- At the intermediate level, on the GENERAL, JOBCARDPARMS, and JOBCARDSTMTS suboptions
- At the lowest level, on individual settings, for example JCLEDIT or MSGCLASS

COND (conditional) means the selected value is used as the default setting on the display if the user has not already specified a different setting. That is, it applies to a first-time user, or whenever the user clears the setting on the display.

UNCOND (unconditional) means the selected value is used as the current setting on the display at the start of any Fault Analyzer ISPF interface session, regardless of the previous setting. During the session, the setting can be changed to any other valid value, which remains in effect until the next session.

COND is the default at all levels, unless UNCOND is specified at a higher level. That is, the higher-level specification is by default propagated to each lower level.

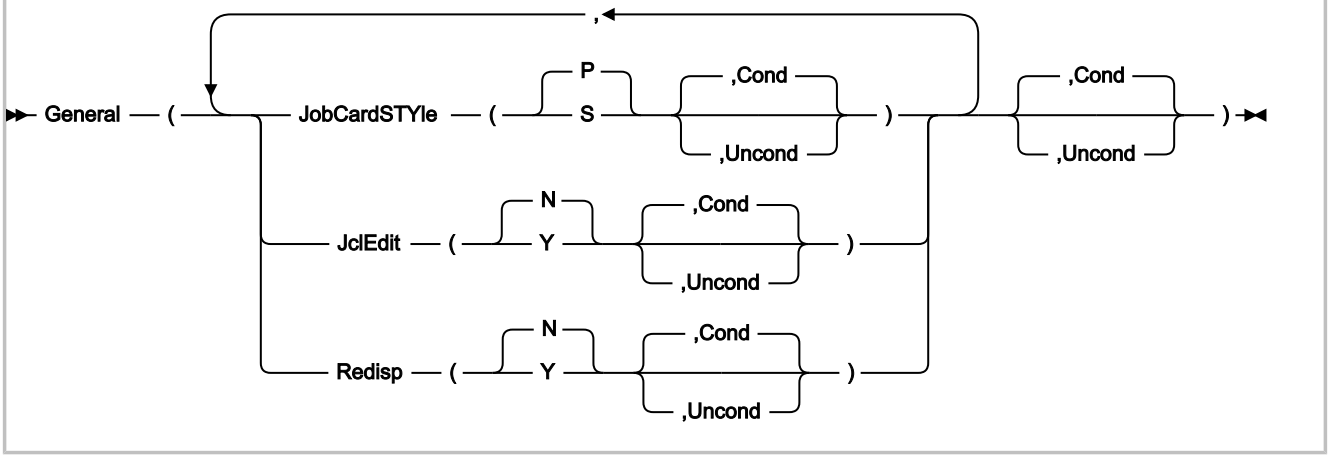
In the following example:

```
FAISPFOPTS(BATCHOPTS(GENERAL(JOBCARDSTYLE(P),JCLEDIT(Y),JOBCARDSTMTS(<stmt1>,<stmt2>),
    JOBCARDPARMS(MSGCLASS(A),COND),UNCOND))
```

- `JOBCARDPARMS(MSGCLASS(A))` is conditional.
- `GENERAL(JOBCARDSTYLE(P)`, `JCLEDIT(Y)`, and `JOBCARDSTMTS(<stmt1>,<stmt2>)` are unconditional.

BatchOpts General suboptions

Figure 314. Syntax



The GENERAL suboptions apply to the **General Options** section of the Batch Reanalysis Options display.

JOBCARDSTYLE(P | S [, COND | UNCOND]))

JCSTY(P | S [, C | U])

Specifies the value for the **Job card style** field in the Batch Reanalysis Options display:

P

Parameters

S

Statements

If not specified, the default is P.

JCLEDIT(Y | N [, COND | UNCOND]))

JE(Y | N [, C | U])

Specifies the value for the **Display panel to edit generated JCL** field in the Batch Reanalysis Options display:

Y

Yes

N

No

If not specified, the default is N.

REDISP(Y | N [, COND | UNCOND]))

R(Y | N [, C | U])

Specifies the value for the **Redisplay this panel before each reanalysis** field in the Batch Reanalysis Options display:

Y

Yes

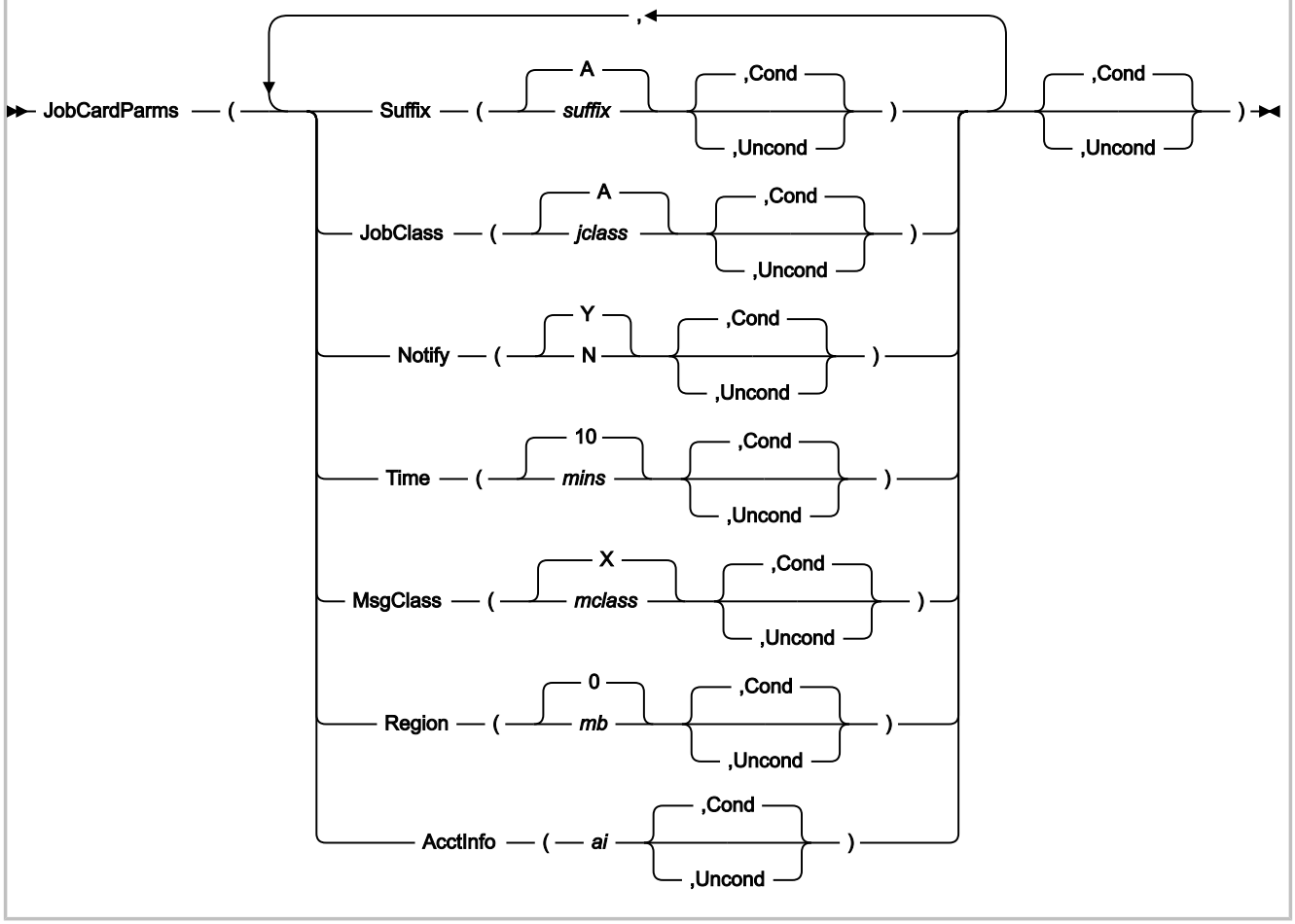
N

No

If not specified, the default is N.

BatchOpts JobCardParms suboptions

Figure 315. Syntax



The JOBCARDPARMS suboptions apply to the **Job Card Parameters** section of the Batch Reanalysis Options display.

SUFFIX(*suffix*[, COND | UNCOND]))

S(*suffix*[, C | U])

Specifies the value of the **Job name suffix** field in the Batch Reanalysis Options display:

suffix

A-Z, 0-9, @, #, or $

If not specified, the default is A.

JOBCLASS(*jclass*[, COND | UNCOND]))

JC(*jclass*[, C | U])

Specifies the value for the **Job class** field in the Batch Reanalysis Options display:

jclass

A-Z or 0-9

If not specified, the default is A.

NOTIFY(Y | N [, COND | UNCOND]))

N(Y | N [, C | U])

Specifies the value for the **Job notify** field in the Batch Reanalysis Options display:

Y

Yes

N

No

If not specified, the default is Y.

TIME(*mins*[, COND | UNCOND]))

T(*mins*[, C | U])

Specifies the value for the **Job time minutes** field in the Batch Reanalysis Options display:

mins

0-99

If not specified, the default is 10.

MSGCLASS(*mclass*[, COND | UNCOND]))

MC(*mclass*[, C | U])

Specifies the value for the **Message class** field in the Batch Reanalysis Options display:

mclass

A-Z or 0-9

If not specified, the default is X.

REGION(*mb*[, COND | UNCOND]))

R(*mb*[, C | U])

Specifies the value for the **Region megabytes** field in the Batch Reanalysis Options display:

mb

0-2047

If not specified, the default is 0.

ACCTINFO(*ai*[, COND | UNCOND]))

AI(*ai*[, C | U])

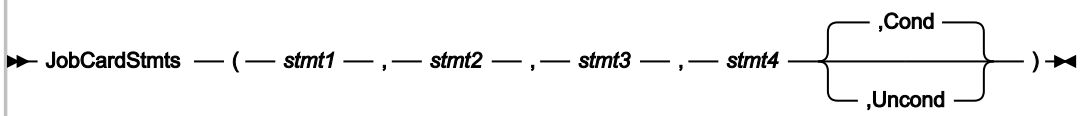
Specifies the value for the **Accounting info** field in the Batch Reanalysis Options display:

ai

Up to 50 characters.

BatchOpts JobCardStmts suboptions

Figure 316. Syntax



You can specify up to four comma-delimited JCL statements.

- If a statement is not specified (for example, if you need only 3 statements), you must still specify the delimiting comma of the omitted statement.
- If a statement contains blanks or commas, enclose it in either single quotes or double quotes. If the statement already contains quotes of the same type, double them.

  For example, to create the JCL statement:

  ```
  //MYJOB JOB (123),'X Y Z',
  ```

  You can specify any of the following:

  ```
  '//MYJOB JOB (123),"X Y Z",'
  ```

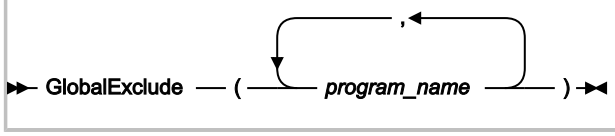
  ```
  "//MYJOB JOB (123),'X Y Z',"
  ```

  ```
  '//MYJOB JOB (123),''X Y Z'','
  ```

  In the last example, two single quotes surround `XYZ`.

GlobalExclude

Figure 317. Syntax



The GlobalExclude option can be used to provide an installation-wide default list of program names, for which no prompt for side files should be performed during interactive reanalysis.

The program names must be valid PDS or PDSE member names, but can include the wildcard characters '*' (zero, one or more characters) and '%' (a single required character).

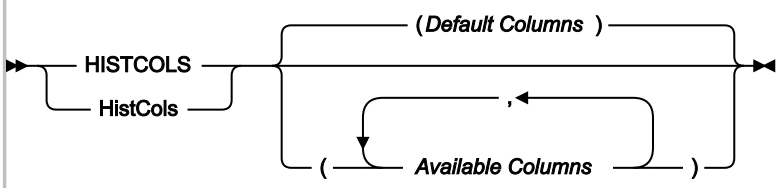
The specified program names are not case-sensitive.

The following are examples of valid program name specifications:

```
*XMAI*
PAYROLL0
SELOPT%
SUBRTN*
TZ%%C*
```

HistCols

Figure 318. Syntax

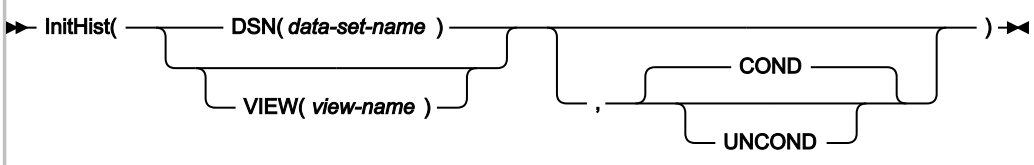


The FAISPFopts(HistCols(...)) option customizes the columns of information that are shown on the Fault Entry List display of the Fault Analyzer ISPF interface. By adding this option to the IDICNFxx config member, an installation can provide a customized Fault Entry List display for all users. Individual users can change their own Fault Entry List display regardless of the specification of this option.

For more information about customizing the Fault Entry List display, and for explanations of the individual columns that can be specified using the FAISPFopts(HistCols(...)) option, see Fault entry list column configuration on page 65.

InitHist

Figure 319. Syntax



The InitHist option can be used to specify that a given history file or view should initially be displayed by all users of the Fault Analyzer ISPF interface.

DSN(*data-set-name*)

Specifies the data set name of a history file that is to be used for the initial ISPF interface display.

VIEW(*view-name*)

Specifies the name of a view member in the IDIVIEWS concatenation that is to be used for the initial ISPF interface display.

COND

The specified history file or view is only used for the initial Fault Entry List display shown when starting the Fault Analyzer ISPF interface, if the user has not previously selected a different history file or view.

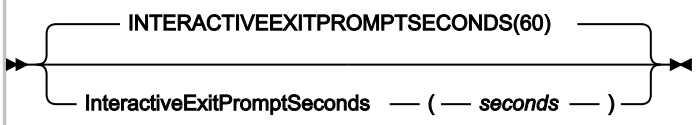
This value is the default.

UNCOND

The specified history file or view is always used for the initial Fault Entry List display shown when starting the Fault Analyzer ISPF interface, regardless of whether the user has previously selected a different history file or view.

InteractiveExitPromptSeconds

Figure 320. Syntax



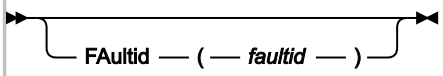
The InteractiveExitPromptSeconds option specifies the minimum number of elapsed seconds that the interactive reanalysis must have taken before the exit prompt panel is shown when leaving the interactive report display. If the interactive reanalysis took less than the specified number of seconds, then no prompt is shown. Otherwise, the user is requested to press the Enter key to confirm exit from the interactive report.

The valid *seconds* range is 0 - 99999:

- If 0 is specified, then the prompt is displayed unconditionally.
- If 99999 is specified, then the prompt is never displayed.

FaultID

Figure 321. Syntax



The FaultID option identifies a fault entry by its assigned fault identifier (*faultid*), and is used when performing fault reanalysis. The fault identifier must be in the format:

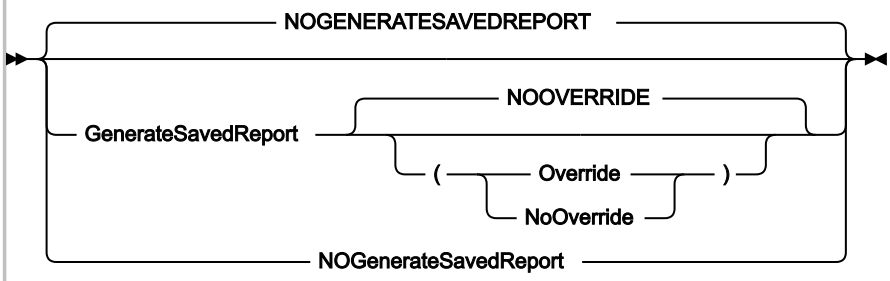
where *ccc* is the assigned history file prefix (1 to 3 characters) and *nnnnn* is a 5-digit decimal fault number.

Note:

1. The fault identifier F00000 signifies a 'null' ID that cannot be used to select a fault entry for reanalysis. This identifier might be assigned to a fault entry if no available fault numbers exist.
2. No default setting of the FaultID option is appropriate. It should only be used if building your own JCL for batch reanalysis of a history file fault entry. If performing reanalysis through the Fault Analyzer ISPF interface, then the correct use of this option is automatic.

GenerateSavedReport

Figure 322. Syntax



The GenerateSavedReport option can be used to create or replace a saved report in a fault entry during reanalysis.

If the Override suboption is used, then a new saved report is generated, regardless of whether one already exists or not.

If the NoOverride suboption is used, then a saved report is generated only if one does not already exist. This value is the default if the GenerateSavedReport option is specified without a suboption.

If the GenerateSavedReport option is in effect, then source information data sets (compiler listings or side files) used during real-time processing are automatically included during the reanalysis.

A practical application of this option might be to use it with automated batch reanalysis of newly created fault entries, which due to the DeferredReport option having been in effect during real-time processing for performance reasons, do not already include a saved report. By, for example, submitting such a batch reanalysis job from a Notification user exit, a saved report can be made to exist prior to users issuing the 'V' or 'S' line command from the Fault Entry List display. A sample Notification user exit which can be used for this purpose has been provided in Example 4 on page 505.

If the GenerateSavedReport option is used with batch analysis of an MVS™ dump data set specified using the DumpDSN option, then this option causes a fault entry to be created in the current history file, subject to the user's access authorization. If the fault entry is to be created in a history file, other than the default history file, then the history file can be designated using any of the normal methods, such as specifying an IDIHIST DD statement or using the DataSets(IDIHIST(*dsn*)) option. If a fault entry is successfully created, then message IDI0164I on page 732 is issued.

This option is only applicable to reanalysis. It is ignored for all other modes of execution.

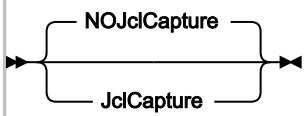
This option is included in the section of the fault analysis report that shows options in effect only if reanalysis with either GenerateSavedReport(Override) or GenerateSavedReport(NoOverride) in effect.

Include

See Exclude/Include on page 595.

JclCapture

Figure 323. JclCapture syntax



Use the JCLcapture option to specify that Fault Analyzer is to attempt to obtain the JCL of the abending job and save it in the fault entry. Only JES2 is supported. If successful, the JCL is available through a point-and-shoot field in the interactive reanalysis report "Abend Job Information" section.

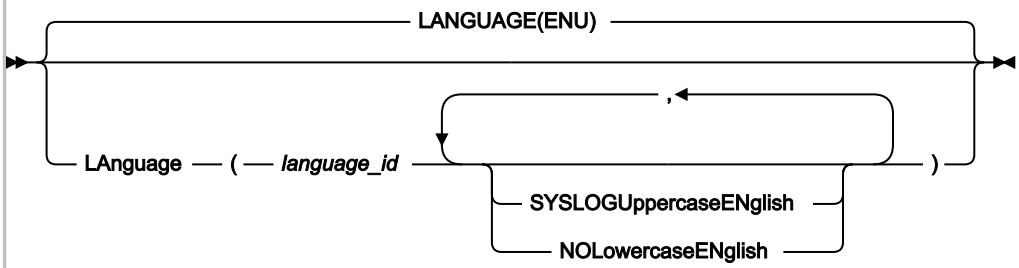
The default is NOJclCapture.

This option is applicable only to real-time analysis. For all other modes of execution, it is ignored.

This option is included in the section of the fault analysis report that shows options in effect only if real-time analysis with JCLcapture is in effect.

Language

Figure 324. Syntax



The Language option specifies the national language ID which is used to select appropriate language-dependent messages.

The following language IDs are permitted:

ID

Language

ENU

U.S. English (default)

JPN

Japanese (available only if the Japanese feature of Fault Analyzer is installed)

You can specify the following optional suboptions:

SyslogUppercaseEnglish

Where possible, causes all WTO messages to be in uppercase English only.

This suboption can be abbreviated to SYSLOGUEN.

NoLowercaseEnglish

Where possible, folds English lowercase characters to uppercase in all output written by Fault Analyzer. This suboption can, for example, be used with 3270 terminals in 'KANA' mode to prevent lowercase English characters from being unreadable.

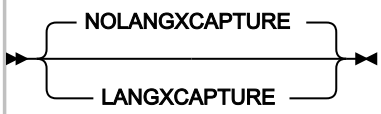
This suboption can be abbreviated to NOLEN.

If the Language option is specified in the IDICNFxx parmlib configuration member, then it should be the first option that is specified. This permits the maximum number of language-dependent messages to be issued by Fault Analyzer.

Note: If the Fault Analyzer IDIS subsystem is used, then any Language option specified by the user is ignored as far as explanations for abends, messages, reason codes, and similar, are concerned. These explanations are always provided in accordance with the Language option in effect for the IDIS subsystem. This is because the IDIS subsystem is used to cache explanations in order to improve performance and reduce I/O.

LangxCapture

Figure 325. Syntax



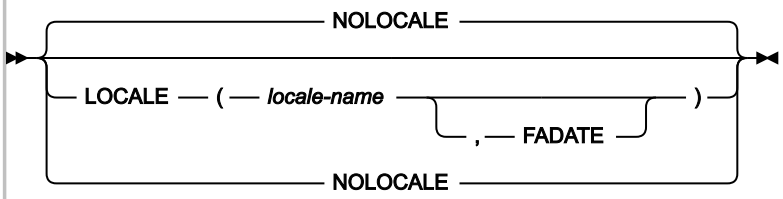
The LangxCapture option writes source-level debugging information from compiler listings or side files to the fault entry. Once the source-level debugging information is captured in the fault entry, you no longer need the compiler listings or side files.

This option has no effect if the DeferredReport option or the NoSource option is specified. Mismatching side files are not captured. The LangxCapture option is included in the section of the fault analysis report that shows options in effect only if interactive reanalysis or real-time analysis with LangxCapture is in effect.

From within the interactive reanalysis report, you can view captured source for the current fault entry from the Services menu (see Services-> LANGP Side File Formatting Utility on page 96). You can also set deferred breakpoints.

Locale

Figure 326. Syntax



The Locale option specifies the locale to be used for cultural environment-dependent presentation. Affected are things like date and time formatting, collating sequences of sorted information, and determination of non-printable characters, which are shown as periods.

The locale name that is specified as *locale-name* can be one of those supplied with z/OS® C/C++ for the setlocale() runtime function. A list of locale names can be found in the *z/OS® C/C++ Programming Guide*, "Appendix D. Locales Supplied with z/OS® C/C++".

If the optional FADATE suboption is specified, then all dates presented are in the standard Fault Analyzer format of YYYY/MM/DD and all time values in the standard Fault Analyzer 24-hour format of HH:MM:SS, regardless of the locale used.

Specifying the NoLocale option is the equivalent to specifying Locale(C,FADATE).

Note: The Fault Analyzer specification of *locale-name* overrides any IBM Application Delivery Foundation for z/OS Common Components Locale option specification in the IPVCNF00 parmlib member. For information about the IPVCNF00 parmlib member, see *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*.

LoopProtection

Figure 327. Syntax

This option is used to activate or deactivate the Fault Analyzer loop/wait protection feature. With this feature active (the default condition), Fault Analyzer terminates processing if the maximum expected elapsed time for any one of several internal checkpoints has been exceeded. If this termination happens, then message IDI0092S on page 715 is issued.

A user exit can deactivate the loop/wait protection feature by setting the ENV.LOOPPROTECTION_OPT field to N. For more information about this field, see ENV - Common exit environment information on page 653.

Note: Even with the NoLoopProtection option in effect, there is no guarantee that an analysis, that might otherwise be terminated with the IDI0092S on page 715 message, successfully completes. This possible lack of completion

is because the job might still be subject to normal installation-imposed MVS™ execution time limits, or be in a never-ending loop or wait condition that eventually requires the job to be canceled.

This option is only applicable to real-time analysis. For all other modes of execution, it is ignored.

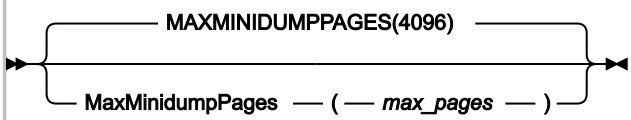
This option is only included in the section of the fault analysis report that shows options in effect if real-time analysis with NoLoopProtection in effect.

Note: If *max_pages* is exceeded during real time analysis, but Fault Analyzer was able to spill enough pages to the XDUMP, then the MaxMinidumpPages exceeded situation no longer exists.

MaxMinidumpPages

The MaxMinidumpPages option specifies the maximum number of 4K pages that can be saved as a minidump in each history file fault entry.

Figure 328. Syntax



If the use of an XDUMP data set has been enabled, some of the storage pages referenced during real-time analysis might be written to an associated XDUMP data set, while the rest are written to the fault entry minidump. If the use of an XDUMP data set has not been enabled, all referenced pages are written to the fault entry minidump.

A fault entry minidump is required to perform reanalysis of a fault entry created during real-time processing of Fault Analyzer.

If the number of referenced pages destined for the minidump exceeds *max_pages*, then a fault entry with a saved report is still created, but it will not include a minidump, nor will it have an XDUMP data set associated with it.

Note: If max_pages is exceeded during real time analysis, but Fault Analyzer was able to spill enough pages to the XDUMP, then the MaxMinidumpPages exceeded situation no longer exists.

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 applies to real time processing or batch reanalysis of MVS® dump data sets with option GenerateSavedReport in effect.

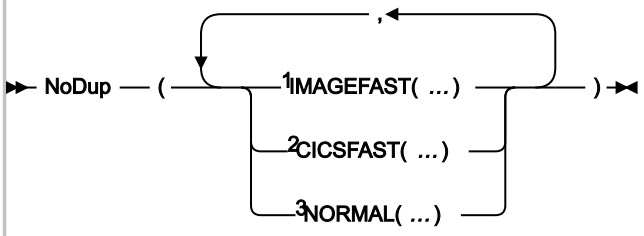
The section of the fault analysis report that shows options in effect includes this option and further indicate if the limit was exceeded, and if so, by how many pages. If it was exceeded, then message IDI0052I on page 706 is also issued.

The valid *max_pages* range is 0 - 524288.

Note: The use of an Analysis Control or End Processing user exit can effectively override the MaxMinidumpPages option in effect.

NoDup

Figure 329. Syntax



Notes:

[1] For details, see NoDup(ImageFast(...)) on page 618.
[2] For details, see NoDup(CICSFAST(...)) on page 621.
[3] For details, see NoDup(NORMAL(...)) on page 624.

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 621).

    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 618).

    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 624).

  The purpose of normal duplicate detection is to prevent multiple identical faults from being recorded separately in a history file, thus assisting with DASD space preservation as well as maintaining a history file containing only entries which represent unique problems.

Regardless of the method used to designate a fault as a duplicate, the fault entry of which it is a duplicate has its duplicate count incremented accordingly.

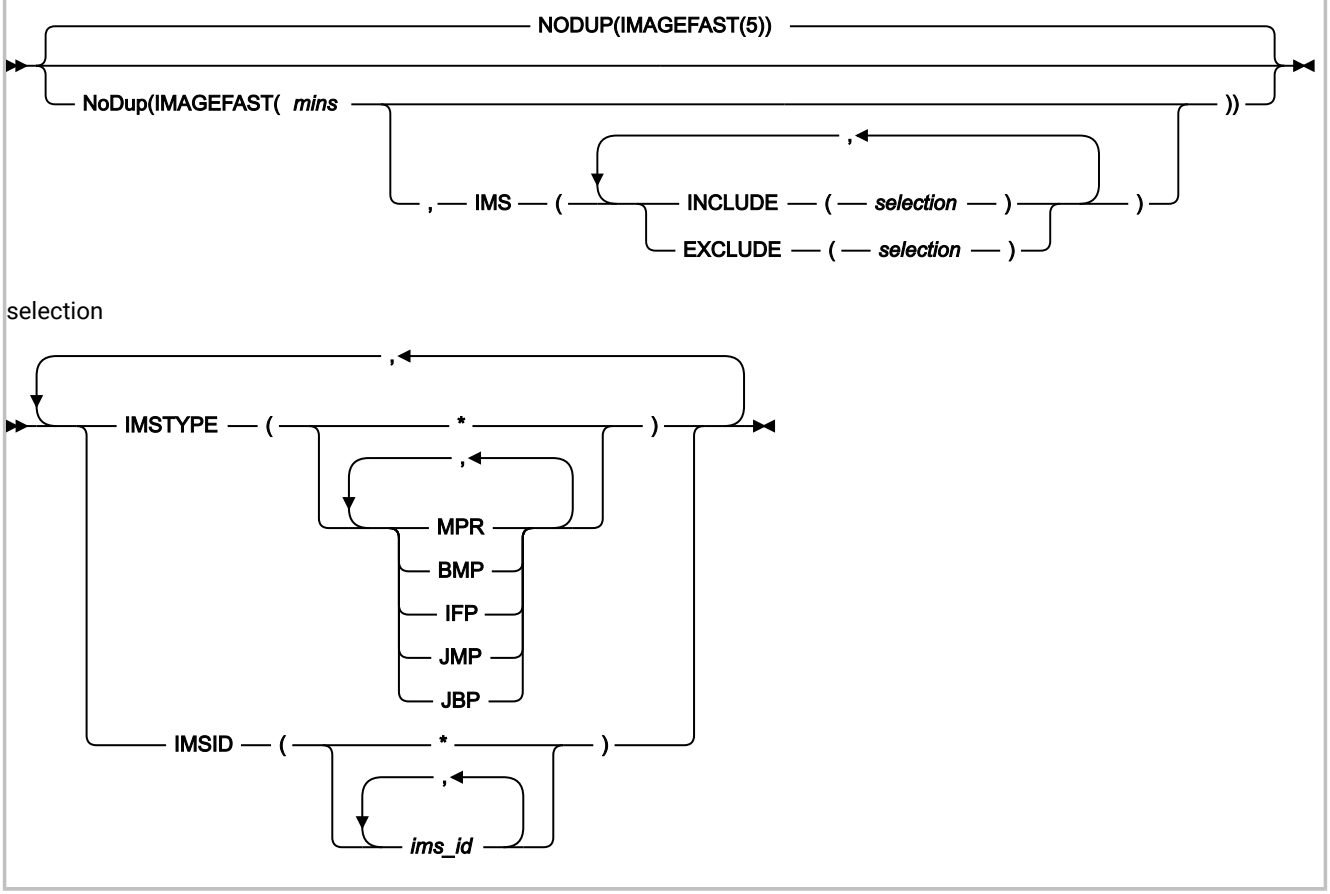
This option applies to the real-time analysis only.

If NoDup(NORMAL(*hours*, ...)) is in effect, with a non-zero value of *hours*, then this option is included in the section of the fault analysis report that shows options in effect. In addition, information about whether no duplicate fault was determined, or the fault ID of any identified duplicate, is provided. NoDup(CICSFAST) or NoDup(ImageFast) option information is not included, since no report is written if the associated duplicate criteria matched.

While the syntax for each duplicate subtype is shown separately in the following, they can be specified together in a single NoDup option if desired.

NoDup(ImageFast(...))

Figure 330. Syntax



This suboption is used to specify the number of minutes elapse time, since the last occurrence of a fault, during which subsequent invocations of Fault Analyzer for other faults *in the same MVS™ image* are deemed duplicates of the last fault, if they satisfy the appropriate fault characteristics criteria. This type of fault suppression is referred to as *"IMS™ fast duplicate fault suppression"*.

For details, see Using the Fault Analyzer IDIS subsystem on page 318.

The valid range of the minutes value specified in *mins* is 0 - 10080 (10080 is equivalent to one week). Specification of 0 minutes means that "IMAGEFAST" duplicate faults are not detected.

The default elapse time is 5 minutes.

In the IMS™ environment, a fault is considered a duplicate of another if the faults occurred within the specified elapse time (*mins*), and the following fault details are identical:

- IMS™ program name
- IMS™ subsystem ID
- Last IMS™ status code
- Last DB2® SQLCODE
- Failing program name
- Failing program compile date
- Offset to error in failing program
- Length of failing program
- Abend code
- User title specified for IDISNAP invocation
- Call chain

NoDup(ImageFast) signatures are kept in the IDIS subsystem on each MVS™ image. Changes to the NoDup(ImageFast) option take effect only after stopping and restarting the Fault Analyzer IDIS subsystem. For details, see Using the Fault Analyzer IDIS subsystem on page 318.

By default, all IMS™ jobs are eligible for IMS™ fast duplicate fault suppression. This approach is the equivalent of having specified the option

```
NoDup(ImageFast(minutes,IMS(INCLUDE(IMSTYPE(*),IMSID(*)))))
```

Use the INCLUDE or EXCLUDE suboptions of the NoDup(ImageFast(*minutes*,IMS(...))) option to restrict eligibility for IMS™ fast duplicate fault suppression. Here are the specification rules:

- An IMSTYPE criterion matches if the specified values include the IMS™ region type that is associated with the current fault.

  For example, if the following option is specified:

  ```
  NoDup(ImageFast(5,IMS(EXCLUDE(IMSTYPE(MPR,BMP,IFP)))))
  ```

  and the current fault is an IMS™ BMP region, then the IMSTYPE criterion matches, and the current fault deemed ineligible for IMS™ fast duplicate fault suppression.
- An IMSID criterion matches if one or more of the specified values for *ims_id* match the IMS™ ID associated with the current fault.

  For example, if the following option is specified:

```
NoDup(ImageFast(5,IMS(EXCLUDE(IMSID(ABC1,ABC2,ABC3)))))
```

and the current fault is associated with the IMS™ subsystem ID ABC2, then the IMSID criterion matches, and the current fault deemed ineligible for IMS™ fast duplicate fault suppression.

- Each specification of an INCLUDE or EXCLUDE suboption is tested against the current fault and, if all criteria of the specified suboption match, changes the eligibility state accordingly.

  For example, if the following option is specified:

  ```
  NoDup(ImageFast(5,IMS(EXCLUDE(IMSTYPE(MPR,BMP),IMSID(ABC1,ABC2,ABC3)))))
  ```

  and the current fault is an IMS™ BMP region with subsystem ID ABC4, then the IMSTYPE criterion matches, but the IMSID criterion does not, resulting in the EXCLUDE criteria not matching.

