

IMS  
15.6.0

*Diagnosis*  
*(2025-06-16 edition)*



**Note**

Before you use this information and the product it supports, read the information in [“Notices” on page 669](#).

2025-06-16 edition.

© **Copyright International Business Machines Corporation 1974, 2025.**

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

---

# Contents

<b>About this information.....</b>	<b>xi</b>
Prerequisite knowledge.....	xi
How new and changed information is identified.....	xi
How to read syntax diagrams.....	xi
Accessibility features for IMS 15.6.....	xiii
How to send your comments.....	xiii
 <b>Chapter 1. Collecting IMS diagnostic information.....</b>	<b>1</b>
Standard IMS diagnostic information.....	1
Managing standard diagnostic information.....	2
Preserving the z/OS console (syslog).....	2
Preserving the JES JOBLLOG.....	3
Preserving the IMS master console log.....	3
Preserving the SYS1.LOGREC.....	3
Preserving memory dumps.....	4
Preserving the IMS OLDS and SLDS.....	4
Manual intervention for dump creation.....	4
Deciding when to create a memory dump.....	5
Creating IMS memory dumps.....	5
IEADMCxx, z/OS SYS1.PARMLIB.....	5
IMS sysplex dump considerations.....	6
 <b>Chapter 2. Collecting data about specific problems.....</b>	<b>9</b>
Control region wait or hang.....	9
Control region or DL/I region loop.....	10
IMS dependent region wait or loop.....	11
Formatting a BPE trace entry.....	12
DBRC-related problems.....	15
DBRC security override.....	15
DBCTL-related problems.....	15
VTAM-related problems.....	16
OTMA-related problems.....	17
APPC-related problems.....	19
IMS Connect problems.....	20
ISC TCP/IP problems.....	21
CQS-related problems.....	23
CSL-related problems.....	23
Repository Server audit log records.....	24
Managing Repository Server audit log records.....	25
How the Repository Server handles z/OS logger errors.....	25
Printing Repository Server audit log records.....	26
ESAF and DB2 ESS interface problems.....	27
Database problems.....	27
Collecting data about z/OS Resource Recovery Services problems.....	28
MSC-related problems.....	29
 <b>Chapter 3. Collecting data about abends by using IMS abend search and notification.....</b>	<b>31</b>
Enabling IMS abend search and notification.....	31
Researching information about abends dynamically.....	31

<b>Chapter 4. How to search problem-reporting databases.....</b>	<b>33</b>
Developing search arguments.....	33
Creating a search argument.....	34
Selecting the keywords.....	34
Component identification keyword procedure.....	34
Type-of-failure keyword.....	35
IMS keyword dictionary.....	63
Dependency keywords.....	65
Searching the database.....	66
Procedures for preparing an APAR.....	68
<b>Chapter 5. Data areas and record formats.....</b>	<b>71</b>
Finding more information on modules, control blocks, and record formats.....	71
Table of control block definitions.....	72
IMS Control Block Table (CBT) Pools.....	81
Control block interrelationship diagrams.....	88
DL/I record formats.....	127
HSAM and SHSAM database.....	127
HISAM and SHISAM database.....	128
HDAM, HIDAM, PHDAM, or PHIDAM database.....	129
OSAM and VSAM ESDS block format.....	131
VSAM LRECL for a primary index.....	132
Secondary index or PSINDEX database (VSAM only).....	133
Variable-length segments.....	135
<b>Chapter 6. CQS - Common Queue Server service aids.....</b>	<b>137</b>
Diagnosing a CQS related problem.....	137
CQS additional manual intervention for dump creation.....	138
CQS structure dump contents.....	138
CQS - z/OS log stream example.....	140
CQS structure recovery data set.....	140
CQS checkpoint problems.....	140
CQS structure rebuild problems.....	141
CQS trace records.....	142
CQS log records.....	146
Printing CQS log records.....	148
Copying CQS log records for diagnostics.....	149
<b>Chapter 7. CSL - Common Service Layer service aids.....</b>	<b>151</b>
CSL trace records.....	151
RM trace record example.....	154
<b>Chapter 8. DB - Database service aids.....</b>	<b>155</b>
Job control block trace.....	155
Sample JCB trace.....	155
JCB trace call function codes.....	156
DL/I test program - DFSDDLTO.....	157
COMPARE statement SNAPS.....	157
SNAPS on exceptional conditions.....	158
DL/I call image capture.....	159
Batch environment.....	159
Online environment.....	160
How to retrieve DL/I call image capture data from the log data set.....	160
DL/I analysis.....	160
IMS abends.....	160
Dump analysis introduction.....	161

Dump analysis - detailed.....	162
Generalized DL/I problem analysis.....	163
Locating database-related traces.....	164
DL/I trace.....	165
Using the DL/I trace facility.....	166
DL/I trace formats.....	167
Delete/Replace - DL/I trace information.....	207
Retrieve trace.....	208
Online Recovery Manager trace.....	212
Starting the Online Recovery Manager trace.....	213
Format of the Online Recovery Manager Trace.....	213
Online Recovery Manager trace example.....	219
Program isolation-related problem analysis.....	219
Limiting locking resources used by an application program.....	220
Program isolation trace.....	220
DL/I call image capture program.....	221
Log analysis (database related).....	221
Sequential buffering service aids.....	227
SBSNAP option.....	228
SBESNAP option.....	228
Testing algorithms using the SB Test utility (DFSSBHD0).....	229
SB COMPARE option.....	229
GSAM control block dump - DFSZD510.....	230
Example of a formatted GSAM control block dump.....	230
Example of an unformatted GSAM control block dump.....	231
Recovering from Sx37 abends on GSAM data sets.....	234
<b>Chapter 9. DBRC - Database Recovery Control service aids.....</b>	<b>239</b>
Diagnosing from a RECON list.....	239
RECON record types.....	239
DBRC internal trace.....	243
DBRC trace input.....	244
Locating the DBRC trace.....	244
DBRC trace output.....	245
DBRC trace header record.....	245
Module call, module return, and DSPSTACK trace entries.....	245
BGNCABN0, DSPCABN0, BGNRETRY, DSPCRTR0, and CRTROXIT trace entries.....	247
DSPURI00 trace entries.....	249
Trace entries related to parallel RECON access.....	253
DBRC group services entries.....	254
Unformatted DBRC internal trace example.....	255
DBRC external trace.....	263
Examples of DBRC router processing and RECON I/O error processing output.....	263
Samples of JCL to create trace output.....	264
<b>Chapter 10. BPE-based DBRC service aids.....</b>	<b>267</b>
BPE-based DBRC trace records.....	267
Trace record examples for BPE-based DBRC.....	268
Unformatted BPE-based DBRC internal trace example.....	273
<b>Chapter 11. DC - Data Communication Service Aids.....</b>	<b>277</b>
Terminal communication task trace.....	277
Analyzer entry points.....	277
Trace records.....	278
Trace output.....	279
DC trace.....	279
Starting the trace.....	279

Stopping the trace.....	280
Printing the trace records.....	281
Content of the trace records.....	282
Diagnosing line and terminal problems.....	291
Diagnosing problems in the Queue Control Facility Message Requeuer.....	297
IBM IMS Queue Control Facility for z/OS interface.....	298
SCRAPLOG diagnostic records.....	300
6701-MRQE diagnostic records.....	302
Obtaining diagnostics in addition to SCRAPLOG and 6701-MRQE.....	304
Determining when messages are successfully queued.....	305
Diagnosing message routing problems.....	306
DFS070 UNABLE TO ROUTE MESSAGE RSN=xyyy .....	306
DFSMSCE0 TM/MSD Message Routing exit trace.....	312
Diagnosing routing errors by using the transaction trace or program trace.....	315
Diagnosing routing problems by using the DC LINE/NODE/LINK TRACE.....	316
Using 01/03 log record trace.....	316
IMS transaction trace.....	317
Receive-any buffer analysis.....	321
IMS IPCS Dump Formatter panel.....	321
Manual process to determine receive-any buffers space issues.....	322
Finding the active save set.....	323
IMS VTAM interface.....	323
IBM 3270 error recovery analysis.....	324
Diagnosing Message Format Service problems.....	324
Message Format Service module traces.....	326
Tracing errors in module DFSCNXA0.....	328
Location codes for DFSCNXA0 error messages.....	328
Qualifier codes.....	335
IDC0 trace table entries.....	336
APPC/IMS diagnostic aids.....	339
LU manager trace.....	339
LU 6.2 module-to-code cross-reference table.....	348
APPC/MVS verb-to-code cross-reference table.....	350
DFS1959E message information.....	351
DFS1965E APPC/MVS call failures.....	364
Diagnostics for use with synchronous APPC and OTMA with shared queues.....	364
SNAPs and dumps.....	365
OTMA diagnostic aids.....	365
OTMA trace.....	365
OTMA trace entries for parallel RESUME TPIPE request processing (X'5A20').....	369
OTMA trace entry for user exits.....	375
OTMA trace entry for synchronous callout.....	378
OTMA trace entry for synchronous program switch.....	383
OTMA module-to-code cross-reference table.....	388
OTMA verb-to-code cross-reference table.....	389
DFS1269E message information.....	390
OTMA log records.....	390
SNAPs and dumps.....	391
Diagnosing Fast Path problems related to print data set options: IMS Spool API support.....	391
Understanding parsing errors.....	391
Debugging and diagnostic aids provided by IMS Spool API.....	395
<b>Chapter 12. DRA - Database Resource Adapter service aids.....</b>	<b>397</b>
DRA dumps.....	397
DRA SDUMP output.....	397
SNAP dump output.....	398
Recovery tokens.....	398

Analyzing DRA problems.....	398
<b>Chapter 13. FP - Fast Path service aids.....</b>	<b>401</b>
Diagnosing Fast Path problems.....	401
ABENDU1026 analysis.....	401
Fast Path Transaction Retry.....	404
DEDB control interval (CI) problem assistance aids.....	405
Locating Fast Path control blocks and tables.....	408
Locating IMS blocks and work areas by using load list elements.....	410
Fast Path external trace.....	415
Trace activation.....	416
Trace deactivation.....	416
Diagnostic data.....	416
Fast Path external trace examples.....	417
<b>Chapter 14. IMS Connect service aids.....</b>	<b>427</b>
IMS Connect Dump Formatter.....	427
Accessing the IMS Connect Dump Formatter.....	427
Initializing a dump by using the IMS Connect Dump Formatter.....	428
IMS Connect traces.....	429
Configuring BPE for an external trace of IMS Connect.....	431
Starting an external trace for IMS Connect.....	432
Stopping an external BPE trace of IMS Connect.....	433
Displaying the status of an external trace of IMS Connect.....	433
Formatting the trace data from an external trace of IMS Connect.....	434
Tracing to the HWSRCDR data set.....	434
Recorder log record mappings.....	434
<b>Chapter 15. IRLM service aids.....</b>	<b>441</b>
IRLM dumps.....	441
SYS1.LOGREC records.....	441
z/OS component trace.....	442
<b>Chapter 16. MSC - Multiple Systems Coupling service aids.....</b>	<b>445</b>
Multiple Systems Coupling communication task trace.....	445
Multiple Systems Coupling device-dependent module.....	445
Multiple Systems Coupling traces.....	447
Diagnosing link problems.....	448
Diagnosing link problems by using MSC link statistics.....	453
MSS1 and MSS2 records.....	454
MSS3 and MSS4 records.....	457
Channel-to-channel access method trace stack (LXB trace).....	458
DFSCMC00 (MSC analyzer).....	459
DFSCMC50 (shutdown processing routine).....	459
DFSCMC40 (attention DIE routine).....	459
DFSCNC40 (I/O request DIE routine).....	460
DFSCMC10 (channel-end appendage).....	461
DFSCMC10 (abnormal-end appendage).....	461
DFSCMC10 (shutdown appendage).....	462
MSC routing trace - BUFMSVID.....	464
<b>Chapter 17. ODBA - Diagnosing hung threads and UORs.....</b>	<b>465</b>
<b>Chapter 18. SYS - System service aids.....</b>	<b>469</b>
Log records.....	469
Format of X'29' log record.....	492

Format of X'67' log record.....	503
Printing log records and trace table entries.....	511
Log Merge utility.....	512
Formatting IMS dumps offline.....	512
Overview of the IMS Offline Dump Formatter.....	513
Solving IMS problems by using the IMS Offline Dump Formatter.....	514
Syntax restrictions on the FMTIMS statement.....	530
Contents formatted for FMTIMS options.....	531
Formatted dump contents.....	541
Edited command buffer format.....	549
IMS Dump Formatter.....	550
Dump formatting options.....	551
IMS Dump Formatter menus.....	556
IMS components formatting panels.....	559
Using IMS enhanced dump analysis.....	563
Formatting log records for use with log analysis tools.....	565
Formatting a dump for analysis.....	566
Formatting IMS dumps online.....	566
Formatted dump for the CTL address space.....	567
Formatted dump for the DL/I address space.....	570
SNAP call facility.....	572
/DIAGNOSE command SNAP function.....	573
Type-1 trace table interface.....	574
Finding type-1 trace tables in a dump.....	575
Format of type-1 trace records.....	577
IMS type-1 trace function codes.....	577
Common Service Layer trace.....	582
Dispatcher trace.....	587
ITASK ECB posting.....	595
System post codes.....	595
External subsystem trace.....	596
Layout of the X'57' variable section.....	599
Layout of the X'58' variable section.....	604
Resource Recovery Services trace.....	611
Scheduler trace.....	626
Storage manager trace.....	629
Latch trace.....	630
Queue manager trace.....	634
Shared queues interface trace.....	638
Fast Path trace.....	639
Fast Path trace entries.....	641
Type-2 trace tables.....	665
Type-2 trace records.....	665
IMS shutdown trace table.....	666
<b>Notices.....</b>	<b>669</b>
<b>Programming interface information.....</b>	<b>671</b>
<b>Trademarks.....</b>	<b>673</b>
<b>Terms and conditions for product documentation.....</b>	<b>675</b>
<b>IBM Online Privacy Statement.....</b>	<b>677</b>
<b>Bibliography.....</b>	<b>679</b>



<b>Index.....</b>	<b>681</b>
-------------------	------------



## About this information

---

These topics provide guidance and reference information for setting up an IMS system for diagnosis, collecting information to help diagnose IMS problems, and searching problem-reporting databases. These topics also describe how to use keywords to develop a failure description that you can use to search problem-reporting databases and communicate with IBM® Software Support.