  Conceptionally, IMSTYPE or IMSID values can be considered logically OR'ed, while INCLUDE or EXCLUDE suboptions can be considered logically AND'ed, in order to determine the resulting match status. If | represents a "logical OR" operation, and & represents a "logical AND" operation, then the previous option specification could be interpreted as

  ```
  NoDup(ImageFast(5,IMS(EXCLUDE(IMSTYPE(MPR | BMP) & IMSID(ABC1 | ABC2 | ABC3)))))
  ```

  A single INCLUDE or EXCLUDE suboption should only ever contain a single IMSTYPE and/or a single IMSID criterion, since a match is otherwise not possible.
- Any number of INCLUDE or EXCLUDE suboptions can be specified. Specifying multiple INCLUDE or EXCLUDE suboptions within a single NoDup(ImageFast(*minutes*,IMS(...))) option is equivalent to specifying these as separate options.

  For example, specifying

  ```
  NoDup(ImageFast(5,IMS(INCLUDE(IMSTYPE(MPR),IMSID(ABC1)))))
  ```

  is equivalent to specifying

  ```
  NoDup(ImageFast(5,IMS(INCLUDE(IMSTYPE(MPR)))))
  NoDup(ImageFast(5,IMS(INCLUDE(IMSID(ABC1)))))
  ```

  Note: If different, only the last specified value of *mins* takes effect.

- A wildcard (*) can be used as complete substitution for a value, that is, with no characters preceding the asterisk.
- Processing of INCLUDE or EXCLUDE suboptions occur in the order specified, and in accordance with the hierarchy of options sources defined in Options on page 571. For example, the IDICNFxx parmlib member is read prior to any IDIOPTS user options file that was used.
- The most generic criteria should be specified first, followed by more specific ones.

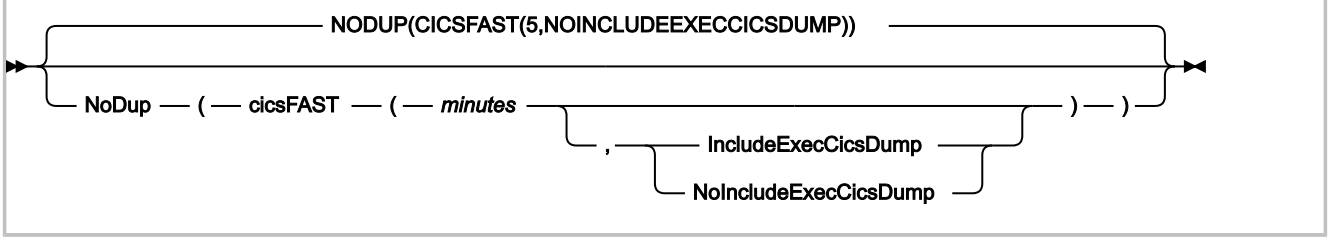
  For example, if an installation wants to use IMS™ fast duplicate fault suppression for only IMS™ MPR jobs using subsystem IDs other than ABC1, then specifying the following options would achieve this:

  ```
  NoDup(ImageFast(5,IMS(EXCLUDE(IMSTYPE(*)    /* Exclude everything */
                        INCLUDE(IMSTYPE(MPR)) /* Include only IMS MPR regions */
                        EXCLUDE(IMSID(ABC1))  /* Exclude subsystem ID ABC1 */
                        )))
  ```

When a fault is deemed a duplicate of another based on the fault characteristics and the NoDup(ImageFast(...)) option in effect, the fault analysis is skipped, the duplicate count associated with the original fault is incremented by one, and message IDI0121I on page 722 is issued.

NoDup(CICSFAST(...))

Figure 331. Syntax

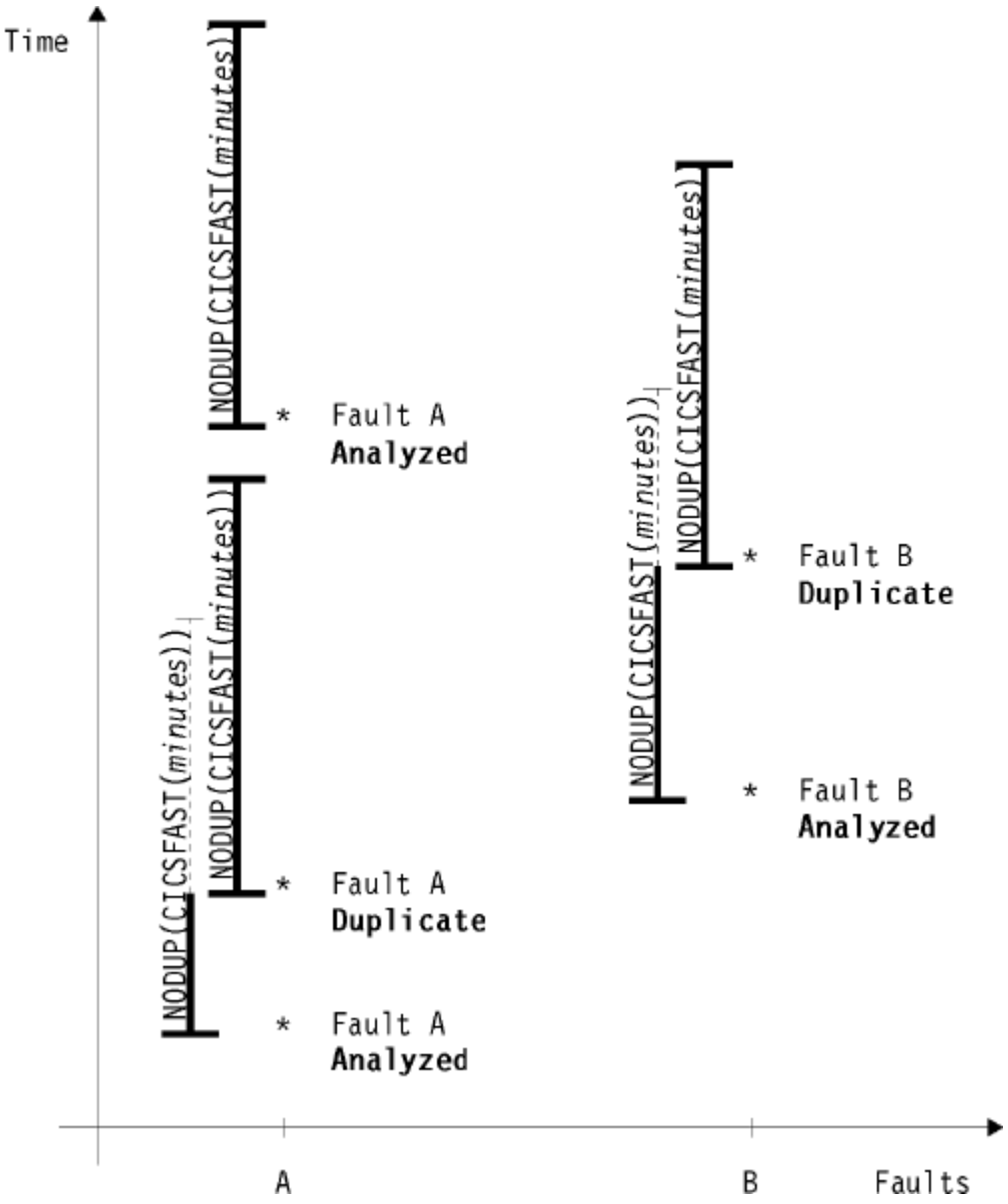


This suboption is used to specify the number of minutes elapse time, since the last occurrence of a fault, during which subsequent invocations of Fault Analyzer for other faults *in the same job step* are deemed duplicates of the last fault if they satisfy the appropriate fault characteristics criteria.

An illustration of this type of processing is shown in Figure 332: NoDup(CICSFAST) illustration on page 622, 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 332. NoDup(CICSFAST) illustration



This option is currently used under CICS® to prevent multiple identical transaction abends that occur within a short period of time from all being analyzed by Fault Analyzer with subsequent risk of exhausting system resources.

The valid range of *minutes* is 0 - 10080 (10080 is equivalent to one week). Specification of 0 minutes means that "CICSFAST" duplicate faults are not detected.

The default elapse time is 5 minutes.

In the CICS® environment, a fault is considered a duplicate of another if the faults occurred within the specified elapse time (*minutes*), and the following fault details are identical:

- Transaction IDs
- CICS® abend codes
- Failing program names
- Request IDs
- System and user sense codes
- Operating system abend codes
- Offsets to the point of error
- PSWs on entry to abend

NoDup(CICSFAST) signatures are kept in the CICS® region.

All fault details are obtained from the transaction abend control block (TACB), except for the transaction IDs. Only TACB fields that are valid for both TACBs are included in the duplicate comparison.

By default, invocations of Fault Analyzer using the EXEC CICS® DUMP command are not subject to NoDup(CICSFAST) duplicate determination. However, if the IncludeExecCicsDump suboption is specified, then NoDup(CICSFAST) duplicate determination is also performed when using the EXEC CICS® DUMP command:

- If a TACB is available, then the duplicate determination is performed on the same basis as for CICS® abends shown above (with the CICS® dump codes used instead of the CICS® abend codes).
- If a TACB is not available, then the duplicate determination is performed on the basis of:
  - Transaction IDs
  - CICS® dump codes
  - Failing program names

When a fault is deemed a duplicate of another based on the fault characteristics and the NoDup(CICSFAST(…)) option in effect, the writing of the fault entry is suppressed, the duplicate count associated with the last fault is incremented by one, and message IDI0066I on page 710 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 408.

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 402.

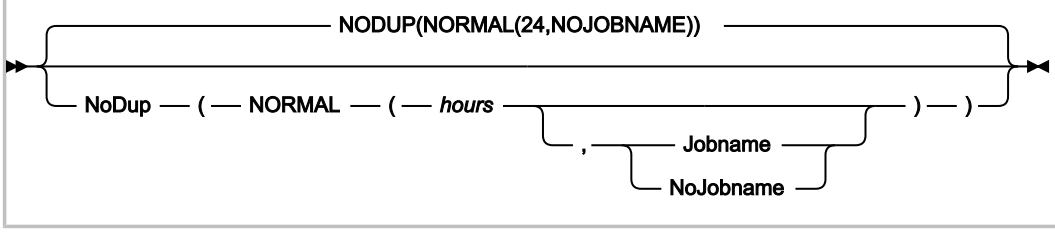
Changing the NoDup(CICSFAST) option

Typically, an IDIOPTS DD statement is used in CICS® procedures to facilitate changes to options without the need to cycle CICS®. However, as far as the NoDup(CICSFAST) option is concerned, a limitation exists. The actual processing of the IDIOPTS data set is performed by the main Fault Analyzer module, IDIDA. This module is the main program that runs in the MVS™ attached subtask. The NoDup(CICSFAST) processing is done in the 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 on page 710), the attach of IDIDA and the reading of the options data set does *not* occur. Another IDIDA attach must happen before the NoDup(CICSFAST) option change is reflected down to the exit for NoDup(CICSFAST) action. This means either waiting for another unique fault (forced by creating a new abend with CECI to perform an EXEC CICS® ABEND) or the NoDup(CICSFAST) time-out to occur.

Alternatively, you can cause changes to this option to take effect immediately by using the CFA transaction to uninstall and reinstall all CICS invocation exits. See Controlling CICS transaction abend analysis on page 400.

NoDup(NORMAL(...))

Figure 333. Syntax



This suboption is used to specify the number of hours elapse time, inside of which invocations of Fault Analyzer, *using the same history file*, can be deemed duplicates of a previously recorded fault. This mechanism is the standard duplicate fault detection mechanism used, for example, for abending batch jobs.

The valid range of *hours* is 0 - 168 (168 is equivalent to one week). Specification of 0 hours means that "NORMAL" duplicate faults are not detected.

The default elapse time is 24 hours.

A fault is considered a duplicate of another already recorded in the same history file if the faults occurred within the specified elapse time (*hours*), the point-of-failure has been determined, and the following fault details are identical:

- Initial abend codes
- CICS® transaction IDs (applicable to CICS® transaction faults only)
- Point-of-failure module names
- Point-of-failure CSECT (or program) names
- Point-of-failure offsets from start of the CSECT (or program)

- Point-of-failure module link-edit date and time (if available for the current fault as well as in the fault entry for the duplicate fault)
- User titles.
- The jobname (only if the Jobname suboption is specified; otherwise, the jobname is not included in the duplicate fault determination)

NoDup(NORMAL) signatures are based on the existing fault entries in a history file.

When a fault is deemed a duplicate of another based on the fault characteristics and the NoDup(NORMAL(*hours*)) option in effect, the default is to suppress both the writing of the fault entry and any dumps for the duplicate fault. However, the final decision on suppression can be overridden by an End Processing user exit.

When it has been determined that the fault is a duplicate of another fault, the duplicate count that is associated with the existing fault is incremented by one and message IDI0044I on page 705 is issued.

Trace information

By adding an IDITRACE DDname to your job, Fault Analyzer can provide you with trace information that might help you understand why a particular fault is, or is not, being considered a "normal" duplicate of another fault during real-time analysis. For example:

```
//IDITRACE DD SYSOUT=*
```

(See IDITRACE under CICS on page 402 for an alternative method of activating this trace under CICS®.)

Note: Only NoDup(NORMAL) trace information is available, not NoDup(CICSFAST).

The trace information is written to the IDITRACE DDname destination. An example of a NoDup option-processing trace follows:

Figure 334. Sample NoDup option-processing trace

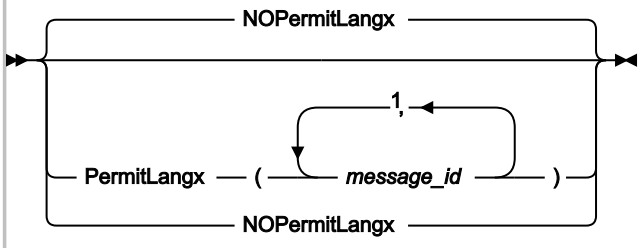
```
NoDup(NORMAL(24)) option processing match values:
  Abend Code . . . . . : S0C7
  CICS Transaction ID. . . . . : n/a
  Module Name. . . . . : IDISCBL1
  CSECT Name . . . . . : IDISCBL1
  Offset . . . . . : X'3D4'
  Module Link-Edit Date. . . . : 2002/05/06
  Module Link-Edit Time. . . . : 15:56:35
Fault ID F00615 module link-edit time 15:55:09 did not match
No duplicate fault exists
```

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

Figure 335. Syntax



Notes:

[1] Either comma or blank character is permitted as delimiter.

Some compilers issue messages during the compilation which result in a return code greater than 4, while still producing a valid object module. However, since IDILANGX processing by default treats all messages that cause a return code greater than 4 as an error, this option can be used to specify a list of message IDs which should be ignored when found in the listing from a compilation ending with a return code greater than 4.

The PermitLangx option is applicable to COBOL or PL/I compiler listings only.

PL/I message IDs must be specified with a length of 8 characters. For example, if the following message was written to the compiler listing

```
Message     Statement  Message Description
IBM1352I E         11  The statement element PUT is invalid. The statement
                       will be ignored.
```

then specifying the PermitLangx option as follows allows IDILANGX processing to be performed:

```
PermitLangx(IBM1352I)
```

OS/VS COBOL message IDs must be specified with a length of 8 characters. For example, if the following message was written to the compiler listing

```
CARD   ERROR MESSAGE
20     IKF1150I-C     ILLEGAL CHARACTER IN COLUMN 7, BLANK ASSUMED.
```

then specifying the PermitLangx option as follows allows IDILANGX processing to be performed

```
PermitLangx(IKF1150I)
```

All other COBOL message IDs (other than OS/VS COBOL) must be specified with a length of 9 characters. For example, if the following message was written to the compiler listing

```
LineID  Message code  Message text
     1  IGYDS0017-E   "IDENTIFICATION" should begin in area "A".  It was
                      processed as if found in area "A".
```

then specifying the PermitLangx option as follows allows IDILANGX processing to be performed

```
PermitLangx(IGYDS0017)
```

Using wildcards

To specify that all messages of a given severity should be permitted, the following notation can be used:

```
XXX-C
```

where

XXX

The message prefix.

C

The message severity level.

The following example shows how all error-level messages (E) can be permitted with Enterprise PL/I (IBM®):

```
PermitLangx(IBM-E)
```

The message severity to which the specified option applies, is the one that follows the message ID in the compiler listing. For example, in the case of Enterprise PL/I message IBM1352I, the compiler listing might show:

```
Message     Statement  Message Description
IBM1352I E         11  The statement element SKIP is invalid. The statement
                       will be ignored.
```

In this case, the message severity level is E, which generally implies a return code of 8.

Only messages with the specified message severity level are affected.

The following example shows how to specify that both Enterprise PL/I and Enterprise COBOL error-level messages should be ignored:

```
PermitLangx(IBM-E,IGY-E)
```

Combining multiple option specifications

Only the last specification of the PermitLangx option is used. That is, multiple specifications are not cumulative. For example, if the following options are specified in the IDICNFxx parmlib member

```
PermitLangx(IBM-E)
PermitLangx(IGYDS0017,IBM1352I)
PermitLangx(IGY-E)
```

then only the option

```
PermitLangx(IGY-E)
```

remains in effect.

To retain all three options that are specified, combine all suboptions into a single option specification, for example:

```
PermitLangx(IBM-E,
            IGYDS0017,IBM1352I,
            IGY-E)
```

Maximum length of suboptions

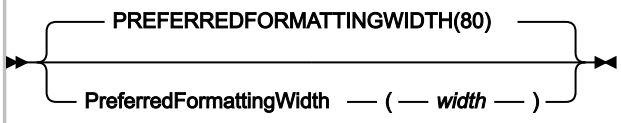
The "normalized length", which is defined as the sum of the lengths of all message IDs, plus the number of message IDs minus one, must not exceed 75. This length is equivalent to the most compact specification possible, with only a single delimiter separating suboptions and no unnecessary blank-padding used. For example, the normalized length of

```
PermitLangx(IKF1150I     IBM-E,IGY-E   )
```

is 20. This value is calculated as follows: The total number of characters in the message IDs is 8 + 5 + 5 = 18. Since 3 message IDs were specified, the total "normalized length" becomes 18 + 3 - 1 = 20.

PreferredFormattingWidth

Figure 336. Syntax



This option specifies the real-time or batch reanalysis report line width to be used for flowing paragraph text and hexadecimal storage formatting. Other parts of the report are not affected by this option.

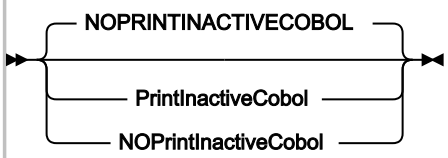
Hexadecimal storage formatting requires a minimum formatting width of 128 characters before changing from the default 16 bytes per line to 32 bytes per line.

The valid range of *width* is 80 - 132.

This option is not applicable to interactive reanalysis.

PrintInactiveCOBOL

Figure 337. Syntax



The PrintInactiveCOBOL option can be used with real-time or batch reanalysis to request that storage for inactive COBOL programs (programs that are not in the current save-area chain) is included in the analysis report.

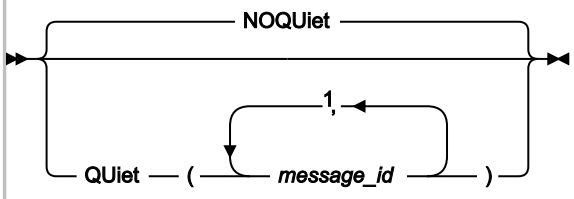
Note: Depending on the number and size of inactive COBOL programs, this option might increase the size of the report significantly.

When the NoPrintInactiveCOBOL option is in effect (the default), then storage for inactive COBOL programs is not included.

This option is not applicable to interactive reanalysis, which is always capable of displaying storage for inactive COBOL programs.

Quiet

Figure 338. Syntax



Notes:

[1] Either comma or blank character is permitted as delimiter.

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 on page 696.

If NoQuiet is specified, all write-to-operator messages can be issued.

The last specification of the Quiet or NoQuiet option is in effect. For example, if the IDICNFxx 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 by IBM® Support, while gathering documentation for a reported problem, might override this option.

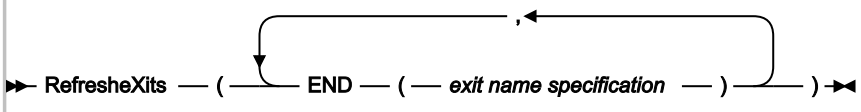
Some messages are not able to be suppressed using the Quiet option, such as message IDI0001I on page 695.

It is not possible to suppress S-level (severe) messages, or messages written by the Fault Analyzer IDIS subsystem.

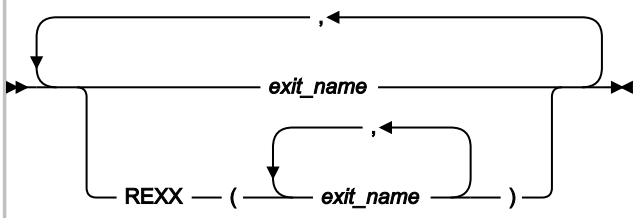
For changes to this option related to message numbers IDI0034I on page 703, IDI0066I on page 710, or IDI0118W on page 721 to take effect, uninstall and reinstall all CICS® invocation exits using the CFA transaction. See Controlling CICS transaction abend analysis on page 400.

RefreshExits

Figure 339. Syntax



exit name specification



The RefreshExits option specifies the types and names of user exits to be invoked during the first batch reanalysis of a dump registration fault entry. Any number of exit names can be specified for a given exit type, and all exits are attempted invoked.

Multiple specifications of the RefreshExits option are cumulative.

Exits can be either REXX EXECs or load modules:

- REXX EXECs must be specified as

  ```
  REXX(exit_name_1, exit_name_2, …)
  ```

  and be available via the IDIEXEC DDname.
- Load module exits must be available via the standard system search path (LPA, LINKLIST, or JOBLIB/STEPLIB JCL statement).

The possible exit types are:

END

End Processing user exit. This exit can be used to request suppression of the Fault Analyzer minidump or the update of the entire fault entry. For details, see End Processing user exit (Fault entry refresh) on page 497.

The exit name that is specified as *exit_name* can be any valid TSO/E REXX EXEC or load module name. However, certain names are reserved for special purposes:

NONE

The special name 'NONE' represents a 'null' exit that is not invoked and causes further attempts to invoke exits of the specified type to be terminated.

-DROPCNF-

The special name '-DROPCNF-' is used to drop exit specifications from the IDICNFxx parmlib member. For details, see Dropping IDICNFxx parmlib member user exit specifications on page 602.

If one or more exits have been specified via the RefreshExits option, information about the exits is written to the *"Options in Effect"* section of the analysis report. In this section, all specified exits are listed, and those of each type that were invoked (if any) are identified.

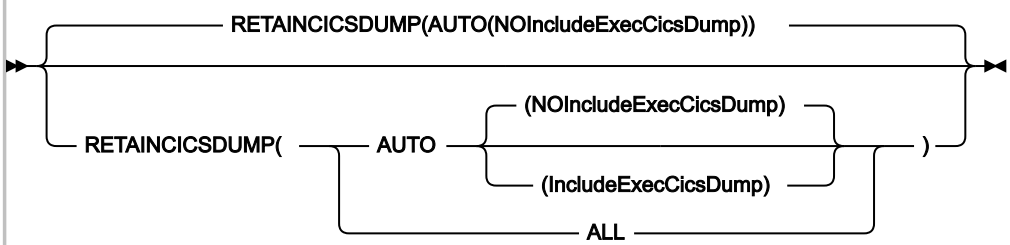
An example of the information written to the *"Options in Effect"* section of the analysis report follows:

```
Exits:

  The following user exits were specified via RefreshExits options.

  Type     Name     Type Invoked
  -------- -------- ---- ----------------------
  END      ABC1     LMOD Yes
```

This example indicates that a single End Processing user exit (ABC1) had been specified as a load module. This user exit was invoked.

RetainCICSDump

Figure 340. Syntax



The RetainCICSDump option specifies that a CICS® transaction dump should be retained based on the analysis success (AUTO) or unconditionally (ALL).

The analysis success (AUTO) setting means that if the Fault Analyzer analysis completes normally, then the transaction dump is suppressed. However, if there is a significant problem with the analysis, then the transaction dump is retained. AUTO(NOIncludeExecCicsDump) is used to prevent suppression of the CICS® transaction dump issued for an EXEC CICS® DUMP call, even if analysis is successful. This is the default. AUTO(IncludeExecCicsDump) is used to suppress all CICS® transaction dumps, including for EXEC CICS® DUMP calls, if the analysis is successful.

This option applies to real-time analysis of CICS® transaction faults only, and requires that Fault Analyzer is installed in either the XPCABND or XDUREQ global user exits. For changes to this option to take effect, uninstall and reinstall all CICS invocation exits using the CFA transaction. See Controlling CICS transaction abend analysis on page 400.

Note: Transaction dumps as a result of an EXEC CICS® DUMP command are not, by default, suppressed by Fault Analyzer. This lack of suppression is due to the SUPPRESSED response being passed back to the issuing application program, which if not handled correctly, can lead to AEXW abends.

This option does not affect the writing of the fault entry to the history file.

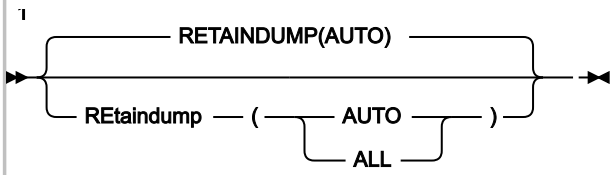
This option is included in the section of the fault analysis report that shows options in effect for real-time analysis of CICS® transaction faults only.

Note:

1. The use of an End Processing user exit can effectively override the RetainCICSDump option in effect.
2. To control the retention of MVS™ dump data sets, use the RetainDump option instead.
3. See Dump suppression on page 32 for more information about dump suppression.

RetainDump

Figure 341. Syntax



Notes:

[1] Either comma or blank character is permitted as delimiter.

The RetainDump option specifies that the Fault Analyzer IEAVTABX change options/suppress dump exit (IDIXDCAP) is to retain the SYSABEND dump, SYSMDUMP, or SYSUDUMP of the abending job step, based on the analysis success (AUTO) or unconditionally (ALL).

The analysis success (AUTO) setting means that if the Fault Analyzer analysis completes normally, then the MVS™ dump is suppressed. However, if there is a significant problem with the analysis, then the MVS™ dump is retained.

This option applies to real-time analysis of non-CICS transaction faults only, and is only applicable to the IEAVTABX change options/suppress dump exit, IDIXDCAP. This option does not affect the writing of the fault entry to the history file.

If an IEAVTABX_EXIT exit is installed ahead of IDIXDCAP, and this exit has requested, via its return code, that dump options should be changed or the dump suppressed, then Fault Analyzer honors the request regardless of options settings.

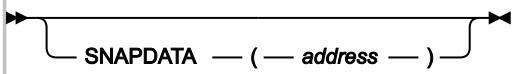
This option is included in the section of the fault analysis report that shows options in effect for real-time analysis of non-CICS transaction faults only.

Note:

1. The use of an Analysis Control or End Processing user exit can effectively override the RetainDump option in effect.
2. To control the retention of CICS® transaction dumps, use the RetainCICSDump option instead.
3. See Dump suppression on page 32 for more information about dump suppression.

Snapdata

Figure 342. Syntax



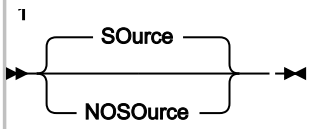
Use the Snapdata option to specify a storage address for the program SNAP interface (IDISNAP) to store the final version of the ENV data area in a user-specified data area. *address* specifies a storage address in 8-byte character format.

This option cannot be specified in the IDICNFxx PARMLIB configuration member or in the IDIOPTS DD statement.

For information about how to specify this option in a call to IDISNAP, see Using the SNAPDATA option on page 41. For information about the ENV data area, see ENV - Common exit environment information on page 653.

Source

Figure 343. Syntax



Notes:

[1] Either comma or blank character is permitted as delimiter.

The Source option is used to control whether real-time source code analysis is to be performed:

- If Source is in effect, then real-time source code analysis is performed as usual. This value is the default.
- If NoSource is in effect, then Fault Analyzer does not access any compiler listings or side files and source code information might not be available in the real-time report. Assuming that either a minidump or an associated MVS dump data set data set has been written, then it is still possible to provide source code information by performing reanalysis against the fault entry.

  This option might be useful as a means of improving real-time analysis performance in installations where a large number of compiler listing or side file data sets are used, and where performing reanalysis to provide the extra source code information is an acceptable alternative.

SpinIDIREPRT

Figure 344. Syntax

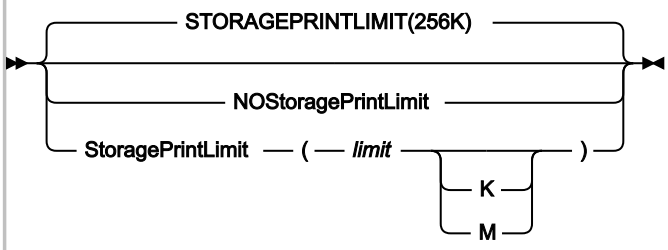
The NoSpinIDIREPRT option can be used with real-time analysis to request that dynamic allocation of the IDIREPRT DDname is not explicitly deallocated at the end of processing. Use this option, for example, to ensure that all JES spool data for a given real-time analysis is capable of being sent as a single file across to a different system using NJE.

When the SpinIDIREPRT option is in effect (the default), then dynamic allocation of the IDIREPRT DDname is explicitly deallocated at the end of processing.

This option is not applicable to batch or interactive reanalysis.

StoragePrintLimit

Figure 345. Syntax



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 17. StoragePrintLimit option behavior

Compiler listing or side file available[1]	StoragePrintLimit not exceeded	StoragePrintLimit exceeded
Yes	Complete event-related source view formatting	Addressable[2] event-related source view formatting
No	Complete event-related hex-dump formatting	Addressable[2] event-related hex-dump formatting

1. This includes source obtained from the load module due to TEST option usage.
2. Storage within +4 kilobytes of a general purpose register value.

To determine if the specified limit is exceeded, it is tested against the sum of all hex-dumped event-related associated storage areas and CICS® transaction storage areas being reported on, with due consideration given to overlapping storage ranges. The SystemWidePreferred(StorageAreas(Hex)) option is assumed in effect for the accumulation of storage ranges, regardless of its actual setting.

This option affects the real-time or batch reanalysis reports only; it does not affect the interactive reanalysis report. Even if used in real time, the full data is still available from reanalysis of the minidump, which is not reduced in size by the StoragePrintLimit option.

Valid specification of *limit*:

- Bytes: 0-2147483648
- Kilobytes: 0K-2097152K
- Megabytes: 0M-2048M

NoStoragePrintLimit is the equivalent to specifying StoragePrintLimit(2147483648).

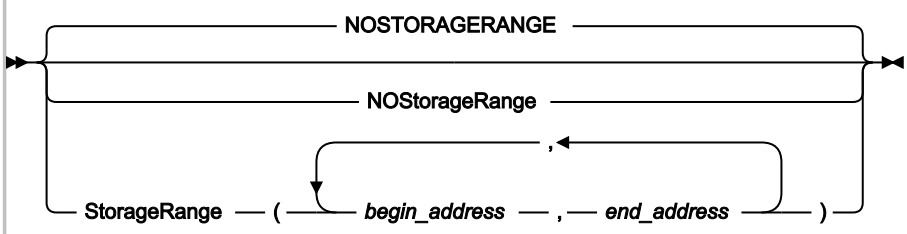
Information about the accumulated storage amount, which is used to determine whether or not the storage print limit in effect has been exceeded, is written to IDITRACE.

In the "Options in Effect" section of the report, this option is shown using the largest possible unit type regardless of the actual specification of the option. For example, if StoragePrintLimit(1024K) was specified, then the option is shown as StoragePrintLimit(1M). If instead StoragePrintLimit(1025) was specified, then the option is shown as StoragePrintLimit(1025) also, since 1025 is not an even number of megabytes or kilobytes.

Information about whether the storage print limit was exceeded or not is included in the "Options in Effect" section of the report. If the limit was exceeded, then the amount of storage by which the limit was exceeded is also shown.

StorageRange

Figure 346. Syntax



This option can be used to request that one or more specific areas of storage are shown in the analysis report, instead of the default event-associated or system-wide hex-dumped storage.

The primary usage of this option is expected to be with IDISNAP calls from application programs, where only a certain area (or areas) of storage is of interest. For example, a COBOL programmer can choose to call IDISNAP with a StorageRange option that specifies a particular variable in the program's working-storage section. Instead of the analysis report containing all of the program's associated storage, it instead contains only the area requested.

For information about how to specify this option in a call to IDISNAP, see Using the program SNAP interface (IDISNAP) on page 37.

If this option is in effect for real-time analysis, then the same specification is in effect during any subsequent reanalysis of the fault entry, without the ability to override the option.

The begin address is the address of the first byte of the storage area, while the end address is the address of the byte immediately following the last byte of the storage area. That is, the end address minus the begin address equals the length of the storage area to be included.