This information is available in [IBM Documentation](#).

## Prerequisite knowledge

---

You will be most successful in using this information if you have a basic understanding of:

- IMS concepts and externals
- How to access an IBM Software Support database
- Dump analyses
- z/OS® diagnostic practices
- Telecommunications
- System Network Architecture (SNA)

To learn about z/OS, see [z/OS Basic Skills](#). For more resources, see [IBM Z Education and Training](#).

To learn about IMS, see the IBM Press publication *An Introduction to IMS*, the resources listed for [IBM Information Management System](#), and the variety of options available in [IBM Training](#).

## How new and changed information is identified

---

For most IMS library PDF publications, information that is added or changed after the PDF publication is first published is denoted by a character (revision marker) in the left margin. The *Program Directory* and *Licensed Program Specifications* do not include revision markers.

Revision markers follow these general conventions:

- Only technical changes are marked; style and grammatical changes are not marked.
- If part of an element, such as a paragraph, syntax diagram, list item, task step, or figure is changed, the entire element is marked with revision markers, even though only part of the element might have changed.
- If a topic is changed by more than 50%, the entire topic is marked with revision markers (so it might seem to be a new topic, even though it is not).

Revision markers do not necessarily indicate all the changes made to the information because deleted text and graphics cannot be marked with revision markers.

## How to read syntax diagrams

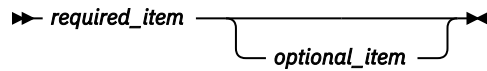
---

The following rules apply to the syntax diagrams that are used in this information:

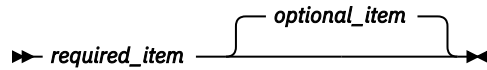
- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following conventions are used:
  - The >>--- symbol indicates the beginning of a syntax diagram.
  - The ---> symbol indicates that the syntax diagram is continued on the next line.
  - The >--- symbol indicates that a syntax diagram is continued from the previous line.
  - The --->< symbol indicates the end of a syntax diagram.
- Required items appear on the horizontal line (the main path).

➤ *required\_item* ➤

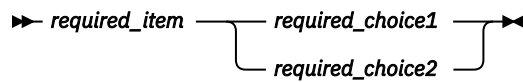
- Optional items appear below the main path.



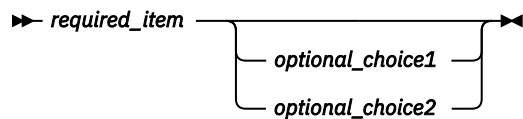
If an optional item appears above the main path, that item has no effect on the execution of the syntax element and is used only for readability.



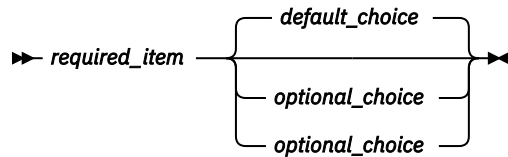
- If you can choose from two or more items, they appear vertically, in a stack.
- If you *must* choose one of the items, one item of the stack appears on the main path.



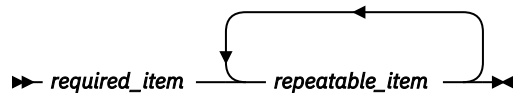
If choosing one of the items is optional, the entire stack appears below the main path.



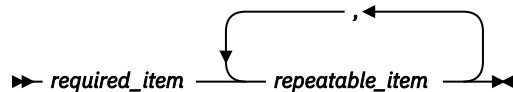
If one of the items is the default, it appears above the main path, and the remaining choices are shown below.



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

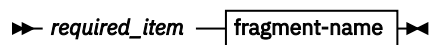


If the repeat arrow contains a comma, you must separate repeated items with a comma.

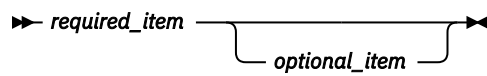


A repeat arrow above a stack indicates that you can repeat the items in the stack.

- Sometimes a diagram must be split into fragments. The syntax fragment is shown separately from the main syntax diagram, but the contents of the fragment should be read as if they are on the main path of the diagram.



**fragment-name**



- In IMS, a b symbol indicates one blank position.
- Keywords, and their minimum abbreviations if applicable, appear in uppercase. They must be spelled exactly as shown. Variables appear in all lowercase italic letters (for example, *column-name*). They represent user-supplied names or values.
- Separate keywords and parameters by at least one space if no intervening punctuation is shown in the diagram.
- Enter punctuation marks, parentheses, arithmetic operators, and other symbols, exactly as shown in the diagram.
- Footnotes are shown by a number in parentheses, for example (1).

## Accessibility features for IMS 15.6

---

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

### Accessibility features

The following list includes the major accessibility features in z/OS products, including IMS 15.6. These features support:

- Keyboard-only operation.
- Interfaces that are commonly used by screen readers and screen magnifiers.
- Customization of display attributes such as color, contrast, and font size.

### Keyboard navigation

You can access IMS 15.6 ISPF panel functions by using a keyboard or keyboard shortcut keys.

For information about navigating the IMS 15.6 ISPF panels using TSO/E or ISPF, refer to the *z/OS TSO/E Primer*, the *z/OS TSO/E User's Guide*, and the *z/OS ISPF User's Guide Volume 1*. These guides describe how to navigate each interface, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

### Related accessibility information

Online documentation for IMS 15.6 is available in IBM Documentation.

### IBM and accessibility

See the *IBM Human Ability and Accessibility Center* at [www.ibm.com/able](http://www.ibm.com/able) for more information about the commitment that IBM has to accessibility.

## How to send your comments

---

### About this task

Your feedback is important in helping us provide the most accurate and highest quality information. If you have any comments about this or any other IMS information, you can take one of the following actions:

## Procedure

- Submit a comment by using the DISQUS commenting feature at the bottom of any [IBM Documentation](#) topic.
- Send an email to [imspubs@us.ibm.com](mailto:imspubs@us.ibm.com). Be sure to include the book title.
- Click the **Contact Us** tab at the bottom of any [IBM Documentation](#) topic.

## What to do next

To help us respond quickly and accurately, please include as much information as you can about the content you are commenting on, where we can find it, and what your suggestions for improvement might be.

# Chapter 1. Collecting IMS diagnostic information

Before you report a problem to IBM Software Support, collect information to help document the problem at your installation. Having this information available when you call IBM can save you time because you might not need to create the problem again.

As a result of its complexity, IMS can experience problems that must be diagnosed and corrected. Examples of problems that you might encounter while running IMS include an abnormal end (known as an *abend*) occurs in processing, a job hangs in the system and does not process, a process repetitively loops through a series of instructions, or processing slows down.

For these types of problems, IMS displays symptoms that can help you with your diagnosis, but, in order to obtain that information, you need to gather all of the correct data to diagnose a problem.

To collect data about a system problem:

1. Collect the symptom data and determine what type of problem it is.
2. Use the procedures recommended to diagnose the problem to determine whether the problem is an IMS problem or a user problem.
3. If the problem is an IMS or system problem, build a search argument from the data that you collect as a result of following the procedure for that problem. For example, the data you gather from a control region wait can be helpful in building a search argument to search the symptom database with.
4. Search the symptom database. You might need to refine your search with more data from the problem.
5. If you cannot find a known problem with the same symptoms, report the problem to IBM Software Support.

## Related concepts

[Setting up IMS for diagnostics \(System Definition\)](#)

## Standard IMS diagnostic information

If you contact IBM Software Support for assistance with a problem, you might be asked to collect a standard set of logs, data sets, and dumps that can help them determine the source of your problem. Collecting this information before you contact IBM Software Support will shorten the amount of time required to resolve your problem.

The following table describes the types of information that are most commonly requested by IBM Software Support:

*Table 1. Information IBM Software Support might need*

Log, data set, or dump name	Purpose	Which version to save
SYSLOG	The SYSLOG is useful when the dumped MTRACE buffer is not large enough to contain all necessary error messages.	Save the SYSLOG from time of IMS start up.
LOGREC data set	z/OS failures are logged internally.	Save the LOGREC data set from IMS start up time.
IMS master console log	The master console log provides a different message set than the SYSLOG.	Save the master console log from IMS start up time.
IMS log data sets (OLDS)	The IMS log data sets track IMS transaction and database activity.	Save the IMS online data sets that are active at the time of the error.

Table 1. Information IBM Software Support might need (continued)

Log, data set, or dump name	Purpose	Which version to save
IMS system log data sets (SLDS)	The IMS system log data sets track IMS transaction and database activity.	Save the SLDS from IMS start up time.
JES job log of jobs related to failure	The JES job log provides JCL start up parameters and isolated system messages.	Save the JES job log from IMS start up time.
All dumps created near the time of IMS failure	Multiple SYS1.DUMPs might be created for related failures. SYSDUMP for the IMS Control, DLI/SAS, and DBRC regions might be created if the primary SYS1.DUMP encounters problems. Also, look for related SYSUDUMPs for IMS dependent regions	Save copies of these dump datasets.
z/OS log data sets	The z/OS log data sets provide information for structure rebuild and checkpoint related problems.	Save the current z/OS log data sets for the failing CQS job stream.

#### Related tasks

[“Collecting data about IMS Connect problems” on page 20](#)

If a problem occurs during IMS Connect execution, you need to collect logs, data sets, and dumps to determine the source of the problem.

## Managing standard diagnostic information

You can preserve documentation that can be helpful near the time of error.

### About this task

Consider implementing normal operating procedures for the following tasks:

## Preserving the z/OS console (syslog)

The z/OS Console should be saved to view relevant system messages.

### About this task

- The ideal time frame:
  - Back to the last IMS restart
  - z/OS Console from the prior clean execution (for comparison)
- The moderate time frame:
  - 24 hours of z/OS Console messages
- The minimum time frame:
  - Two IMS system checkpoint intervals



## Preserving the JES JOBLOG

Preserve the JES JOBLOG to view relevant job-related messages.

### About this task

- Save the JES JOBLOGs for:
  - The IMS control region
  - The IMS DLI/SAS region
  - The IMS DBRC region
  - Any suspicious IMS dependent regions
  - The CQS regions
  - The OM region
  - The RM region
  - The SCI regions
- The ideal time frame:
  - JES JOBLOG from the current error execution
  - JES JOBLOG from the prior clean execution (for comparison)
- The moderate time frame:
  - 24 hours of JES JOBLOG
- The minimum time frame:
  - Two IMS system checkpoint intervals, or two hours, whichever is greater

## Preserving the IMS master console log

The IMS Master Console Log should be saved to view relevant IMS messages:

### About this task

- The ideal time frame:
  - IMS Master Console Log from the current error execution
  - IMS Master Console Log from the prior clean execution (for comparison)
- The moderate time frame:
  - 24 hours of IMS Master Console
- The minimum time frame:
  - Two IMS system checkpoint intervals or two hours, whichever is greater

## Preserving the SYS1.LOGREC

The SYS1.LOGREC should be saved to view system failures logged internally.

### About this task

- The ideal time frame:
  - Back to the last IMS restart
- The moderate time frame:
  - 48 hours of SYS1.LOGREC data
- The minimum time frame:

- Current SYS1.LOGREC data set

## Preserving memory dumps

Retain all IMS memory dumps generated at or near the time of a problem.

### About this task

- SYS1.DUMP data sets should be examined:
  - Multiple dumps might be created.
  - Keep all dumps at time of failure, regardless of the subsystem.
- SYSMDUMP for the IMS Control, DLI/SAS, and DBRC regions need to be examined in case of primary SYS1.DUMP failures.
  - Save these data sets, if a dump was produced.
- SYSUDUMP should be saved for IMS dependent regions.

**Note:** If you use the z/OS-provided JCL BLSJDPFD in SYS1.SAMPLIB to generate a redacted SVC dump or a stand-alone dump (SADUMP) to send to IBM or another software vendor, ensure that you keep the original complete dump until your case has been resolved. Data removed by the tool might be necessary for solving a problem. IBM might request you to provide specific information from the original dump, even if you do not send the complete dump to IBM for data privacy reasons.

## Preserving the IMS OLDS and SLDS

The IMS OLDS and SLDS should be saved in case IMS log analysis is required.

### About this task

- The ideal time frame:
  - From the time of the last IMS restart
  - Prior execution
- The moderate time frame:
  - 24 hours of IMS log records
- The minimum time frame:
  - Active IMS OLDS

## Manual intervention for dump creation

---

IMS produces SDUMPs for some internal errors without manual intervention. However, IMS Wait/Loop or partial loss-of-function conditions require manual intervention to produce an SVC dump.

IMS hangs can be caused by interaction with many address spaces, including those shown in the list below:

- IMS control region
- IMS DLI/SAS region
- DBRC region
- IRLM region
- CQS
- Operations Manager
- Resource Manager
- Structured Call Interface

- Troublesome IMS dependent regions
- CCTL regions
- ODBA
- IXGLOGRC
- z/OS Resource Recovery Services
- APPC
- VTAM®
- WLM
- TCPIP
- WebSphere® Application Server
- ESAF - Db2 for z/OS, IBM MQ, others
- Other regions
- Other IMSplex members with all their related regions

## Deciding when to create a memory dump

Because of the complex interactions between these address spaces, it is difficult to determine exactly where the source of the problem lies without a dump of the associated address spaces.

Omission of any interrelated address space adds to the possibility that the dump might not be sufficient to solve the problem.

The time that is required to produce the dump must be weighed against the possibility that there might not be sufficient data to solve the problem, adding to the possibility that the problem could recur.

## Creating IMS memory dumps

IMS SVC dumps can be requested using three different commands.

- z/OS SYS1.PARMLIB IEADMCxx
  - DUMP command parmlib member
- z/OS SYS1.PARMLIB IEASLPxx
  - SLIP command parmlib member
- z/OS DUMP command
- Customized JCL can be built and submitted

## IEADMCxx, z/OS SYS1.PARMLIB

You can use the IEADMCxx, member of the z/OS SYS1.PARMLIB to create an SVC dump.

### About this task

The following are characteristics of the IEADMCxx member of the z/OS SYS1.PARMLIB data set:

- DUMP command parmlib member
- Can be used to customize IMS memory dumps prior to error event.
- Simple operator interface.
- Create SYS1.PARMLIB members called IEADMCxx for each customized dump command.

### Related reference

[z/OS: Syntax for the IEADMCxx member of the z/OS SYS1.PARMLIB data set](#)

## IEADMCxx example for IMS

This example shows creating a SYS1.PARMLIB member called IEADMC11. DUMP parameters are given.

### About this task

Create a SYS1.PARMLIB member called IEADMC11 containing the following DUMP parameters:

```
JOBNAME=(j1,j2,j3,j4),SDATA=(CSA,PSA,RGN,SQA,SUM,TRT,GRSQ)
```

Where:

**j1**

IMS Control region job name.

**j2**

IMS DL/I region job name.

**j3**

DBRC region job name.

**j4**

IRLM region job name.

Create a second SYS1.PARMLIB member called IEADMC12 containing the following DUMP parameters:

```
JOBNAME=(j5,j6,j7),SDATA=(CSA,PSA,RGN,SQA,SUM,TRT)
```

Where:

**j5**

IMS CCTL region 1.

**j6**

IMS CCTL region 2.

**j7**

IMS CCTL region 3.

## IEADMCxx DUMP activation

When you request a dump from the IEADMC11 and IEADMC12 parmlib members, two dump data sets are created on the z/OS image from which the dump command was entered.

### About this task

To request a dump from the IEADMC11 and IEADMC12 parmlib members, enter the following z/OS command:

```
DUMP TITLE=(DUMP OF IMS and CCTL Regions ),PARMLIB=(I1,I2)
```

## IMS sysplex dump considerations

IMS produces SDUMPs for some internal errors without manual intervention. However, IMS Wait/Loop or partial loss-of-function conditions require manual intervention to produce an SVC dump. IMS hangs can be caused by interaction with many address spaces.

The following are considerations for IMS sysplex dumps:

- IMS sysplex implementations need to consider the possibility that a hang or problem on one IMSplex member might be due to a problem originating from another member.
- Problems such as IMS Wait/Loops or partial loss-of-function conditions which require manual intervention to produce an SVC dump, should include SVC dumps from other members of the IMSplex.
- Ensure that a dump is taken for all necessary address spaces on each system.

## Sysplex IEADMCxx example

This example shows how to create a SYS1.PARMLIB member containing various DUMP parameters.

Create a SYS1.PARMLIB member called IEADMCI1 containing the following DUMP parameters:

```
JOBNAME=(j1,j2,j3,j4),SDATA=(CSA,PSA,RGN,SQA,SUM,TRT,GRSQ),  
REMOTE=(SYSLIST=(*('j1','j2','j3','j4'),SDATA))
```

Where:

**j1**

IMS Control region job name.

**j2**

IMS DLI region job name.

**j3**

DBRC region job name.

**j4**

IRLM region job name.

Create a second SYS1.PARMLIB member called IEADMCI2 containing the following DUMP parameters:

```
JOBNAME=(j5,j6,j7),SDATA=(CSA,PSA,RGN,SQA,SUM,TRT,XESDATA),  
REMOTE=(SYSLIST=(*('j5','j6','j7'),SDATA))
```

Where:

**j6**

CCTL region 1.

**j7**

CCTL region 3.

**j8**

CCTL region 2.

**Note:** The XESDATA and REMOTE parameters are for use in sysplex environments.

## Sysplex IEADMCxx DUMP activation

Activating a sysplex dump using the z/OS command **DUMP** is shown.

### About this task

To request a dump from the IEADMCI1 and IEADMCI2 parmlib members, enter the following z/OS command:

```
DUMP TITLE=(IMS/CCTL SYSPLEX Dumps),PARMLIB=(I1,I2)
```

Two dump data sets are created on each z/OS image in the sysplex matching the REMOTE parameter specifications for the JOBNAMEs.



---

## Chapter 2. Collecting data about specific problems

Occasionally, there are problems in specific environments or certain problem types that require special handling.

### About this task

---

## Collecting data about a control region wait or hang

When an IMS control region waits or hangs, IMS can appear frozen or lose partial function. The most critical piece of information in diagnosing such problems is the z/OS SVC dump.

### About this task

**Recommendation:** Do not use the z/OS **MODIFY dump (F jobname,DUMP)** command as a source of IMS diagnostic information. This command adds unnecessary complexity to the dump while processing the modify abends.

Obtain a z/OS SVC dump by issuing this series of commands:

```
DUMP COMM=(dump title) R id JOBNAME=(j1,j2,j3,j4,j5,j6),  
SDATA=(CSA,PSA,RGN,SQA,SUM,TRT),END
```

In the previous example,

- j1**  
is the IMS CTL or DBCTL region job name
- j2**  
is the IMS DL/I region job name
- j3**  
is the suspicious IMS dependent region job name, if any
- j4**  
is the suspicious CCTL (CICS®) region name, if any
- j5**  
is the IRLM region job name (if IRLM DB locking is used)
- j6**  
is the DBRC region job name

Also, consider dumping related regions:

- IMS Control region
- IMS DLI/SAS region
- DBRC region
- IRLM region
- CQS
- Operations Manager
- Resource Manager
- Structured Call Interface
- Troublesome IMS dependent regions
- CCTL regions
- ODBA

- IXGLOGRC
- z/OS Resource Recovery Services
- APPC
- VTAM
- WLM
- TCPIP
- WebSphere Application Server
- ESAF - Db2 for z/OS, IBM MQ, others
- Other Regions
- Other IMSplex members with all their related regions

Most likely, a dump of the IMS CTL, DL/I, and a suspicious dependent region or CCTL region is sufficient to solve wait or hang problems. Occasionally, the DBRC and IRLM (if used for DB locking) regions are a factor, so include DBRC and IRLM.

If IMS is not completely stopped (for example, IMS commands can still be entered, BMPs are still processing, and some transactions still process), taking a second z/OS SVC dump will help differentiate normal IMS processing from the problem.

## Collecting data about a control region or DL/I region loop

Occasionally, there are problems in specific environments or certain problem types, that require special handling. If IMS can accept commands, you can set a trace.

### Procedure

1. If IMS can accept commands, use the following IMS command to set up the internal trace environment:

```
/TRA SET ON TABLE nnnn
```

where *nnnn* is the DISP, SCHD, DLI, LOCK or LATCH parameter. Each parameter must be entered in a separate /TRA command.

2. Set the z/OS system trace table size to 999K and turn on branch tracing by issuing the following command:

```
TRACE ST,999K,BR=ON
```

3. Obtain two z/OS SVC dumps of the CTL, DL/I, suspicious dependent region, or CCTL, DBRC, and IRLM regions. Taking a second z/OS SVC dump will help differentiate normal IMS processing from the problem. Obtain a z/OS SVC dump with this series of commands:

```
DUMP COMM=(dump title)
R id JOBNAM=(j1,j2,j3,j4,j5,j6),
SDATA=(CSA,PSA,RGN,SQA,SUM,TRT),END
```

In the previous example,

**j1**

is the IMS CTL or DBCTL region job name

**j2**

is the IMS DL/I region job name

**j3**

is the suspicious IMS dependent region job name, if any

**j4**

is the suspicious CCTL (CICS) region name, if any



**j5**

is the IRLM region job name (if IRLM DB locking is used)

**j6**

is the DBRC region job name

4. Reset the z/OS system trace table to its original settings.

## Results

**Note:** IMSplex partner dumps are probably not required for loop problems, unless they are also looping.

## Collecting data about an IMS dependent region wait or loop

Occasionally, there are problems in specific environments or certain problem types, that require special handling. If IMS can accept commands, you can set a trace.

### About this task

If the dependent region appears to be looping, follow these steps:

### Procedure

1. If IMS can accept commands, use the following IMS command to set up the internal trace environment:

```
/TRA SET ON TABLE nnnn
```

where *nnnn*= can be DISP, SCHD, DLI, LOCK, or LATCH. Each must be entered separately.

2. Set the z/OS system trace table size to 999K and turn on branch tracing with this command:

```
TRACE ST,999K,BR=ON
```

3. If the problem is a wait, obtain two z/OS SVC dumps of the CTL, DL/I, suspicious dependent region, or CCTL, DBRC, and IRLM regions. If the problem is a loop, obtain two z/OS SVC dumps of the CTL, DL/I, suspicious dependent region, or CCTL, DBRC, and IRLM regions. Obtaining a second z/OS SVC dump will help differentiate normal IMS processing from the problem. Obtain a z/OS SVC dump with this series of commands:

```
DUMP COMM=(dump title)
R id JOBNAME=(j1,j2,j3,j4,j5),
SDATA=(CSA,PSA,RGN,SQA,SUM,TRT),END
```

**j1**

is the IMS CTL or DBCTL region job name

**j2**

is the IMS DL/I region job name

**j3**

is the suspicious IMS dependent region job name, if any

**j4**

is the IRLM region job name (if IRLM DB locking is used)

**j5**

is the DBRC region job name

In the previous example,

4. Reset the z/OS system trace table to its original settings.

## Results

**Note:** IMSplex partner dumps are probably not required for loop problems, unless they are also looping.

## Formatting a BPE trace entry

You can format a BPE trace entry by using either the Interactive Problem Control System (IPCS) or a batch job.