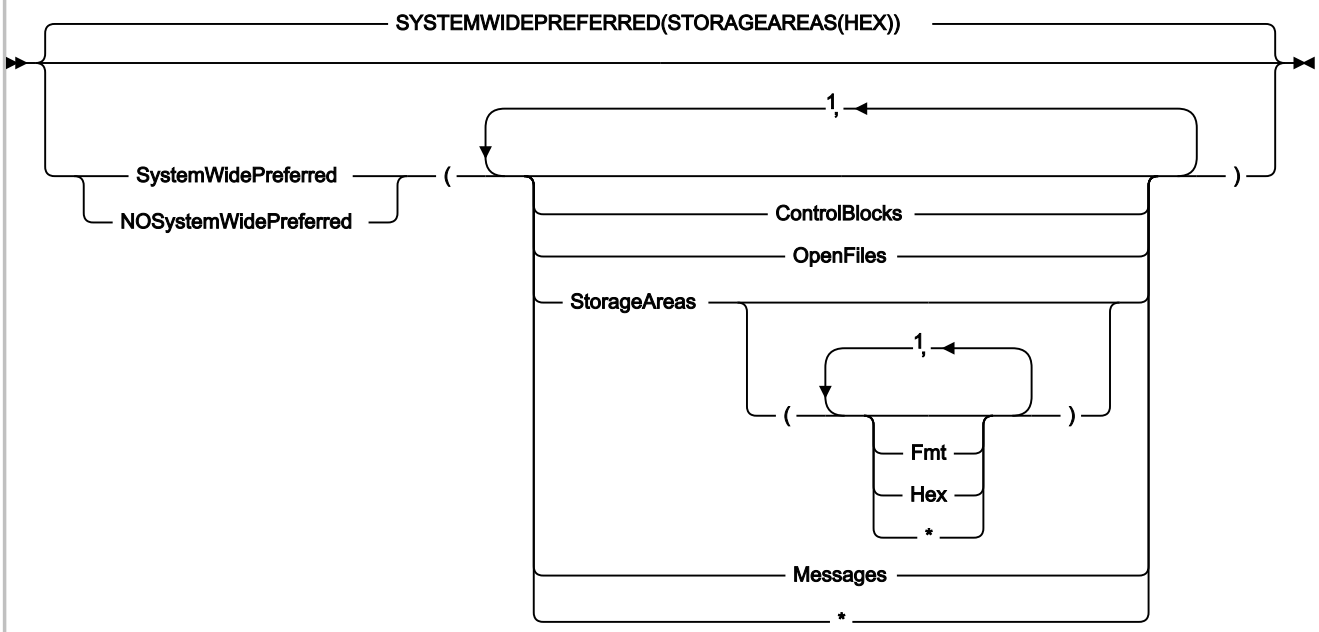
Multiple storage ranges can be specified in any order, as long as the end address is greater than the begin address. Any invalid storage ranges are ignored.

Multiple specifications of the StorageRange option replace any prior specifications, that is, the option is not cumulative.

The StorageRange option is shown in the "Options in Effect" section of the analysis report only if it has been specified in real time with at least one valid storage range.

SystemWidePreferred

Figure 347. Syntax



Notes:

[1] Either commas or blank characters are permitted as delimiters when repeating suboptions or values.

This option is applicable to the real-time or batch reanalysis reports only. It does not affect the interactive reanalysis report.

Fault Analyzer by default places information that is generally considered associated with an event in the "Event Details" section for that event. However, by using the SystemWidePreferred option, Fault Analyzer can be directed to instead placing such information in the "System-Wide Information" section of the report.

The following suboptions are available to change the placement of various event-related areas of the report:

ControlBlocks

Specifies that all event-related data that is usually placed under the heading "Associated Control Blocks" in the "Event Details" section of the report are to be placed in the "System-Wide Information" section instead. This data includes the following control blocks for the various execution environments:

Table 18. 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 are presented formatted in the report as opposed to hex-dumped. These areas are given separate subheadings within the "Storage Areas" section for easier identification.

This suboption can be abbreviated to F.

Hex

This suboption specifies storage areas that are presented in hex-dump format. These types of storage areas are combined for all events and placed in ascending address sequence under the common subheading "Hex-Dumped Storage" within the "Storage Areas" section, with labeling of addresses to indicate their origin. This suboption is the default.

This suboption can be abbreviated to H.

*****

Specifying an asterisk (*) implies specification of all suboptions.

This suboption can be abbreviated to SA.

Messages

Specifies that all event-related messages that are usually placed under the heading "Associated Messages" in the "Event Details" section of the report are to be placed in the "System-Wide Information" section instead.

This suboption can be abbreviated to M.

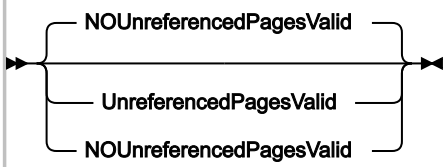
*****

Specifying an asterisk (*) implies specification of all suboptions.

Each specification of the SystemWidePreferred option is cumulative. To reset all SystemWidePreferred suboptions at once, use the NoSystemWidePreferred option.

UnreferencedPagesValid

Figure 348. Syntax



By default, Fault Analyzer treats all unreferenced storage pages as invalid storage. This is done to reduce the size of minidumps and to avoid showing large amounts of storage in the report that the application never used. However, PL/I TEST option data might sometimes reside in unreferenced storage pages, which if treated as invalid, can prevent Fault Analyzer from providing complete formatting of the program's working storage.

Use the NoUnreferencedPagesValid option to treat all unreferenced storage pages as invalid storage. This is the default.

Use the UnreferencedPagesValid option to treat all unreferenced storage pages as valid storage.

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 UnreferencedPagesValid in effect.

Chapter 35. Data areas

The following provides descriptions of data areas available to user exits. For information about user exits, refer to Customizing Fault Analyzer by using user exits on page 464.

Softcopy versions of the following data areas are available in the softcopy samples data set (IDI.SIDISAMP) for Assembler, COBOL, C, and PL/I as members IDISXPLA, IDISXPLB, IDISXPLC, and IDISXPLP respectively.

Notes relating to the following data areas:

- The 'Access' column provides information about a user exit's ability to update individual fields as follows:

  R/W

  Read/write. These fields can be updated by the user exit.

  R/O

  Read only.

- Numerical fields are identified by '(nnn)' and are, on input to the user exit, always padded on the left with zeroes to the total width of the field.

  Note: Leading zeroes are generally not included in data area fields provided to REXX user exits.

- All fields for which Fault Analyzer does not provide an initial value are initialized to blanks.

  Note: Blank-padding is generally not included in character-format data area fields provided to REXX user exits, unless the fields might contain hexadecimal characters.

- Unless otherwise indicated, all fields are translated to upper case by Fault Analyzer.
- Unless otherwise indicated, all R/W fields can be truncated using a null character (X'00') as delimiter. Truncation is never used by Fault Analyzer when initializing the parameter lists.
- For COBOL, substitute all underscores ('_') with dashes ('-').

To see the initial values of any of these fields for an abending job, add the following JCL statements to produce an exit trace:

```
//IDITRACE DD SYSOUT=*
//IDIOPTS  DD *
  Exits(exit_type(NONE))
/*
```

where *exit_type* is either CONTROL, LISTING, FORMAT, REPORT, MSGXPL, END, or NOTIFY, depending on the data area that you are interested in. For more information about tracing, see Diagnostic tracing on page 470. For more information about the Exits option, see Exits on page 600.

Non-REXX user exit buffered data format

The information that is provided in this section is only applicable to user exits that are not written in REXX.

If the data length for certain character fields exceed the maximum field size, then Fault Analyzer instead allocates a buffer large enough for all of the field data, and provide information about the buffer in three fullwords starting at relative offset zero of the field as follows:

- The first byte of the first fullword is set to X'FF' to indicate that this field contains buffer information. The remaining 3 bytes are not used, but are set to X'00' by Fault Analyzer.
- The second fullword is the address of a buffer containing the data in the same format as when the data is provided in the field itself.
- The third fullword is the allocated length of the buffer.

The data fields that this buffered format is applicable to are identified separately in the data area descriptions.

The Analysis Control user exit cannot free any buffer that is allocated by Fault Analyzer for the field. Instead, if the field or buffer size is inadequate, the exit can allocate its own buffer and place the address and length information in the three fullwords at relative offset zero of this field. Fault Analyzer is not dependent on any original buffer address to be retained for later release of allocated storage.

Freeing of buffers that were allocated by user exits is the responsibility of the user. This freeing can be achieved be utilizing the ENV.USER_1 or ENV.USER_2 fields to point to an area of storage containing information about any buffer allocations made. A later exit, such as the End Processing user exit, can then be used to perform the release of allocated storage.

For REXX user exits, Fault Analyzer automatically handles the allocation and freeing of any necessary buffers to accommodate data lengths in excess of the field size.

Data area descriptions

The following describes the various user exit data areas.

CTL - Analysis Control user exit parameter list

Table 19. CTL data area

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
0	(0)	CHAR	R/O	4	**VERSION** Parameter list version (currently 0002).
4	(4)	CHAR	R/W	1	**EXCLUDE** Exclude from analysis (Y/N). If this field is set to 'Y', no analysis of the current fault is performed, and no updates are made to the history file.

Table 19. CTL data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					Only applicable to real-time processing and dump registration.
5	(5)	CHAR	R/W	1	**DETAIL_OPT** Detail option (S/M/L). Refer to Detail on page 591 for information about this option. Not applicable to interactive reanalysis.
6	(6)	CHAR	R/W	1	**DEFERREDREPORT_OPT** DeferredReport option (Y/N). Refer to DeferredReport on page 589 for information about this option. Only applicable to real-time processing.
7	(7)	CHAR	R/W	4	**DETAIL_OPT_EXTRA_SOURCE** 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	**RETAINDUMP_OPT** RetainDump option. Refer to RetainDump on page 632 for information about this option. (The format provided in this field is identical to that of the normal RetainDump option syntax, that is, "AUTO" or "ALL"). Only applicable to real-time processing.
28	(1C)	CHAR	R/O	44	(Reserved)
72	(48)	CHAR	R/O	120	**INCLUDE_CRITERION** Include criterion. The last matching Include option criterion processed.

Table 19. CTL data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					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 639. The format of this field is transparent to users of REXX exits. Only applicable to real-time processing.
192	(C0)	CHAR	R/O	120	**EXCLUDE_CRITERION** 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 639. 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	**SOURCE_OPT** Source option (Y/N). Refer to Source on page 633 for information about this option. Only applicable to real-time processing.
318	(13E)	CHAR	R/O	6	(Reserved)
324	(144)	CHAR	R/W	1	**PRINTINACTIVECOBOL_OPT** PrintInactiveCOBOL option (Y/N). Refer to PrintInactiveCOBOL on page 628 for information about this option. Not applicable to interactive reanalysis.
325	(145)	CHAR	R/O	51	(Reserved)

Table 19. CTL data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					Field format and usage: • The following information is applicable to all IDI*_PRE, IDI*_JOB, and IDI*_CFG fields in this data area, with the exception of IDIJAVA_* fields: In each field, the data set names are provided left justified and blank padded on a 45-character boundary. Any unused space is set to all blanks. Fault Analyzer permits a buffered data format that can be used if the number of data sets exceed the maximum number that can be contained in this field. The buffer, when allocated by Fault Analyzer, is allocated large enough to hold an extra 25 data set names. For details, see Non-REXX user exit buffered data format on page 639. The format of this field is transparent to users of REXX exits. • The following information is applicable to all IDI*_PRE fields in this data area: An Analysis Control user exit can choose to unallocate any of these data sets, or allocate extra ones. These fields contain the preallocated data set names, and are provided for information only. Changes to the data set names in this field are not acted upon by Fault Analyzer. Instead, Fault Analyzer redetermines the preallocated data sets using standard operating system interfaces. • The following information is applicable to all IDI*_JOB or IDI*_CFG fields in this data area: An Analysis Control user exit can add to, delete, or alter any or all of these data set names. Upon return from the exit, data set names must be separated by one or more blanks or commas, and can be aligned on any boundary. For these data set names, Fault Analyzer uses the returned list of names.
376	178	CHAR	R/O	5400	**IDIADATA_PRE** 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	**IDIADATA_JOB**

Table 19. CTL data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					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 *"Field format and usage"* on page 643 for more information.
11176	(2BA8)	CHAR	R/W	5400	**IDIADATA_CFG** IDIADATA data sets specified in IDICNFxx config member. This field is initialized by Fault Analyzer to contain all IDIADATA data sets specified via DataSets options in the IDICNFxx config member.
16576	(40C0)	CHAR	R/O	5400	**IDILC_PRE** IDILC preallocated data sets. This field is initialized by Fault Analyzer to contain all data sets preallocated to the IDILC DDname.
21976	(55D8)	CHAR	R/W	5400	**IDILC_JOB** 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.
27376	(6AF0)	CHAR	R/W	5400	**IDILC_CFG**

Table 19. CTL data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					IDILC data sets specified in IDICNFxx config member. This field is initialized by Fault Analyzer to contain all IDILC data sets specified via DataSets options in the IDICNFxx config member. Refer to *"Field format and usage"* on page 643 for more information.
32776	(8008)	CHAR	R/O	5400	**IDILCOB_PRE** IDILCOB preallocated data sets. This field is initialized by Fault Analyzer to contain all data sets preallocated to the IDILCOB DDname.
38176	(9520)	CHAR	R/W	5400	**IDILCOB_JOB** 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.
43576	(AA38)	CHAR	R/W	5400	**IDILCOB_CFG** IDILCOB data sets specified in IDICNFxx config member. This field is initialized by Fault Analyzer to contain all IDILCOB data sets specified via DataSets options in the IDICNFxx config member.
48976	(BF50)	CHAR	R/O	5400	**IDILCOBO_PRE**

Table 19. CTL data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					IDILCOBO preallocated data sets. This field is initialized by Fault Analyzer to contain all data sets preallocated to the IDILCOBO DDname. Refer to *"Field format and usage"* on page 643 for more information.
54376	(D468)	CHAR	R/W	5400	**IDILCOBO_JOB** 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.
59776	(E980)	CHAR	R/W	5400	**IDILCOBO_CFG** IDILCOBO data sets specified in IDICNFxx config member. This field is initialized by Fault Analyzer to contain all IDILCOBO data sets specified via DataSets options in the IDICNFxx config member.
65176	(FE98)	CHAR	R/O	5400	**IDILANGX_PRE** IDILANGX preallocated data sets. This field is initialized by Fault Analyzer to contain all data sets preallocated to the IDILANGX DDname.
70576	(113B0)	CHAR	R/W	5400	**IDILANGX_JOB**

Table 19. CTL data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					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 *"Field format and usage"* on page 643 for more information.
75976	(128C8)	CHAR	R/W	5400	**IDILANGX_CFG** IDILANGX data sets specified in IDICNFxx config member. This field is initialized by Fault Analyzer to contain all IDILANGX data sets specified via DataSets options in the IDICNFxx config member.
81376	(13DE0)	CHAR	R/O	5400	**IDILPLI_PRE** IDILPLI preallocated data sets. This field is initialized by Fault Analyzer to contain all data sets preallocated to the IDILPLI DDname.
86776	(152F8)	CHAR	R/W	5400	**IDILPLI_JOB** 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.
92176	(16810)	CHAR	R/W	5400	**IDILPLI_CFG**

Table 19. CTL data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					IDILPLI data sets specified in IDICNFxx config member. This field is initialized by Fault Analyzer to contain all IDILPLI data sets specified via DataSets options in the IDICNFxx config member. Refer to *"Field format and usage"* on page 643 for more information.
97576	(17D28)	CHAR	R/O	1024	(Reserved)
98600	(18128)	CHAR	R/O	5400	**IDILPLIE_PRE** IDILPLIE preallocated data sets. This field is initialized by Fault Analyzer to contain all data sets preallocated to the IDILPLIE DDname.
104000	(19640)	CHAR	R/W	5400	**IDILPLIE_JOB** 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.
109400	(1AB58)	CHAR	R/W	5400	**IDILPLIE_CFG** IDILPLIE data sets specified in IDICNFxx config member. This field is initialized by Fault Analyzer to contain all IDILPLIE data sets specified via DataSets options in the IDICNFxx config member.
114800	(1C070)	CHAR	R/W	256	**LOCALE**

Table 19. CTL data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					Locale option locale name. Refer to Locale on page 614 for information about this option.
115056	(1C170)	CHAR	R/W	1	**FADATE** Locale option FADATE suboption (Y/N). Refer to Locale on page 614 for information about this suboption.
115057	(1C171)	CHAR	R/O	5400	**IDISYSDB_PRE** IDISYSDB preallocated data sets. This field is initialized by Fault Analyzer to contain all data sets preallocated to the IDISYSDB DDname.
120457	(1D689)	CHAR	R/W	5400	**IDISYSDB_JOB** 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.
125857	(1EBA1)	CHAR	R/W	5400	**IDISYSDB_CFG** IDISYSDB data sets specified in IDICNFxx config member. This field is initialized by Fault Analyzer to contain all IDISYSDB data sets specified via DataSets options in the IDICNFxx config member.
131257	(200B9)	CHAR	R/W	56	**IDITRACE**

Table 19. CTL data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					Use to start or stop IDITRACE tracing dynamically: **ONx** Start tracing to SYSOUT class x. If x is not specified, its default is '*'. **<dsn>** Start tracing to the specified data set name. The data set name must be fully qualified but not surrounded by single quotation marks. If the data set is a PDS or PDSE, a member name must be specified in parentheses. **OFF** Stop tracing.
131313	(200F1)	CHAR	R/O	5400	**IDIJAVA_PRE** IDIJAVA preallocated path names. This field is initialized by Fault Analyzer to contain all path names preallocated to the IDIJAVA DDname. The path names are provided as a blank-delimited list. If a path name contains single quotation marks, commas, or blanks, the name will be surrounded by single quotation marks. Any single quotation marks within the path name will be doubled up. Fault Analyzer permits the use of a buffered data format if the list of path names exceeds the maximum size that can be contained in this field. For details, see Non-REXX user exit buffered data format on page 639. The format of this field is transparent to users of REXX exits. An Analysis Control user exit can choose to deallocate any of these path names, or allocate extra ones. Changes to the path names in this field are not acted upon by Fault Analyzer. Instead, Fault Analyzer

Table 19. CTL data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					redetermines the preallocated path names using standard operating system interfaces.
136713	(21609)	CHAR	R/W	5400	**IDIJAVA_JOB** IDIJAVA path names specified in the user options file (IDIOPTS). This field is initialized by Fault Analyzer to contain all IDIJAVA path names specified through DataSets options in the user options file. The path names are provided as a blank-delimited list. If a path name contains single quotation marks, commas, or blanks, the name will be surrounded by single quotation marks. Any single quotation marks within the path name will be doubled up. If an exit is providing a path name containing single quotes, commas, or blanks, then the exit must provide the path name surrounded by single quotation marks and with all single quotation marks within the path name doubled up. Fault Analyzer permits the use of a buffered data format if the list of path names exceeds the maximum size that can be contained in this field. For details, see Non-REXX user exit buffered data format on page 639. The format of this field is transparent to users of REXX exits. An Analysis Control user exit can add to, delete, or alter any or all of these path names. Path names must be separated by one or more blanks or commas, and must protect single quotation marks as described above. For these path names, Fault Analyzer uses the returned list of names.
142113	(22B21)	CHAR	R/W	5400	**IDIJAVA_CFG** IDIJAVA path names specified in IDICNFxx config member. This field is initialized by Fault Analyzer to contain all IDIJAVA path names specified through DataSets options in the IDICNFxx config member.

Table 19. CTL data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					The path names are provided as a blank-delimited list. If a path name contains single quotation marks, commas, or blanks, the name will be surrounded by single quotation marks. Any single quotation marks within the path name will be doubled up. If an exit is providing a path name containing single quotes, commas, or blanks, then the exit must provide the path name surrounded by single quotation marks and with all single quotation marks within the path name doubled up. Fault Analyzer permits the use of a buffered data format if the list of path names exceeds the maximum size that can be contained in this field. For details, see Non-REXX user exit buffered data format on page 639. The format of this field is transparent to users of REXX exits. An Analysis Control user exit can add to, delete, or alter any or all of these path names. Path names must be separated by one or more blanks or commas, and must protect single quotation marks as described above. For these path names, Fault Analyzer uses the returned list of names.
147513	(24039)	CHAR	R/O	5400	**STEPLIB** STEPLIB data set concatenation (or JOBLIB data set concatenation, if STEPLIB is not defined). This is the concatenation as at the time of the abend, except for batch reanalysis, where it is the STEPLIB data sets for the reanalysis job. The data set names are provided as a blank-delimited list. Fault Analyzer permits the use of a buffered data format if the list of path names exceeds the maximum size that can be contained in this field. For details, see Non-REXX user exit buffered data format on page 639. The format of this field is transparent to users of REXX exits.

ENV - Common exit environment information

Table 20. ENV data area

Offsets		Type	Access	Len	Name and description	Note
Dec	**Hex**					
0	(0)	CHAR	R/O	4	**VERSION** Parameter list version (currently 0005). Summary of changes: **0003** Support for 6-digit CICS® system abend codes. **0004** Change of LOCK_FLAG from 1 to 2 characters to support fault entry expiration control.	
4	(4)	CHAR	R/O	1	**EXIT_CALL_TYPE** User exit call type that indicates the type of exit being invoked as one of the following: **C** Analysis Control **L** Compiler Listing Read **M** Message and Abend Code Explanation **F** Formatting **E** End Processing **N** Notification	

Table 20. ENV data area

(continued)

Offsets		Type	Access	Len	Name and description	Note
Dec	**Hex**					
					I IDIUTIL Import **D** IDIUTIL Delete **H** IDIUTIL ListHF **P** IDIUTIL ListHFDup **X** Dump registration Analysis Control exit **Y** Dump registration Notification exit **Z** Fault entry refresh End Processing exit This information permits a user exit to be common across exit types.	
5	(5)	CHAR	R/O	8	**FAULT_ID** 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	**ABEND_DATE**	

Table 20. ENV data area

(continued)

Offsets		Type	Access	Len	Name and description	Note
Dec	**Hex**					
					Date of abend in the format YYYY/MM/DD.	
23	(17)	CHAR	R/O	8	**ABEND_TIME** Time of abend in the format HH:MM:SS (24-hour clock value).	
31	(1F)	CHAR	R/O	1	**REALTIME** Real-time execution (Y/N).	
32	(20)	CHAR	R/O	8	**SYSTEM_NAME** System name: • When ENV.VERSION is 1, this field contains the APPLID for CICS® transaction faults and the MVS™ system name for all other fault types. • 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.	
40	(28)	CHAR	R/O	8	**JOB_NAME** Job/started task name.	
48	(30)	CHAR	R/O	8	**EXEC_PGM_NAME** EXEC program name.	See note 2
56	(38)	CHAR	R/O	8	**USER_ID** User ID.	See note 2
64	(40)	CHAR	R/O	4	(Reserved)	
68	(44)	CHAR	R/O	8	**ABEND_MODULE_NAME**	

Table 20. ENV data area

(continued)

Offsets		Type	Access	Len	Name and description	Note
Dec	**Hex**					
					ABEND module name. This value identifies the module name where the initial (if more than one) abend occurred.	
76	(4C)	CHAR	R/O	4	**CICS_TRANSACTION_ID** CICS® transaction ID.	
80	(50)	CHAR	R/O	5	**CICS_TASK_NUMBER** CICS® task number.	
85	(55)	CHAR	R/O	1	**JOB_TYPE** The type of job being analyzed as one of the following: **B** Batch job or MVS™ dump analyzed interactively by way of the **File** menu option 5. **S** Started task **T** TSO **C** CICS® transaction **I** CICS® system dump analysis, including dump registration of CICS® system dump **D** Dump registration (other than CICS® system dump)	
86	(56)	CHAR	R/O	1	**JOB_CLASS** Job execution class.	See note 2

Table 20. ENV data area

(continued)

Offsets		Type	Access	Len	Name and description	Note
Dec	**Hex**					
87	(57)	CHAR	R/O	3	**ACCOUNTING_FIELDS** Number of job accounting fields (nnn) (from JCT ACTJNFLD).	See note 2
90	(5A)	CHAR	R/O	144	**ACCOUNTING_INFO** Job accounting information (from JCT ACTACCNT). For job accounting, this field contains a one-byte length field, followed by the content of the field (repeated as many times as the number of fields shown in ENV.ACCOUNTING_FIELDS). For step accounting, this field contains these items (in order) for each step: • A three-byte maximum step running time. • A one-byte number of fields in step. • A one-byte length field. • The content of the field. The last two fields repeat as many times as the number of fields in byte 4 indicates. Any non-printable characters (such as the binary field length value) are shown as periods.	See note 2
234	(EA)	CHAR	R/W	4	**USER_1** 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.	See note 7
238	(EE)	CHAR	R/W	4	**USER_2**	See note 7

Table 20. ENV data area

(continued)

Offsets		Type	Access	Len	Name and description	Note
Dec	**Hex**					
					User field 2. Same as USER_1.	
242	(F2)	CHAR	R/O	1	(Reserved)	
243	(F3)	CHAR	R/W	1	**LOOPPROTECTION_OPT** LoopProtection option (Y/N). • When set to Y, this value is equivalent to the LoopProtection option being in effect. • When set to N, this value is equivalent to the NoLoopProtection option being in effect. It is only possible to deactivate the loop/wait protection feature of Fault Analyzer by setting this field to N. Setting this field to Y is ignored. Refer to LoopProtection on page 614 for more 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	**WRITE_ROUTINE_EP** Write routine entry-point address.	
252	(FC)	CHAR	R/O	4	(Reserved. Always contains X'00000000')	
256	(100)	CHAR	R/O	1	**INVOCATION_EXIT** The type of invocation exit used to invoke Fault Analyzer implicitly for real-time abend analysis, or explicitly using IDISNAP: **C** CICS® XPCABND exit (IDIXCX53) **D** CICS® XDUREQ exit (IDIXCX53)	

Table 20. ENV data area

(continued)

Offsets		Type	Access	Len	Name and description	Note
Dec	**Hex**					
					E CICS® LE CEECXTAN exit, IDIXCCEE **L** LE CEEEXTAN exit, IDIXCEE **M** MVS™ IEAVTABX change options/suppress dump exit, IDIXDCAP **S** Fault Analyzer program SNAP interface (IDISNAP) **P** MVS™ IEAVTSEL post dump exit, IDIXTSEL (Fault Analyzer dump registration)	
257	(101)	CHAR	R/O	8	**STEP_NAME** Job/started task step name.	See note 2
265	(109)	CHAR	R/O	8	**JOB_ID** JES job ID.	See note 2
273	(111)	CHAR	R/O	8	**IMS_PROGRAM_NAME** The name of the IMS™ program. This name is available if the environment has a DFSPRPX0 module loaded.	See note 2
281	(119)	CHAR	R/W	8	**USER_NAME** User name field.	See note 5
289	(121)	CHAR	R/W	40	**USER_TITLE**	See note 5

Table 20. ENV data area

(continued)

Offsets		Type	Access	Len	Name and description	Note
Dec	**Hex**					
					User title field.	
329	(149)	CHAR	R/O	8	**APPLID** 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	**TERMID** CICS® terminal ID.	
341	(155)	CHAR	R/O	8	**NETNAME** CICS® terminal netname.	
349	(15D)	CHAR	R/O	8	**TCB_ADDRESS** Analyzed TCB address. **Note:** The TCB address for CICS® transaction abends points to the QR TCB, which is no longer running the abending transaction.	
357	(165)	CHAR	R/O	8	**CSA_ADDRESS** CICS® CSA address. This field is only available for CICS® transaction abend or CICS® system dump analysis.	
365	(16D)	CHAR	R/O	8	**TCA_ADDRESS** CICS® TCA address. This field is only available for CICS® transaction abend analysis.	
373	(175)	CHAR	R/W	44	**IDIHIST**	

Table 20. ENV data area

(continued)

Offsets		Type	Access	Len	Name and description	Note
Dec	**Hex**					
					Fault history file name.	
417	(1A1)	CHAR	R/O	6	**ABEND_CODE** The initial (or only) abend code: • If ENV.JOB_TYPE = C, then this code is a 4-character CICS® transaction abend code. • Else if ENV.JOB_TYPE = I, then this code is a 6-character CICS® system abend code. • Otherwise, it is either a system abend code (Sxxx) or a user abend code (Unnnn).	
423	(1A7)	CHAR	R/O	6	**CPU_HSECONDS** Total CPU time used by Fault Analyzer in 1/100s of a second at the end of generating the analysis report.	

Table 20. ENV data area

(continued)

Offsets		Type	Access	Len	Name and description	Note
Dec	**Hex**					
					Note: 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 319).	
429	(1AD)	CHAR	R/O	9	**CICS_VRM** CICS® release level in VnnRnnMnn format	See note 3
438	(1B6)	CHAR	R/O	9	**DB2_VRM** DB2® release level in VnnRnnMnn format	See note 3
447	(1BF)	CHAR	R/O	9	**IMS_VRM** IMS™ release level in VnnRnnMnn format	See note 3
456	(1C8)	CHAR	R/O	9	**ZOS_VRM** z/OS® release level in VnnRnnMnn format	See note 3
465	(1D1)	CHAR	R/W	2	**LOCK_FLAG** Fault entry lock flag. The purpose of this flag is to provide a mechanism that prevents accidental deletion of the current fault entry. For more information about this flag, including specification of fault entry expiration control, see Viewing fault entry information on page 129. Changes to this field are only reflected in the history file fault entry when an Analysis Control 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.	See note 5

Table 20. ENV data area

(continued)

Offsets		Type	Access	Len	Name and description	Note
Dec	**Hex**					
					By default, the lock flag is set to a blank, which does not prevent deletion of the fault entry. Any printable character can be specified for this field: • If a non-printable character is specified, then it is changed to a '/'. • If a lowercase character is specified, then it is translated to uppercase. The final value of the lock flag is subject to any changes made by the IDIXLOCK lock flag control exit. See Controlling fault entry lock flag values on page 128.	
467	(1D3)	CHAR	R/O	5	**DUPLICATE_COUNT** Total number of duplicates, not including the current fault. This total is determined by accumulating all instances of duplicate recorded faults, in the same history file, going back as far as the NoDup(Normal(...)) time period in effect. Any recorded faults encountered whose duplicate criteria match the current fault account for one instance, and if duplicates have been recorded against the fault, the duplicate count of that fault are also added.	
472	(1D8)	CHAR	R/O	8	**POF_MODULE_NAME** Point-of-failure module name.	
480	(1E0)	CHAR	R/O	10	**POF_MODULE_LKED_DATE** Point-of-failure module link-edit date in the format YYYY/MM/DD.	
490	(1EA)	CHAR	R/O	8	**POF_MODULE_LKED_TIME**	

Table 20. ENV data area

(continued)

Offsets		Type	Access	Len	Name and description	Note
Dec	**Hex**					
					Point-of-failure module link-edit time in the format HH:MM:SS.	
498	(1F2)	CHAR	R/O	8	**POF_CSECT_NAME** Point-of-failure program or CSECT name. If the program name length exceeds 8 characters, then this field will only contain the first 8 characters of the name. In this case, use the POF_LONG_PGM_NAME field instead.	
506	(1FA)	CHAR	R/O	10	**POF_CSECT_OFFSET** Point-of-failure entry-point, program/CSECT or load module offset (decimal).	
516	(204)	CHAR	R/O	44	**POF_LOADED_FROM** Data set name from where the point-of-failure program was loaded.	
560	(230)	CHAR	R/O	44	**EXEC_LOADED_FROM** Data set name from where the EXEC program was loaded.	
604	(25C)	CHAR	R/O	10	**DUP_DATE** Date of most recent duplicate fault in the format YYYY/MM/DD.	
614	(266)	CHAR	R/O	8	**DUP_TIME** Time of most recent duplicate fault in the format HH:MM:SS (24-hour clock value).	
622	(26E)	CHAR	R/O	8	**GROUP_ID** Security server default group ID.	See note 2
630	(276)	CHAR	R/O	6	**INVOCATION_ABEND_CODE**	

Table 20. ENV data area

(continued)

Offsets		Type	Access	Len	Name and description	Note
Dec	**Hex**					
					The final (or only) abend code, which is the abend code for which Fault Analyzer was invoked: • If ENV.JOB_TYPE = C, then this code is a 4-character CICS® transaction abend code. • Else if ENV.JOB_TYPE = I, then this code is a 6-character CICS® system abend code. • Otherwise, it is either a system abend code (Sxxx) or a user abend code (Unnnn).	
636	(27C)	CHAR	R/O	10	**MINIDUMP_PAGES** Number of minidump pages (nnnnnnnnnn). **Note:** For real-time processing, more minidump pages resulting from storage that is referenced during execution of a Formatting user exit are not included in this value.	
646	(286)	CHAR	R/O	8	**IDIRLOAD_DD** The IDIRLOAD DDname. During initial RFR fault entry reanalysis initiated by an I or B line command on the Fault Entry List display, if the STEPLIB or JOBLIB data set concatenation was saved, then Fault Analyzer allocates the STEPLIB or JOBLIB concatenation to a system-generated DDname (SYS*nnnnn*) and initializes this field with that DDname. The purpose of this is to automatically facilitate the Binder CSECT mapping that could not be performed during the real-time analysis. In all other cases, this field is initialized by default to IDIRLOAD. An Analysis Control user exit may choose one of the following actions:	See note 1

Table 20. ENV data area

(continued)

Offsets		Type	Access	Len	Name and description	Note
Dec	**Hex**					
					• Clear this field to disable automatic processing during the initial RFR fault entry reanalysis. • Allocate one or more load libraries to a user-determined DDname and change this field to that DDname. • Allocate one or more load libraries to the IDIRLOAD DDname. Upon return from the Analysis Control user exit, Fault Analyzer will use any data sets that are allocated to the DDname specified in this field as the IDIRLOAD allocation. For more information, see Using the IDIRLOAD DDname for CSECT mapping on page 388. Changes made to this field by user exit types other than Analysis Control are ignored. The specified DDname is not case sensitive.	
654	(28E)	CHAR	R/O	8	**ABEND_REASON_CODE** Hexadecimal reason code that is associated with ENV.ABEND_CODE. Available to all user exits that run following "Analysis" processing as per Figure 231: Fault Analyzer analysis exit points (IDIDA) on page 465.	
662	(296)	CHAR	R/O	8	**LOCK_USERID** The user ID who last changed the ENV.LOCK_FLAG field.	
670	(29E)	CHAR	R/O	10	**ORIGINAL_DATE** Date of original fault in the format YYYY/MM/DD.	See note 4
680	(2A8)	CHAR	R/O	8	**ORIGINAL_TIME**	See note 4

Table 20. ENV data area

(continued)

Offsets		Type	Access	Len	Name and description	Note
Dec	**Hex**					
					Time of original fault in the format HH:MM:SS (24-hour clock value).	
688	(280)	CHAR	R/O	1	**ASSOCIATED_DUMP_TYPE** The type of dump data set associated with this fault entry, as one of the following: **P** SLIP dump **S** SDUMP (SVC dump) **T** TDUMP (transaction dump) **X** XDUMP (extended dump)	See note 6
689	(281)	CHAR	R/O	44	**ASSOCIATED_DUMP_DSN** The name of the dump data set associated with this fault entry.	See note 6
733	(2DD)	CHAR	R/O	32	**POF_LONG_PGM_NAME** Point-of-failure long (up to 32 characters) program name. This field is only initialized if the program name length exceeds 8 characters. Otherwise, use the POF_CSECT_NAME field instead.	
765	(2FD)	CHAR	R/O	775	(Reserved)	

Notes:

1

Not applicable to real-time processing.

2

Not available to dump registration user exits.

3

Not available at the time of calling the Analysis Control user exit.

4

Only available if the fault is a duplicate.

5

Updates are only saved in the fault entry during real-time processing when using exits that are invoked before report generation.

6

When a Notification user exit is called during the creation of an RFR fault entry with an associated TDUMP, these fields are initialized with the expectation that the TDUMP will be successfully written after the return of the exit.

7

Not preserved in Notification user exit called during RFR fault entry creation.

EPC - End Processing user exit parameter list

Table 21. EPC data area

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
0	(0)	CHAR	R/O	4	**VERSION** Parameter list version (currently 0002).
4	(4)	CHAR	R/O	4	(Reserved)
8	(8)	CHAR	R/W	1	**IS_DUPLICATE** This is a duplicate fault (Y/N).
9	(9)	CHAR	R/W	1	**SUPPRESS_MINIDUMP** Suppress minidump (Y/N).

Table 21. EPC data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					This flag is set by Fault Analyzer based on the MaxMinidumpPages option setting and the expected minidump pages for this fault. It can be overridden by an End Processing user exit.
10	(A)	CHAR	R/O	1	(Reserved)
11	(B)	CHAR	R/W	1	**SUPPRESS_FAULT_ENTRY** 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	**MINUTES_SINCE_LAST_DUP** Number of minutes elapsed since recording of last duplicate fault (nnnnn). If blank, no duplicate fault was found. **Note:** The maximum value of 99999 is used whenever the number of minutes exceeds the limit of this field.
31	(1F)	CHAR	R/O	1	**ANALYSIS_SUCCESSFUL** Successful analysis (Y/N). The criteria for successful analysis are the following: • No error messages issued. • Identification of the source line of code for the point of failure.
32	(20)	CHAR	R/O	88	(Reserved)
120	(78)	CHAR	R/W	1	**SUPPRESS_DUMP** Suppress dump (Y/N).

Table 21. EPC data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					This flag affects the suppression of the MVS™ system dump or CICS® transaction dump. For details, see Dump suppression on page 32.
121	(79)	CHAR	R/O	63	(Reserved)

LST - Compiler Listing Read user exit parameter list

Table 22. LST data area

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
0	(0)	CHAR	R/O	4	**VERSION** Parameter list version (currently 0001).
4	(4)	CHAR	R/O	8	**MODULE_NAME** Module name. This name is the name of the load module containing the CSECT identified in LST.CSECT_NAME.
12	(C)	CHAR	R/O	8	**CSECT_NAME** CSECT name. This name is the name of the CSECT containing the program identified in LST.PROGRAM_NAME.
20	(14)	CHAR	R/O	256	**EP_NAME** Entry point name (truncated to 256 chars).
276	(114)	CHAR	R/O	10	**COMPILE_DATE** Compile date in the format YYYY/MM/DD.

Table 22. LST data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
286	(11E)	CHAR	R/O	8	**COMPILE_TIME** Compile time in the format HH:MM:SS.
294	(126)	CHAR	R/O	1	**LISTING_TYPE** Compiler listing or assembler SYSADATA file (L), or side file (S).
295	(127)	CHAR	R/O	12	**LANGUAGE_TYPE** Language type: • Assembler • C/C++ • COBOL • OS/VS COBOL • PL/I • Entprs PL/I
307	(133)	CHAR	R/O	4	**RECFM** Record format.
311	(137)	CHAR	R/O	5	**LRECL** Logical record length (nnnnn).
316	(13C)	CHAR	R/W	5	**DATA_LENGTH** Data length of variable length record (nnnnn). This field specifies the length of the record placed in DATA_BUFFER.
321	(141)	CHAR	R/W	1	**DISREGARD_EXIT_LISTING** 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.

Table 22. LST data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
322	(142)	CHAR	R/O	8	**PROGRAM_NAME** Program name. If the program name length exceeds 8 characters, then this field will only contain the first 8 characters of the name. In this case, use the LONG_PROGRAM_NAME field instead.
330	(14A)	CHAR	R/O	10	**PROGRAM_LENGTH** Program length in bytes (decimal). If the program name length exceeds 8 characters, then this field will only contain the first 8 characters of the name. In this case, use the LONG_PROGRAM_NAME field instead.
340	(154)	CHAR	R/W	1	**DATA_BUFFER_DSN** Data buffer contains data set name (Y/N)
341	(155)	CHAR	R/O	44	**LOAD_MODULE_DSN** Load module data set name. This name is the name of the data set from which the load module identified in LST.MODULE_NAME was loaded.
385	(181)	CHAR	R/O	5	(Reserved)
390	(186)	CHAR	R/W	8188	**DATA_BUFFER** 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

Table 22. LST data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					as appropriate for LISTING_TYPE. Refer to Compiler Listing Read user exit on page 480 for extra data set name requirements.
8578	(2182)	CHAR	R/O	32	**LONG_PROGRAM_NAME** Long (up to 32 characters) program name. This field is only initialized if the program name length exceeds 8 characters. Otherwise, use the PROGRAM_NAME field instead.

NFY - Notification user exit parameter list

Table 23. NFY data area

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
0	(0)	CHAR	R/O	4	**VERSION** Parameter list version (currently 0002).
4	(4)	CHAR	R/O	45	(Reserved)
49	(31)	CHAR	R/O	1024	**SYNOPSIS** 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 639. The format of this field is transparent to users of REXX exits.
1073	(431)	CHAR	R/O	1	**NFYTYPE**

Table 23. NFY data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					C Fault created **R** Recovery fault recording **N** NoDup(Normal) duplicate **F** NoDup(CICSfast) or NoDup(ImageFast) duplicate
1074	(432)	CHAR	R/O	8	**DUPCOUNT** The number of new duplicates during this 30-second recording period when NFYTYPE is set to 'F'. Always 1 when NFYTYPE is set to 'N'. Not applicable for other values of NFYTYPE.
1082	(43A)	CHAR	R/O	55	(Reserved)

UFM - Formatting user exit parameter list

Table 24. UFM data area

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
0	(0)	CHAR	R/O	4	**VERSION** Parameter list version (currently 0001).
4	(4)	CHAR	R/W	100	**USEROPTIONTITLE** Report section heading for output from all Formatting user exits run using the Exits option. Initialized to the heading set by any previously

Table 24. UFM data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					called Formatting user exit. The initial default for the batch report is "U S E R", and for the interactive reanalysis report it is "User".
104	(68)	CHAR	R/O	91	(Reserved)
195	(C3)	CHAR	R/O	5	**NUM_EVENTS** Total number of events (decimal).
All fields from here on are populated with data for a single event only. To populate with data for another event, use the IDIEventInfo command.					
200	(C8)	CHAR	R/W	5	**EVENT_NO** Current® event number (nnnnn).
205	(CD)	CHAR	R/O	5	**NEXT_EVENT_NO** Next available event number (decimal).
210	(D2)	CHAR	R/O	5	**PREVIOUS_EVENT_NO** Previous available event number (decimal).
215	(D7)	CHAR	R/O	1	**POF** Point of failure (Y/N).
216	(D8)	CHAR	R/O	30	**EVENT_TYPE** 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 639. The format of this field is transparent to users of REXX exits. If Language(JPN) is in effect, then the event type description provided in this field is subject to translation into Japanese.

Table 24. UFM data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
246	(F6)	CHAR	R/O	12	**MODULE_NAME** 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 639. The format of this field is transparent to users of REXX exits.
258	(102)	CHAR	R/O	8	**MODULE_ADDRESS** Module address.
266	(10A)	CHAR	R/O	8	**MODULE_LENGTH** Module length (hexadecimal).
274	(112)	CHAR	R/O	12	**PROGRAM_NAME** 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 639. The format of this field is transparent to users of REXX exits.
286	(11E)	CHAR	R/O	8	**PROGRAM_ADDRESS** Program address.
294	(126)	CHAR	R/O	8	**PROGRAM_LENGTH** Program length (hexadecimal).
302	(12E)	CHAR	R/O	12	**EP_NAME** 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

Table 24. UFM data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					page 639. The format of this field is transparent to users of REXX exits.
314	(13A)	CHAR	R/O	8	**EP_ADDRESS** Entry point address.
322	(142)	CHAR	R/O	64	**EVENT_LOCATION** 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 639. The format of this field is transparent to users of REXX exits.
386	(182)	CHAR	R/O	44	**LOADED_FROM** 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 639. The format of this field is transparent to users of REXX exits.
430	(1AE)	CHAR	R/O	8	**INSTRUCTION_ADDRESS** The event instruction address.
438	(1B6)	CHAR	R/O	2	**AMODE** The event addressing mode (24/31/64).
440	(1B8)	CHAR	R/O	16	**PSW** The event PSW.

Table 24. UFM data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
456	(1C8)	CHAR	R/O	8	**GPREG0** General purpose register 0.
464	(1D0)	CHAR	R/O	8	**GPREG1** General purpose register 1.
472	(1D8)	CHAR	R/O	8	**GPREG2** General purpose register 2.
480	(1E0)	CHAR	R/O	8	**GPREG3** General purpose register 3.
488	(1E8)	CHAR	R/O	8	**GPREG4** General purpose register 4.
496	(1F0)	CHAR	R/O	8	**GPREG5** General purpose register 5.
504	(1F8)	CHAR	R/O	8	**GPREG6** General purpose register 6.
512	(200)	CHAR	R/O	8	**GPREG7** General purpose register 7.
520	(208)	CHAR	R/O	8	**GPREG8** General purpose register 8.
528	(210)	CHAR	R/O	8	**GPREG9** General purpose register 9.
536	(218)	CHAR	R/O	8	**GPREG10**

Table 24. UFM data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					General purpose register 10.
544	(220)	CHAR	R/O	8	**GPREG11** General purpose register 11.
552	(228)	CHAR	R/O	8	**GPREG12** General purpose register 12.
560	(230)	CHAR	R/O	8	**GPREG13** General purpose register 13.
568	(238)	CHAR	R/O	8	**GPREG14** General purpose register 14.
576	(240)	CHAR	R/O	8	**GPREG15** General purpose register 15.
584	(248)	CHAR	R/O	8	**AREG_DATA_ADDRESS** Address of storage area containing access registers in hexadecimal format (AR0 through AR15).
592	(250)	CHAR	R/O	122	(Reserved)
714	(2CA)	CHAR	R/O	16	**BEAR** Breaking event address register.
730	(2DA)	CHAR	R/W	5	**DATA_LENGTH** Data length (nnnnn). This field specifies the length of the record placed in UFM.DATA_BUFFER.
735	(2DF)	CHAR	R/W	1024	**DATA_BUFFER** Data buffer.

Table 24. UFM data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					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. **Note:** 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 490. 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	**FPREG0** Floating-point register 0.
1776	(6F0)	CHAR	R/O	16	**FPREG1** Floating-point register 1.
1792	(700)	CHAR	R/O	16	**FPREG2** Floating-point register 2.
1808	(710)	CHAR	R/O	16	**FPREG3** Floating-point register 3.
1824	(720)	CHAR	R/O	16	**FPREG4** Floating-point register 4.
1840	(730)	CHAR	R/O	16	**FPREG5** Floating-point register 5.
1856	(740)	CHAR	R/O	16	**FPREG6**

Table 24. UFM data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					Floating-point register 6.
1872	(750)	CHAR	R/O	16	**FPREG7** Floating-point register 7.
1888	(760)	CHAR	R/O	16	**FPREG8** Floating-point register 8.
1904	(770)	CHAR	R/O	16	**FPREG9** Floating-point register 9.
1920	(780)	CHAR	R/O	16	**FPREG10** Floating-point register 10.
1936	(790)	CHAR	R/O	16	**FPREG11** Floating-point register 11.
1952	(7A0)	CHAR	R/O	16	**FPREG12** Floating-point register 12.
1968	(7B0)	CHAR	R/O	16	**FPREG13** Floating-point register 13.
1984	(7C0)	CHAR	R/O	16	**FPREG14** Floating-point register 14.
2000	(7D0)	CHAR	R/O	16	**FPREG15** Floating-point register 15.
2016	(7E0)	CHAR	R/O	8	**FPCR** Floating-point control register.

Table 24. UFM data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
2024	(7E8)	CHAR	R/O	1	**GPREGS_64BIT** 64-bit general purpose registers available (Y/N).
2025	(7E9)	CHAR	R/O	16	**GPREG0_64BIT** General purpose register 0 (64-bit).
2041	(7F9)	CHAR	R/O	16	**GPREG1_64BIT** General purpose register 1 (64-bit).
2057	(809)	CHAR	R/O	16	**GPREG2_64BIT** General purpose register 2 (64-bit).
2073	(819)	CHAR	R/O	16	**GPREG3_64BIT** General purpose register 3 (64-bit).
2089	(829)	CHAR	R/O	16	**GPREG4_64BIT** General purpose register 4 (64-bit).
2105	(839)	CHAR	R/O	16	**GPREG5_64BIT** General purpose register 5 (64-bit).
2121	(849)	CHAR	R/O	16	**GPREG6_64BIT** General purpose register 6 (64-bit).
2137	(859)	CHAR	R/O	16	**GPREG7_64BIT** General purpose register 7 (64-bit).
2153	(869)	CHAR	R/O	16	**GPREG8_64BIT** General purpose register 8 (64-bit).
2169	(879)	CHAR	R/O	16	**GPREG9_64BIT**

Table 24. UFM data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					General purpose register 9 (64-bit).
2185	(889)	CHAR	R/O	16	**GPREG10_64BIT** General purpose register 10 (64-bit).
2201	(899)	CHAR	R/O	16	**GPREG11_64BIT** General purpose register 11 (64-bit).
2217	(8A9)	CHAR	R/O	16	**GPREG12_64BIT** General purpose register 12 (64-bit).
2233	(8B9)	CHAR	R/O	16	**GPREG13_64BIT** General purpose register 13 (64-bit).
2249	(8C9)	CHAR	R/O	16	**GPREG14_64BIT** General purpose register 14 (64-bit).
2265	(8D9)	CHAR	R/O	16	**GPREG15_64BIT** General purpose register 15 (64-bit).
2281	(8E9)	CHAR	R/O	32	**VFREG0** Vector facility register 0.
2313	(909)	CHAR	R/O	32	**VFREG1** Vector facility register 1.
2345	(929)	CHAR	R/O	32	**VFREG2** Vector facility register 2.
2377	(949)	CHAR	R/O	32	**VFREG3** Vector facility register 3.

Table 24. UFM data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
2409	(969)	CHAR	R/O	32	**VFREG4** Vector facility register 4.
2441	(989)	CHAR	R/O	32	**VFREG5** Vector facility register 5.
2473	(9A9)	CHAR	R/O	32	**VFREG6** Vector facility register 6.
2505	(9C9)	CHAR	R/O	32	**VFREG7** Vector facility register 7.
2537	(9E9)	CHAR	R/O	32	**VFREG8** Vector facility register 8.
2569	(A09)	CHAR	R/O	32	**VFREG9** Vector facility register 9.
2601	(A29)	CHAR	R/O	32	**VFREG10** Vector facility register 10.
2633	(A49)	CHAR	R/O	32	**VFREG11** Vector facility register 11.
2665	(A69)	CHAR	R/O	32	**VFREG12** Vector facility register 12.
2697	(A89)	CHAR	R/O	32	**VFREG13** Vector facility register 13.
2729	(AA9)	CHAR	R/O	32	**VFREG14**

Table 24. UFM data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					Vector facility register 14.
2761	(AC9)	CHAR	R/O	32	**VFREG15** Vector facility register 15.
2793	(AE9)	CHAR	R/O	32	**VFREG16** Vector facility register 16.
2825	(B09)	CHAR	R/O	32	**VFREG17** Vector facility register 17.
2857	(B29)	CHAR	R/O	32	**VFREG18** Vector facility register 18.
2889	(B49)	CHAR	R/O	32	**VFREG19** Vector facility register 19.
2921	(B69)	CHAR	R/O	32	**VFREG20** Vector facility register 20.
2953	(B89)	CHAR	R/O	32	**VFREG21** Vector facility register 21.
2985	(BA9)	CHAR	R/O	32	**VFREG22** Vector facility register 22.
3017	(BC9)	CHAR	R/O	32	**VFREG23** Vector facility register 23.
3049	(BE9)	CHAR	R/O	32	**VFREG24** Vector facility register 24.

Table 24. UFM data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
3081	(C09)	CHAR	R/O	32	**VFREG25** Vector facility register 25.
3113	(C29)	CHAR	R/O	32	**VFREG26** Vector facility register 26.
3145	(C49)	CHAR	R/O	32	**VFREG27** Vector facility register 27.
3177	(C69)	CHAR	R/O	32	**VFREG28** Vector facility register 28.
3209	(C89)	CHAR	R/O	32	**VFREG29** Vector facility register 29.
3241	(CA9)	CHAR	R/O	32	**VFREG30** Vector facility register 30.
3273	(CC9)	CHAR	R/O	32	**VFREG31** Vector facility register 31.
The following GPREGn_VALID fields apply to both GPREGn and GPREGn_64BIT general purpose register values. That is, if the value for register n is 'N', then neither GPREGn nor GPREGn_64BIT are valid.					
3305	(CE9)	CHAR	R/O	1	**GPREG0_VALID** General purpose register 0 valid (Y/N).
3306	(CEA)	CHAR	R/O	1	**GPREG1_VALID** General purpose register 1 valid (Y/N).
3307	(CEB)	CHAR	R/O	1	**GPREG2_VALID**

Table 24. UFM data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					General purpose register 2 valid (Y/N).
3308	(CEC)	CHAR	R/O	1	**GPREG3_VALID** General purpose register 3 valid (Y/N).
3309	(CED)	CHAR	R/O	1	**GPREG4_VALID** General purpose register 4 valid (Y/N).
3310	(CEE)	CHAR	R/O	1	**GPREG5_VALID** General purpose register 5 valid (Y/N).
3311	(CEF)	CHAR	R/O	1	**GPREG6_VALID** General purpose register 6 valid (Y/N).
3312	(CF0)	CHAR	R/O	1	**GPREG7_VALID** General purpose register 7 valid (Y/N).
3313	(CF1)	CHAR	R/O	1	**GPREG8_VALID** General purpose register 8 valid (Y/N).
3314	(CF2)	CHAR	R/O	1	**GPREG9_VALID** General purpose register 9 valid (Y/N).
3315	(CF3)	CHAR	R/O	1	**GPREG10_VALID** General purpose register 10 valid (Y/N).
3316	(CF4)	CHAR	R/O	1	**GPREG11_VALID** General purpose register 11 valid (Y/N).
3317	(CF5)	CHAR	R/O	1	**GPREG12_VALID** General purpose register 12 valid (Y/N).

Table 24. UFM data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
3318	(CF6)	CHAR	R/O	1	**GPREG13_VALID** General purpose register 13 valid (Y/N).
3319	(CF7)	CHAR	R/O	1	**GPREG14_VALID** General purpose register 14 valid (Y/N).
3320	(CF8)	CHAR	R/O	1	**GPREG15_VALID** General purpose register 15 valid (Y/N).

UTL - IDIUTIL Batch Utility user exit parameter list

Table 25. UTL data area

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
0	(0)	CHAR	R/O	4	**VERSION** Parameter list version (currently 0002).
4	(4)	CHAR	R/O	44	(Reserved)
48	(30)	CHAR	R/W	1	**PERFORM_ACTION** Perform the utility action on this fault entry (Y/N).
49	(31)	CHAR	R/W	100	**IMPORT_DUMP_DSN** This value is applicable only when importing a single fault entry from a staging data set created using the IDIROBOT exec. If the fault entry being imported was exported along with an associated dump data set, the original associated dump data set name is provided in ENV.ASSOCIATED_DUMP_DSN. An IDIUTIL user exit can then respond in the following ways:

Table 25. UTL data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					• It can ignore the associated dump data set by setting the UTL.IMPORT_DUMP_DSN field to "NULLFILE". • It can initialize UTL.IMPORT_DUMP_DSN to the original dump data set name provided in ENV.ASSOCIATED_DUMP_DSN. ❶ • It can initialize UTL.IMPORT_DUMP_DSN to a different dump data set name. ❶ • It can leave the UTL.IMPORT_DUMP_DSN field blank, in which case the appropriate data set name specified in the RFRDSN, SDUMPDSN, or XDUMPDSN options of the IDIOPTLM configuration-options module will be used. The name specified in UTL.IMPORT_DUMP_DSN will be used for the dump data set allocated for, and associated with, the imported fault entry.
83	(131)	CHAR	R/O	1	**DUP_TYPE** This field is applicable only when using the ListHFDup exit. Possible values are: **Blank** Base fault entry **N** Normal duplicate **F** CICSfast duplicate **I** Imagefase ducplicate

Note:

❶

The data set name must be appropriate for the system on which the fault entry is being imported and the user ID under which IDIUTIL is running.

XPL - Message and Abend Code Explanation user exit parameter list

Table 26. XPL data area

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
0	(0)	CHAR	R/O	4	**VERSION** Parameter list version (currently 0001).
4	(4)	CHAR	R/O	125	**MESSAGE_TEXT1** Single-line WTO or first multi-line WTO message text.
129	(81)	CHAR	R/O	70	**MESSAGE_TEXT2** Multi-line WTO message text line 2.
199	(C7)	CHAR	R/O	70	**MESSAGE_TEXT3** Multi-line WTO message text line 3.
269	(10D)	CHAR	R/O	70	**MESSAGE_TEXT4** Multi-line WTO message text line 4.
339	(153)	CHAR	R/O	70	**MESSAGE_TEXT5** Multi-line WTO message text line 5.
409	(199)	CHAR	R/O	70	**MESSAGE_TEXT6** Multi-line WTO message text line 6.
479	(1DF)	CHAR	R/O	70	**MESSAGE_TEXT7** Multi-line WTO message text line 7.
549	(225)	CHAR	R/O	70	**MESSAGE_TEXT8**

Table 26. XPL data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					Multi-line WTO message text line 8.
619	(26B)	CHAR	R/O	70	**MESSAGE_TEXT9** Multi-line WTO message text line 9.
689	(2B1)	CHAR	R/O	70	**MESSAGE_TEXT10** Multi-line WTO message text line 10.
759	(2F7)	CHAR	R/O	4	(Reserved)
763	(2FB)	CHAR	R/O	8	**ABEND_REASON_CODE** ABEND reason code.
771	(303)	CHAR	R/O	8	**ABEND_MODULE_NAME** ABEND module name.
779	(30B)	CHAR	R/O	1	**ABEND_TYPE** ABEND type: **C** CICS® transaction abend **D** CICS® dump code **S** System **U** User
780	(30C)	CHAR	R/W	5	**DATA_LENGTH** Data length (nnnnn). This field specifies the length of the record placed in DATA_BUFFER.
785	(311)	CHAR	R/W	256	**DATA_BUFFER**

Table 26. XPL data area

(continued)

Offsets		Type	Access	Len	Name and description
Dec	**Hex**				
					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. **Note:** 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 485.
1041	(411)	CHAR	R/O	1	**EXPLANATION_AVAILABLE** 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	**ABEND_CODE** ABEND code.
1048	(418)	CHAR	R/O	58	(Reserved)

Chapter 36. Return codes

This topic describes the return codes issued by Fault Analyzer utilities.

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 that are suffixed by 'I').

2 or 4

- One or more warning messages has been issued (message numbers that are suffixed by 'W').
- Informational messages might also have been issued.

8

- One or more error messages has been issued (message numbers that are suffixed by 'E').
- Informational and warning messages might also have been issued.

12

- One or more severe messages has been issued (message numbers that are 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 warning messages written to the SYSPRINT ddname.

8

One or more error messages written to the SYSPRINT ddname.

IDILANGX

For return codes issued by IDILANGX, see *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*.

Chapter 37. Messages

This chapter describes messages issued by Fault Analyzer and related utilities.

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 under *"IPVLANGX messages"* in the *IBM Application Delivery Foundation for z/OS Common Components: Customization Guide and User Guide*.

Fault Analyzer messages

These messages are issued by Fault Analyzer.

IDI0001I

Fault Analyzer *version-info* invoked by *exit-name* using *config-member*

Explanation:

Fault Analyzer was invoked for real-time analysis. In the message text:

- *version-info* provides information about the version of Fault Analyzer used and the current maintenance level.
- *exit-name* identifies the exit which invoked Fault Analyzer.
- *config-member* is the data set and member name containing the parmlib configuration options, or indicates the use of default options if no configuration member could be found.

System action: Normal processing continues.

User response: None

IDI0002I

analysis-summary

Explanation:

Fault analysis has completed. In the message text, *analysis-summary* provides a brief summary of the problem. The analysis summary is typically presented in the format:

```
[point-of-failure:] symptom
```

where:

point-of-failure

The point in the user application at which the error occurred, or where control left the user application prior to the error. If available, module name, program or CSECT name, offset, and source line number or compiler listing statement number is provided.

symptom

The type of error that occurred (for example, the initial abend code).

System action: Normal processing continues.

User response: None

IDI0003I

Fault ID *faultid* assigned in history file *history-file-name*

Explanation: Fault Analyzer has completed real-time analysis and has assigned the fault identifier *faultid* in the history file *history-file-name* to this abend.

System action: Processing has ended.

User response: None

IDI0004S

The input dump data set *data-set-name* could not be opened because: *reason*

Explanation: The dump data set identified by *data-set-name* could not be opened for reanalysis for the reason that is identified in *reason*.

System action: Processing terminates.

User response: Correct the data set name and resubmit the job.

IDI0005S

module-name:*line-number* Storage allocation for *dec-count* (X'*hex-count*') bytes failed - processing terminated

Explanation: An out-of-storage condition has occurred.

System action: Processing terminates.

User response: Specify a larger region size and resubmit the job—see Storage recommendations on page 296 for more information. If using TSO, specify a larger region size on the logon panel. Fault Analyzer deliberately attempts not to use up all available 31-bit storage before issuing this message, as using up all storage might cause MVS™ to use 24-bit storage instead with possibly disastrous consequences for non-terminating address spaces, such as CICS® regions.

IDI0006E

Open of *context* data set *data-set-name* failed because: *reason*

Explanation: The data set identified by *data-set-name* could not be opened for the reason that is identified in *reason*. The context in which the data set was attempted opened is provided in *context*. This context might be an associated DDname, or some other description of the type of data set involved.

When *reason* is in the form of

```
DYNALLOC error=error-code info=info-code
```

then refer to *z/OS®: MVS™ Programming: Authorized Assembler Services Guide* for information about the error and info codes.

The reasons for some of the more common DYNALLOC info codes are as follows:

Info code

Reason

210

Data set is allocated to another job.

1708

Data set not found.

System action: Processing attempts to continue without the use of this data set.

User response: Determine the reason for the open failure and resubmit the job.

IDI0007S

GETMAIN failed *resource-name* rc=*return-code*

Explanation: An out-of-storage condition has occurred.

System action: Processing terminates.

User response: Specify a larger region size and resubmit the job.

IDI0008E

CSVINFO error, rc=*return-code*

Explanation: An error occurred using the CSVINFO service.

System action: Processing terminates.

User response: This might be an internal error. Contact IBM Support.

IDI0009E

IEWBUFF error

Explanation: An error occurred while using the Binder.

System action: Processing terminates.

User response: This might be an internal error. Contact IBM Support.

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 IBM Support.

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 IBM Support.

IDI0012S

Abend *abend-code* occurred in abend exit processing,

Explanation: An abend occurred in Fault Analyzer. This message is followed by message IDI0013S on page 698.

System action: Processing terminates.

User response: This is an internal error. Contact IBM Support. Ensure that the SVC dump written by Fault Analyzer is kept for later analysis.

IDI0013S

R15=*r15-value* PSW=*program-status-word* DCAPSUB=*base-reg-value*

Explanation: This message follows message IDI0012S on page 698.

System action: See message IDI0012S on page 698.

User response: See message IDI0012S on page 698.

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 might be an internal error. Contact IBM Support.

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 IBM Support.

IDI0016E

Abend *abend-code* during fault history file OPEN/READ processing

Explanation: During real-time processing, an abend occurred while attempting to OPEN or READ the history file.

System action: Processing continues, however, no fault entry is created and no duplicate counts is updated.

User response: Determine the reason for the abend.

IDI0018W

Parmlib member read error: *reason*

Explanation: A problem occurred while attempting to open or read the IDICNFxx parmlib configuration member. The reason for the error is provided in *reason*.

System action: Processing continues without the use of the IDICNFxx parmlib member.

User response: Ensure that an IDICNFxx member exists in the logical parmlib concatenation, or in the alternative parmlib data set specified via the IDIOPTLM configuration-options module CNFDSN setting. For details, see Customize Fault Analyzer by using an IDIOPTLM configuration-options module on page 337.

IDI0019W

options-source syntax error on line *line-number* column *column-number*: *reason*

Explanation:

A syntax error was encountered while processing options that are specified in *options-source*, where *options-source* is one of the following:

Analysis Control user exit

The option was provided by an Analysis Control user exit.

Environment variable _IDI_OPTS

The option was provided through the _IDI_OPTS environment variable.

Options line for interactive reanalysis

The option was specified in the "Options line for interactive reanalysis" field on the Interactive Reanalysis Options display, which is shown when selecting "Interactive Reanalysis Options..." from the action bar Options pull-down menu.

PARM field

The option was specified in one of these places:

- The PARM field of the EXEC card for PGM=IDIDA in a batch reanalysis job.
- The **Options line for batch reanalysis** field on the Batch Reanalysis Options display. This field is shown when selecting **Batch Reanalysis Options...** from the action bar Options pull-down menu.

Parmlib config member

The option was specified in the IDICNFxx parmlib member, or in a data set and member identified by an IDICNFUM user-options module.

User options file

The option was specified through the IDIOPTS DDname.

System action: Processing continues.

User response: Correct the error and resubmit your job.

IDI0020W

options-source contained invalid option *option*

Explanation: An invalid option was encountered while processing options that are specified in *options-source*. For the possible values of *options-source*, see message IDI0019W on page 699.

System action: The option *option* is ignored and processing continues.

User response: Correct the error and resubmit your job.

IDI0021W

Allocation failed: DD=*ddname* DSN=*data-set-name* RC=*return-code* Error=*error-code* Info=*info-code*

Explanation: Dynamic data set allocation failed.

System action: Processing continues without the data set identified in *data-set-name*.

User response: Refer to *z/OS®: MVS™ Programming: Authorized Assembler Services Guide* for information about the return code, error code, and info code.

IDI0022W

Suboption(s) missing for *options-source* option *option*

Explanation: An option with missing required suboptions(s) was encountered while processing options that are specified in *options-source*. For the possible values of *options-source*, see message IDI0019W on page 699.

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 on page 699.

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 *sub-opt* 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 on page 699.

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 IPVLANGX or IPVLANGP (or their aliases IDILANGX or IDILANGP), then these modules are provided by ADFz Common Components and must be available via LINKLIST or LPA. See Making Fault Analyzer modules available on page 306 for details.

System action: If *module-name* is not IPVLANGX or IPVLANGP (or their aliases IDILANGX or IDILANGP), or this message was not issued while using the IBM Fault Analyzer plug-in for Eclipse, processing continues without module *module-name*. However, depending on the functionality of the subject load module, processing might be incomplete. If *module-name* is IPVLANGX or IPVLANGP (or their aliases IDILANGX or IDILANGP), and this message was issued while using the IBM Fault Analyzer plug-in for Eclipse, the Fault Analyzer process of the ADFz server terminates, resulting in the inability to use the IBM Fault Analyzer plug-in for Eclipse.

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 might be an internal error. Contact IBM® Support.

IDI0029W

options-source options syntax error at offset *offset*: *reason*

Explanation:

A syntax error was encountered while processing options that are specified in *options-source*, where *options-source* is one of the following:

PARM field

This field is the JCL EXEC statement PARM field specified in a job that executes program IDIDA.

Environment variable _IDI_OPTS

This environment variable is set by the abending application.

System action: Processing continues.

User response: Correct the error and resubmit your job.

IDI0030W

Allocation to SYSOUT=(*class*,,*form-name*) failed: DD=*ddname* RC=*return-code* Error=*error-code* Info=*info-code*

Explanation: Dynamic allocation of JES spool data set failed.

System action: Processing continues without the DDname identified in *ddname*.

User response: Refer to *z/OS®: MVS™ Programming: Authorized Assembler Services Guide* for information about the return code, error code, and info code.

IDI0031W

Open of DD=*ddname* failed: *reason*

Explanation: An unsuccessful attempt was made to open DDname *ddname*.

System action: Processing continues without the DDname identified in *ddname*.

User response: Refer to *reason* for a possible explanation.

IDI0032W

I/O error writing report: *reason*

Explanation: An I/O error occurred while writing the analysis report.

System action: Processing continues, however, the analysis report might be missing information.

User response: Refer to *reason* for a possible explanation.

IDI0033E

I/O error writing to *history-file-dsn*: System abend *abend-code-reason-code reason-text*

Explanation: An I/O error occurred while writing a fault history record to the history file data set shown in *history-file-dsn*.

System action: Processing continues, however, the history file does not contain information for the current job.

User response: Refer to *abend-code*, *reason-code* and *reason-text* for a possible explanation.

IDI0034I

Fault analysis skipped due to: *reason*

Explanation:

A job was excluded from fault analysis for the reason that is identified in *reason* as one of the following:

- **EXCLUDE option specification (FAST)** A match for the current fault was found during fast Exclude options processing (see Fast Exclude options processing on page 372).
- **Parmlib config member Exclude option specification** An Exclude option that is specified in the IDICNFxx configuration member matched the current fault attributes. No subsequent Include or Exclude option specifications in a user options file matched, and no attempt was made to override the exclusion by an Analysis Control user exit.
- **User options file Exclude option specification** An Exclude option that is specified in the IDIOPTS user options file matched the current fault attributes. No attempt was made to override the exclusion by an Analysis Control user exit.
- **Analysis Control user exit request** No Exclude option specification in either the IDICNFxx configuration member or the IDIOPTS user options file matched the current fault attributes. Instead, an Analysis Control user exit requested the exclusion.
- **Abend *abend-code* during options processing** An abend occurred during real-time options processing.

System action: No fault analysis is performed.

User response: If an abend occurred, and the problem persists, contact IBM Support.

IDI0035W

No dump records found in *data-set-name*

Explanation: An attempt was made to read a dump data set during reanalysis, but the dump data set did not contain any data.