### About this task

Before you begin to format BPE trace entries, ensure that the user ID that you use to format and print the external BPE trace records is authorized by RACF® to access the external trace data sets.

To format BPE trace entries by using the IPCS:

### Procedure

1. Select option 0 from the IPCS Primary Option menu to specify the generation data group (GDG) to analyze.
2. Specify the data set name for the GDG in the **Source** field of the IPCS Default Values menu. For example, specify: DSN=IMS.SDFSRESL,DISP=SHR
3. Select option 2.6 from the IPCS Primary Option menu to display a list of the dump component analysis tools.
4. Select option DFSAAMP from the IPCS MVS Dump Component Data Analysis menu to display options for the IMS Dump Formatter.
5. Select option 6 from the IMS Dump Formatting Primary menu to display formatting for other IMS components, such as BPE.
6. Select one of the options from the IMS Component Selection Dump Formatting menu to display formatting options for an IMS component (BPE, CQS, DBRC, ODBM, OM, REPO, RM, SCI, or IMS Connect). For example, select option **B** to display general BPE formatting options.
7. Select option 4 from the component subsystem dump formatting menu, for example the IMS BPE Subsystem Dump Formatting menu, to display the options for external trace formatting.
8. Specify the various formatting options for the external trace data from the subsystem's external trace formatting menu, for example the BPE External Trace Formatting menu.

## Results

You can also format BPE external trace records by using a batch job. The following figure shows sample JCL for formatting the BPE external trace records from a batch job.

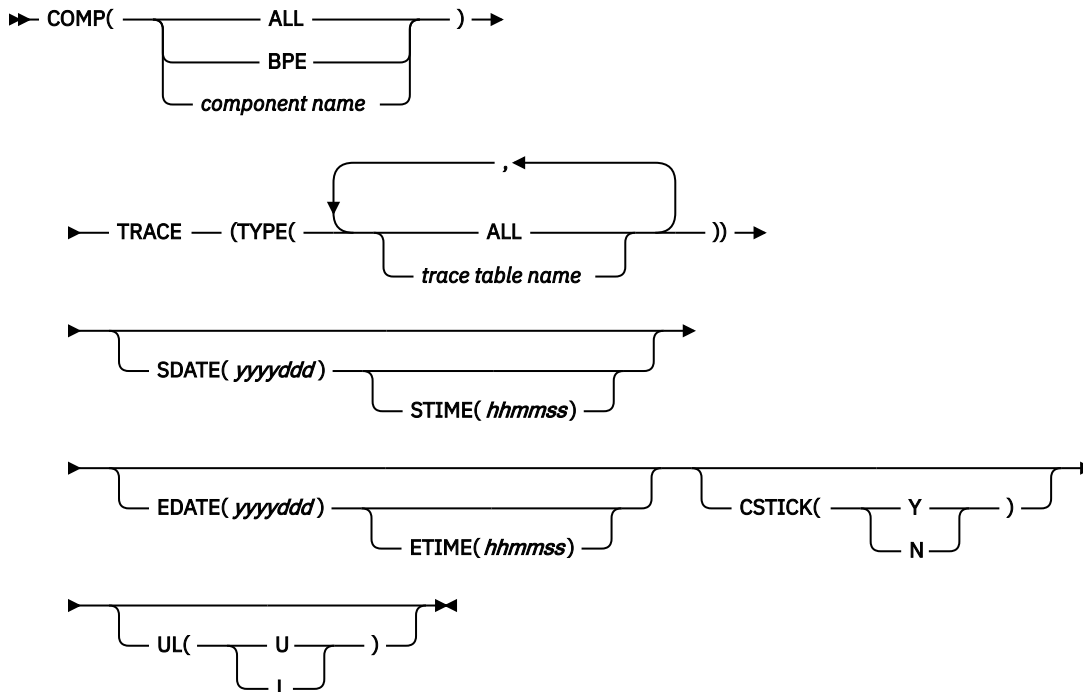
*Figure 1. Batch JCL for formatting BPE external trace records*

```
//BPEEXTPR JOB ...
//*****
//* Job to print all traces in a BPE external trace data set. */
//*****
//JOB LIB DD DSN=IMS.SDFSRESL,DISP=SHR
//IPCS DMP EXEC PGM=IKJEFT01,REGION=8M
//SYSTSPRT DD SYSOUT=*
//IPCS PRNT DD SYSOUT=*
//INDEX DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//IPCS PARM DD DSN=USER.PARMLIB,DISP=SHR
// DD DSN=SYS1.PARMLIB,DISP=SHR
//SYSTSIN DD *
DELETE 'SYS1.IPCSDDIR'
ALLOC SP(1) TRACK VOL(333333)
DEFINE CLUSTER (NAME('SYS1.IPCSDDIR') +
  VOLUMES(333333)) +
  INDEX(NAME('SYS1.IPCSDDIR.DDX')) +
  TRACKS(1 1)) +
  DATA( NAME('SYS1.IPCSDDIR.DDD')) +
```

```

CYLINDERS(1 1) BUFSP(X'10000') KEYS(128 0) CISZ(X'1000')
IPCSDDIR 'SYS1.IPCSDDIR'
ALLOC FILE(IPCSDDIR) +
  DA('SYS1.IPCSDDIR') +
  REUSE SHR
ALLOC FILE(INFILE) +
  DA('BPEEXTRC.GDG01.G0001V00') +
  REUSE SHR
IPCS NOPARM
SETDEF DSN('BPEEXTRC.GDG01.G0001V00') +
  NOPROBLEM PRINT NOTERMINAL
VERBX BPETRFM0 +
  'COMP(HWS) +
  TRACE(TYPE(RCTR)) +
  SDATE(2008080) STIME(110909) +
  EDATE(2008090) ETIME(140000) +
  UL(L) +
  CSTCK(Y)'
END
DELETE 'SYS1.IPCSDDIR'
/*

```



## BPETRFM0 Parameter Keywords

### COMP()

Specify 'ALL' to format all trace records, 'BPE' to format BPE trace records, or a specific component name (BPE, CQS, DBRC, ODBM, OM, REPO, RM, SCI, or HWS) to format trace records for only that component.

### TYPE()

Specify 'ALL' to format trace records for all trace tables or specify a specific trace table name to format records only for that trace table type.

### SDATE()

Specify a starting date for the trace entries in Julian format (yyyyddd). Trace entries with a store clock (STCK) value prior to the specified date are filtered and not printed.

### STIME()

Specify a starting time for the trace entries in 24 hour format (hhmmss). Trace entries with an STCK value prior to the specified time are filtered and not printed. SDATE() is required with STIME().

### EDATE()

Specify an ending date for the trace entries in Julian format (yyyyddd). Trace entries with an STCK value after the specified date are filtered and not printed.

## ETIME()

Specify an ending time for the trace entries in 24 hour format (hhmmss). Trace entries with an STCK value after the specified time are filtered and not printed. EDATE() is required with ETIME().

## CSTCK()

Specify 'Y' to have the value for each trace entry printed in JDAYTIME format (DDD HHMMSS.thmiju).

## UL()

Specify 'L' if the specified filtering time is based on the local time in the trace record or 'U' if the specified filtering time is based on UTC.

## Example

### BPETRFM0 formatted BPE external trace record header output

```
-----
--- BPE  ERRV Trace Table ---
-----

ETHD: 00000000
+0000 LL..... 4010      ZZ..... 0000      TYPE.... 01      SUBTYPE.. 02      VERSION.. 0001
RESERVED. 00000000
+000C NAME..... ERRV      LENGTH... 00000080  UDATALEN. 00000000  TDATALEN. 00003F80  NUMPGS... 0008
ENTLEN... 0020
+0020 ENTSKPD.. 00000000  BVERS.... 010700  RESERVED. 00      UTYPE.... HWS      UVERS.... 0B0100
RESERVED. 00
+0030 USYSNAME.      UTRMOD... 00000000  00000000      FLG1..... 48      FLG2..... 00
LEVEL.... 04
+0043 IDX..... CB      CYCLECT.. 00000000  OFFSET... 00000000  BYTELOST. 00000000  LDT0..... FFFFA21F  68400000
+0058 LS0..... 00000000  00000000      STCK..... C233AA77  1692DF60      RESERVED. 00000000
NEXT..... 00000000
+0070 FIRST.... 0BC01C60  TOKVAL... 00000000  ID..... ETHD END
Flag analysis for ethd_flg1:
  ethd_f1_system (40x) - Trace table is a BPE system table
  ethd_f1_var (08x) - Trace type creates variable length entries
Trace level for this table is: HIGH

ETSF: 00004000
+0000 STCK.... C233AA7D  AE68F180      SEQNUM... 00000000  00000004

Variable trace table entries follow with oldest entry printed first
```

### Formatted BPE external trace variable trace entry

```
ERRV trace table entry:
  Code:      SSRV      Record #: 1
  Subcode: *BPEDYA10 ERROR  Continue: 0
  TimeStmp: 096 183544.830765

TTVE: 00000000

  Variable entry prefix:
  LL..... 1FC0      REC#..... 00000001
  ZZ..... 0000      CONT..... 00000000
  VLEN..... 00001F90

TTE: 00000010

  Variable entry fixed section:
  CODE..... C6      WD02..... 00000002
  SCDE..... 1B      WD03..... 00000003
  B1B2..... C1C2      WD04..... 00000004
  WD01..... 00000001  WD05..... 00000005
  STCK..... C233AA77  1692DF60
  Data +00: |F.AB.....|  Data +10: |.....B....k.-|

Data: 00000030  Length: 8080

  Variable entry variable section:
Offset  0      4      8      C      0      4      8      C      EBCDIC Data
-----
+000000  81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 |
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa|
  LINES +000020 TO +001F7F SAME AS ABOVE
+001F80  81818181 81818181 81818181 81818181 81818181 81818181 81818181 |
aaaaaaaaaaaaaaaa|
```

### Related concepts

[“IMS Connect traces” on page 429](#)

You can trace two types of information about IMS Connect: information about the messages that are processed by IMS Connect and information about the IMS Connect subsystem.

### Related tasks

[“Formatting the trace data from an external trace of IMS Connect” on page 434](#)

The following example shows the JCL that can be used to format the trace data from an external trace of IMS Connect.

## Collecting data about DBRC-related problems

---

DBRC related problems can cause a variety of symptoms, including waits and loops. If you need to create the problem again, copies of the RECON listing, before and after the problem occurred, are most useful.

### About this task

To diagnose a DBRC related problem, you need the following information:

- A listing of the DBRC RECON data sets for the time that is as close as possible to the time of the failure.
  - Use the DBRC LIST . RECON command to obtain the listing.
- A subsystem listing if you cannot obtain a RECON listing because of its size.
  - Use the DBRC LIST . SUBSYS ALL command to obtain a subsystem listing.
- If recreates are possible, obtain the before and after copies of the RECON data sets.
- Use the D GRS , CONTENTION command on each system that shares the RECON data set to determine if the data set is held at the exclusion of other waiters. If so, dump the owning address space by issuing the following command:

```
DUMP COMM=(dump title)
R nn, JOBNAME=(j1), SDATA=(CSA, PSA, RGN, SQA, SUM, TRT, GRSQ), END
```

### Related reference

[z/OS: VSAM Record-Level Sharing \(RLS\) diagnostic aids](#)

## DBRC security override

To set a RCNQUAL value in the RECON that can be used to override DBRC security for copies of RECON data sets, use the DBRC command **CHANGE . RECON** with the CMDAUTH keyword.

### About this task

You can then use the RECON data set for further testing purposes or to send on to an IBM Software Support representative without requiring a zap of the RECON, an exit routine, or requiring that you find an authorized individual to change the authorization level.

## Collecting data about DBCTL-related problems

---

DBCTL-related problems can originate from either the CCTL region or one of the IMS regions (CTL, DL/I, DBRC, or IRLM), so it is important to obtain dumps that relate to all these regions.

### Procedure

1. Issue the following IMS commands (because they include region ID numbers and recovery tokens in their various display output):

```
/DISPLAY ACTIVE
```

```
/DISPLAY CCTL
```

The information that is returned by these commands greatly increases the accuracy and speed that is required to diagnose the problem. The DISPLAY ACTIVE command provides the reasons for waits and region numbers. The DISPLAY CCTL command provides recovery tokens and region numbers. Save the IMS console output.