System action: Fault Analyzer attempts to continue without using the dump data set.

User response: Ensure that the correct dump data set was specified to Fault Analyzer.

IDI0036E

Abend *abend-code* during processing of *exit-type* user exit *exit-name*--refer to message explanation for problem determination information

Explanation: An abend occurred during execution of a user exit. In the message text, *abend-code* is the type of abend that occurred, *exit-type* is the type of user exit that abended, and *exit-name* is the name of the user exit.

System action: Processing continues but no further calls are made to this exit.

User response:

To obtain a dump of this situation for debugging purposes, allocate DDname IDITRACE to anything other than DUMMY and a dump data set to the abending job using, for example, the following JCL statements:

```
//IDITRACE DD SYSOUT=*
//SYSMDUMP DD DISP=SHR,DSN=my.dumpdsn
```

IDI0037E

Incomplete fault history file record detected

Explanation: An error occurred when attempting to open a fault entry for reanalysis.

System action: The reanalysis is terminated.

User response: If the problem persists, contact IBM® Support.

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 in the dump data set name was either more than five minutes earlier than 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 load module name identified in *module-name* is composed of IDIHM followed by the Language option in effect. Either the specified language identifier is unsupported or the load module library in which the module resides was not found in the 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 612) 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 616 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 fault entry is suppressed.

User response: None

IDI0046I

MVS SVC dump registration skipped due to: *reason*

Explanation:

MVS™ SVC dump registration via the Fault Analyzer IDIXTSEL post-dump exit was not performed for the reason that is identified in *reason* as one of the following:

- **Parmlib config member Exclude option specification** An Exclude option that is specified in the IDICNFxx 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 IDICNFxx configuration member matched the current fault attributes. Instead, an Analysis Control (dump registration) user exit requested the exclusion.
- **Abend *abend-code* during options processing** An abend occurred during options processing.

System action: No fault entry is created.

User response: None

IDI0047S

IBM Fault Analyzer internal abend *abend-code*

Explanation: Fault Analyzer terminated abnormally with the abend code shown. This message is equivalent to the IDI0120S on page 721 message, but issued under different circumstances.

System action: Processing terminates.

User response: Contact IBM Support.

A dump taken at the time of the IDI0047S message being issued is required for problem analysis. If no existing recovery fault recording dump is available, one can be obtained by setting the following SLIP trap before recreating the problem:

```
SL SET,ID=xxxx,MSGID=IDI0047S,A=SVCD,END
```

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*, contact IBM Support.

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 that is 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 of these, Fault Analyzer cannot perform fault reanalysis.

System action: Processing terminates.

User response: Specify either the FaultID or the DumpDSN option and retry the reanalysis.

IDI0052I

count page minidump suppressed from the fault entry being created

Explanation: A minidump consisting of *count* 4K pages was suppressed by Fault Analyzer and therefore not written to the history file with the fault entry being created. However, the remainder of the fault entry is still attempted written. The suppression might be due to the maximum number of minidump pages having been exceeded, or a decision made by an invoked End Processing user exit.

System action: Processing continues.

User response: None.

IDI0053I

Fault history file entry suppressed due to: *reason*

Explanation:

No updates were made to the history file for the current fault. The reason for the suppression is provided in *reason* as one of the following:

DUMMY history file specification

The IDIHIST DDname was specified as DUMMY

History file ENQ timeout

Fault Analyzer was unable to serialize on the history file within a set period of time.

History file access error

If a history file open or read failure, it's likely that message IDI0016E on page 699 or IDI0078E on page 712 was issued prior to this message.

Duplicate fault or End Processing user exit

If duplicate fault detection was the reason, then message IDI0044I on page 705 is issued prior to this message. Otherwise, an End Processing user exit requested suppression.

End Processing user exit

During fault entry refresh processing, an End Processing user exit requested suppression.

System action: Processing continues.

User response: Unless the reason for suppression of the fault entry was that it had either been deemed a duplicate, or a user exit had requested suppression, then if the problem persists, the cause of suppression should be investigated. Check if earlier issued messages provide more information, or check if another user or job continues to hold an ENQ against the history file.

IDI0055E

Fault Analyzer processing excluded because *num* MB 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* MB 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 parameter. See Storage recommendations on page 296.

Note:

- 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.
- For CICS®, this entire amount of storage must be provided through the JOB or EXEC JCL statement REGION parameter; *not* through the EDSALIM parameter.

Although Fault Analyzer was unable to perform real-time analysis due to storage constraints, recovery fault recording (see Recovery fault recording on page 51) is attempted in a non-CICS environment.

IDI0056E

REXX environment initialization failed, IRXINIT rc=*rc* reason=*reason*

Explanation: At attempt to initialize a REXX environment failed. In the message text, *rc* is the return code and *reason* is the reason code returned by the REXX initialization routine, IRXINIT. For information about the return and reason codes, see *z/OS®: TSO/E REXX Reference*.

System action: Processing continues. However, no REXX exec user exits are invoked and no diagnostic information is written to the IDITRACE DDname.

User response: Refer to the description of return and reason codes in *z/OS®: TSO/E REXX Reference*.

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 IDICNFxx 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 cannot 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, an attempt will be made to automatically recover the HI segment from the $$BACKUP member when the history file is next updated, for example when a new fault entry is written. Ensure that the data set identified is a valid history file PDS or PDSE, and if necessary, delete the member in error. However, take care not to delete the $$INDEX member, unless the problem persists, as this deletion 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 table lists the possible response codes:

Table 27. Response codes

Response code	Meaning
1	OK
2	Exception
3	Disaster
4	Invalid
5	Kernel error
6	Purged

The reason codes are associated with each command:

Table 28. Reason codes

Reason code	Meaning
FREEMAIN	
(No applicable reason codes)	
GETMAIN	
1	Insufficient storage
INQ_APPLICATION_DATA	
1	DPL program
2	No transaction environment
3	Transaction domain error
4	Invalid function
5	Abend
6	Loop
7	Inquiry failed
INQUIRE_TRANSACTION	

Table 28. Reason codes

(continued)

Reason code	Meaning
1	No transaction environment
3	Buffer too small
7	Invalid transaction token
37	Abend
39	Loop
START_PURGE_PROTECTION	
(No applicable reason codes)	
STOP_PURGE_PROTECTION	
(No applicable reason codes)	
WAIT_MVS	
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 616 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 fault 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 IBM Support. If possible, send the joblog.

IDI0071I

function-name of program *program-name (api,concurrency)* completed successfully

Explanation: This message is a confirmation that the command that was issued has completed.

System action: Processing terminates.

User response: None.

IDI0072I

Program *program-name* is *status*

Explanation: This message provides the current exit program name and status.

System action: Processing continues.

User response: None.

IDI0073I

Usage: CFA <Install|Uninstall>

Explanation: The arguments passed to the CFA transaction were incorrect. The message provides the user with a brief reminder of the correct usage.

System action: Processing continues.

User response: None.

IDI0074E

User not authorized to issue command *command-name*

Explanation: The user that is associated with the issuing task is not authorized to use this command or is not authorized to access resources in the way required by this command.

System action: The attempted command was not successful.

User response: Ensure that the necessary command authority is provided.

IDI0075E

IDIXCCEE CICS exit must abort. CICS SOS pending.

Explanation: Not enough storage was available to initiate analysis. This lack of storage might happen during times of high CICS® system activity or when multiple CICS® transaction abends are being analyzed concurrently by Fault Analyzer.

System action: Processing terminates.

User response: If the problem did not occur due to high CICS® system activity or the analysis of multiple concurrent CICS® transaction abends, try increasing the CICS® DSA size.

IDI0076E

A Fault Analyzer CSVINFO MIPR routine abend occurred

Explanation: An abend occurred during the processing of a CSVINFO macro to retrieve information about a load module.

System action: CSVINFO processing terminates for the load module.

User response: Contact IBM Support.

IDI0077I

Fault Analyzer internal diagnostic: *text*

Explanation: This message is an internal Fault Analyzer message used to display specific diagnostic information.

System action: Processing continues.

User response: None.

IDI0078E

Open of fault history file *data-set-name* failed because: *reason*

Explanation: The history file that was identified by *data-set-name* could not be opened for the reason that is identified in *reason*.

If *reason* is shown as "DDfopenw() rc=X'800*sssrc*'", then this reason indicates an abend during open processing, where:

sss

The hexadecimal MVS™ system abend code.

rc

The associated hexadecimal reason code.

System action: A fault entry might not have been written to the history file.

User response: Determine the reason for the open failure and resubmit the job. Check that the attributes for the file definition are VB LRECL 10000 PDS or PDSE. 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 362 for more information.

IDI0081I

IEWBIND unusual condition *function module-name* rc=*reason-code*

Explanation: An error occurred while using the Binder. A *reason-code* of 83000526 indicates an unusual condition encountered for an input module. This condition occurs when a third-party product adds its own management information to a load module. To suppress this message, use `Quiet(IDI0081I)`.

System action: Processing continues.

User response: The value in *reason-code* is the Binder API reason code. The Binder API reason codes can be found in *z/OS®: MVS™ Program Management: Advanced Facilities*.

IDI0082E

DB2 *subsys-id* Call Level Interface error: *reason*

Explanation: An attempt to obtain DB2® information through the DB2® Call Level Interface (CLI) failed for the DB2® subsystem identified by *subsys-id*. Refer to *reason* for details about the error.

System action: Processing continues but analysis might be incomplete.

User response: Ensure that the DB2® Call Level Interface is installed and required setup has been performed. An application plan to bind DBRMs to packages must exist. A default application plan can be created using the sample job in member DSNTIJCL of the DB2® SDSNSAMP data set (for further information, see Binding DB2 on page 415). 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 Using the Fault Analyzer IDIS subsystem on page 318. Included in the message *reason* is normally the result of calling the SQLError ODBC function, which provides more detailed information about the error. Of particular interest is the 8-digit hexadecimal reason code following the REASON= keyword (for example, REASON=00f30034). The explanation for this reason code can be found in *DB2® for z/OS®: Codes*

IDI0083E

Fault Analyzer SVC error: *reason*

Explanation: An error occurred during the Fault Analyzer SVC execution. Refer to *reason* for details about the error. Intermittent occurrences of this message can be expected since Fault Analyzer traverses subsystem address spaces without holding any locks.

System action: Processing continues but analysis might be incomplete.

User response: Contact IBM Support if the problem persists. If *reason* indicates an abend during the Fault Analyzer SVC execution, an SVC dump should have been written. Retain this dump for diagnostic purposes.

IDI0084E

Fault Analyzer Exit and Main task level mismatch

Explanation: An invocation exit was found to be at a different version or maintenance level than the main Fault Analyzer module, IDIDA.

System action: Processing continues but errors might occur.

User response: Ensure that all Fault Analyzer modules used are at the same version and maintenance level.

IDI0085E

Fault Analyzer purged, detaching subtask TCB

Explanation: An abending CICS® task was running Fault Analyzer when the task was purged by the operator. Fault Analyzer is detaching its analysis subtask TCB.

System action: Processing terminates.

User response: None.

IDI0086E

Fault Analyzer processing excluded because *size* k of 24 bit storage is not currently available

Explanation: Not enough below-the-line storage was available for analysis to be performed. The *size* indicates the amount of storage required in kilobytes.

System action: Fault analysis is excluded for the current fault.

User response: Ensure that sufficient below-the-line storage is available. For more information, see Storage recommendations on page 296. Although Fault Analyzer was unable to perform real-time analysis due to storage constraints, recovery fault recording (see Recovery fault recording on page 51) is attempted in a non-CICS environment.

IDI0087I

size Meg of 31 bit storage could be provided by SETPROG LPA,ADD,MOD(*module-list*),DSN=LNKLST

Explanation: When analysis could not be performed due to insufficient above-the-line storage, and all required modules of significant size have not yet been loaded into LPA, this message is issued to indicate the region size space savings that could be achieved if loading these modules into LPA. For more information, see Storage recommendations on page 296. Message IDI0055E on page 707 precedes this message to indicate the required above-the-line region size that could not be obtained.

System action: Processing terminates.

User response: Ensure that the amount of above-the-line storage indicated in message IDI0055E on page 707 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.

Note: Ensure ECSA size is sufficient to accommodate the additional storage required by the SETPROG LPA,ADD command.

IDI0088E

Fault Analyzer captured IEWBIND abend *abend-code*, processing continues

Explanation: While attempting to map CSECTs in a load module, Fault Analyzer invoked the 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.

IDI0089I

Fault Analyzer Subsystem *task name* RC=*rc*, Rsn=*rsn description*

Explanation: If available, the Fault Analyzer subsystem task name is provided in *task-name*. This message is issued when an unexpected condition occurred in the Fault Analyzer IDIS subsystem interface. Included in the message *description* might be the result of calling the SQLError ODBC function, which provides more detailed information about the error. Of particular interest is the 8-digit hexadecimal reason code following the REASON= keyword (for example, REASON=00f30034). The explanation for this reason code can be found in *DB2® for z/OS®: Codes*

System action: Processing continues, but the analysis might be incomplete.

User response: See if another message, such as IDI0082E on page 713, was also issued, which might help explain the situation.

IDI0090E

Execution of REXX exec *exec-name* failed, IRXEXEC rc=*return-code*

Explanation: This message is issued when the execution of a REXX user exit was unsuccessful.

System action: Processing continues although the execution of the exit failed.

User response: Using IDITRACE, look for REXX messages explaining the problem. Refer to *z/OS®: TSO/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 IDIXTSEL MVS™ post-dump exit.

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 IBM Support if the problem persists.

For problem analysis, a dump that was taken when the IDI0092S message was issued is required. If no existing recovery fault recording dump is available, one can be obtained by setting the following SLIP trap before recreating the problem:

```
SL SET,ID=xxxx,MSGID=IDI0092S,A=SVCD,END
```

To disable the Fault Analyzer wait and loop protection feature for other than short term interval conditions (see "Short term interval" above), the NoLoopProtection option can be used. For details, see LoopProtection on page 614.

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 reported to IBM Support.

System action: Processing continues.

User response: Contact IBM Support.

IDI0096S

CICS Task *task-id* was force purged by an operator while *state* fault analysis

Explanation: Analysis of a CICS® transaction fault has been terminated by operator intervention. In the message text, *task-id* identifies the task that was force purged, and *state* might be either "waiting for" or "performing" to indicate the fault analysis activity that was terminated.

System action: Processing terminates.

User response: None.

IDI0097W

Unsupported REXX execution environment detected - no REXX services available

Explanation: The protection key of the job-step TCB was found to be set to a key other than 8. Since this environment is an environment which is not supported by REXX, then no REXX services are available during the fault analysis. This lack of services includes any calls to REXX user exits and all output to the IDITRACE DDname.

System action: Processing continues without REXX services.

User response: None.

IDI0098S

Exit *exit-name* NOT installed, Fault Analyzer SVC not found

Explanation: The Fault Analyzer invocation exit, identified in *exit-name*), was attempted installed. However, since the Fault Analyzer SVC has not yet been installed, processing is terminated.

System action: Processing terminates.

User response: Install the Fault Analyzer SVC (see step 5 on page 307) 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 307) 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 307) 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 578.

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 IBM Support 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 IBM Support 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 on page 696, but is used for all storage allocation requests involving a negative length.

System action: Processing terminates.

User response: Contact IBM Support.

A dump taken at the time of the IDI0105S message being issued is required for problem analysis. If no existing recovery fault recording dump is available, one can be obtained by setting the following SLIP trap before recreating the problem:

```
SL SET,ID=xxxx,MSGID=IDI0105S,A=SVCD,END
```

IDI0106E

ENQ timed out for *major-name minor-name* held by *jobname* on *system-name*

Explanation: The time limit set for an ENQ against the history file, prior to creating a new fault entry, was exceeded. The time limit is approximately 1 minute. The ENQ resource was held by the job *jobname* on system *system-name*. Although a fault entry is not created, an analysis report is still written to IDIREPRT.

System action: Processing continues.

User response: Determine the reason why the ENQ could not be satisfied. This reason might be a TSO/ISPF user who is editing the history file member.

IDI0107I

transaction-id date time exit status

Explanation: This message is issued whenever the Fault Analyzer Control Transaction (*transaction-id*) is used to alter the status of one or more of the CICS® exits (identified by *exit*): XPCABND, XDUREQ, LE Exit, or SDUMP Exit. The *status* field shows either Uninstalled or Installed.

System action: Processing continues.

User response: None.

IDI0108I

The IDIS subsystem will not process *data-set-name* because it is not a PDSE history file.

Explanation: This message is issued by the Fault Analyzer IDIS subsystem when PARM='UPDINDEX' is used and a history file, which is not a PDSE, is attempted opened. In the message text, *data-set-name* identifies the history file that cannot be cached in the IDIS subsystem. For more information about IDIS subsystem caching, see Caching of history file $$INDEX data on page 319.

System action: Processing continues, but all I/O to the history file $$INDEX member is performed by the requester (real-time analysis, batch or interactive reanalysis, or Fault Analyzer ISPF interface).

User response: If you want to use the potential performance improvements that can be realized by having the Fault Analyzer IDIS subsystem manage the history file $$INDEX member, then a PDSE history file must be used.

IDI0109E

PC recovery entered psw *program-status-word* abend=*abend-word*

Explanation: This message is issued when an error has occurred in the Fault Analyzer subsystem program call interface.

Immediately following this message are usually extra messages that provide more detailed information about the error. For example:

```
IDISSPC PC recovery IDISSPC=16703250 IDISAMAN=16700000
IDISSPC R0-R3     00000001 171EAE20 80FF28FE 00000002
IDISSPC R4-R7     008A6770 008A69E0 008FD0C8 00F7DA00
IDISSPC R8-R11    00000000 00FF24AC 00000C80 00F7DA00
IDISSPC R12-R15   00000000 00000000 80FF2A2A 808A69E0
IDISSPC AR0-AR3   00000000 00000000 00000000 00000000
IDISSPC AR4-AR7   00000000 00000001 00000000 00000000
IDISSPC AR8-AR11  00000000 00000001 00000000 00000000
IDISSPC AR12-AR15 00000000 00000000 00000000 00000000
```

System action: Processing continues, but the subsystem service that failed might cause unexpected results. If it is a S102 abend, then this problem can happen if the job or task that was requesting services from the Fault Analyzer subsystem was canceled before the subsystem request had completed.

User response: Contact IBM Support 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 attempts to query the SYSIBM.SYSDBRM DB2® catalog table for DB2® subsystem name *db2_name* exceeded the expected maximum time indicated by the *num_secs* value. No further attempts to access the SYSIBM.SYSDBRM table on the identified DB2® subsystem is made for 30 minutes.

System action: Processing continues, but some DB2-related information might be missing from the analysis report for the fault that caused this message to be issued, as well as for any subsequent faults involving the same DB2® subsystem.

User response: Create an index on SYSIBM.SYSDBRM for improved performance. For details, see Improving Fault Analyzer DB2 performance on page 415.

IDI0113W

Subtask *tcb-addr*(*description*) return ECB=*ecb-value* during problem reanalysis under CICS®

Explanation: An error occurred while performing interactive reanalysis under CICS®. This error might be caused by an abend, or by an incomplete setup of the interactive reanalysis component under CICS® (see Performing interactive reanalysis under CICS on page 284).

System action: Interactive reanalysis terminated.

User response: Check for other messages that might help explain the problem. Contact IBM Support if the problem persists.

IDI0114W

XMEM POST ABEND S102, requester termination

Explanation: This problem can happen if the job or task that was requesting services from the Fault Analyzer subsystem was canceled before the subsystem request had completed.

System action: The analysis has already been canceled.

User response: Contact IBM Support 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 IBM Support.

IDI0116E

IPVLANGX 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 IBM Support.

IDI0117E

ABEND<*abend-code*> during return POST, request from task *jobname job-id* may have been canceled

Explanation: An abend occurred in the Fault Analyzer subsystem POST processing. This abend might have been caused by a CANCEL of the request from the task identified by *jobname* and *job-id*.

System action: Processing continues.

User response: If the problem persists, contact IBM Support.

IDI0118W

CICS task *task-number* abend *abend-code* analysis skipped due to max waiting exceeded.
CICSFAtasks(*max_slots*,*max_waits*)

Explanation: The maximum number of faults allowed to be queued for analysis had already been reached when another fault occurred. This limit is controlled by the CICSFAtasks suboption of the DeferredReport option, and the current values in effect are provided in the message text.

System action: Analysis is skipped, and normal CICS® transaction dump analysis needs to be performed (that is, not using Fault Analyzer).

User response: Unless the maximum values are already in effect, the CICSFAtasks suboption of the DeferredReport option can be used to permit more parallel execution tasks to be used, or to increase the number of faults that are allowed to be waiting for analysis. See DeferredReport on page 589.

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 IBM Support.

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 on page 705 message, but issued under different circumstances.

System action: Processing terminates.

User response: Check for previously issued severe (S-level) Fault Analyzer messages that might explain the reason for the problem (for example, message IDI0005S on page 696, which indicates a storage shortage problem). If no prior S-level messages were found, contact IBM Support.

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 616.) The earlier occurring fault, of which the current fault was deemed to be a duplicate, is identified by the IMS™ program name (*ims_pgm*), the assigned fault ID (*fault_id*), and the history file data set name (*histfile*).

System action: Processing continues. No fault analysis is performed, but the duplicate count of the earlier occurring fault is incremented, and the duplicate details recorded for later viewing with the Fault Analyzer ISPF interface (see Viewing the fault entry duplicate history on page 134).

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:

Before or during analysis of the abend shown in *abend-code*, Fault Analyzer determined that processing could not proceed for one of the following reasons:

- **SUB=MSTR address space** The address space was started with SUB=MSTR and does therefore not permit SYSOUT allocations.
- **Not full function address space** The current address space was found to be a limited function address space, which does not support required system services, such as allocation of data sets.
- **System address space** The started task was a known system address space for which Fault Analyzer analysis is not appropriate and likely to fail.
- **Abend in system task** The abend occurred under an RB for a system task, such as the IEFIIC initiator TCB.
- **JES not available** The primary subsystem (JES) was found to be not available.
- **Fault Analyzer invoked during JES shutdown** Fault Analyzer determined that JES became unavailable during the processing of an abend.

- **Dynamic Allocation not available** Required Dynamic Allocation services (SVC 99) were found to be not available for the current task.
- **Owning Job-step TCB is abending** If the current TCB is not the job-step TCB, and the owning job-step TCB is terminating, then load failures are likely to occur.
- **Abend in resource manager** A resource manager was active for the current TCB.
- **Concurrent analysis in the address space** To prevent exhausting system resources, only one TCB in a given address space is allowed to perform Fault Analyzer abend analysis at a time. When Fault Analyzer was invoked for the current abend, it was determined that another Fault Analyzer abend analysis was already active, causing the current analysis to be canceled.
- **SYSZTIOT enqueued by TCB *tcb-address*** Fault Analyzer was invoked to analyze an abend with the SYSZTIOT major name already enqueued by the TCB identified by the address in *tcb-address*. An example of when this situation might occur is during shutdown of IMS™ when Fault Analyzer is invoked for a U0002 abend.
- **TCB Not protect KEY 8 or 9** Fault Analyzer was invoked to analyze an abend for a task not running in protect key 8 (non-CICS) or 9 (CICS®).

System action: Processing terminates.

User response: Real-time analysis is not possible under these conditions. Fault Analyzer might still be used to aid with the problem determination by setting a SLIP trap using the MSGID=IDI0123S and A=SVCD parameters, and then performing analysis of the resulting SVC dump by way of the Fault Analyzer ISPF interface **File** menu option 5.

IDI0124E

IDIS subsystem subtask *server-id resource-id* has terminated with abend code *abend-code*

Explanation: A subtask in the Fault Analyzer subsystem terminated abnormally.

System action: The Fault Analyzer subsystem subtask is terminated, but is restarted as necessary. Requester processing continues, but results might be unexpected depending on the type of subsystem request that failed.

User response: If the problem persists, contact IBM Support.

IDI0125W

IDIS subsystem subtask *server-id resource-id* has terminated with return code *rc*

Explanation: A subtask in the Fault Analyzer IDIS subsystem terminated abnormally.

System action: The IDIS subsystem subtask is terminated, but is restarted as necessary. Requester processing continues, but results might be unexpected depending on the type of subsystem request that failed.

User response: If the problem persists, contact IBM Support.

IDI0126I

Recovery fault recording fault ID *faultid* assigned in history file *history-file-name*

Explanation: Fault Analyzer was unable to complete normal real-time analysis, but has created a recovery fault recording fault entry *faultid* in history file *history-file-name*. Reanalysis of this fault entry should provide information about the fault that was being analyzed when Fault Analyzer terminated.

System action: Processing has ended.

User response: None

IDI0127W

Recovery fault recording failed for *job-id*. *reason*

Explanation:

Recovery fault recording processing failed for the job identified by *job-id* and the reason that is identified in *reason* as one of the following:

- **SDUMP suppressed, capture phase of another SVC dump was in progress.** SDUMP failed with return code 8 and reason code 2, indicating that an SVC dump was suppressed because the capture phase of another SVC dump was in progress.
- **SDUMP DASD space or overload error.** SDUMP failed with return code 8 and reason code 3E, indicating that SVC dump is already using the maximum amount of virtual storage (as determined by the installation, using the MAXSPACE parameter on the CHNGDUMP command) to process other dumps. Typically, the reason for this condition is a shortage of DASD space.
- **SDUMP rc=*return-code* reason=*reason-code* error.** SDUMP failed with return code *return-code* and reason code *reason-code*. Refer to *z/OS®: MVS™ Programming: Authorized Assembler Services Reference*, *Volume 3 (LLACOPY-SDUMPX)* for an explanation of these codes.
- **RFR requires the IDIS subsystem to be functioning correctly.** A situation occurred for which recovery fault recording would normally be performed. However, since the IDIS subsystem was not functioning correctly, recovery fault recording was not possible.

System action: Termination continues.

User response: Perform any appropriate action, depending on the reason that is identified. To enable recovery fault recording, the IDIS subsystem must be started and be functioning.

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, contact IBM Support.

IDI0131W

Waiting *mins* minutes for *dsn*(*mbr*) SPFEDIT ENQ *owner*

Explanation: There was a problem 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. If available, the current owner of the ENQ is shown in *owner*. For example: *"held by IDIS on SYS8"*.

System action: The IDIS subsystem continues to wait for the required data set access.

User response: Determine why the identified data set member is not available for update. A possible reason is that a TSO/ISPF user is editing the member.

IDI0132W

MaxWaitSeconds of *seconds* exceeded for transaction *transaction* (task *task*), analysis will be skipped

Explanation: The MaxWaitSeconds value in effect for the DeferredReport(CICS(FATasks(…))) option was exceeded. For details about this option, see DeferredReport on page 589.

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: XDUMPs were not enabled and the DeferredReport option was in effect while the MaxMinidumpPages option limit was exceeded.

System action: No minidump is written to the fault entry, but the analysis is still performed and a report is written to both the fault entry and to IDIREPRT.

User response:

The following might help to prevent this situation:

- Enable the use of XDUMP data sets. See Extended minidump data set (XDUMP) on page 53 for details.
- Ensure that the MaxMinidumpPages option setting is sufficiently high.

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 310).

IDI0136W

Recovery fault recording IEATDUMP not taken because NULLFILE has been selected for the DSN

Explanation: Recovery fault recording processing was unable to write an IEATDUMP data set due to no eligible data set name having been determined.

System action: Processing is terminated.

User response: None.

IDI0137W

I/O Error

Explanation: An error occurred during a data set I/O operation.

System action: Processing continues.

User response: Check for more messages related to this error.

IDI0138S

No minidump or MVS dump data set is available for reanalysis of history file *hist-file* fault ID *fault-id*

Explanation: Batch reanalysis was attempted of a fault entry that did not either include a minidump, or was associated with an existing MVS™ dump data set. Without one or both of these, reanalysis is not possible.

System action: Processing is terminated.

User response: None.

IDI0140S

Processing terminated due to data set open error for DDname *ddname*

Explanation: The open of a data set for a required DDname failed. Normally, this message is preceded by one or more other messages which provide more detailed information about the error (for example, message IDI0006E on page 696).

System action: Processing is terminated.

User response: Check options specification relating to the identified DDname.

IDI0141W

Please use MODIFY-STOP for the IDIS subsystem

Explanation: This message is issued if the CANCEL command was used to stop the IDIS subsystem.

The correct way to stop the IDIS subsystem is to use the MODIFY or STOP command:

```
F name,STOP
```

or

```
P name
```

System action: The IDIS subsystem is stopped.

User response: None.

IDI0142W

Dispatch delay in IDIS subsystem (Priority=*idis-priority*) exceeded *num* seconds for *jobname job-id* (Priority=*job-priority*)

Explanation: This message is issued if the IDIS subsystem response time exceeded expectations for the requester shown.

System action: Processing continues.

User response: Review the dispatch priority of the IDIS subsystem to ensure that it is not less than that of the job identified by *jobname* and *job-id* which was requesting Fault Analyzer subsystem services.

IDI0143W

Binder processing stopped because *num* k of 24 bit LSQA storage is not currently available

Explanation: Not enough 24-bit LSQA storage was available to invoke the Binder.

System action: Processing continues, but source line information might be missing from the analysis report.

User response: None.

IDI0144E

IDIS subsystem TCB *tcb-address detection-location* abended *abend-code*

Explanation: An IDIS subsystem TCB or function abended or detected an error condition.

System action: Processing continues, but information might be missing from the analysis report.

User response:

If the problem persists, contact IBM Support. If a fix is not available, set a SLIP trap as follows and provide the dump for analysis:

```
SL SET,ID=xxxx,MSGID=IDI0144E,A=SVCD,END
```

IDI0145I

message-text

Explanation: This message is used for all status messages that are issued by the IDIS subsystem.

Examples of these messages are:

```
IDI0145I IDISXCFA TCB XCF startup
IDI0145I IDIS subsystem, IDISMAIN Started. V9R1M0
         (MVS 2009/02/03)
IDI0145I Starting Termination.
```

System action: Processing continues.

User response: None.

IDI0146I

IDIS subsystem storage use is at *percent-value* percent of JCL REGION size, running *number-of-tasks* tasks

Explanation: This message is issued by the IDIS subsystem whenever the amount of storage used exceeds 80% of the maximum available storage, as specified in the JCL REGION parameter, or imposed by default. The message is reissued at intervals while the storage usage remains high, but is not issued if the storage usage drops below 80% again.

System action: Processing continues.

User response: If this message is issued regularly, 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 Fault Analyzer SDUMPs

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 310 for information about how to use SDUMPs for improved recovery fault recording performance.

IDI0148E

Unable to create an RFR fault entry due to IDIS subsystem NOUPDINDEX option in effect.

Explanation: This message is issued by the IDIS subsystem when an RFR dump is about to be written, but the subsystem was started with the NOUPDINDEX option.

System action: An SDUMP of the problem might still be taken, but it will not be recorded in a fault entry in a history file.

User response: Perform analysis of the SDUMP, if available, to identify the problem. Restart the subsystem with the UPDINDEX option in effect.

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 that is specified for the IDIMAPS DDname is correct (this name is normally specified using the DataSets option in the IDICNFxx parmlib member), and contains the current data from the SMP/E target library.

IDI0150W

No READ access to DDname *ddname* data set name *dsname*. This data set will not be used.

Explanation: A data set provided to Fault Analyzer, either explicitly or implicitly, was found to be inaccessible due to no security server READ access.

System action: Processing continues, but errors might occur if the data set is critical.

User response: Provide the appropriate access to the data set.

IDI0151W

SDUMP failure *reason* [Job *job_id*]

Explanation:

Recovery fault recording processing or Java™ analysis failed with the reason that is identified in *reason*. If a related job ID is available, then it will be included in the message as *job_id*. For example:

- **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 724 for more information about the SDUMP return and reason codes.
This message is issued by the IDIS subsystem.

System action: IDIS subsystem processing continues.

User response: Perform any appropriate action, depending on the reason that is identified.

IDI0152I

Job *jobname* SDUMP requested for *history-file*(*fault-id*)

Explanation: An SDUMP recovery fault recording dump data set was requested for the job identified by *jobname*, which is to be associated with fault ID *fault-id* in history file *history-file*.

System action: Processing continues.

User response: None.

IDI0153I

Binder processing terminated for member *module-name* because it was created with the LINK=NO option

Explanation: A call to the Binder program for load module *module-name* failed with rc=83000505 due to the use of the LINK=NO linkedit option.

System action: Processing continues without binder information for the identified load module.

User response: None.

IDI0154W

Configuration-options module *module-name* found in non-authorized load library and has been ignored

Explanation: The load module identified by *module-name*, which can be used to provide configuration-options for Fault Analyzer, was found in a load library which was not APF-authorized. The load module name can be either IPVOPTLM or IDIOPTLM. Because it is a requirement that this load module must be placed in an APF-authorized load library in order to be used, it has been ignored.

System action: Processing continues.

User response: Place the load module in an APF-authorized load library.

IDI0155W

The user ID in use for the IDIS subsystem does not have an OMVS segment with a HOME path

Explanation: If an IDIJLIB DDname has not been specified in the IDIS subsystem JCL, then Fault Analyzer instead uses the IDIS subsystem user ID OMVS segment HOME path for work files. In this case, neither an IDIJLIB DDname, nor a HOME path were available.

System action: Processing continues, but without Java™ analysis support.

User response: Ensure that either the IDIJLIB DDname has been specified in the IDIS subsystem JCL, or that an OMVS segment HOME path exists for the IDIS subsystem user ID.

IDI0156W

GETMAIN of *count* bytes from *jobname job-id* address space failed *abend-code*, *idis-module-name history-file-name* IDIS subsystem request failed

Explanation: When the IDIS subsystem was returning data, a cross memory GETMAIN for *count* bytes failed, resulting in abend *abend-code*. This abend occurred because of storage shortage in the requester address space identified by *jobname* and *job-id*.

System action: Processing continues.

User response: If possible, increase the region size of the requester address space to prevent this problem from happening in the future.

IDI0157I

Fault Analyzer about to deliberately abend U0777 and take RFR dump due to IDIRFRON DDname

Explanation: When an IDIRFRON DD statement is used, Fault Analyzer deliberately issues abend U0777 in order to cause a recovery fault recording fault entry to be created. For more information about the use of the IDIRFRON DDname, see Verifying the recovery fault recording setup on page 439.

System action: Processing terminates.

User response: None.

IDI0158W

IDIS subsystem requires restart with STEPLIB containing SMP/E *.SIDIAUT2 to load DLL *dll-name*

Explanation: Analysis of a Java™ fault was attempted but failed due to the absence of data set IDI.SIDIAUT2 in the IDIS subsystem STEPLIB concatenation.

System action: Analysis continues but Java™ information is missing.

User response: Add IDI.SIDIAUT2 to the IDIS subsystem STEPLIB concatenation. For details, see Starting the IDIS subsystem on page 320.

IDI0159I

SDUMP requested for *job-id1 fault-id* will not be taken because *reason* 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 *reason* is one of the following:

- "Current *job-id1*," if the SDUMP rate threshold for the current job has been exceeded.
- "*job-id2*," if the SDUMP threshold for another job (*job-id2*) has been exceeded
- "High system," if the threshold for all jobs combined has been exceeded.

System action: The recovery fault recording SDUMP is not taken.

User response: Contact IBM Support to determine why a high rate of Fault Analyzer recovery fault recording SDUMPs were requested. 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 on page 703, Fault Analyzer was able to reclaim sufficient space in the history file identified by *history-file* to permit the subsequent successful rewriting of the fault entry.

System action: Processing continues normally.

User response: None.

IDI0161W

History file *history-file* I/O error recovery failed: *reason*

Explanation:

An attempt to recover from an I/O error on the history file identified by *history-file* failed for the reason that is identified by *reason*:

- **History file is not a PDSE** I/O error recovery is not available for PDS history files.
- **History file contains 25 or less fault entries** As for AUTO-space management, fault entries are only implicitly deleted if the history file contains more than 25 fault entries.
- **IGWFAMS error** Message IDI0095W on page 716 is issued immediately prior to this message with error-specific information.
- **Unable to provide the required space** No more fault entries could be deleted, but the required amount of space had not yet been made available. This lack of required space might be due to fault entries being locked.
- **Recursive I/O error** An I/O error occurred during the recovery of an earlier I/O error.

System action: Processing continues, but a fault entry is not written to the history file.

User response:

One or more of the following might be appropriate:

- Reallocate the history file with more space.
- Use the IDIUTIL SetMinFaultEntries control statement to change the history file space management setting with a lesser value than the current setting.
- Unlock locked fault entries so they can be implicitly deleted.

IDI0162I

MVS dump taken to extract Java information

Explanation: An MVS™ dump has been taken for the purpose of performing asynchronous extraction of Java™ information for the current fault.

System action: Processing continues.

User response: None.

IDI0164I

Fault ID *fault-id* created in history file *history-file* due to *reason*

Explanation: The batch analysis of an MVS™ dump data set caused the creation of a new fault entry *fault-id* in history file *history-file* due to *reason*.

The possible values for *reason* are:

- GenerateSavedReport option
- IDIRegisterFaultEntry command

System action: Processing continues.

User response: None.

IDI0165A

ddname updates available--restart IDIS subsystem with //IDIDOC2 DD statement for IDI.SIDIDOC2 data set

Explanation: During start-up of the IDIS subsystem, it was determined that one or more updates were available for the VSAM KSDS message and abend code explanation repository, identified by *ddname* as either IDIVSENU, IDIVSJPN or IDIVSKOR. However, no //IDIDOC2 DD statement had been provided to identify the IDI.SIDIDOC2 data set containing the updates.

System action: Processing continues without updating the VSAM KSDS.

User response:

1. Add a DD statement in the IDIS subsystem JCL as follows for the IDI.SIDIDOC2 data set containing the VSAM KSDS updates:
   ```
   //IDIDOC2 DD DISP=SHR,DSN=IDI.SIDIDOC2
   ```
2. Restart the IDIS subsystem.

IDI0166E

Error processing *ddname* update *member*: *reason*

Explanation:

An error occurred while attempting to update the VSAM KSDS message and abend code explanation repository. In the message text:

ddname

Identifies the DDname as either IDIVSENU, IDIVSJPN or IDIVSKOR.

member

The name of the update member in data set IDI.SIDIDOC2.

reason

A description of the error which occurred.

System action: Processing continues, but the VSAM KSDS update is incomplete and is attempted again next time the IDIS subsystem is started.

User response: If the problem persists, contact IBM Support.

IDI0167I

Models processing for CICS *release* returned *count* offsets

Explanation: Under ®CICS, Fault Analyzer needs certain release dependent field offset values. These offsets are determined when Fault Analyzer is installed during CICS® startup, and this message gives a count of the number of offsets found.

System action: Processing continues.

User response: None.

IDI0168E

Models processing for CICS *release* returned 0 offsets

Explanation: Under CICS®, Fault Analyzer needs certain release dependent field offset values. This message is issued when the processing has failed to determine any offset information. This failure prevents Fault Analyzer operation under CICS®.

System action: Fault Analyzer operation under CICS® is not possible without the offset information.

User response: The offset information is extracted from the SIDIMAPS data set member IDIMCICS. Ensure that the IDIMAPS DataSets option has been specified correctly. If the problem persists, set a SLIP trap on this message number and contact IBM Support.

IDI0169E

Module *module-name* loaded from *library-name* is not APF authorized or the concatenation is not APF authorized

Explanation: It is a requirement that the Fault Analyzer load module, identified by *module-name*, is executed from an APF-authorized load library. In this case, the load module was executed from the load library identified by *library-name*, which was either not APF-authorized, or was included in a JOBLIB or STEPLIB concatenation along with one or more other load libraries which were not APF-authorized.

System action: Processing terminates.

User response: Ensure *library-name* is APF-authorized and not included in a concatenation with other load libraries which are not APF-authorized.

IDI0170W

Unable to update *data-set-name* due to no UPDATE access

Explanation: This message is issued by the Fault Analyzer IDIS subsystem if updates are available for the message and abend code explanation repository data set identified by *data-set-name* but the IDIS subsystem does not have UPDATE access to this data set. The updates might be to correct formatting issues or typos in existing explanations, or to add new messages or abend codes.

System action: Processing continues without performing the update.

User response: To perform the update, first grant UPDATE access to *data-set-name* by the IDIS subsystem, then stop and restart the IDIS subsystem.

IDI0171W

Waiting *count* minutes for *history-file-name* PDSE cross-system SHARING contention

Explanation: This message is issued by the Fault Analyzer IDIS subsystem if access to a history file is taking longer than one minute to be obtained, and is issued every minute thereafter until the access has been obtained.

System action: Processing of the history file is suspended until the access is obtained.

User response: If the problem persists, determine who is holding an ENQ on the history file using the SPFEDIT major name.

IDI0172I

Fault Analyzer NameToken anchor built at *storage-addr* by TCB *tcb-addr*

Explanation: This message is issued during Fault Analyzer initialization under CICS®. Its purpose is to provide diagnostic aid for IBM in case of errors.

System action: Processing continues normally.

User response: None.

IDI0173I

Fault Analyzer system DUMP call already issued for Fault Entry *fault-id*

Explanation: During recovery fault recording (RFR) processing, the fault entry identified by *fault-id* was found to already have had a system dump call issued for it. Hence, an extra system dump call is not issued.

System action: Processing continues normally.

User response: None.

IDI0174I

Fault Analyzer Java DUMP fault entry written to *history-file1*(*fault-id*) because IDIS subsystem cannot write to *history-file2*

Explanation: The Fault Analyzer IDIS subsystem was unable to write a Java™ dump fault entry to history file *history-file2* because of insufficient access authorization. Instead, the fault entry *fault-id* was written to the default history file *history-file1*.

System action: Processing continues normally.

User response: None.

IDI0175I

Fault Analyzer DUMP analysis will use *dump-dsn* for *history-file*(*fault-id*)

Explanation: This message is issued by the IDIS subsystem when the dump data set identified by *dump-dsn* has been determined to belong to the fault entry identified by *history-file*(*fault-id*). The fault entry is updated to add the associated dump data set name.

System action: Processing continues normally.

User response: None.

IDI0177E

Java DTFJ processing failed for *history-file*(*fault-id*)

Explanation: Expected Java™ information was not found during reanalysis of fault entry *fault-id* in history file *history-file*. This message suggests that Java DTFJ processing failed to complete successfully.

System action: Processing continues normally, but without Java information.

User response: Determine the reason why DTFJ processing failed. Ensure the MVS™ post-dump exit IDIXTSEL is installed (for details, see Installing the MVS post-dump exit IDIXTSEL on page 412) and the IDIS subsystem is started (for details, see Using the Fault Analyzer IDIS subsystem on page 318).

IDI0178E

Fault Analyzer startup has hung. RFR dump initiated to capture the abend.

Explanation: A hang was detected while attempting to perform real-time analysis.

System action: The real-time analysis is terminated and a recovery fault recording fault entry is written instead.

User response: If the problem persists, contact IBM Support.

IDI0179W

Process error trying to force *side-file-type* side-file use for *program-name*. The side-file will not be used.

Explanation: The "Listing/Side File Mismatch" prompt, "ENTER" key action could not successfully process the side file.

System action: Processing continues without compiler listing or side file support for the program.

User response: If required, check the compile date and the reasons listed for the mismatch, and locate a compiler listing or side file that is a better match for the program.

IDI0180I

Fault Analyzer processing skipped due to CICSDUMPTABLEEXCLUDE(CheckMaxCurr). Abend code *abend-code*.

Explanation: A CICS® transaction dump table entry exists for *abend-code* and the current dump count exceeds the maximum setting. Due to the CICSDUMPTABLEEXCLUDE(CheckMaxCurr) option being specified, Fault Analyzer analysis is skipped.

System action: Processing terminates.

User response: None

IDI0181W

IDIS now stopping and restarting tasks to conserve storage. Increase REGION size and restart.

Explanation: Fault Analyzer has detected a Short on Storage condition within the subsystem. The subsystem is stopping and restarting the IDIS DB2® subtasks when necessary to try to alleviate the condition.

System action: Processing continues.

User response: If this message is regularly issued by the IDIS subsystem, then it is recommended to increase the region size to avoid termination of the IDIS subsystem due to insufficient storage being available.

IDI0182I

Fault Analyzer subsystem error from IDISAREQ: *error_message*

Explanation: A message was passed back from the Fault Analyzer IDIS subsystem due to an unexpected condition. The passed-back message in *error_message* usually includes a separate message ID.

System action: Processing continues.

User response: Refer to the message ID in *error_message* for the problem that occurred in the Fault Analyzer IDIS subsystem call.

IDI0183W

No trace records found in *data_set_name*

Explanation: Mapping of an auxiliary CICS trace data set in *data_set_name* was requested.

System action: The trace mapping is terminated.

User response: Ensure the data set specified is correct.

IDI0184W

Unable to allocate extended minidump data set *data_set_name*: *reason*

Explanation: An attempt to allocate the extended minidump data set named in *data_set_name* failed for the reason shown in *reason*.

System action: Processing continues, but no extended minidump data set will be associated with the fault entry. Below-the-bar XDUMP storage pages will be converted to minidump pages, unless the total number of minidump pages after adding the XDUMP pages exceeds the MaxMinidumpPages option in effect. If this is the case, all storage pages that were destined for the XDUMP data set will be discarded.

User response: Address the reason shown in the message.

IDI0187I

Associated dump data set *data_set_name* not deleted due to insufficient access authorization; dump data set type: *dump-type*, history file *hist-file*, fault ID: *fault-id*, created by jobname: *jobname*, user ID: *user-id*, security server default group ID *group-id*

Explanation: A fault entry was deleted by automatic history file space management during real-time analysis, fault entry creation following interactive MVS dump analysis, or IDIUTIL batch utility IMPORT processing. An attempt was made to also delete the associated data set. However, due to insufficient access authorization, the associated dump data set identified by *data-set-name* could not be deleted.

System action: Processing continues.

User response: Manually delete the identified dump data set. From the information that is provided in this message, determine if security server access authorization to dump data sets of this type should be changed to allow for their automatic deletion along with their associated fault entries.

Additional information, depending on *dump-type*:

- RFR TDUMP and RFR SVC dump: see Managing recovery fault recording data set access on page 309
- SVC dump copy: see Managing copied SDUMP data set access on page 314
- XDUMP: see Managing XDUMP data set access on page 313

IDI0188I

SNAPDATA parameter ignored: *text*

Explanation: An error in the IDISNAP SNAPDATA parameter was detected. *text* contains additional information about the error.

System action: Processing continues.

User response: Refer to *text* for a possible explanation and then correct the problem.

IDI0189W

XDUMP data set *data_set_name* cannot be analyzed separately from its associated fault entry - perform reanalysis of history file *history_file_name* fault ID *fault_id* instead

Explanation: *data_set_name* was selected for analysis as a dump data set, for example from the Fault Entry List display using the File menu option 5. Because the data set is an XDUMP data set, it cannot be analyzed in this manner.

System action: The dump analysis is terminated.

User response: Perform reanalysis of the history file and fault entry in *history_file_name* and *fault_id* instead.

IDI0191W

History file *history-file* has been created without a secondary space allocation

Explanation: The IDIS subsystem detected that the *history-file* data set is allocated without secondary space.

System action: Processing continues.

User response: Fault Analyzer space management is designed to work effectively with secondary space allocations. See Allocating secondary space for history files on page 344, and consider reallocating the history file, as described in Changing the size of a PDSE history file on page 347.

IDI0192I

Dump data set *data_set_name* not deleted *timestamp*: *reason*

Explanation: Fault Analyzer could not delete the *data_set_name* dump data set associated with the deleted fault entry. The *timestamp* indicates the time when the associated fault entry was deleted. At this time, the *data_set_name* dump data set was migrated, and an asynchronous process attempted to delete it. The *reason* indicates the reason deletion failed.

This message is issued once for each dump data set that could not be deleted. The time when the message is issued depends on the *reason*, and might occur at one of the following points:

- At the time when the associated fault entry is deleted.
- When the IDIS subsystem checks for undeleted dump data sets at regular intervals.
- When stopping the IDIS subsystem.

System action: Processing continues.

User response: Delete the dump data set, either manually or by automated operations. Failure to do so leaves the unused data set orphaned. Refer to *reason* for the reason why the deletion failed, or check the SYSLOG for other messages that might have been issued around the time shown in *timestamp*.

IDI0193E

Lock flag control exit load module IDIXLOCK error: *reason*

Explanation: With the lock flag control exit IDIXLOCK installed, the attempted change of a fault entry lock flag failed due to *reason*. The possible values for *reason* are:

Execution abend *abend_code*
Found in non-authorized load library
LOAD abend *abend_code-reason_code*

LOAD error R15=*load_reason_code*
CSVQUERY error R15=*csvquery_reason_code*

System action: Processing continues, but the lock flag remains unchanged.

User response: Contact your systems programmer.

IDI0194W

spilled_pages minidump pages spilled to XDUMP data set *xdump_dsn* due to MaxMinidumpPages(*mmp_pages*) exceeded

Explanation: The MaxMinidumpPages option value in effect (*mmp_pages*) was exceeded. Since XDUMP data sets are enabled, the number of minidump pages that exceeded the MaxMinidumpPages option value (*spilled_pages*) were "spilled" to the XDUMP data set (*xdump_dsn*). While the XDUMP data set exists, this is handled transparently during reanalysis of the fault entry. However, if the XDUMP data set is no longer available (it might have been deleted, renamed, or just not available on the system on which the reanalysis is performed), reanalysis report information might be missing or incorrect. In this case, a warning will be displayed by the Fault Analyzer ISPF interface when using the 'B' or 'I' line commands, or when using the 'V' or 'S' line commands against a fault entry without a saved report. Similar warnings will be inserted into the batch and interactive reanalysis reports. When performing batch reanalysis of a fault entry with a missing XDUMP, message IDI0195W on page 739 will also be issued. To prevent this type of situation:

1. Never delete or rename a tightly coupled associated dump data set, such as an XDUMP. These will automatically be deleted when the fault entry is deleted using Fault Analyzer.
2. Ensure the MaxMinidumpPages option value is sufficient to prevent the need to spill minidump pages to the XDUMP data set.

System action: Processing continues.

User response: Contact your systems programmer.

IDI0195W

Fault entry *fault_id* XDUMP data set *xdump_dsn* containing spilled minidump pages no longer exists - reanalysis report information might be missing or incorrect

Explanation: This message is issued during batch reanalysis of fault entry (*fault_id*), for which message IDI0194W was issued when it was created (see IDI0194W on page 739 for the conditions under which this occurred). Since the associated XDUMP data set (*xdump_dsn*) is no longer available (it might have been deleted, renamed, or just not available on the system on which the reanalysis is performed), reanalysis report information might be missing or incorrect. Refer to IDI0194W on page 739 for additional information.

System action: Processing continues but reanalysis report information might be missing or incorrect.

User response: Contact your systems programmer.

IDI0196W

Unable to perform reanalysis of fault entry *fault_id* in history file *history_file* due to no READ access.

Explanation: This message is issued if batch reanalysis of fault entry, to which the user does not have READ access, is attempted.

System action: Processing is terminated.

User response: Contact your security administrator if access is required.

IDI0197I

Module *module_name* not found during binder processing.

Explanation: The MVS™ binder was unable to locate the specified module during analysis. This occurs during the initial processing of a Recovery Fault Recording (RFR) fault entry due to environmental limitations, such as insufficient storage or module relocation at the time of the fault.

System action: Processing continues.

User response: This is an informational message. If this situation occurs frequently for RFR entries, consider increasing the region size to prevent RFR conditions. To suppress this message, use the Quiet option, see Quiet on page 629 for details on how to use this option.

IDI0198W

MAS member name retrieval error for system *sysname*. RC: *rc* reason of failure: <*reason of failure*>. Batch Routing Disabled.

Explanation: During job routing Fault Analyzer attempted to resolve the MVS™ system name to its corresponding JES2 Multi-Access Spool (MAS) member name. This message indicates that the attempt to obtain the MAS member name based on the fault entry CVTSNAME failed. The *rc* is the return code from the SSI request or the internal SVC call, and <*reason of failure*> provides an internal state or marker (such as `NOT FOUND` or `SVC ERROR`) detailing the point of failure.

System action: Processing continues without routing.

User response: Contact your systems programmer to verify the JES2 configuration and ensure that the Subsystem Interface is functioning correctly. If the problem persists when using a fault entry whose CVTSNAME exists in the current SYSPLEX, retain the message details (RC and reason) and contact IBM Support.

IDI0199S

Static invocation of IEAVTABX exit IDIXDCAP not supported - use dynamic exit invocation instead.

Explanation: This message is issued when Fault Analyzer detects that the MVS™ change options/suppress dump IEAVTABX exit IDIXDCAP was invoked as an IEAVTABX CSECT exit. This method of installation is no longer supported. IDIXDCAP must be installed as an IEAVTABX_EXIT dynamic exit to function correctly.

System action: Processing terminates.

User response: Remove the unsupported IEAVTABX CSECT exit installation of IDIXDCAP. Ensure that IDIXDCAP is properly defined as an IEAVTABX_EXIT dynamic exit. For more information see Installing the MVS change options/suppress dump exit IDIXDCAP on page 333.

Appendix A. 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 preselect the Fault History File or View name being used, or provide a dynamically created MATCH command. A MATCH command can be useful, for example to MATCH on today's date, or a specific PROGRAM name.

An example of how this dynamic tailoring can be achieved is included in the samples data set (IDI.SIDISAM1). The example displays a popup panel which allows the user to supply an optional program name and an Application ID or 'View' name (see Figure 349: Sample display 1 on page 742). A '?' can be placed in the Application/View field to display a list of available applications and views (see Figure 350: Sample display 2 on page 742). In the sample, if the length of the Application/View ID is 2, 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, the ID is assumed to be the name of a Fault Analyzer VIEW.

Once an Application/View ID has been successfully entered, and its existence verified, then the user can press Enter to invoke Fault Analyzer. If a program name was also supplied, then a corresponding MATCH command is also created.

The sample comprises the following files. Each member should be copied to a data set that is concatenated to the DDname indicated in the table.

Table 29. Sample members in the IDI.SIDISAM1 data set

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	ISPPLIB	ISPF panel used for application selection
IDISFEQP	ISPPLIB	Query ISPF panel
IDISFEMP	ISPPLIB	Main ISPF panel for supplying user parameters

Figure 349. Sample display 1

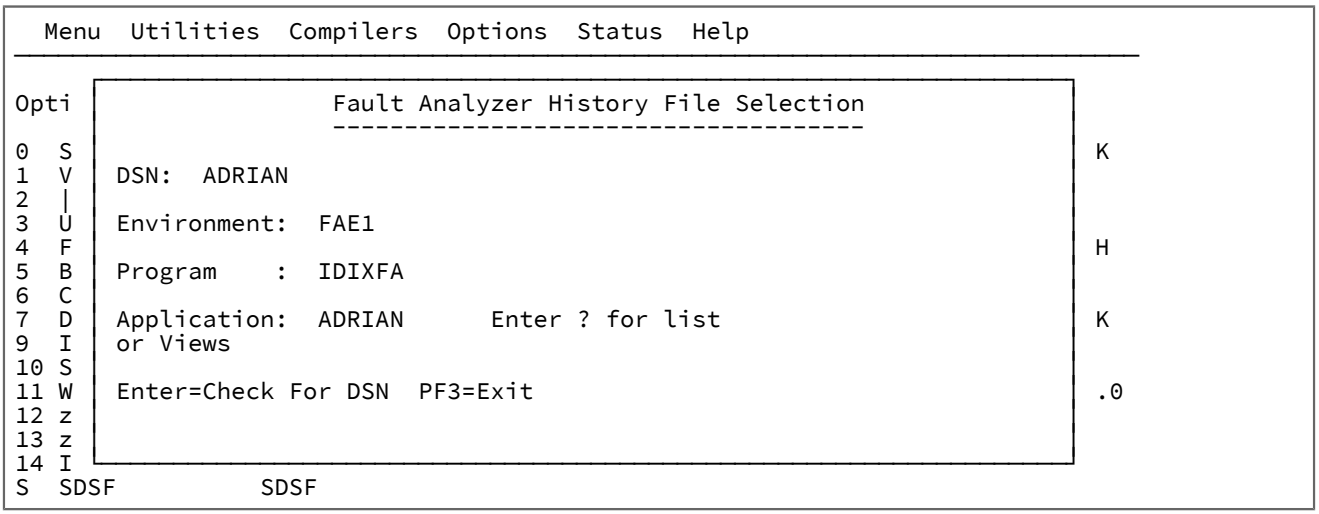
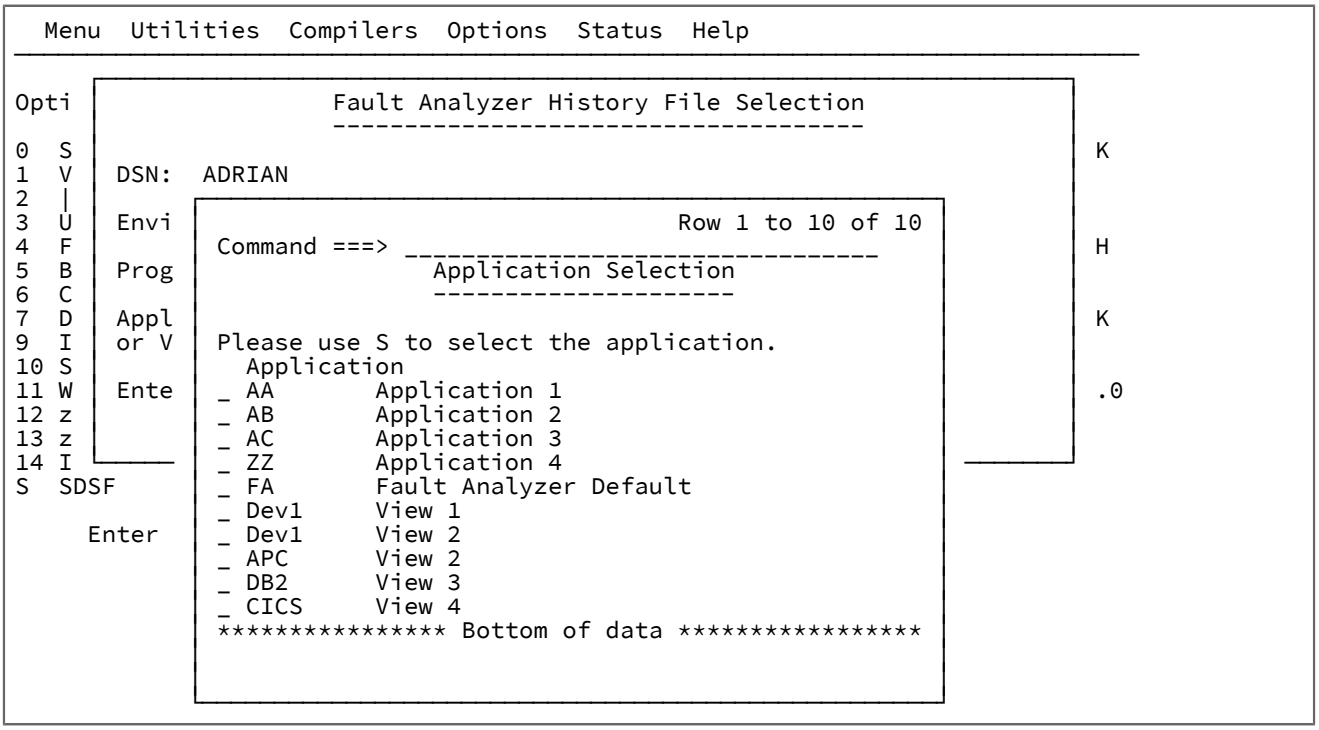


Figure 350. Sample display 2



Once a Fault History File or View name has been selected and verified, the sample code performs the following processing.

Fault History File or VIEW name

In the Fault Analyzer ISPF application (IDI) variable pool, there is a variable called OLDHIST, which is a list of the last 10 accessed Fault History Files or Views. The first item in this list is the History File or View which is opened by the Fault Analyzer ISPF interface. The sample code modifies this list, such that the first item in the list is the History File or View entered by the user. It does this modification by scanning the list to see if the name already exists, in which case the entry is

moved to the top of the list. If the name does not exist, then it is inserted at the top of the list, and all other entries are moved down one position, that is item 10 (if it exists) disappears.

In the list, View names are distinguished from Fault History Files by being enclosed within parentheses.

MATCH on program name

If a program name has been entered, the following command string is created:

```
MATCH PROGRAM <supplied program name>
```

If a program name has not been supplied, a MATCH ALL command is created.

This command string is picked up when the Fault Analyzer ISPF interface starts, and is executed accordingly.

Installing the sample application

It is important that the sample application (IDISFEMA) executes using the same ISPF application ID as the main Fault Analyzer ISPF application. If it does not, then updates made by the sample to ISPF variables are not correctly picked up.
A way to use the sample is to add a new ISPF command, which invokes the main sample program using the same ISPF NEWAPPL application as is used for the 'normal' invocation of Fault Analyzer. For example, add the following command to a 'USER' command table, which is used by Fault Analyzer ISPF users:

```
Verb      T  Action
FASEL     0  SELECT CMD(%IDISFEMA &ZPARM) NEWAPPL(IDI)
```

This command assumes that Fault Analyzer is normally invoked using a NEWAPPL of IDI, for example from an application selection panel using a command similar to the following:

```
9,PGM(IDIPDDIR) NEWAPPL(IDI) SCRNAME(FAULTA)
```

As well as the above, the CLIST IDISFECL must be in one of the data sets that is allocated to the user's SYSPROC concatenation.

How the sample works

So that the MATCH command string can be passed and executed by the Fault Analyzer ISPF interface, an intermediary CLIST is used (IDISFECL). This CLIST is invoked by passing an invocation command in the COMMAND option of an ISPF DISPLAY PANEL command. When the COMMAND option is specified on a DISPLAY PANEL command, the actual PANEL referenced is not displayed, and the command in the COMMAND options is executed. For example, if a program name of MYPROG1 had been entered, the following command string is created:

```
TSO IDISFECL;;MATCH PROGRAM MYPROG1
```

This command string, which is assigned to variable CMDSTACK in the sample code, is referenced in the ISPF DISPLAY command as follows:

```
Address ISPEXEC 'DISPLAY PANEL(IDISFEAP) COMMAND(CMDSTACK)'
```

Appendix B. Java API to download Fault Analyzer report

The Java™ API downloads a Fault Analyzer report from the host and shows it in Eclipse-based products that are developed in Java™.

The download part of the process happens only if the Fault Analyzer report is not already downloaded. This API belongs to the FAAPI package, which is available as part of the Fault Analyzer plug-in for Eclipse products. The assumption is that the specified host ID and port number are defined in the Systems Information view.

Specification

Two interfaces are designed to achieve this function:

```
FAAPI.openReport(String UniqueAdressOfFaultEntry);
        //UniqueAdressOfFaultEntry :"hostId/portNo/historyFile/faultId"

FAAPI.openReport(String hostId, int portNo, String historyFile, String faultId);
```

Both interfaces perform the same function, only the style of argument-specification differs. All arguments are case-sensitive.

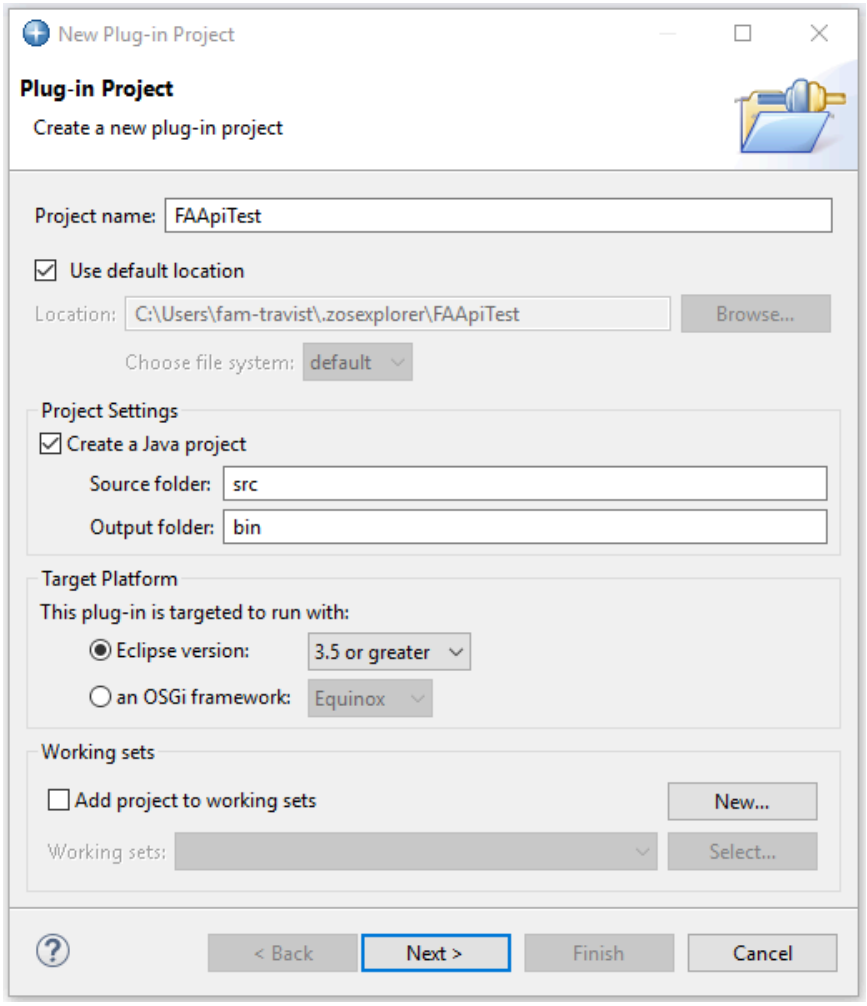
Here are examples using both interfaces of how to open fault entry F03004 in history file DA.DCAT on system pthfae1, which listens on port 7799:

```
FAAPI.openReport("pthfae1/7799/DA.DCAT/F03004");

FAAPI.openReport("pthfae1", 7799, "DA.DCAT", "F03004");
```

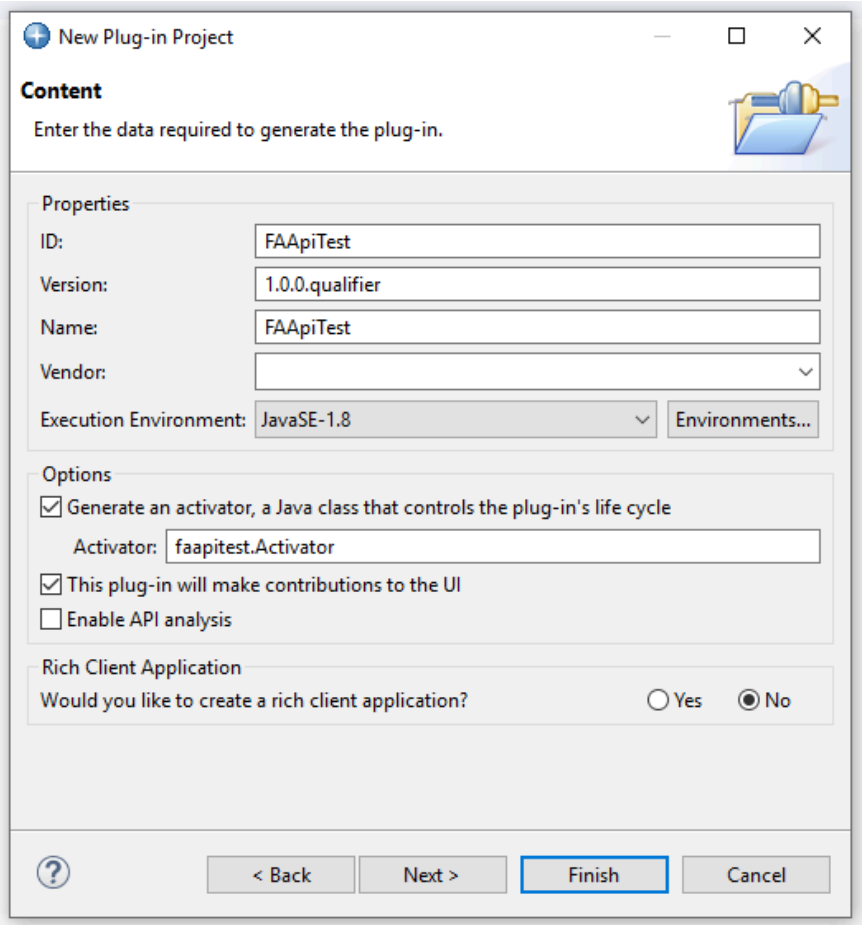
Example

This example shows how to extend the Hello, World Command sample plug-in in Eclipse to open a Fault Analyzer report. Follow these steps to create the Hello, World Command sample plug-in, and then modify it to download and open a Fault Analyzer report.