2. Set the AP portion of the CICS trace to level 1-2. Save this output.
3. Set the FILE CONTROL portion of the CICS trace to level 1-2. Save this output.
4. Obtain the necessary z/OS SVC DUMP of the IMS regions by issuing this series of commands:

```
DUMP COMM=(dump title)
R id JOBNAME=(j1,j2,j3,j4,j5,j6),
SDATA=(CSA,PSA,RGN,SQA,SUM,TRT),END
```

In the previous example,

**j1**

is the IMS CTL or DBCTL region job name

**j2**

is the IMS DL/I region job name

**j3**

is the suspicious IMS dependent region job name, if any

**j4**

is the suspicious CCTL (CICS) region name, if any

**j5**

is the IRLM region job name (if IRLM DB locking is used)

**j6**

is the DBRC region job name

5. Save the IMS online log data set that was active during the failure.

## Collecting data about VTAM-related DC problems

IMS DC-related problems are associated mainly with VTAM or OTMA. Use these guidelines to collect diagnostic information about VTAM when you experience a problem.

### About this task

VTAM dumps are often required to help diagnose problems, but are infrequently obtained by operations personnel. IMS NODE traces, VTAM BUFFER traces, and VTAM INTERNAL traces are often required, in conjunction with the IMS region dumps and VTAM dumps, to solve DC problems.

The IMS log tapes contain much of the transaction data that flows through IMS. This transaction data includes the following IMS records:

- TYPE01
- TYPE03 (MSG queue entries)
- TYPE11 through TYPE16 (SPAs, DIALs, SIGN)

To start the recreate attempt after issuing an IMS /SWITCH OLDS command to have the related data placed on a new OLDS:

### Procedure

1. Issue the following IMS command and save the IMS console output:

```
/DIS NODE nodename
```

2. Turn on the IMS NODE trace by issuing the following command.

```
/TRA SET ON NODE nodename
```

Data is captured in the IMS TYPE6701 log record. Save the IMS online log data set input to the IMS utility programs DFSERA10 and DFSERA30.

3. (Optional) Turn on the VTAM Buffer Trace and VTAM Internal Trace to complement the IMS NODE trace by issuing this series of commands:

```
F NET,TRACE,TYPE=BUF,ID=nodename  
F NET,TRACE,TYPE=VTAM,MODE=EXT,  
  OPT=(API,PIU,MSG)
```

To capture these trace entries, GTF must be active with the USR option specified.

4. Obtain a z/OS dump of the IMS regions by issuing this series of commands:

```
DUMP COMM=(dump title)  
R id JOBNAME=(j1,j2,j3,j4,j5,j6),  
SDATA=(CSA,PSA,RGN,SQA,SUM,TRT),END
```

***j1***

IMS CTL or DBCTL region job name

***j2***

IMS DL/I region job name

***j3***

Suspicious IMS dependent region job name, if any

***j4***

Suspicious CCTL (CICS) region name, if any

***j5***

IRLM region job name (if IRLM DB locking is used)

***j6***

DBRC region job name

In the previous example,

5. Obtain a dump of the VTAM address space by issuing this series of commands:

```
DUMP COMM=(dump title)  
R id JOBNAME=(vtam jobname),  
SDATA=(CSA,PSA,RGN,SQA,SUM,TRT),END
```

6. Save the IMS log tapes created during the error period.

## Collecting data about OTMA-related DC problems

IMS DC-related problems are frequently related to either OTMA or VTAM. Use these guidelines to collect diagnostic information about OTMA when you experience a problem.

### About this task

There are four main tools that you can use to diagnose problems with OTMA message traffic:

- OTMT table trace
- /DISPLAY commands
- OTMA tmember and tpipe traces
- IMS log records

## Procedure

1. Enable the OTMT trace table.

Issue the following command:

```
/TRA SET ON TABLE OTMT OPTION LOG VOLUME HIGH
```

This trace data is used by IBM Software Support to diagnose OTMA problems. You can also specify OTMADB=Y in the DFSPBxxx member of the IMS.PROCLIB data set to get more trace data. However, using OTMADB=Y causes a large amount of WTO output to be written to the MVS console, and the setting can be removed only by recycling IMS.

**Recommendation:** Do not set OTMADB=Y unless instructed to do so by IBM Software Support.

2. Get the current status of OTMA clients and servers.

Issue the following command:

```
/DISPLAY OTMA
```

Save the console output.

3. Enable the OTMA tmember trace.

Issue the following command:

```
/TRA SET ON TMEMBER XXXXXXXX
```

Where XXXXXXXX is the tmember name. If you cannot identify a specific OTMA tmember, specify ALL for the tmember name to enable the trace for all members.

4. If you know that the problem is associated with a specific OTMA tmember, you can get the current status for that tmember.

Issue the following command:

```
/DISPLAY TMEMBER XXXXXXXX TPIPE ALL
```

Where XXXXXXXX is the tmember name. Save the console output.

5. If you know that the problem is associated with a specific OTMA tpipe, you can enable tracing at the tpipe level.

Issue the following command:

```
/TRA SET ON TMEMBER XXXXXXXX TPIPE YYYYYYYY
```

Where XXXXXXXX is the tmember name and YYYYYYYY is the tpipe name.

6. For CM1 and CM0 problems that are associated with a specific PSB, enable DL/I tracing.

Issue the following command:

```
/TRA SET ON PGM pppppppp
```

Where pppppppp is the PSB name.

## What to do next

After you enable the required traces, preserve the X'67D0' and X'6701' log records and the console output from the /DISPLAY commands.



## Collecting data about APPC-related DC problems

APPC problems that originate from IMS dependent regions and that make calls explicitly rely heavily on the dependent region dumps.

### About this task

To diagnose an APPC-related IMS problem:

### Procedure

1. Turn on the IMS LUMI trace for the external trace data set by issuing the following IMS /TRACE commands:

```
/TRACE SET ON TABLE LUMI OPTION LOG
```

The LOG option can be set up to cause the output to be sent to the external trace data set with this /TRACE command:

```
/TRACE SET ON LUNAME XXXXXXXX INPUT
```

```
TRACE SET ON LUNAME XXXXXXXX OUTPUT
```

where XXXXXXXX is the partner LU

2. Turn on the VTAM buffer trace and VTAM internal trace to complement the IMS LUMI trace by issuing the following commands:

```
F NET,TRACE,TYPE=BUF,ID=luname
```

```
F NET,TRACE,TYPE=VTAM,MODE=EXT,  
OPT=(API,PIU,MSG)F
```

To capture these trace entries, GTF must be active with the USR option specified

3. Turn on the program trace to trace TPPCB DL/I calls, so that the APPC component trace can send its trace buffers to a SYS1.DUMP data set when it stops. Turn on the program trace by issuing the following command:

```
/TRACE SET ON PROGRAM pppppppp
```

where pppppppp is the program name of the application.

4. Turn on the z/OS APPC component trace by issuing the following command:

```
TRACE CT,ON,200M,COMP=SYSAPPC
```

**Note:** If the command above fails, omit the ON keyword.

5. Start the recreate attempt after issuing an IMS /SWITCH OLDS command to have related data placed in a new OLDS. Save the IMS log tapes that are created during the error period. IMS log records are not as useful for explicit APPC applications as they are for implicit APPC applications because little information is logged about explicit APPC applications.
6. Reply to the z/OS outstanding reply with the following response:

```
nn,OPTIONS=(GLOBAL),END
```

7. When the problem has been recreated, stop the component trace with this command:

```
TRACE CT,OFF,COMP SYSAPPC
```

You can use the following IPCS commands to format the trace:

- For one-line entries:

```
CTRACE COMP SYSAPPC SHORT
```

- Summary of each entry:

```
CTRACE COMP SYSAPPC FULL
```

8. Obtain a z/OS SVC dump of the IMS regions with this series of commands:

```
DUMP COMM=(dump title)
R id JOBNAME=(j1,j2,j3,j4,j5,j6),
SDATA=(CSA,PSA,RGN,SQA,SUM,TRT),END
```

In the previous example,

**j1**

is the IMS CTL or DBCTL region job name

**j2**

is the IMS DL/I region job name

**j3**

is the suspicious IMS dependent region job name, if any

**j4**

is the suspicious CCTL (CICS) region name, if any

**j5**

is the IRLM region job name (if IRLM DB locking is used)

**j6**

is the DBRC region job name

9. Obtain a dump of the APPC, APPC Scheduler, and VTAM address spaces with this series of commands:

```
DUMP COMM=(dump title)
R id JOBNAME=(j1,j2,j3),SDATA=(CSA,PSA,RGN,SQA,SUM,TRT),END
```

In the previous example,

**j1**

is the APPC job name

**j2**

is the APPC scheduler job name

**j3**

is the VTAM job name

## Collecting data about IMS Connect problems

If a problem occurs during IMS Connect execution, you need to collect logs, data sets, and dumps to determine the source of the problem.

### About this task

For almost all problems that might occur during IMS Connect execution, you can take the following action to collect the data that is needed to diagnose the problem:

### Procedure

- Create a dump of the IMS Connect address space.
- Save the IMS Connect joblog, which contains only IMS Connect messages. IMS Connect messages are also written to the MVS system log (SYSLOG), but the IMS Connect messages can be difficult to find in the SYSLOG, because of the other z/OS subsystem messages that the SYSLOG contains.

- Save the MVS SYSLOG. The SYSLOG is useful for seeing what messages other subsystems issued before and after the IMS Connect messages were issued, because the MVS SYSLOG contains messages from all z/OS subsystems, including TCP/IP, IMS, and IMS Connect.
- Turn on the IMS Connect Recorder Trace facility and attempt to re-create the problem.

## What to do next

In addition to collecting data about IMS Connect, you might need to collect data about other components or products if they appear to be related to the problem in some way. For example:

- If the problem appears to be network related, you might also need to initiate a TCP/IP packet trace to trace the IP packets flowing to and from a TCP/IP stack on the z/OS Communications Server. For more information about initiating packet traces, see *z/OS Communications Server IP Diagnosis Guide*.
- If the problem is related to IMS Connect support for an IMS TM system, where the connections to IMS are through OTMA, you might also need to collect data about the IMS system.
- If the problem is related to IMS Connect support for an IMS DB system, where the connections to IMS are through an instance of the Open Database Manager (ODBM), which runs in its own address space as a component of the IMS Common Service Layer (CSL), you might also need to collect data about both the IMS system and any ODBM instance that might be associated with the problem.
- If the problem is related to two-phase-commit processing, you might also need to collect data related to z/OS Resource Recovery Services.

## Related tasks

[“Collecting data about CSL-related problems” on page 23](#)

The Common Service Layer address spaces, Open Database Manager (ODBM), Operations Manager, Structured Call Interface, and Resource Manager, produce SDUMPs for internal errors. The CSL dumps are in the SYS1.DUMP data sets.

[“Collecting data about z/OS Resource Recovery Services problems” on page 28](#)

RRS provides a system resource recovery platform such that applications that run on z/OS can have access to local and distributed resources and have system-coordinated recovery management of these resources.

[“IMS Connect service aids” on page 427](#)

The service aids for IMS Connect include the IMS Connect Dump Formatter and trace options.

## Related reference

[“Standard IMS diagnostic information” on page 1](#)

If you contact IBM Software Support for assistance with a problem, you might be asked to collect a standard set of logs, data sets, and dumps that can help them determine the source of your problem. Collecting this information before you contact IBM Software Support will shorten the amount of time required to resolve your problem.

## Collecting data about ISC TCP/IP link problems

If a problem occurs on an ISC link that uses TCP/IP, you might need to collect diagnostic information from IMS, the Structure Call Interface (SCI) component of the IMS Common Service Layer (CSL), IMS Connect, TCP/IP, and IBM CICS Transaction Server for z/OS.

## Before you begin

**Prerequisite:** Review error messages that were issued by IMS, the SCI, IMS Connect, and CICS around the time of the error to determine at which point in the ISC link the error might have occurred. If you can determine that any components are unrelated to the error, you might not need to collect information about that component.

## About this task

To collect the data that is needed to diagnose a problem on an ISC link that uses TCP/IP:

## Procedure

- For IMS, the information that you can collect includes:
  - The IMS system console sheet
  - The IMS job log
  - The z/OS system log (SYSLOG)
  - A dump of the IMS control region
  - A DC trace
- For SCI, the information that you can collect includes:
  - An SCI region dump
  - The z/OS SYSLOG from the logical partition in which SCI is running.
  - A z/OS SVC dump of the CSL address spaces
- For IMS Connect, the information that you can collect includes:
  - A dump of the IMS Connect address space
  - The IMS Connect job log
  - The z/OS SYSLOG
  - Output from the IMS Connect Recorder Trace facility
- If the problem appears to be network related, you might also need to initiate a TCP/IP packet trace to trace the IP packets flowing to and from a TCP/IP stack on the z/OS Communications Server. For more information about initiating packet traces, see *z/OS Communications Server IP Diagnosis Guide*.
- For CICS, refer to the CICS documentation for current information about collecting diagnostic information. Information that might be helpful to collect includes:
  - The CICS system console sheet
  - The CICS job log
  - The CICS message log including any DFHISnnnn messages
  - A z/OS system dump taken at the point of failure
  - A CICS internal trace

## Related tasks

[“DC trace” on page 279](#)

The data communication (DC) trace enables you to obtain information about the program flow within the communications analyzer and between the analyzer and the device dependent modules (DDMs).