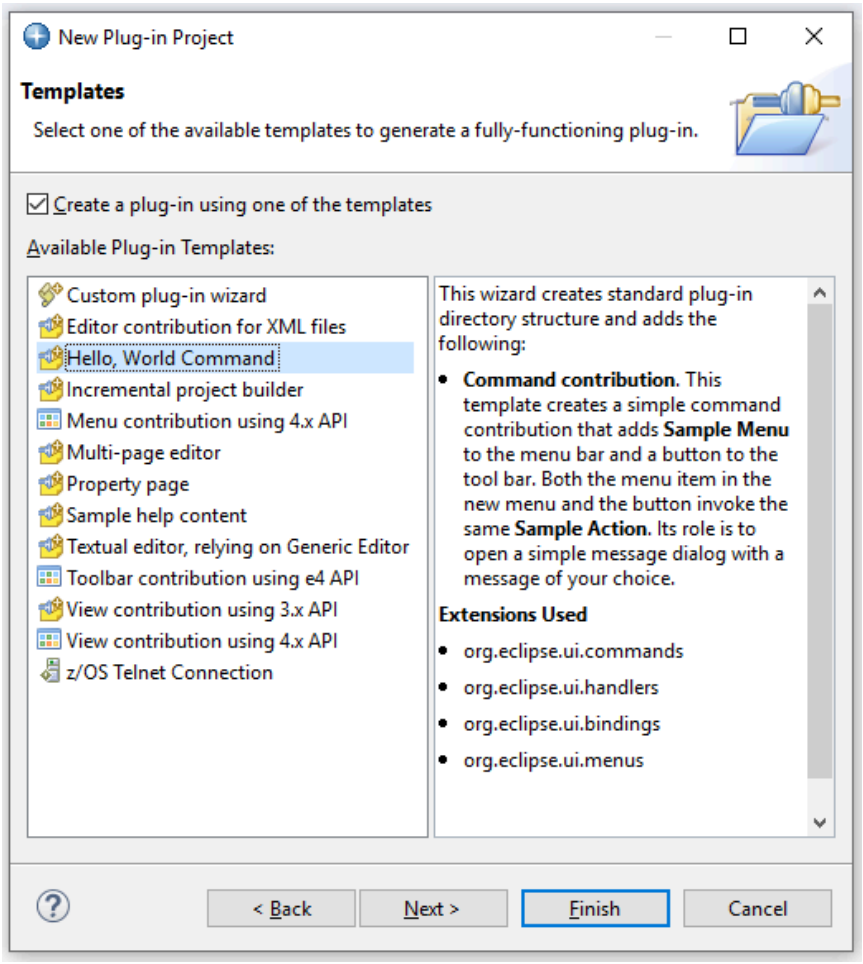
1. In IBM® Explorer for z/OS®, activate the Java™ perspective.
2. Activate the plug-ins for the ADFz family of products.
3. Click **New** -> **Plug-in Project**.



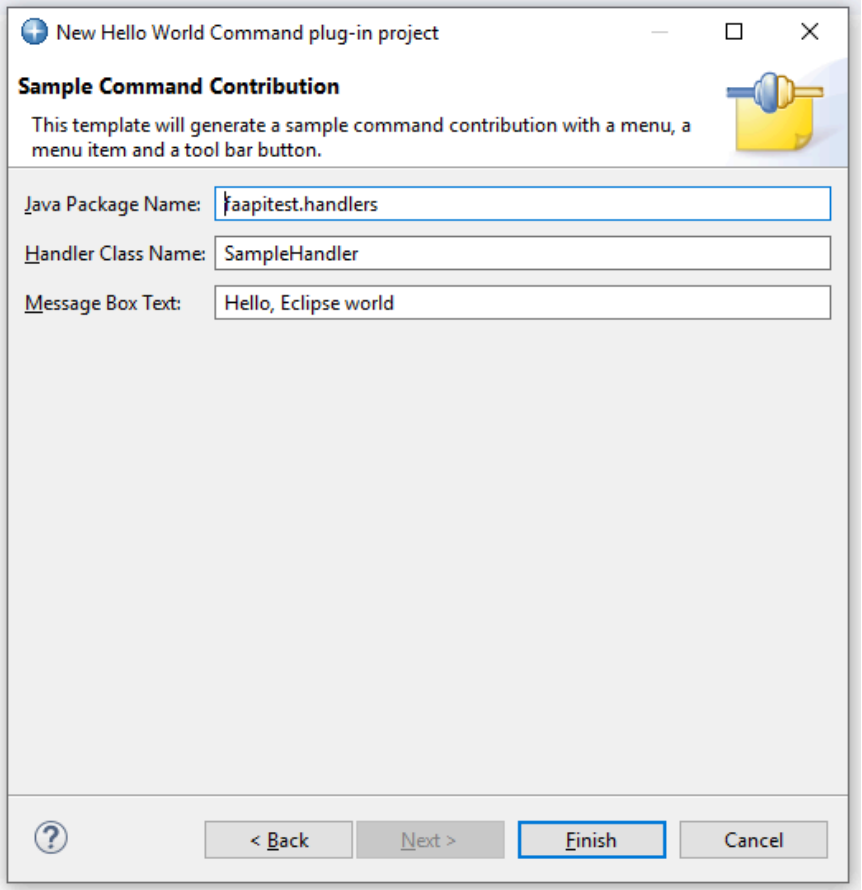
4. Enter a Project name (in this sample FAAPITest) and then click **Next**.


New Plug-in Project
Content
Enter the data required to generate the plug-in.
Properties
ID:
FAApiTest
Version:
1.0.0.qualifier
Name:
FAApiTest
Vendor:
Execution Environment:
JavaSE-1.8
Environments...
Options
Generate an activator, a Java class that controls the plug-in's life cycle
Activator:
faapitest.Activator
This plug-in will make contributions to the UI
Enable API analysis
Rich Client Application
Would you like to create a rich client application?
Yes
No
< Back
Next >
Finish
Cancel

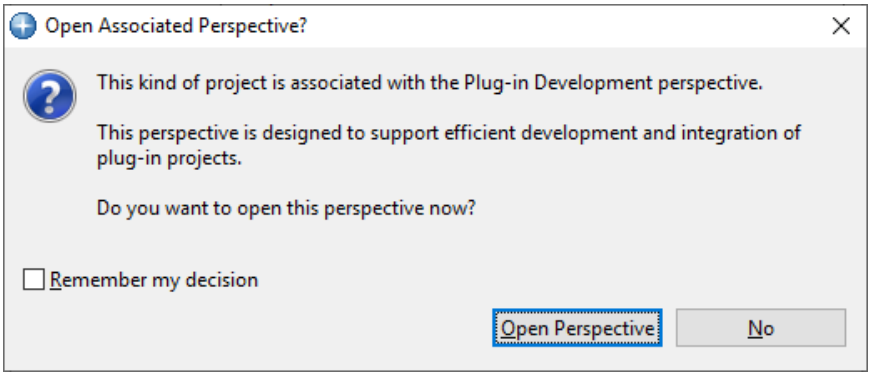
5. Click **Next**.



6. Select the **Hello, World Command** template and click **Next**.

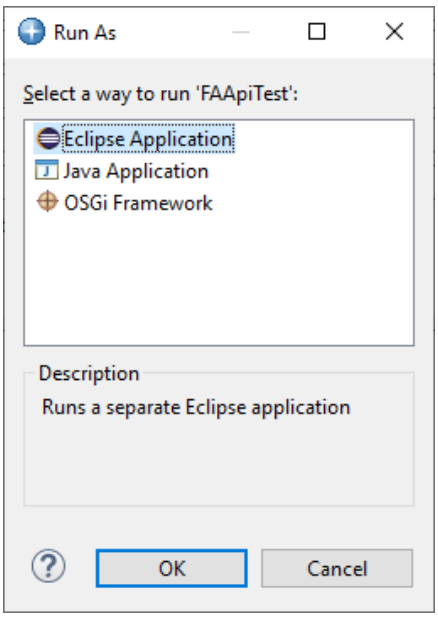


7. Click **Finish**. If you receive the following message, click **Yes**.

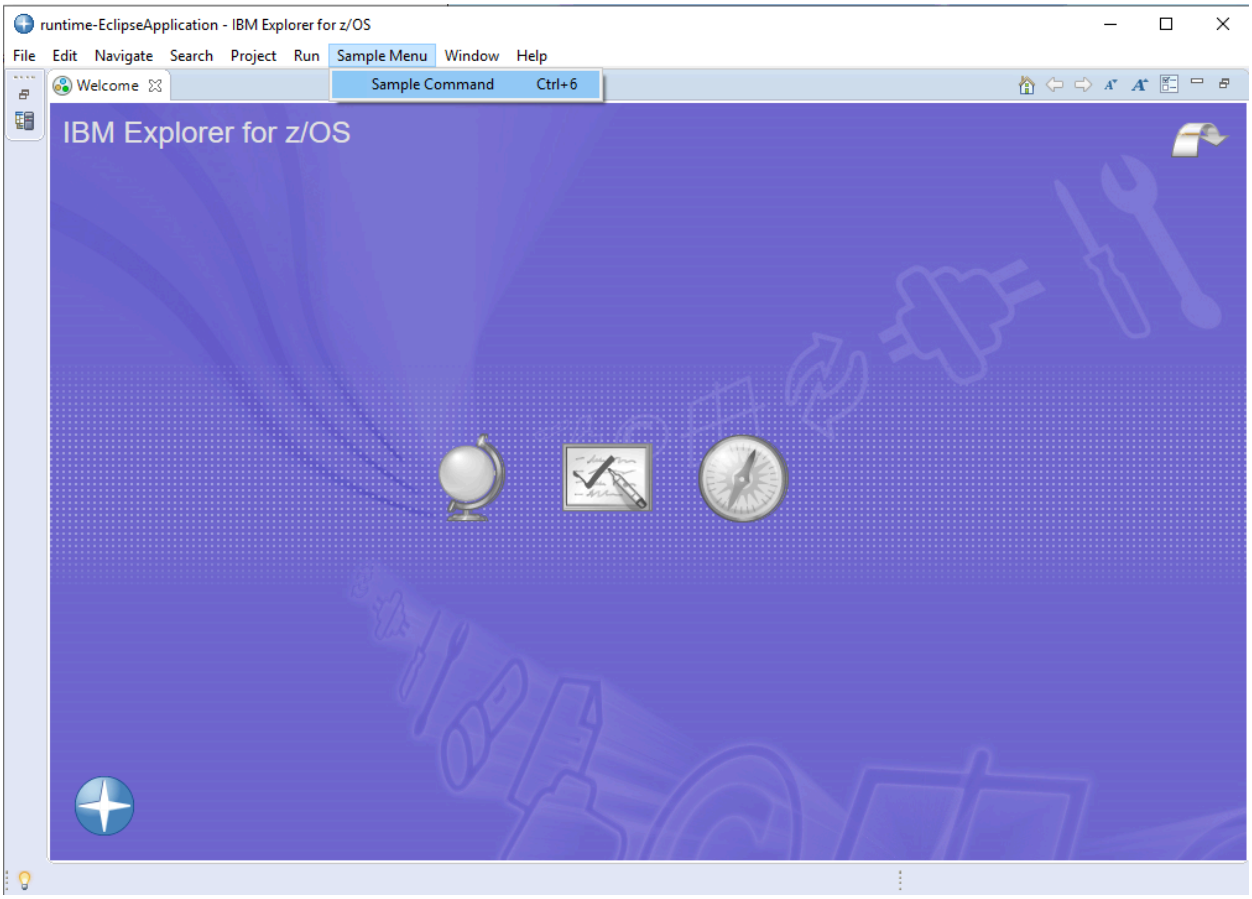


At this stage, FAAPITest is created. To run the generated plug-in at this stage, follow these steps:

1. Click **FAAPITest** in Package Explorer.
2. Click **Run** from the main menu and choose Run.
3. Select **Eclipse Application** and click **OK**.



4. The running environment displays. the following window. Click **Sample Menu > Sample Command**.



5. The following message appears on the screen:

6. Click **OK** and close the running Eclipse environment.

Now change the FAAPITest plug-in to show a fault entry when you click **Sample Menu** -> **Sample Command**:

1. Open MANIFEST.MF. This file exists in the META-INF folder of the FAAPITest plug-in.
2. Click the **Dependencies** tab. Then click **Add...** for required plug-ins and select **com.ibm.etools.fa.pdtclient.ui**. Click **OK** and save the MANIFEST file.
3. Open **faapitest.handlers.SampleHandler.java** and modify the run method as shown in the following example:

```
package faapitest.handlers;
```

```
import org.eclipse.core.commands.AbstractHandler;
import org.eclipse.core.commands.ExecutionEvent;
import org.eclipse.core.commands.ExecutionException;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.handlers.HandlerUtil;
import org.eclipse.jface.dialogs.MessageDialog;
import com.ibm.etools.fa.pdtclient.ui.util.FAAPI;

public class SampleHandler extends AbstractHandler {

 @Override
 public Object execute(ExecutionEvent event) throws ExecutionException {
  IWorkbenchWindow window = HandlerUtil.getActiveWorkbenchWindowChecked(event);
  MessageDialog.openInformation(
    window.getShell(),
    "FAApiTest",
    "Hello, Eclipse world");
  return null;
 }
}
```

4. Run the modified plug-in as explained earlier.
5. After Eclispe has initialized, make sure that the system is defined in the Systems Information view. If not, add the system.
6. Click **Sample Menu** -> **Sample Command**. The report is downloaded and displayed in the current perspective in the Eclipse editor area.

Appendix C. Technical details for screen size adjustments

To support the widest range of terminal characteristics, use a DLOGMOD specification of D4C32XX3, in the IBM® supplied MODETAB of ISTINCLM. Further information about this and Telnet server requirements can be found in z/OS® Communications Server publications.

Users of IBM® Personal Communications should reference IC71220: PCOM: HOW CAN YOU USE A 62X160 SCREEN SIZE? on the IBM® Support Portal.

Appendix D. Support resources

Use these resources to find product details, fixes, and support.

Search knowledge bases

- Download a Program Directory:
  - Program Directory for Application Delivery Foundation for z/OS Common Components
  - Program Directory for IBM File Manager for z/OS
  - Program Directory for IBM Fault Analyzer for z/OS
- Get up-to-date details about installing, customizing, and using these products:
  - Application Delivery Foundation for z/OS Common Components Customization Guide and User Guide
  - Fault Analyzer User's Guide and Reference
  - File Manager Customization Guide
  - File Manager User's Guide and Reference
  - File Manager User's Guide and Reference for DB2
  - File Manager User's Guide and Reference for CICS
  - File Manager User's Guide and Reference for IMS

Get the latest PTFs

- ADFz Common Components
- Fault Analyzer for z/OS
- File Manager for z/OS
- z/OS Debugger
- IBM Developer for z/OS Enterprise Edition
- Application Performance Analyzer for z/OS

Collect diagnostic data

Before you contact Support, be ready to answer these questions:

- What software versions are you running?
- Do you have logs, traces, and messages related to the problem?
- Can you re-create the problem? If so, how do you re-create the problem?
- Did you make hardware, operating system, or networking software changes?
- Do you have a workaround for the problem?

Contact Support

Open a case, chat with Support, or connect to resources and communities through https://www.ibm.com/mysupport.

Appendix E. Accessibility features for Fault Analyzer

Accessibility features assist users who have a disability, such as restricted mobility or limited vision, to use information technology content successfully.

Overview

Fault Analyzer includes the following major accessibility features:

- Keyboard-only operation
- Operations that use a screen reader

Fault Analyzer uses IBM® z/OS® SMP/E to install the product. You can read about the accessibility features of IBM® z/OS® SMP/E at https://www.ibm.com/docs/en/zos/2.5.0?topic=documentation-accessibility-features-zos.

Fault Analyzer uses IBM® Explorer for z/OS® to install the optional Fault Analyzer Eclipse plug-in. You can read about the accessibility features of IBM® Explorer for z/OS® at https://www.ibm.com/docs/en/explorer-for-zos/3.2?topic=documentation-accessibility.

Fault Analyzer uses the latest W3C Standard, WAI-ARIA 1.0 (www.w3.org/TR/wai-aria/), to ensure compliance with US Section 508 (www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-section-508-standards/section-508-standards) . To take advantage of accessibility features, use the latest release of your screen reader that is supported by Fault Analyzer.

PDF files have limited accessibility support. With PDF documentation, you can use optional font enlargement, high-contrast display settings, and can navigate by keyboard alone.

Keyboard navigation

For information about navigating ISPF panels by using TSO/E or ISPF, refer to https://www.ibm.com/docs/en/zos/2.5.0?topic=documentation-accessibility-features-zos. This topic describes how to navigate through the interface, including how to use keyboard shortcuts or function keys.

Interface information

The Fault Analyzer user interfaces do not have content that flashes 2 - 55 times per second.

The Fault Analyzer web user interface relies on cascading style sheets to render content properly and to provide a usable experience. The application provides an equivalent way for low-vision users to use system display settings, including high-contrast mode. You can control font size by using the device or web browser settings.

The Fault Analyzer web user interface includes WAI-ARIA navigational landmarks that you can use to quickly navigate to functional areas in the application.

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access the online product documentation with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that the screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element

The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element

The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In the example, if you include the FILE keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, the default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

*** indicates an optional syntax element that is repeatable**

The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3* , 3 HOST, 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write HOST STATE, but you cannot write HOST HOST.
3. The * symbol is equivalent to a loopback line in a railroad syntax diagram.

+ indicates a syntax element that must be included

The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loopback line in a railroad syntax diagram.

Related accessibility information

In addition to standard IBM® help desk and support websites, IBM® has a TTY telephone service for use by deaf or hard of hearing customers to access sales and support services:

TTY service 800-IBM®-3383 (800-426-3383) (within North America)

For more information about the commitment that IBM® has to accessibility, see IBM® Accessibility (www.ibm.com/able).

Notices

This information was developed for products and services offered in the U.S.A.

Licensed Materials - Property of HCL Technologies Ltd.

© Copyright IBM Corporation 2000, 2016. © Copyright HCL Technologies Ltd. 2017, 2026. U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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, MD-NC119
Armonk, NY 10504-1785
US

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.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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 jurisdictions 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 document at any time without notice.

Any references in this information to non-IBM® websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM® product and use of those websites is at your own risk.

IBM® may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

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 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.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

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.

Statements regarding Fault Analyzer for z/OS®'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. The sample programs are provided "AS IS", without warranty of any kind. IBM® shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year).
Portions of this code are derived from IBM Corp. and/or
HCL Technologies Ltd. sample programs.
© Copyright IBM Corp. 2000, 2016. © Copyright HCL Technologies Ltd. 2017, 2026.

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

IBM®, the IBM® logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM® or other companies. A current list of IBM® trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM® website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM®.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM®.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM® reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM®, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM® MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at http://www.ibm.com/privacy and IBM's Online Privacy Statement at http://www.ibm.com/privacy/details in the section entitled “Cookies, Web Beacons and Other Technologies,” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

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*.

A

ABDUMP

SYSUDUMP, SYSMDUMP, or SYSABEND

analysis

The methodical investigation of a problem, and the separation of the problem into smaller related units for further study.

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.

ADFz

IBM® Application Delivery Foundation for z/OS® family of products.

ADFzCC

IBM Application Delivery Foundation for z/OS Common Components, which provides common functions to ADFz products.

C

cache

A buffer storage that contains frequently accessed instructions and data; it is used to reduce access time.

compiler listing

A printout produced by compiling a program or creating a file and that optionally includes, for example, a line-by-line source listing, cross-reference list, diagnostic information, and for programs, a description of externally described files.

D

dump

To copy the contents of all or part of virtual storage to collect error information.

E

expert system

A system that provides for solving problems in a particular application area by drawing inferences from a knowledge base acquired by human expertise.

F

fault entry

The information about a fault saved as a member in a *history file* at the end of *real-time analysis*. The fault entry might also include a *minidump*.

H

history file

A *PDSE* data set containing *fault entries* as individual members.

L

LANGX side file

A compiler listing or SYSADATA file, converted to a binary format using the ADFz Common Components IPVLANGX utility, or one of its aliases CAZLANGX, EQALANGX, or IDILANGX.

listing

See *compiler listing*.

logical history file size

The number of 4K pages allocated to a PDSE history file at the time of becoming auto-managed or explicitly specified in the fault history file settings.

loosely coupled dump data set

A dump data set that is not uniquely linked with a particular fault entry. The fault entry was created as a result of analysis of an already existing dump data set.

M

minidump

The storage pages that were referenced during real-time analysis by Fault Analyzer and saved in the fault entry. A minidump permits later reanalysis of the fault even if no associated MVS dump data set 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 or PDSE

A data set that can be either a *partitioned data set (PDS)* or a *partitioned data set extended (PDSE)*.

R

real-time analysis

The *analysis* undertaken by Fault Analyzer immediately after a program has *abended*.

reanalysis

A second or subsequent *analysis*.

Recovery fault recording (RFR)

A feature of Fault Analyzer which enables abnormal termination of a Fault Analyzer real-time analysis to still create a fault entry of the original application abend.

RFR dump

A SDUMP or IEATDUMP data set written during recovery fault recording processing, and associated with an RFR fault entry.

RFR fault entry

A history file fault entry written during recovery fault recording processing, and associated with an RFR dump data set.

S

saved report

The report that is contained within the fault entry and which can be viewed from the ISPF interface without the need to perform reanalysis.

side file

A condensed version of a *listing*, readable by computers (but not humans).

spilled minidump pages

Storage pages that would normally be included in the minidump, but were written to the XDUMP instead due to MaxMinidumpPages exceeded.

system dump

See *dump*.

T

TDUMP

Synonymous with IEATDUMP. A dump data set type used by Fault Analyzer recovery fault recording processing.

tightly coupled dump data set

A dump data set that is uniquely linked with a particular fault entry. It was originally created by Fault Analyzer during real-time processing.

V

view

A member of a *PDS or PDSE* containing a collection of fault history file data set names that are to be displayed simultaneously using the Fault Analyzer ISPF interface.

X

XDUMP

Extended minidump data set.

Index

Special Characters

_IDI_OFF environment variable 463
_IDI_OPTS environment variable options 576
_IDI_OPTSFILE environment variable options 575
-DROPCNF-
  data set name 585
  user exit name 602
-HistCols keyword 348
-Match 349
&SYSCLONE MVS system symbol
  IDICNFxx parmlib member suffix 369
++IDIOPT2 342
$$BACKUP history file data set member 26
$$INDEX data
  caching 319
$$INDEX history file data set member 25
$$UFMTX example 234

Numerics

3270 screen buffer
  full-color display 98
  plain-text display 178
64-bit addresses
  specifying 142

A

ABCODE keyword on EXEC CICS ABEND command 400
abend code explanations
  displaying 123
  repository 308
  user-defined 456
abend codes
  expanding 206
abend data, saving 515
Abend Intelligence
  configuring 567
abend job information display example 200
abend S913 339
abend trapping 54
access requirements
  compiler listings 384
  side file 384
accessibility features 755
ADATA option
  producing SYSADATA file 374
AdditionalIDIOffDD option 577
ADDR tag 542
address space
  selecting for analysis 249
address TSO REXX commands 467
ADFzCC
  IPVCNF00 parmlib member 571
  IPVOPTLM configuration-options module 571
ADFzCC server
  customizing for plugin 561
AI
  configuring 567
AI analysis report 281
allocate ISPF data sets 327
analysis
  real time *see* real-time analysis
Analysis Control user exit 475
Analysis Control user exit (MVS SVC dump registration) 479
analysis engine 19
analysis process 19
analysis report, real-time 35
analyze MVS dump data set display example 247
application error handling, effect on Fault Analyzer invocation 304
application-specific descriptions, providing 455
AREA tag 542
Artificial Intelligence
  configuring 567
assembler ADATA option 374
assembler exits
  CICS NoDup(CICSFAST) override 408
assembler invocation
  IDISNAP 46
assembler programs
  verifying Fault Analyzer 428
associated dump data sets 25
associated file control blocks display example 175
associated storage areas display
  example 202
  with hex value column collapsed example 203
  with level 88 items collapsed example 204
AUTO-managed PDSE history files 88, 345
available dump status 134

B

batch options line 150
batch reanalysis 23
  data sets 154
  initiating 154
  JCL control statements 331
  options 149
  purpose 149
  return codes 693
  submitting job 81
  viewing saved report 120
batch reanalysis options display example 149, 152
batch TSO receive, sample EXEC 350
batch utility
  IDIUTIL 441
BatchOpts option 116, 603
binder-related dependencies 29
BindExclude option 577
blank lines
  adding 122
  removing 122
BPX014I message 307

C

C programs
  CSECT naming 379
  Java sample 270
  verifying Fault Analyzer 431
CA-Panexec
  exit for 335
call depth
  maximum 278
CallEqaueDat option 578
CE command 98
CEEWUCHA
  special processing 30
CEEWUCHA LE user condition handler 272
CFA 400
  FORCEPURGE currently analyzed task 399
CFA CICS transaction 397
CFA Exit Options display example 402
CFA IVP testing display example 432
CFA transaction 399
CFA transaction display example 401
change fault history file settings display example 89
checklist
  customizing Fault Analyzer 291
  installing Fault Analyzer 291
CICS
  duplicate fault criteria 622
  IDIOPTS DDname 411
  IVP testing 432
  performing interactive reanalysis 284
  protection key support 26
  required programs 397
  sample CICS program definition job 397
  startup PLT
    adding programs 398
    installing Fault Analyzer 398
  supported versions 26
  task storage areas 99
  transaction abend analysis 400, 408
  verifying customization 432
  XPCABND global user exit 400, 408
CICS 3270 screen buffer
  full-color display 98
  plain-text display 178
CICS auxiliary trace data set 257
CICS channels 186
CICS commareas 186
CICS containers 186
CICS dump data set
  CICS transaction LE dump output 410
CICS environment
  customizing 396
CICS fast duplicate fault suppression 495, 623
CICS global user exits 33
CICS information display example 176
CICS levels 186
CICS NoDup(CICSFAST) override assembler exit (IDINDFUE) 408
CICS open (L9) TCB
  capturing abends 411
CICS region SDSF job data set display example 402
CICS sample program definition job 399
CICS shutdown PLT *see* shutdown PLT
CICS Storage Accounting Area (SAA) overlays 177, 412
CICS system abend analysis
  creating history file entry 255
  selecting dump data set 247
  setting options 247
  user exit usage 247
CICS system abend dump analysis 247
CICS system abend interactive reanalysis report
  abend job information display 252
  CICS system information display 252
  display example 250
  fastpath commands 251
  options in effect display 254
  setting options 157
  synopsis display 251
CICS system initialization parameters

displaying during interactive reanalysis 188
displaying with SIT command 111
CICS task trace table 186
CICS trace
display example 183
formatting system-wide information 180
preventing wrap from LE 410
storage area size 410
CICS trace selection parameters display example 180
CICS trace, summarized
system-wide information 179
CICS transaction abend analysis
CFA 399
defining to CICS 397
enabling dynamic control 399
maximizing performance 412
CICS transaction abends
special handling 340, 340
CICS transaction LE dump output
directing to CICS dump data set 410
CICS transaction storage summary display example 177
CICS XDUREQ global user exit
stopping Fault Analyzer analysis 340, 340
CICSD command 98
CICSDumpTableExclude option 578
CICSSTG command 99
CICSTranAnalysisUser option 580
CMDS command 99
COBOL
base locators 205
Java sample 269
supported versions 26
COBOL Explorer 241
example 242
COBOL invocation
IDISNAP 42, 43
COBOL programs
verifying Fault Analyzer 429
COBOL Report Writer Precompiler 375
COBOL suppressed copybooks 274
COBOL SYSDEBUG file usage 377, 380
COLS command 99
column
hex-value *see* hex-value column
columns
-HistCols keyword 348
fault entry list display 67
source of configuration 66
specifying default layout 348
com.ibm.faultanalyzer.Snap.dump method
real-time analysis report 35
commands
CE 98
CICSD 98
CICSSTG 99
CMDS 99
COLS 99
COPY 100
CUROPTS 100
DISASM 100
DSECT 101
DUPS 101
EDIT 101
EXEC 101
FIND 102
HTTP 105
IDISINFO 105
INFO 106
ISPF interface 98
JCL 106
LOOKUP 106
MATCH 108
NEXT 108
NOTE 108
NOTELIST 109
PREV 109
QUIT 109
reading syntax diagrams xii
REFRESH 110
RESET 110
RPTFIND 110
RUNCHAIN 110
SHOW 111
SIT 111
STCK 112
VIEWS 112
XML 112
CompileDateCheck option 580
Compiler Listing display example 207, 207, 208
Compiler Listing Not Found display example 235
Compiler Listing Read user exit 480
compiler listings
access requirements 384
activating search trace 383
attributes 387
example of providing for reanalysis 156
locating 380
naming 378
prompting display 235
providing 374
selection criteria 30
temporary data set prohibition 387
compiler options required for IDILANGX 375
configuring
Abend Intelligence 567
IBM Fault Analyzer for z/OS Abend Intelligence 567
MCP server 564, 565
Confirm Exit display example 167
Confirm Fault Entry Deletion option 126
Confirm Fault Entry Deletion setting 113
confirm sysmdump open display example 159
COPY command 100
copyright
displaying 126
create fault entry display example 255
criteria
compiler listing selection 30
side file selection 30
successful analysis 669
cross-system coupling facility 363
Cross-system Coupling Facility (XCF) 319
CSECTs
naming 379
CSV042I message 307
CTL data area 640
cultural environment
specifying 373
CUROPTS command 100
cursor match 80
customization
verifying 428
customization checklist 291

D

data areas
converting from decimal 553
converting from hexadecimal 554
displaying 211
displaying chained 227
data set members *see* sample data set members
data set name substitution symbols 585
data set security server access requirement
IDIDOC 308
IDIVSENU 308
IDIVSJPN 309
data sets
batch reanalysis 154
compiler listing
temporary data set restriction 387
dropping from DDname 585
interactive reanalysis 240
ISPF *see* ISPF data sets
reanalysis 584
side file
temporary data set restriction 387
DATA tag 543
DataSets option 581
DB2
binding 415
IDIS subsystem requirements 323, 325
improving pre-V10 performance 415
LE considerations 415
supported versions 26
system-wide information display example 188
verifying Fault Analyzer 434
DB2 IVP
C 435
COBOL 436
DD tag 543
DDnames
IDIRLOAD 388
DEBUG option considerations 378
default options 369
Deferred Breakpoints Feature 209, 246
DeferredReport option 589
delete confirmation display 126
DELETE control statement 444
descriptor codes
WTO messages 31
Detail option 591
detailed PCB display 196
DFHRPL concatenation 399
DISASM command 100
display area
increasing 331
display information
setting preferred formatting width 122
displays
abend job information 200
analyze MVS dump data set 247
associated file control blocks 175
associated storage areas 202
associated storage areas with hex value column collapsed 203
associated storage areas with level 88 items collapsed 204
batch reanalysis options 149, 152
CFA IVP testing 432
CFA transaction 401, 402
change fault history file settings 89
CICS abend job information 252
CICS information 176
CICS interactive reanalysis options in effect 254
CICS levels, commareas, and channels 186
CICS region SDSF job data set 402

CICS system abend interactive reanalysis report 250
CICS system abend synopsis 251
CICS system information 252
CICS trace 182
CICS trace selection parameters 180
CICS Transaction Storage Summary 177
compiler listing 207, 208
Compiler Listing 207
Compiler Listing Not Found 235
Confirm Exit 167
confirm fault entry deletion 126
Confirm Java Fault Entry Reanalysis 260
confirm sysmdump open 159
create fault entry 255
Create Java Fault Entry 259
DB2 information 188
dump storage 209
event details 170
event summary 168
exclude program from side file search 239
excluded program names 161
Fault Analyzer options 201
Fault Analyzer Preferences 113
fault entry information 129
fault entry list 57, 57
fault entry list column configuration 65
fault entry list with updates pending 92
File Browse 64
file information 174
format CICS auxiliary trace data set 257
Formatting User Exit Selection List 233
hex-dumped storage 198
history file properties 84
history file updates pending 93
IMS information 191
IMS summary information 195
interactive reanalysis options 157
interactive reanalysis report 165
interactive reanalysis status 164
Java event details 264
Java event summary 262
Java information 264
Java interactive reanalysis report 262
last accessed history file entries 62
last accessed history files or views 61
last CICS 3270 screen buffer 178
last CICS 3270 screen buffer hex 179
LE heap analysis information 198, 199
level 88 items 205
listing/side file mismatch 237
Lookup Search and Browse 124
message explanation 206
message ID look-up 123
MTRACE records 199
new history file allocation 87
preferred formatting width 122
real-time report 120
specify compiler listing or side file 236
specify move/copy options 138
STCK conversion 231
storage disassemble 229
storage DSECT mapping entry 224
storage DSECT mapping map 224
storage RUNCHAIN command entry 227
summarized CICS trace 179
suppressed dump storage 212
synopsis 168
system-wide messages 188
system-wide open files 173
system-wide storage areas 197
user note list 215
user notes update prompt 216
view list 63
XMIT options 140
DL tag 543
DROPCNF
data set name 585
user exit name 602
DSECT command 101
DSECT data sets ($DINDEX member)
indexing 226
DSECT indexing utility (IDIPDSCU) 226
DSECT information
mapping storage areas 223
DSNACLI default DB2 plan 415
DT tag 544
DTJF processing 266
DUMMY data set
specifying via DataSets option 584
DUMP abending transaction option 408
dump data set
associated 25
Fault Analyzer effect on size 304
name, viewing 83
dump registration
Analysis Control user exit 479
Fault Analyzer IDIS subsystem usage 318
identification of fault entry 69
IDIXTSEL invocation exit 302
indication of invocation exit in ENV data area 659
Notification user exit 506
process description 47
specifying user exits for 593
verifying 439
dump status
available 134
not found 134
dump storage
locate with IDIXDLOC function 551
locate with IDIXXLOC function 558
dump storage display example 209
dump storage suppressed display example 212
dump suppression 32
controlling with user exit 494
DUMP tag 545
DUMPA tag 546
DumpDSN option 592
DumpRegistrationExits option 593
duplicate fault count 133
duplicate fault detection
overview 616
duplicate fault determination
controlling with user exit 494
criteria 624
duplicate fault processing
overview 50
duplicate history 82, 134
DUPS command 101
dynamic SVC update 306