[“Collecting data about IMS Connect problems” on page 20](#)

If a problem occurs during IMS Connect execution, you need to collect logs, data sets, and dumps to determine the source of the problem.

[“Collecting data about CSL-related problems” on page 23](#)

The Common Service Layer address spaces, Open Database Manager (ODBM), Operations Manager, Structured Call Interface, and Resource Manager, produce SDUMPs for internal errors. The CSL dumps are in the SYS1.DUMP data sets.

## Related information

[Collect troubleshooting data \(MustGather\) for CICS products](#)

[CICS troubleshooting and support](#)

## Collecting data about CQS-related problems

---

CQS problems can appear in various ways and, like the IMS control region, they can manifest themselves in the form of WAITs, HANGs, LOOPs, or some other type of internal error that results in an SDUMP being taken.

### About this task

These dumps are in the SYS1.DUMP data sets. CQS can also produce LOGREC data set entries for these types of errors.

If an isolated event type within CQS encounters an error, then IBM Software Support might request additional CQS-trace level settings for the various trace types.

For a CQS WAIT problem, one or more inflight dumps might be required. Multiple dumps might need to be taken if the problem is a LOOP. If a structure rebuild or structure checkpoint related problem occurs, you will also need to dump the CQS address spaces for any CQS associated with the given structure, and save the associated SRDS (structure recovery data set) for the CQS structure checkpoints and CQS system checkpoints.

## Collecting data about CSL-related problems

---

The Common Service Layer address spaces, Open Database Manager (ODBM), Operations Manager, Structured Call Interface, and Resource Manager, produce SDUMPs for internal errors. The CSL dumps are in the SYS1.DUMP data sets.

### About this task

You might need to collect one or more of the following types of information to diagnose CSL related problems:

#### **SYSLOG**

To determine the sequence of events, collect the SYSLOG from every logical partition (LPAR) where a CSL member resides. CSL address spaces issue messages that begin with "CSL"

- ODBM messages - CSLDxxxx
- OM messages - CSLOxxxx
- RM messages - CSLRxxxx
- SCI messages - CSLSxxxx
- CSL common messages - CSLZxxxx

#### **QUERY IMSPLEX SHOW(ALL) command output**

Issue the QUERY IMSPLEX command to display the members of the IMSplex and their status.

If there are problems accessing OM or RM services, verify that at least one OM or RM is active in the IMSplex and that an active SCI resides on every LPAR where a CSL address space resides.

#### **Obtain z/OS SVC dumps**

Obtain a z/OS SVC dump of the CSL address spaces that appear to have a problem, are waiting, or are looping. CSL dumps contain the CSL traces, which can be very useful for diagnosing CSL related problems. Dump all of the CSL address spaces that appear to have a problem with the following series of commands:

```
DUMP COMM=(dump title)
R id JOBNAME=(odbm1,om1,rm1,sci1)
SDATA=(CSA,PSA,RGN,SQA,SUM,TRT),END
```

In the previous example:

#### **odbm1**

An ODBM address space.

**om1**

An OM address space.

**rm1**

An RM address space.

**sci1**

An SCI address space.

For some CSL problems, IBM Software Support might request additional trace level settings for the various trace types.

**Related tasks**

[“Collecting data about IMS Connect problems” on page 20](#)

If a problem occurs during IMS Connect execution, you need to collect logs, data sets, and dumps to determine the source of the problem.

**Related reference**

[“CSL - Common Service Layer service aids” on page 151](#)

Common Service Layer (CSL) and Resource Manager (RM) trace records can help you analyze problems in CSL.

## Repository Server audit log records

---

If AUDIT=YES is specified in the FRPCFG member of the IMS PROCLIB data set, the Repository Server (RS) writes to the z/OS logger audit log stream name specified in the AUDIT\_LOG= parameter.

Maintaining the RS audit log is optional.

The RS audit log contains information about selected events that occur during server execution.

The audit access rule identifies the information that can be written to the audit log. The default access rule can be specified in the FRPCFG member of the IMS PROCLIB data set using the AUDIT\_DEFAULT parameter. This specification is applicable to all IMS repositories managed by the RS. The value for the AUDIT\_DEFAULT parameter can be overridden by setting the AUDITACCESS parameter in the CSLRIxxx member of the IMS PROCLIB data set.

If an audit access rule is already set for a given repository type, setting it again replaces the rule. Setting AUDITACCESS=DEFAULT removes the audit rule for the given repository type. The audit rule is then set to the default access rule specified in the FRPCFG member.

Audit log entries can be triggered in the audit log by the following events:

- RS and repository events
- ADMIN and CONTROL requests
- Registration, connection, and UOW events
- Events relating to repository members during a client member session:
  - Access failure due to a security restriction
  - Update
  - Read
  - System-level read

Refer to the FRPLGREC macro that maps the RS audit log records to view the event types and subtypes that are written and the data in each log record type.

**Related concepts**

[Overview of the IMSRSC repository \(System Definition\)](#)

**Related reference**

[“Log records” on page 469](#)

To diagnose some problems, you need to examine the content of log records to determine what was going on in the system before the problem occurred. By knowing the layout of the log records, you can set up a DFSERA10 job that will produce the specific log records that you need to examine.

[FRPCFG member of the IMS PROCLIB data set \(System Definition\)](#)

[CSLRxxxx member of the IMS PROCLIB data set \(System Definition\)](#)

## Managing Repository Server audit log records

Issuance of Repository Server (RS) audit log records can be controlled by specifying parameters in the FRPCFG and CSLRxxxx members of the IMS PROCLIB data set and through the **F reposervername,AUDIT** RS command.

Auditing can be enabled for the RS to connect to the z/OS log stream by specifying AUDIT=Y in the FRPCFG member of the IMS PROCLIB data set, but no audit records are written if AUDIT\_LEVEL=NONE is specified.

Auditing can be restarted after an error during RS initialization if AUDIT\_FAIL=CONTINUE is specified.

You can also dynamically change the value specified for the AUDIT\_LEVEL parameter in the FRPCFG member by issuing the RS command **F reposervername,AUDIT**.

You can control which types of events and member access are audited during client member sessions by specifying the AUDIT\_DEFAULT= parameter in the FRPCFG member or the AUDITACCESS= parameter in the CSLRxxxx member. Audit access can also be modified through the **UPDATE RM** command. An audit access rule applies to all members of a specified repository type.

A read request from an authorized client that is done as a part of the update request is identified as a *system read* request. With an audit access rule of READ, system read requests do not cause a read audit record to be generated. With an audit access rule of SYSTEMREAD, all read requests, including system read requests, are audited.

### Related concepts

[Overview of the IMSRSC repository \(System Definition\)](#)

### Related reference

[FRPCFG member of the IMS PROCLIB data set \(System Definition\)](#)

[CSLRxxxx member of the IMS PROCLIB data set \(System Definition\)](#)

[F reposervername,AUDIT \(Commands\)](#)

## How the Repository Server handles z/OS logger errors

The Repository Server (RS) uses the ENF48 exit to detect when system logger resources become unavailable or available.

This process enables the RS to resume logging when transient logger errors are resolved.

If AUDIT\_FAIL=CONTINUE is specified in the FRPCFG member of the IMS PROCLIB data set, during RS startup, the RS starts after an error and no audit log is enabled.

If AUDIT\_FAIL=ABORT is specified, during RS startup, the RS terminates on an error.

During a client request, if AUDIT\_FAIL=CONTINUE is specified, the RS continues processing as if the audit log is not enabled.

If AUDIT\_FAIL=ABORT is specified, the client request is rejected.

### Related concepts

[Overview of the IMSRSC repository \(System Definition\)](#)

### Related reference

[FRPCFG member of the IMS PROCLIB data set \(System Definition\)](#)

## Printing Repository Server audit log records

To print the audit log records from the Repository Server (RS) audit log stream on the z/OS system logger, use the IMS File Select and Formatting Print utility (DFSERA10) with exit routine CSLRERA3.

### DD statements

The DD statements for printing RS audit log records are:

#### STEPLIB DSN=

Points to IMS.SDFSRESL, which contains the IMS File Select and Formatting Print utility, DFSERA10.

#### SYSPRINT

Describes the output data set to which the formatted print records and control messages are to be directed. SYSPRINT is usually defined as SYSOUT=A. DCB should not be specified on the SYSPRINT DD.

#### SYSUT1 DSN=

Points to the RS audit log stream name that is specified on the AUDIT\_LOG= parameter in the FRPCFG member of the IMS PROCLIB data set.

### Control statements

The control statements for printing the RS audit log records are:

#### H=

Specifies the number of log records to print. H=EOF specifies to print all of the log records.

#### EXITR=

Identifies the Resource Manager (RM) audit log record exit routine that is called to format each log record. EXITR=CSLRERA3 specifies to print the records in a memory dump format, including the record type and time-stamp information for each record.

### Limiting log data to a specific time range

To limit the log data to a specific time range, use the FROM and TO parameters on the SUBSYS statement, as shown in the following example. This DD card prints log records from 11:00 to 12:00 on day 42 of the year 2010:

```
//SYSUT1    DD    DSN=SYSLOG.REPO.AUDIT.LOG
//          SUBSYS=(LOGR,IXGSEXIT,
//          'FROM=(2010/042,11:00:00),TO=(2010/042,12:00:00)',
//          DCB=(BLKSIZE=32760)
```

Dates and times are specified in Greenwich mean time (GMT). The seconds field in the time value is optional. To use local dates and times, add the LOCAL keyword, as shown in the following example:

```
//SYSUT1    DD    DSN=SYSLOG.REPO.AUDIT.LOG,
//          SUBSYS=(LOGR,IXGSEXIT,
//          'FROM=(2007/042,11:00:00),TO=(2007/042,12:00:00),LOCAL',
//          DCB=(BLKSIZE=32760)
```

### Sample

The following JCL shows what is required to print RS log records from the RS audit log stream:

```
//CSLERA1    JOB    MSGLEVEL=1,MSGCLASS=A,CLASS=K
//STEP1      EXEC   PGM=DFSERA10
//STEPLIB    DD     DISP=SHR,DSN=IMS.SDFSRESL
//SYSPRINT   DD     SYSOUT=A
//SYSUT1     DD     DSN=SYSLOG.REPO.AUDIT.LOG,
//          SUBSYS=(LOGR,IXGSEXIT),
//          DCB=(BLKSIZE=32760)
//SYSIN      DD
CONTROL      CNTL   H=EOF
OPTION       PRINT  EXITR=CSLRERA3
END
//
```



## Related concepts

[Overview of the IMSRSC repository \(System Definition\)](#)

## Related reference

[File Select and Formatting Print utility \(DFSERA10\) \(System Utilities\)](#)

# Collecting data about ESAF and DB2 ESS interface problems

ESAF (External Subsystem Attach Facility) interface problems can be diagnosed by using the external trace data set. You can use the IMS /TRACE command to direct and control the tracing of internal IMS events.

## About this task

The IMS external subsystem (ESS) trace impacts performance. Activate the trace only when you notice a problem or if you need to re-create a problem. IBM software support may ask you to provide dumps of IMS CTL, z/OS SVC and the IMS online log data set.

To document problems that involve the Db2 for z/OS ESS interface:

## Procedure

1. Use the following **TRACE** command to turn on the IMS ESS trace and to direct its output to the external trace data set:

```
/TRACE SET ON TABLE SUBS OPTION LOG
```

2. Obtain dumps of the IMS CTL and involved dependent regions, before and after the failure, by issuing this series of commands:

```
DUMP COMM=(dump title)
R id JOBNAME=(j1,j2,j3,j4,j5),
SDATA=(CSA,PSA,RGN,SQA,SUM,TRT),END
```

3. Obtain a z/OS SVC dump of the DB2® MSTR and DBM1 regions by issuing this series of commands:

```
DUMP COMM=(dump title)
R id JOBNAME=(dbtmstr,dbwdbm1),
SDATA=(CSA,PSA,RGN,SQA,SUM,TRT),END
```

4. Save the IMS online log data set that was active during the failure because IMS TYPE5501, 08, 07, 56 and other log records can be critical to diagnosis. The IMS TYPE5501 records are updated by DB2. The internal buffer for these records is stored at the location described by the CDE entry named WAL in the IMS regions.
5. If the IMS monitor is started, issue the following command to monitor the IMS data set:

```
/TRACE SET ON MONITOR ALL
```

# Collecting data about database problems

The first step to diagnosing an IMS problem is to collect data about the problem. If you call an IBM service representative, you will be asked for documentation about the problem.

## About this task

For database problems, obtain the following information:

- The damaged database data set.
- The database image copy of the damaged database in a state prior to damage.
- The image copy of logically related databases.
- The IMS OLDS from all data-sharing IMS subsystems.

- Save from the last good database image copy of damaged database.
- If possible, and not already set, use the following IMS commands and save the output:

```
/TRA SET ON TABLE DLI OPTION LOG
```

and

```
/TRA SET ON TABLE LOCK OPTION LOG
```

- The SYSOUT from the Pointer Checker jobs for the damaged database.
- The SYSOUT from batch jobs that accessed the damaged database.
- The LIST.RECON and LIST.HISTORY DBD from the damaged database.
- The SMF 60, 62, and 64 records from all data-sharing systems back to the last good image copy of damaged database.
- For VSAM data sets:
  - Issue IDCAMS LISTC for the damaged VSAM data set.
  - Issue IDCAMS DIAGNOSE and IDCAMS EXAMINE for the damaged VSAM KSDS data sets.

## Collecting data about z/OS Resource Recovery Services problems

RRS provides a system resource recovery platform such that applications that run on z/OS can have access to local and distributed resources and have system-coordinated recovery management of these resources.

### About this task

If you use RRS with your IMS system:

- Take an SVC dump of the standard IMS regions using one of the methods discussed earlier: CTL, DL/I, DBRC, suspicious dependent regions, IRLM, and so on.
  - In addition, include the z/OS RRS address space and the z/OS logger address space (IXGLOGR).
  - Consider setting the following SLIP trap to supplement standard IMS/RRS ABENDU0711 diagnostics:

```
SLIP SET,C=U0711,JOBLIST=(ctljname,rrsjname,IXGLOGR),
SDATA=(CSA,PSA,RGN,SQA,SUM,TRT,GRSQ,LPA,ALLNUC),
ID=nnnn,DSPNAME=('RRS'.*),END
---
```

In the previous example:

#### **ctljname**

IMS control region job name

#### **rrsjname**

RRS region job name

#### **nnnn**

Name used to recognize this SLIP

- Turn on the RRS component trace.
  - Place the following statements in the CTIRRSxx PARMLIB member:

```
TRACEOPTS
ON
BUFSIZE(500M)
OPTIONS('EVENTS(URSERVS,LOGGING,CONTEXT,EXITS,STATECHG,RRSAPI,RESTART)')
```

- Place the following statement in the z/OS COMMNDxx SYS1.PARMLIB member:

```
TRACE CT,ON,COMP=SYSRRS,500M,PARM=CTIRRSxx
```

**Note:** This statement allows the trace to be active at IPL.

- Use the D TRACE,COMP=SYSRRS command to view the current trace setting.
- RRS component trace is present in the RRS address space.
- Format the trace by using IPCS CTRACE COMP(SYSRRS) FULL command.
- Save the IMS OLDS
  - IMS 67D0 log records are produced for some ABENDU0711 abends.
    - Print these records by using the IMS utility programs DFSERA10 and DFSERA30.
  - Other RRS related records that are produced:
    - TYPE4098 - Checkpoint for RRS log name.
    - TYPE5615 - IMS restarted with RRS.
    - TYPE5616 - Start of protected UOW.
  - Issue two or three IMS DISPLAY UOR ALL commands to show status about the IMS UOR for protected resources on the RRS recovery platform.
    - The RRS-URID provided by RRS and the IMS recovery token are displayed.
- If the problem is recreatable, then:
  - Turn on the RRS component trace:

```
TRACE CT,ON,500M,COMP=SYSRRS  
nn,OPTIONS=(EVENTS(ALL)),END
```

- When the problem has been recreated, stop the component trace:

```
TRACE CT,OFF,COMP=SYSRRS
```

- RRS component trace is present in the RRS address space.
- Format the trace by using the IPCS CTRACE COMP(SYSRRS) FULL command.
- Issue two or three IMS DISPLAY UOR ALL commands to show status about the IMS UOR for protected resources on the RRS recovery platform.
  - The RRS-URID provided by RRS and the IMS recovery token are displayed.

### Related tasks

[“Collecting data about IMS Connect problems” on page 20](#)

If a problem occurs during IMS Connect execution, you need to collect logs, data sets, and dumps to determine the source of the problem.

## Collecting data about MSC-related problems

If you experience IMS Multiple Systems Coupling (MSC) related problems, you can collect diagnosis data with SVC dumps.

### About this task

If you use IMS MSC and experience a related problem, complete the following tasks:

- Create an SVC dump of the coupled IMS regions (minimally, the CTL regions, but the problem might reside in any IMS-related region). In addition, ensure that the VTAM address space is also included. Create this dump as close to the time of the problem as possible, before you try to fix the problem.
- Save the IMS OLDS for both coupled systems from the time of the message creation.

- To show the status and queue counts for the logical link, issue the type-1 commands **/DISPLAY LINK ALL** and **/DISPLAY LINK ALL MODE**, or issue the type-2 command **QUERY MSLINK NAME(linkname) SHOW(ALL)**.

- If the problem is recreatable:

- Turn on the VTAM Internal Trace for both coupled systems:

```
F NET,TRACE,TYPE=VTAM,OPT=(API,PIU,MSG),DSPSIZE=5,SIZE=999
```

- Using the options shown above, the VIT (VTAM internal trace) is created in a VTAM data space. After the problem has been recreated, the dump parameters should also include the VTAM data space:

```
DSPNAME=('NET'.ISTITDS1)
```

- Turn on the MSC LINK trace for both coupled systems by issuing either the type-1 command **/TRACE SET ON LINK link# LEVEL 3 MODULE ALL** or the type-2 command **UPDATE MSLINK NAME(linkname) START(TRACE)**.
- The type-2 command **UPDATE MSLINK NAME(linkname) START(TRACE)** uses the same level and module settings that were used the last time the command **/TRACE SET ON LINK link#** was issued. If a **/TRACE SET ON LINK link#** command has not been issued since the last cold start, this command defaults to **MODULE ALL** and **LEVEL 4**.

# Chapter 3. Collecting data about abends by using IMS abend search and notification

IMS abend search and notification enables IMS to send an email or text message to a designated email address when an abend occurs.

When an abnormal termination (or *abend*) occurs, IMS sends a message that contains an abend code to an operator's console or to the master terminal operator (MTO). This process delays a response because the person who can respond to the abend might not be near the console when the message is sent. Therefore, action in response to the abend is delayed until the message reaches the correct person.

IMS abend search and notification enables IMS to send an email or text message to a designated email address when an abend occurs, in order to:

- Notify the correct person of an abnormal termination (abend).
- Provide a web address (URL) to the location of additional informational resources, such as IMS product documentation, technical notes in an IBM technical support database, and information in the preventive service planning (PSP) database.

You can also use IMS abend search and notification to dynamically research abends and build links to web addresses that provide information that relates to the abend, such as online product documentation and technical support databases.

## Enabling IMS abend search and notification

Before you can use IMS abend search and notification to search online documents and technical support databases for information related to IMS abends, you must enable it.

### About this task

Complete the setup procedures for the IMS abend search and notification, as described in "Setting up IMS for diagnostics" in *IMS Version 15.5 System Definition*.

## Researching information about abends dynamically

You can use IMS abend search and notification to dynamically generate emails for specific abends, specify research criteria, and replicate the search criteria that IMS used in a particular abend event and generate an email.

### About this task

To research information about abends dynamically:

### Procedure

1. Start IMS abend search and notification by using one of the following methods:

Option	Description
<b>Select IMS abend search and notification from the IMS Application Menu.</b>	The IMS Abend Search and Notification panel is displayed.
<b>From the ISPF option 6, enter the following command, where <i>hlq</i> is the high-level qualifier used to install IMS:</b>  <code>exec 'hlq.SDFSEEXEC(DFSASNO)' 'HLQ(hlq)'</code>	The IMS Abend Search and Notification panel is displayed.

2. Type 2 in the IMS ASN On-Demand Interface field and press **Enter**.

The IMS Abend Search and Notification panel - on demand interface panel is displayed, as shown in the following figure.

```
-----
                                IMS Abend Search and Notification - on demand interface
COMMAND ==>

*Product Name . . . . . REQUIRED (example: IMS)
Provide one or more of the following web search arguments
  PSP for FMID . . . . . (example: JMK7701)
  Abend Code . . . . . (example: S0C4)
  Return Code (RC) . . . . . (example: 44)
  Module Name. . . . . (example: DFSSAMP0)
  APAR Number (PE or other). . . . . (example: PQ99999)
  Message ID . . . . . (example: EFS555i)
  Generic Search Argument . . . . . (example: SQL+OVERFLOW)
Recipient Information
  Recipient email Address. . . . . (example: name@company.com)
  Specify Additional Addresses? _ (Y-Yes/N-No)
JOB JCL Statement. . . . . (E-Edit/Y-Yes/N-No)
-----
```

*Figure 2. IMS Abend Search and Notification - on demand interface panel*

3. Type your search criteria and press Enter.

---

## Chapter 4. How to search problem-reporting databases

After you obtain background information about the problem you are diagnosing, you can then use that information to create search arguments to search problem-reporting databases for known problems that describe an aspect of a program failure.

You use keyword strings to search an IBM Technical Support database for documents such as Authorized Program Analysis Reports (APARs), Preventative Service Planning buckets, or Technotes for information about the resolution of reported problems. If the search is successful, you will find a similar problem description, and usually a fix or recommendation. If the failure is one that is not known, you will use the keywords to describe the failure when you contact IBM Product Software Support for assistance.

Some optional search tools might require keywords in a structured database (SDB) format.

---

### Developing search arguments

A keyword describes one aspect of a program failure. A set of keywords, called a *keyword string*, describes a specific problem in detail. Because you use a keyword string to search a database, a keyword string is also called a *search argument*.

The keywords you use to search for problems in IMS are:

- The component identification

This is the first keyword in the string. A search of the database with this keyword alone detects all reported problems for that version of IMS.

- The type of failure

The second keyword specifies the type of failure that occurred. Its values can be:

- ABENDxxx
- ABENDUxxxx
- DOC
- PERFM
- MSGx
- INCORROUT
- WAIT/LOOP

- Symptom keywords

These can follow the keywords above and supply additional details about the failure. You select these keywords as you proceed through the type-of-failure keyword procedure that applies to your problem.

Add symptom keywords to the search argument gradually so that you receive all data matches or *hits*, which are problem descriptions that might match your problem. If you receive too many problem descriptions to examine, you can add AND or OR operators to additional keywords in various combinations to the keyword string to reduce the number of hits.

- Dependency keywords

These are program or device dependent keywords that define the specific environment that the problem occurred in. When added to your set of keywords, they can help reduce the number of problem descriptions you need to examine. See [“Dependency keywords” on page 65](#) for a list.

## Creating a search argument

---

After you have performed some analysis on the problem you are diagnosing, you can then use that information to search problem reporting databases. To do that, you create a search argument comprised of keyword strings. If that search technique is unsuccessful, you can prepare an Authorized Program Analysis Report (APAR).

### About this task

To build the keyword string and search the IBM software support database for a problem similar to the one you are experiencing, follow these steps:

### Procedure

1. Begin with [“Component identification keyword procedure” on page 34](#) to determine the failing IMS component.
2. Follow the sequential steps in one of the "Type-of-Failure Keyword" procedures until you build a keyword string.
3. Go to [“Searching the database” on page 66](#), to learn how to search the IBM software support database with your completed string.
  - a) Optional: Use the [“IMS keyword dictionary” on page 63](#), which provides guidance on translating free-form keywords into structured database (SDB) format.
  - b) If needed, you could use [“Dependency keywords” on page 65](#), which are used to narrow search arguments.
4. If your search is unsuccessful, go to [“Procedures for preparing an APAR” on page 68](#).

## Selecting the keywords

---

You select the proper keywords to search the IBM Software Support database for a problem similar to the one you are experiencing. The keywords you select depend on the component that is experiencing the problem and the type of failure that occurred.

### Related tasks

[“Analyzing DRA problems” on page 398](#)

To analyze DRA problems, first investigate any external conditions that might have caused the problem. If you can eliminate external causes, an unexpected DBCTL return code or another IMS function might have caused the problem.