E

Eclipse plug-in
ADFzCC server configuration 561
overview 283, 284
EDIT command 101
End Processing user exit 494
End Processing user exit (fault entry refresh) 497
Enterprise COBOL
Java sample 269
Enterprise PL/I
64-bit Java sample 269
Java sample 268
SYSDEBUG file usage 377, 380
ENV data area 653
environment variable _IDI_OFF 463
environment variables
STDENV DD statement 569
EPC data area 668
EQAUEDAT exit
locating SYSDEBUG files 381
error display example 57
error handling effect on Fault Analyzer invocation 304
ErrorHandler option 594
Evaluate REXX command 517
event information
obtaining through IDIXEINF function 552
event summary
details 170
display example 168
Exclude option 595
exclude processing
jobs analyzed with 371
exclude program from side file search display example 239
excluded history files 144
excluded program names display example 161
EXEC command 101
exits
CICS 302
for invoking Fault Analyzer 298
Language Environment 299, 302
MVS 298, 302
user *see* user exits
EXITS control statement 450
Exits option 600
EXPORT control statement 449
extended minidump (XDUMP) data set 53

F

FA standard date format 649
fast Exclude options processing 372
fastpath commands for CICS system dump analysis 251
fault
reanalyzing 55
fault analysis report
viewing 83
Fault Analyzer 415
authorizing 306
CPU time consumed 513, 516
customizing
for CICS 396
for ISPF 327
preparation 291
through IDIOPTLM configuration-options module 337
through user exits 464
through USERMODs 333
disabling 461
displaying copyright information 126
displaying license information 126
enabling implicit invocation from PL/I V2R3 applications 335
enabling invocation 333
invoking
from CICS 397
from Java try-catch block 418
Java dump events 421

PL/I PLIDUMP 335
SDSF 330
through Language Environment for CICS 397
legal notices dcclix
maintaining 459
migrating
from earlier version 287
from V10.1 to V11.1 290
from V11.1 to V12.1 289
from V12.1 to V13.1 288
from V13.1 to V14.1 288
from V14.1 to V15.1 287
from V15.1 to V16.1 287
from V16.1 to V17.1 287
from V9.1 to V10.1 290
registering in IFAPRDxx parmlib member 316
report 272
running with similar third-party products 304
stopping analysis 340, 340
turning off (IFAPRDxx parmlib member) 462
turning off (JCL switch IDIOFF) 462
turning off environment variable 463
uninstalling temporarily 461
verifying customization 428
Fault Analyzer DB2 performance improving 415
Fault Analyzer ISPF interface 55
Fault Analyzer modules
making available 306
Fault Analyzer options 201
Fault Analyzer plug-in for Eclipse
installing 283, 284, 561
Fault Analyzer preferences 113
Fault Analyzer system requirements 26
Fault Analyzerterminating 398
fault entries
collecting to analyze 515
deleting 126
expiration control 94
fault entry prefix 88
finding 77
ISPF line commands 81
locking
exits 127
flag 113
highlighting 113, 127
methods 127
to prevent deletion 94
matching 77
minimum/maximum 88
packaging 141
selecting 77
sending from one system to another 359
sorting 77
transmitting 140
fault entry duplicate history 134
fault entry information
refreshing 92
viewing 129
fault entry information display example 129
fault entry list column configuration display example 65
fault entry list display
customizing 373
fault entry list display example 57
fault entry list display with updates pending example 92
fault entry refresh
End Processing user exit 497
fault entry refresh processing 240
fault entry selection
specifying initial criteria 349
fault history entries
copying 138
cursor match 80
moving 140
moving range 140
wildcard 80
fault history files 24
selecting during real-time execution 33
fault identifier
format 610
fault reanalysis 22
batch 149
creating your own job 155
interactive 157
FaultID option 610
File Browse display example 64
file information display example 174
File Manager for z/OS
ISPF data set allocations required 327
FILES control statement 442
FIND command 102
differences between display types 104
FND area
clearing 402
Format CICS Auxiliary Trace Data Set display example 257
format tags
ADDR (address) 542
AREA (area) 542
DATA (data) 543
DD (definition description) 543
DL (definition list) 543
DT (definition term) 544
DUMP (EBCDIC dump) 545
DUMPA (ASCII dump) 546
HP (highlighted phrase) 547
L (line) 547
LI (list item) 547
NOTEL (note list) 548
P (paragraph) 548
TH (table heading) 549
U (underline) 549
UL (unordered list) 550
formatted text
write to report with IDIXWRIT function 557
Formatting user exit 490
samples 493
Formatting User Exit Selection List example 233
fragments, syntax diagrams xii

G

general report information 272
GenerateSavedReport option 611
global environment data area (ENV) 467
global resource serialization 363
GlobalExclude option 608
GRS 363

H

help text
adding 122
removing 122
hex-dumped storage display example 198
hex-value column
hiding 203
HistCols option 609
history entries
fault *see* fault history entries
history file entries
creating 255
moving 140
moving range 140
history file fault entries
creating 259
XFACILIT resource class 363
history file properties display example 84
history file selection
controlling with user exit 494
history file settings
restricting change 307
history file updates
controlling with user exit 494, 497
history file updates pending display example 93
history files
accessing through non-ISPF interfaces 283
allocating 343
auto-management 345
contents 24
converting between PDS and PDSE 345
copying entry 81
default name 347
deleting entry 82
deleting fault entries 126
determining size 343
excluded 144
faults *see* fault history files
initiating interactive reanalysis 82
logical size 345
managed 143
managing access 363
managing across MVS systems without shared DASD 350
managing with IDIUTIL 441
matching faults 77
moving entry 83
PDSE-managed 343
refreshing 92
renaming 343
resetting access information 91
selected 145
selecting for display 59
setting name 347, 584
setting up 343
sharing across sysplex 362
showing duplicate history information 82
size 347
submitting batch dump reanalysis 81
transmitting entry 83
viewing dump data set name 83
viewing saved fault analysis report 83
HLASM 46
HP tag 547
HTTP command 105

I

IBM Fault Analyzer for z/OS Abend Intelligence
configuring 567
ICH408I message 339
ICH420I message 307
ICH422I message 307
identification of column configuration source 66
IDI_SDUMP_ACCESS
XFACILIT resource class 310, 418
IDI.SIDISAM1 sample member 415
IDI* messages 695
IDIADATA

attributes 387
data sets 643, 643, 644
DD statement 35
specifying through DataSets option 581
IDIADATA DD statement 374
IDIALLOC REXX command 521
IDIBOPT DDname 574
IDICHKI utility 291, 460
IDICNFUM user-options module 573, 574
IDICNFxx
&SYSCLONE parmlib member suffix 369
alternative parmlib data set 339
dropping
data sets 585
user exits 602
options file 573
parmlib member 369, 573
IDIDATST 389
IDIDDTEST REXX command 525
IDIDOC
READ access requirement 309
specifying through DataSets option 581
IDIDOxxx
specifying through DataSets option 581
IDIDSECT
concatenation 226
specifying through DataSets option 581
IDIDSECTdsn REXX command 526
IDIDSNTEST REXX command 527
IDIEventInfo REXX command 528
IDIEXEC
specifying through DataSets option 581
IDIEXEC DDname 467, 593, 601, 630
IDIFREE REXX command 529
IDIGET
REXX command 530
IDIGSVRJ sample member 561
IDIHIST
data set 661
specifying through DataSets option 581
IDIHIST DD statement 347
IDIHIST_GROUP_DSN
XFACILIT resource class 363
IDIHIST_USERID_DSN
XFACILIT resource class 363
IDIHUSRM 455
IDIJAVA
specifying through DataSets option 581
IDIJAVA option 391
examples 392
IDIJVM DD statement 324
IDIJVM11 DD statement 324
IDIJVM6 DD statement
JVM
specifying default 324
IDILANGX
attributes 387
data sets 646, 646, 647
DD statement 35
return codes 693
specifying through DataSets option 581
IDILC
attributes 387
data sets 644, 644, 644
DD statement 35
specifying through DataSets option 581
IDILCOB
attributes 387
data sets 645, 645, 645
DD statement 35
specifying through DataSets option 581
IDILCOBO
attributes 387
data sets 645, 646, 646
DD statement 35
specifying through DataSets option 581
IDILPLI
attributes 387
data sets 647, 647, 647
DD statement 35
specifying through DataSets option 581
IDILPLIE
attributes 387
data sets 648, 648, 648
DD statement 35
specifying through DataSets option 581
IDIMAPS
specifying through DataSets option 581
IDIModQry REXX command 530
IDINDFUE CICS NoDup(CICSFAST) override assembler exit 408
IDIOFF DD statement 462
IDIOPTLM
configuration-options module 337
sample job 337
IDIOPTLM configuration-options module 339
IDIOPTS
DD statement 156
options file 411
IDIOPTS DDname 572, 573
IDIOPTS options file 575
IDIPANEX sample member 335
IDIPUT
REXX command 531
IDIRegisterFaultEntry REXX command 532
IDIREPRT DD statement 35
changing SYSOUT class 36
combining reports 36
DUMMY allocation 37
dynamic allocation 36
preventing dynamic deallocation 634
suppressing report 37
user exit allocation 36
IDIRFR_TDUMP_HLQ
XFACILIT resource class 311
IDIRLOAD DDname
CSECT mapping 388
IDIROBOT sample EXEC 350
IDIS subsystem 318
displaying status 142
requirements for DB2 323
requirements for deletion 325
requirements for Java 324
starting 320
stopping 325
storage requirements 322
IDIS$NDX sample member 26
IDISCICS sample member 399
IDISCLST EXEC 379
IDISCMDS command table 328
IDISCNFU sample member 574
IDISCPRO EXEC 379
IDISDB2B sample member 436
IDISF* messages 695
IDISFA sample member 329
IDISFE* sample members 741
IDISHIST sample member 343
IDISINFO command 105, 142
IDISISPF sample member 327
IDISJAV1 sample member 268
IDISJAV2 sample member 269
IDISJAV3 sample member 269
IDISJAV4 sample member 269
IDISJAV5 sample member 270
IDISJAV6 sample member 270
IDISJCTL skeleton member 331
IDISNAP 37, 659
assembler invocation example 46
COBOL invocation example 42, 43
entry specifications 38
IDISNAP sample members 38
input parameter list 38
invocation 38
PL/I 31-bit DLL invocation example 45
PL/I 64-bit DLL invocation example 46
PL/I non-DLL invocation example 44
return specifications 41
Snapdata option 38, 633
IDISPDM sample member 335
IDISPLI sample member 335
IDISPLI USERMOD 335
IDISPLIA sample member 335
IDISPLIA USERMOD 335
IDISRC1 sample member 431
IDISROBT sample member 354
IDISTSOB sample member 358
IDISUFM3 sample member 539
IDISUFMn sample member 493
IDISUFMX sample member 234
IDISUSI exit sample 298
IDISUSI sample member 298
IDISUTL1 sample member 514
IDISVENU sample member 308
IDISVJPN sample member 309
IDISXNFY sample member 351
IDISXPLA sample member 468
IDISXPLB sample member 468
IDISXPLC sample member 468
IDISXPLP sample member 468
IDISYSDB
attributes 387
data sets 649, 649, 649
DD statement 35
specifying through DataSets option 581
IDISYSLB concatenation 223
IDITRACE
activating compiler listing/side file search trace 383
controlling dynamically 650
writing to 556
IDITRACE DD statement 470, 574, 599, 625, 713
IDITRACE under CICS 402
IDIUTIL batch utility 441
return codes 693
IDIUTIL batch utility user exit samples 453
IDIUTIL Delete user exit 510
IDIUTIL Import user exit 507
IDIUTIL ListHF user exit 511
IDIUTIL ListHFDUP user exit 515
IDIVIEWS DDname 59
IDIVPASM sample member 428, 428
IDIVPBLE sample member 432
IDIVPC sample member 431
IDIVPCOB sample member 429, 432
IDIVPDB2 sample member 435
IDIVPDBB sample member 436
IDIVPPLE sample member 430
IDIVPPLI sample member 430, 432
IDIVSENU
READ access requirement 309
IDIVSJPN
READ access requirement 309

IDIVSxxx
specifying through DataSets option 581
IDIWCIDI sample member 563
IDIWRITE REXX command 533
IDIWTO REXX command 535
IDIWTSEL sample member 412
IDIXCCEE 659
IDIXCEE 659
IDIXCEE LE exit enablement
verifying 432
IDIXCX53 exit 400, 658
IDIXDCAP 659
installing 333
IDIXDCAP real-time analysis
suppressing 342
IDIXDLOC function 551
IDIXEINF function 552
IDIXFA 400
IDIXFXIT
entry specifications 366
example 368
input parameter list 367
return specifications 368
user exit purpose 366
IDIXGETN function 553
IDIXJAVA 418
IDIXLIST function 555
IDIXMIT sample member 359
IDIXSFOR
assembler example 387
input parameter list 385
IDIXSFOR exit
compiler listing 384
side file 384
IDIXTSEL 659
IDIXUFMT load module 550
functions 551
IDIXDLOC function 551
IDIXEINF function 552
IDIXGETN function 553
IDIXGETS function 553
IDIXGETX function 554
IDIXLIST function 555
IDIXNOTE function 556
IDIXTRCE function 556
IDIXWRIT function 557
IDIXWTO function 558
IDIXXLOC function 558
IDIXWRIT function 557
IDIXWTO function 558
IDIXXLOC function 558
IEATDUMP
changing default name for recovery fault recording 339
dump title 52
IEAVTABX MVS exit 35
IFAPRDxx parmlib member 462
registering Fault Analyzer 316
IGZIUXB exit
locating SYSDEBUG files 380
implicit refresh 92
IMPORT control statement 448
IMS
Java sample 269
LE considerations 417
supported versions 26
IMS environment
customizing 417
IMS fast duplicate fault suppression 618
IMS information
system-wide information 191
IMS information display example 191
IMS information summary display
example 195
Include option 595
INFO command 106
input parameter list
IDISNAP 38
installation checklist 291
installation status, verifying 291
installation-wide default options 369, 573
interactive displays
copying to file 125
interactive options line 158
interactive reanalysis 24
data sets 240
initiating 82, 164
options 157
performing 157
performing under CICS 284
user notes 147, 212
interactive reanalysis report display
example 165
interactive reanalysis status display
example 164
interactive reanalysis under CICS 563
interactive report
abend job information 200
COBOL Eplorer 241
deferred breakpoints feature 246
displaying associated storage areas 202
displaying Java analysis 262
displaying source code 206
displaying storage locations 209
event summary 168
exit 167
expanding abend codes 206
expanding messages 206
Fault Analyzer options 201
general information 165
synopsis 168
user notes 201
InteractiveExitPromptSeconds option 610
introduction 19
invocation for CICS transaction abends 302
IPL requirement 306
IPVCNF00 parmlib member 339, 341, 571
IPVLANGX
messages 695
IPVOPTLM configuration-options module 341, 571
ISPF
verifying Fault Analyzer 438
ISPF data sets
allocating for Japanese feature 426
ISPF environment
modifying 327
ISPF interface 55
action-bar pull-down menus 94
commands 98
customizing 373
fault entry commands 81
invoking 56
line commands 81
on-line help 56
providing defaults for new users 330
security considerations 141
split screen support 56
ISPF packed data format 390
ISPF selection panel
updating 328
ISPLIBD TSO/ISPF command 327
ISRDDN TSO/ISPF command 327
ISRFIND TSO/ISPF command 327
IVP testing
CICS 432

J

J9 VM issues 425
Japanese feature
allocating ISPF data sets 426
customizing 426
Java
compiling for optimal debugging 393, 394
IDIS subsystem requirements 324
invoking Fault Analyzer 418
JVTMI agent 424
reporting limitations 267
supported versions 26
Java analysis 259
displaying in interactive report 262
setting options 259
Java API
download reports 744
Java application abends
examples 392
providing Java source information 391
required LE options 303
Java capture SDUMP data sets
managing with XFACILIT resource class 418
Java dump analysis
creating history file fault entry 259
Java dump data set
selecting 259
Java dump events
invoking Fault Analyzer 421
Java event 266
Java event details display example 264
Java event summary display example 262
Java fault entry reanalysis 260
Java information
system-wide information 198
Java information display example 264
Java interactive reanalysis report display
example 262
Java sample
64-bit Enterprise PL/I 269
C++ program 270
data set 268
Enterprise COBOL 269
Enterprise PL/I 268
IMS batch processing 269
requirements 268
temporary files 268
wrapper utility 270
Java try-catch block
sample 270
Java wrapper utility 422
sample 270
JCL command 106
JclCapture option 612
JVM V9 issues 425
JVTMI agent 424

K

keywords, syntax diagrams xii

L

L tag 547
Language Environment *see* LE
Language Environment abnormal termination exit for CICS
enabling 397

Language Environment for CICS
configuring to invoke Fault Analyzer 397
Language Environment options required for invocation of Fault Analyzer 303
Language option 612
LangxCapture option 613
last accessed history file entries display example 62
last accessed history files or views display example 61
last CICS 3270 screen buffer hex display example 179
LE
DB2 considerations 415
IMS considerations 417
preventing wrap of CICS trace 410
LE abnormal termination exit
enabling 333
MVS change options/suppress dump exit 299
LE heap analysis
information display example 198, 199
system-wide information 198
LE options 303, 410
for CICS abends 304
for non-CICS abends 303
to capture Java application abends 303
legal notices
Fault Analyzer dcclix
level 88 items
collapsing 204
level 88 items display example 205
LI tag 547
Liberty or Java dump analysis 259
library names after installation 295
license information
displaying 126
license inquiry dcclix
line command for ISPF 3.4 data set list 329
LINKLIST
modules in 306
List REXX command 536
LISTHF control statement 442
LISTHFDUP control statement 443
listing/side file mismatch display example 237
listings
pointing to 373
load modules
obtaining from CA-Panexec 335
LOADER restriction 32
locale name 648
Locale option 614
lock flag 94, 131, 662
Lock Flag Value default setting 113
logical history file size 345
LOOKC command 108
LOOKC ISPF command 330
LOOKUP command 106
invoking through cursor selection 330
Lookup Search and Browse display example 124
LoopProtection option 614
LPA
modules in 306
placing modules in 459
LST data area 670

M

main report sections 275
managed history files 143
Marketplace, Visual Studio 283
MATCH ALL match condition 79
MATCH command 77, 79, 108
MATCH CSR match condition 80
matching
cursor-selecting match value 77
over-typing existing values 79
maximum call depth 278
maximum fault entries 88
MaxMinidumpPages option 615
MCP server
configuring 564, 565
Message and Abend Code Explanation user exit 485
message explanations
display example 206
repository 308
user-defined 455
message ID look-up display example 123
messages
expanding 206
Fault Analyzer 695
IDISF* 695
IPVLANGX 695
system-wide information 188
migrating 287
from V15.1 to V16.1 287
from V16.1 to V17.1 287
minidumps
concept 19
limiting size 373
minimum fault entries 88
minimum storage requirements 296
MQSeries
support requirements 26
MTRACE records
system-wide information 199
MTRACE Records display example 199
multicultural support 426, 612, 614
MVS change options
installing 333
MVS change options/suppress dump exit
LE abnormal termination exit 300
MVS console
write to with IDIXWTO function 558
MVS dump data set size 304
MVS IEAVTABX exit 35
MVS open prompt 159
MVS post-dump exit IDIXTSEL
installing 412
MVS SVC dump registration
Analysis Control user exit 479
Notification user exit 506

N

national language
setting 426
new history file allocation display example 87
NEXT command 108
NFY data area 673
NODUMP keyword on EXEC CICS ABEND command 400
NoDup option 616
NoDup(CICSFAST) option
changing 624
NoDup(ImageFast) option
changing 618
NoErrorHandler option 594
NOIPVOPT setting 341
non-CICS transaction abends
invoking analysis 298
non-ISPF interface
installing 561, 561
non-ISPF interfaces
accessing history files 283
non-LE run time
applications 335
non-REXX logging routine
calling from REXX 502
non-REXX user exit buffered data format 639
NoPrintInactiveCOBOL option 628
NoQuiet option 629
NoSource option 633
NoSpinIDIREPRT option 634
not found dump status 134
NOTE command 108
Note REXX command 538
NOTEL tag 548
NOTELIST command 109
NOTEST(DWARF) option limitations 378
Notification user exit 498
Notification user exit (MVS SVC dump registration) 506

O

object code
disassembling with DISASM command 229
OMVS home path 319
open files
system-wide information 173
operating environment
customizing 306
option changes 576
options
_IDI_OPTS environment variable 576
_IDI_OPTSFILE environment variable 575
AdditionalIDIOffDD 577
batch reanalysis 149
BatchOpts 116, 603
BindExclude 577
CallEqaueDat 578
CICSDumpTableExclude 578
CICSTraceMax 579
CICSTranAnalysisUser 580
CompileDateCheck 580
cumulative 572
DataSets 373, 581
default *see* default options
DeferredReport 589
Detail 372, 591
displaying current 96, 100, 147
DumpDSN 592
DumpRegistrationExits 593
ErrorHandler 594
Exclude 371, 595
Exits 600
FaultID 610
GenerateSavedReport 611
GlobalExclude 608
HistCols 609
IDICNFxx parmlib member 573
IDIOPTS 575
Include 371, 595
installation-wide 573
interactive reanalysis 157
InteractiveExitPromptSeconds 610
JclCapture 612
Language 426, 612
LangxCapture 613
Locale 614
LoopProtection 614
MaxMinidumpPages 373, 615
NoDup 616

NoErrorHandler 594
NoPrintInactiveCOBOL 628
NoQuiet 629
NoSource 633
NoSpinIDIREPRT 634
PARM field 576
PermitLangx 626
PreferredFormattingWidth 628
PrintInactiveCOBOL 628
processing order 571
purpose 571
Quiet 373, 629
RefreshExits 630
RetainCICSDump 631
RetainDump 632
setting 426
setting for CICS system abend analysis 247
setting for Java analysis 259
Snapdata 633
Source 633
specifying 573
SpinIDIREPRT 634
StoragePrintLimit 634
StorageRange 635
syntax rules 572
SystemWidePreferred 636, 638
user options file 575
user-options module IDICNFUM 574
where set 571
organization of this document xii

P

P tag 548
packed data format
ISPF *see* ISPF packed data format
PARM field options 576
parmlib data set
specifying alternative for IDICNFxx 339
PDS or PDSE
allocating as history file 343
converting history file 345
PermitLangx option 626
PF keys
showing 57
PL/I
64-bit Java sample 269
Java sample 268
supported versions 26
verifying Fault Analyzer 430
PL/I invocation
31-bit DLL example 45
64-bit DLL example 46
non-DLL example 44
PL/I ON ERROR usage 304
PL/I TEST (NOSEPERATE NOSOURCE) option limitations 378
PL/I V2R3 applications
enabling implicit Fault Analyzer invocation 335
PL/I version 2 release 3
USERMOD to facilitate invocation of Fault Analyzer 335
PLIDUMP
invoking Fault Analyzer 335
PLT
CICS startup
installing Fault Analyzer 398
required programs 398
plug-in for Eclipse
installing 561
overview 283, 284
point of failure 669
point-and-shoot fields 166
preferences, Fault Analyzer 113
preferred formatting width display
example 122
PreferredFormattingWidth option 628
PREV command 109
PrintInactiveCOBOL option 628
program control access
defining 307
program descriptions
user-defined 457
program SNAP interface 37
programs
defining to CICS 399
setting up for fault analysis 29
prompting
controlling 240

Q

Quiet option 629
QUIT command 109

R

READ access requirement
IDIDOC 309
IDIVSENU 309
IDIVSJPN 309
real-time abend analysis 19
real-time analysis 32
controlling with options 34
displaying captured JCL 82
displaying debugging information 81
real-time analysis report
viewing saved 120
real-time analysis reports 35
changing SYSOUT class 36
combining 36
dynamic allocation 36
preventing dynamic IDIREPRT deallocation 634
suppressing 37
real-time SNAP analysis 21
reanalysis
batch 81, 149, 154
interactive 157, 240, 284
recording area
clearing NoDup(CICSFAST(...)) 402
recovery fault recording 310
changing default IEATDUMP data set name 339
Fault Analyzer IDIS subsystem usage 318
overview 51
verifying setup 439
recovery fault recording data set access
managing 309
REFRESH command 92, 110
refresh processing 240
RefreshExits option 630
region size
extra required 30
region size requirements 296
registering Fault Analyzer 316
repeatable items, syntax diagrams xii
report
contents 272
report detail
controlling 372
report examples 282
report sections
abend job information 281
AI analysis report 281
epilog 282
event details 279
main 275
options 281
prolog 275
summary 275
synopsis 275
system-wide information 281
user section 281
report user exit *see* Analysis Control user exit
reports
download with Java API 744
increasing display area 331
real-time *see* real-time reports
required programs
defining to CICS 397
RESET command 110
RetainCICSDump option 631
RetainDump option 632
return codes
batch reanalysis 693
IDIDA 693
IDILANGX utility 693
IDIUTIL batch utility 693
return specifications
IDISNAP 41
REXX
calling non-REXX logging routine 502
REXX commands 517
Evaluate 517
IDIALLOC 521
IDIDDTEST 525
IDIDSECTdsn 526
IDIDSNTEST 527
IDIEventInfo 528
IDIFREE 529
IDIGET 530
IDIModQry 530
IDIPUT 531
IDIRegisterFaultEntry 532
IDIWRITE 533
IDIWTO 535
List 536
Note 538
REXX exec libraries
pointing to 373
REXX SAY instruction 474
REXX TRACE instruction 474
RFR dump titles 52
RFR SDUMP data sets
XFACILIT resource class 310
RFR TDUMP XFACILIT example 312
routing codes
WTO messages 31
row count 58
RPTFIND command 110
RUNCHAIN command 110

S

S913 abend 339
SAA overlays 177, 412
sample customized ISPF interface front-end 741
sample data set members
IDI.SIDISAM1 415
IDICNFxx 369
IDIGSVRJ 561
IDIOPTLM 337
IDIPANEX 335
IDIS$NDX 26

IDISCICS 399
IDISCNFU 574
IDISDB2B 436
IDISFE* 741
IDISHIST 343
IDISISPF 327
IDISJAV1 268
IDISJAV2 269
IDISJAV3 269
IDISJAV4 269
IDISJAV5 270
IDISJAV6 270
IDISPDM 335
IDISPLI 335
IDISPLIA 335
IDISRC1 431
IDISROBT 354
IDISTSOB 358
IDISUFM3 539
IDISUFMn 493
IDISUFMX 234
IDISUSI 298
IDISUTL1 514
IDISVENU 308
IDISVJPN 309
IDISXNFY 351
IDISXPLA 468
IDISXPLB 468
IDISXPLC 468
IDISXPLP 468
IDIVPASM 428, 428
IDIVPBLE 432
IDIVPC 431
IDIVPCOB 429, 432
IDIVPDB2 435
IDIVPDBB 436
IDIVPPLE 430
IDIVPPLI 430, 432
IDIWCIDI 563
IDIWTSEL 412
IDIXMIT 359
sample reports 282
sample XFACILIT implementation 365, 365, 366
saved report
viewing 120
saved report display example 120
SAY
REXX instruction 474
screen size 753
SDUMP
data set name patterns 341
SDUMP data set
managing access 314
SDUMP recovery fault recording data sets 310
SDUMP SVC
screening 399
SDUMPDSN setting 341
security considerations 141
selected history files 145
SETFAULTPREFIX control statement 446
SETMAXFAULTENTRIES control statement 446
SETMINFAULTENTRIES control statement 447
SETPROG command
managing modules 459
SFA command 330
SHOW command 111
shutdown PLT 398
side file
access requirements 384
side file availability test utility 389
side files
activating search trace 383
advantage over listings 29
attributes 387
definition 235
locating 380
naming 378
prompting display 235
providing 374
selection criteria 30
temporary data set prohibition 387
SIT command 111
SLIP traps 305
SLIP,COMP=0C4 trap 53
SMF type 89 record 50, 461
SMP/E APPLY 459
SMP/E RESTORE 459
snap dump method
invoking Fault Analyzer from 418
Snapdata option 633
source code
displaying 206
Source option 633
special history file data set members 25
$$BACKUP 26
$$INDEX 25
special processing of Language Environment CEEWUCHA abends 30
specify compiler listing or side file display example 236
SpinIDIREPRT option 634
SQLCA 191, 637
staging data set
IDIROBOT sample EXEC 350
IMPORT control statement 448
startup PLT
adding programs 398
installing Fault Analyzer 398
status pop-up display 164
STCK command 112
STCK conversion display example 231
STDENV DD statement
environment variables 569
Storage Accounting Area (SAA) overlays 177, 412
storage areas
displaying associated 202
mapping with DSECT information 223
report with IDIXLIST function 555
system-wide information 197
Storage Disassemble display example 229
storage DSECT mapping entry display example 224
storage DSECT mapping map display example 225
storage locations
displaying 209
storage recommendations 296
storage requirements 296
storage RUNCHAIN command entry display example 227
StoragePrintLimit option 634
StorageRange option 635
STORE CLOCK values (STCK command)
converting 231
substitution symbols in data set names 476, 585
subsystem for Fault Analyzer
user exits running from 593
subsystems
IDIS 318
successful analysis
criteria 669
summarized CICS trace
system-wide information 179
summarized CICS trace display example 179
summary of exit usage 300
suppress dump exit IDIXDCAP
installing 333
suppressed copybooks 274
SVC 51
screening 399
SVC dump data set display 247
SVC dump exit 412
SVC dump registration exit 302
SVC dump screening 399
SVC installation
performing dynamically with IDICZSVC 306
SVCDUMP dump title 52
symbol substring specification 588
symbols in data set names 476, 585
synopsis 168
synopsis display example 168
syntax diagrams, how to read xii
SYSADATA DD statement 374
SYSCLONE MVS system symbol
IDICNFxx parmlib member suffix 369
SYSDEBUG files
locating with EQAUEDAT exit 381
locating with IGZIUXB exit 380
syslog messages
suppressing noncritical 373
SYSLOG summary 37
SYSMDUMP
ASA printer control characters 592
dump data set display 247
logical record length 592
specifying as input to batch reanalysis 592
SYSOUT class of real-time reports
controlling 36
sysplex
sharing history files 362
Sysplex-wide subsystem inter-communication 319
system dump suppression
controlling with user exit 494
system requirements 26
system-wide information
CICS 176
summarized CICS trace 179
CICS information
CICS trace formatting 180
DB2 188
IMS information 191
Java Information 198
LE heap analysis 198
messages 188
MTRACE records 199
open files 173
storage areas 197
system-wide messages display example 188
system-wide open files display example 173
system-wide storage areas display example 197
SystemWidePreferred option 636, 638
SYSUDUMP SYSOUT class
allocation of IDIREPRT 35

T

table displays
matching 253
sorting 253

TACB
duplicate determination under CICS 623
tags *see* format tags
TCB protection key support 26
TDUMP recovery fault recording data sets 311
temporary data sets
compiler listing restriction 387
side file restriction 387
TEST option considerations 377
TEST(NONE,SYM,SEPARATE) COBOL compiler option 377, 380
restriction with display of pseudo-assembler instructions 208
TEST(STMT,SYM,NOHOOK,SEPARATE) Enterprise PL/I compiler option 377, 380
TH tag 549
third-party products 304
title 111, 743
total rows displayed 58
TRACE
REXX instruction 474
transaction abend analysis
CICS *see* CICS
transaction abend control block *see* TACB
trapping abends 54
try-catch block
sample 270
TSO
REXX environment restrictions 467

U

U tag 549
UFM data area 674
UL tag 550
updates pending 92
user exit type specific data area 467
user exits 464
Analysis Control 475
Analysis Control (MVS SVC dump registration) 479
CICS system abend analysis 247
Compiler Listing Read 480
data area version checking 469
dropping 602
End Processing 494
End Processing (fault entry refresh) 497
for multiple exit types 469
Formatting 490
IDIUTIL Delete 510
IDIUTIL Import 507
IDIUTIL ListHF 511
IDIUTIL ListHFDUP 515
invocation parameters 467
Message and Abend Code Explanation 485
Notification 498
Notification (MVS SVC dump registration) 506
substitution symbols in data set names returned 476
supported programming languages 467
tracing 470
user modifications
parmlib data set name 369
user note list display example 215
user notes 201
creating 212
managing 212
recovering 147
split screen limitations 56
user notes update prompt example 216
user options file 575
user-defined
abend code explanations 456
message explanations 455
program descriptions 457
user-options file 572
user-options module IDICNFUM 574
user-selected message explanations
displaying 123
user-specific report formatting (EXEC command) 233
USERMODs
IDISPLI 335
IDISPLIA 335
IDIWTSEL 412
re-applying 459
restoring 459
USRHDLR(CEEWUCHA)
special processing 30
UTL data area 688

V

variables, syntax diagrams xii
version and maintenance level, displaying 126
view considerations
without XFACILIT resource class 350
view list display example 63
views 59
selecting for display 59
setting up 347
VIEWS command 112
VS Code extension 283

W

WebSphere Liberty 259
width
setting preferred formatting 122
wildcards
matching fault history entries 80
wrapper utility
sample 270
write-to-operator messages *see* WTO
WTO
descriptor codes 31
routing codes 31

X

XCF 319, 363
XDUMP data set 53
managing access 313
name pattern 340
XFACILIT resource class 313
XDUMPs
verifying Fault Analyzer 428
XDUREQ CICS global user exit 340
XEIIN global user exit 412
XFACILIT implementation example 365, 365, 366
XFACILIT resource class 313
XFACILIT resource classes
IDI_SDUMP_ACCESS 310, 418
IDIHIST_GROUP_DSN 363
IDIHIST_USERID_DSN 363
IDIRFR_TDUMP_HLQ 312
managing history file fault entries 363
managing Java capture SDUMP data sets 418
managing recovery fault recording SDUMP data sets 310
managing TDUMP RFR data sets 311
XMIT options display example 140
XML command 112
XPCABND CICS global user exit 400, 408
XPL data area 690

Z

z/OS
real-time dump processing before V2.2 35
supported versions 26