## Component identification keyword procedure

Use a component identification number with at least one other keyword to search the IBM software support database.

The component identification numbers for IMS appear in the following table.

---

*Table 2. IMS component identification numbers*

---

Identification number	Description
5635A05	IMS Services Database Manager Transaction Manager Extended Terminal Option (ETO) Recovery-level Tracking Database-level Tracking
569516401	Internal Resource Lock Manager (IRLM) version 2

---



Some of the procedures in these topics contain offsets within control blocks. Be aware that maintenance might change the offsets in these control blocks. For a current version of the layout of the control blocks for your system, assemble the DFSADSCT module that is in the IMS.ADFSSMPL library.

#### **Related concepts**

[“PERFM procedure” on page 40](#)

Most performance problems are related to system tuning and should be handled by system programmers.

#### **Related tasks**

[“Type-of-failure keyword” on page 35](#)

You choose the keyword that best describes the program failure you experienced, and then go to the procedure for that type of failure.

## **Type-of-failure keyword**

You choose the keyword that best describes the program failure you experienced, and then go to the procedure for that type of failure.

#### **Related reference**

[“Component identification keyword procedure” on page 34](#)

Use a component identification number with at least one other keyword to search the IBM software support database.

## **ABENDxxx procedure**

Use this procedure when the system terminates abnormally with a system abend completion code.

### **About this task**

#### **Keyword: ABENDxxx**

Compare the completion code and PSW address in both the z/OS-formatted section of the dump and the IMS-formatted section of the dump. If the code and address do not match, use only the data from the IMS-formatted section, because the system dump data might be produced if an abend occurs during abend processing.

Replace the xxx part of the ABENDxxx keyword with the abend code from either the termination message or the abend dump.

#### **Keyword: RCxx**

This keyword applies only if the abend has an associated return code as described in *z/OS MVS System Commands*.

Replace the xx part of the RCxx keyword with the return code.

#### **Keyword: module name**

You can determine the name of the module that received the abend in one of the following ways:

- Check both the dump title and message DFS629I, which might contain the name of the module that ended abnormally.
- Check the summary section, called "Diagnostic Area", in the offline formatted dump.
- Find the PSW address at the time of abend. Locate this address in the storage section of the dump, and scan backward through the eye catchers until you find a module identifier.

#### **Module-specific keyword: Failing instruction, register**

You can use these module-specific keywords to further narrow the field of hits.

#### **Failing Instruction**

The PSW address at the time of abend usually points to the next instruction to be executed. If ABEND0C4 or ABEND0C5 occurs and the INTC (interrupt code) field on the PSW AT ENTRY TO

ABEND line contains X'0011' (segment exception) or X'0010' (page translation exception), the PSW points directly to the instruction that failed.

### Register in Error

Examine the code near the failure to determine the register that is invalid or in error, if possible.

For example, if the failing instruction is BALR (05EF), look at registers 14 (E) and 15 (F). If register 15 (F) contains zeros, the program cannot branch to that location. Therefore, register 15 is in error.

In performing system-abend analysis, another module might have passed the register in error. You might be able to determine this by looking at the registers on entry to the failing module. If the incorrect value is in one of the registers, that value might have been passed.

### Module-specific keyword: Search argument example

If, for example, ABEND0C4 occurred in IMS module DFSFXC30 on a BALR (05EF) instruction because register 15 (F) contained zeros, use the following search argument:

```
5655J3800 ABEND0C4 DFSFXC30
```

For a structured database search, use the following search argument:

```
PIDS/5655J3800 AB/S00C4 RIDS/DFSFXC30
```

With this search argument, you might receive numerous hits, which would most likely include the APAR that describes your problem. You can add module-specific keywords to narrow the field of hits received. Use the OR operator with these additional keywords at first.

The additional keywords for this example are:

```
BALR | R15 ZEROS
```

For a structured database search, use the following search argument:

```
OPCS/BALR | REGS/GR15 VALU/H000000000
```

### Related concepts

[“ABENDUxxxx procedure” on page 36](#)

Use this procedure when an IMS user abnormal termination occurs. For user abends, you must gather more information before calling IBM Software Support.

## ABENDUxxxx procedure

Use this procedure when an IMS user abnormal termination occurs. For user abends, you must gather more information before calling IBM Software Support.

A message usually precedes a user abend. First, find the message and then the abend code in *IMS Version 15.5 Messages and Codes, Volume 1: DFSMessages* or *IMS Version 15.5 Messages and Codes, Volume 2: Non-DFS Messages*. Then, if you need further diagnostic information (such as return codes) that you can use to build the search argument, or information about why the abend was issued, refer to the *IMS Version 15.5 Messages and Codes, Volume 3: IMSAbend Codes*.

If you cannot solve the problem by using the information in the *IMS Version 15.5 Messages and Codes, Volume 3: IMSAbend Codes*, develop a search argument.

## ABENDUxxxx keywords

Replace the xxxx part of the ABENDUxxxx keyword with the user abend code from either the termination message or the abend dump. User abends are always represented in decimal.

## Keyword: module name

You can determine the name of the module that received the abend in either of the following ways:

- Check both the dump title and message DFS629I, which might contain the name of the module that ended abnormally.
- Use the PSW address at the time of abend. You can find this address in the IMS-formatted section of the dump under the diagnostic area or in the z/OS-formatted section. From the PSW address, scan backward through the eye catchers until you find a module identifier.

Use the module name in the search argument for standard user abends only. For pseudoabends, do not include the module name as part of the argument. *IMS Version 15.5 Messages and Codes, Volume 3: IMSAbend Codes* indicates whether the abend is a pseudoabend or a standard abend.

## Abend-specific keywords

By examining the information in *IMS Version 15.5 Messages and Codes, Volume 3: IMSAbend Codes*, you might gather additional keywords that can be pertinent to the problem, such as:

- User call function
- Internal call function
- Database organization
- Messages

Replace the xxxxxx part of keyword MSGxxxxxx with the actual message identifier (for example, the keyword for message DFS053I is MSGDFS053I).

- Return codes

Replace the xx part of keyword RCxx with the associated hexadecimal return code (for example, the keyword for return code C is RC0C).

- Function codes

Replace the xxxx part of keyword FCxxxx with the associated hexadecimal function code (for example, the keyword for function code 13 is FC0013).

## Search argument example

If, for example, ABENDU3046 occurred in IMS module DFSPCC20 with message DFS3624I indicating function code 291 and return code 4, the search argument to use is:

```
5655J3800 ABENDU3046
```

For a structured database search, use this search argument:

```
PIDS/5655J3800 AB/U3046
```

With this search argument, you might receive numerous hits, which would most likely include the APAR describing your problem. You can add keywords from the section [“Type-of-failure keyword” on page 35](#) to narrow the field of hits received. It is a good idea to use the **OR** operator on these additional keywords at first. Module name DFSPCC20 is not included as part of the search argument because ABENDU3046 is a pseudoabend.

The additional keywords for the above scenario are:

```
MSGDFS3624I | RC04 | FC0291
```

For a structured database search, use this search argument:

```
MS/DFS3624I PRCS/00000004 OPCS/0291
```

## Additional documentation

IBM Software Support might ask you to obtain certain information to determine and resolve the problem. At times you might need to create the problem again in order to gather this documentation.

For database problems, ensure that you have access to the following documentation before calling IBM Software Support:

- A dump of the problem
- DBDGENS
- PSBGENS
- A copy of the databases involved in the error
- Logs and archive tapes that might have activity against the databases
- Output from both the DL/I and LOCK traces
- When tracing to the log, a printout of the traces
- A current CDS list or a current SMP/E target zone
- A current assembly listing of DFSADSCT from IMS.ADFSSMPL (control block DSECTs)

Problems can be resolved more quickly if the documentation listed above is available.

## IRLM procedure

Use this procedure when the IRLM terminates abnormally.

1. Locate the PSW and register contents at entry to the abend either from the software LOGREC entry or from the RTM2WA summary in the formatted section of the SDUMP.
  - a. If the PSW is not within an IRLM module (prefixed with DXR), determine the system component in which the abend occurred and use the diagnostic procedure for that component to resolve the problem.
  - b. If the RTM2WA summary entry shows that the IRLM was terminated by an abend completion code of U2017, U2018, U2019, U2020, U2022, U2023, U2024, U2025, U2027, U2031 (X'7E1', X'7E2', X'7E3', X'7E4', X'7E6', X'7E7', X'7E8', X'7E9', X'7EB', or X'7EF'), the IRLM task was terminated because of an error either in a subtask or in an SRB related to the IRLM. To diagnose the problem, use the software LOGREC entry or the RTM2WA summary entry for the original error in the subtask or related SRB.
2. Register 12 normally contains the base register contents for the module that was in control at the time of the error.
3. Register 9 normally contains the address of the RLMCB if the error occurred during IRLM processing.
4. Using the module name, find the function keyword and locate the function and subfunction keywords.

## Examples

An example of a search argument for an IRLM problem is:

```
569516401 ABEND0C4 DXRRL200
```

For a structured database search, an example is:

```
PIDS/569516401 AB/S00C4 RIDS/DXRRL200
```

## Related tasks

[“ABENDxxx procedure” on page 35](#)

Use this procedure when the system terminates abnormally with a system abend completion code.

## DOC procedure

Use this procedure if you find a deficiency in documentation through omission or inaccuracy.

### Keyword: order-number

Your feedback is important in helping us provide the most accurate and highest quality information. If you have any comments about this or any other IMS information, you can take one of the following actions:

- From any topic in IBM Documentation, click the **Feedback** link at the bottom of the page and complete the form.
- Send your comments by email to [imspubs@us.ibm.com](mailto:imspubs@us.ibm.com). Be sure to include the title, the part number of the title, the version of IMS, and, if applicable, the specific location of the text on which you are commenting (for example, a page number in the PDF or a heading in IBM Documentation).

Corrections resulting from readers' comments are included in future editions of the manual, but are not included in the software support database.

If a problem can have severe results or cause lost time for many other users, contact IBM Software Support to initiate a documentation change.

APARs are not generally accepted for documentation errors. However, APARs that correct a programming error can result in documentation changes.

Use this keyword to search for all changes to a specific manual. The format for the order-number is *ppnnnnnnnee*, where *pp* is the alphabetic prefix, *nnnnnn* is the 6-digit base publication number, and *ee* is the edition number. For example, the order number for *IMS Version 15.5 Messages and Codes, Volume 1: DFSMessages* is GC18-9712-00. Replace *ppnnnnnnnee* with GC18971200. The edition number is optional. To broaden the search to include all editions of a publication, either omit the edition number or replace it with two asterisks (\*\*).

### Search argument example

Use this search argument to search for all changes to any edition of *IMS Version 15.5 Messages and Codes, Volume 1: DFSMessages*:

```
5655J3800 GC189712**
```

For a structured database search, use this search argument:

```
PIDS/5655J3800 PUBS/GC269712**
```

You can add more keywords to narrow the search. For example, if you cannot find message DFS3007 in *IMS Version 15.5 Messages and Codes, Volume 1: DFSMessages*, add this keyword to the above search argument:

```
MSGDFS3007
```

For a structured database search, use this search argument:

```
MS/DFS3007
```

If you do not find an APAR that adds message DFS3007, you can report the omission to IBM by clicking the **Feedback** link at the bottom of any topic in IBM Documentation.

## PERFM procedure

Most performance problems are related to system tuning and should be handled by system programmers.

### Keyword: PERFM or PERFORMANCE

Always use the keywords PERFM and PERFORMANCE for performance problems. You should use the **OR** operator to link them together in the search argument.

You can use the following search argument to check for all performance APARs in IMS Fast Path:

```
5655J3800 PERFM | PERFORMANCE FAST | PATH | FASTPATH
```

For a structured database search, you can use this search argument:

```
PIDS/5655J3800 PERFM | PERFORMANCE RIDS/FASTPATH
```

You can add the **OR** operator to the general component identifier together with the Fast Path component identifier. With this search argument, the resulting number of hits could be very large, but would include APARs describing performance problems in Fast Path.

You can add more keywords to narrow the number of hits. For example, if the performance problem occurs because of an excessive number of file opens and closes, you can add the **OR** operator with the following keywords to the above search argument:

```
OPEN | CLOSE
```

For a structured database search, use this search argument:

```
PCSS/OPEN | PCSS/CLOSE
```

If you cannot find an appropriate APAR with these search arguments, contact IBM Software Support.

Appropriate documentation for performance problems might include:

- Traces, such as DL/I, lock, dispatcher, scheduler, external subsystem, and others, depending on the area of the performance problem
- Dumps of the problem during the period of performance degradation
- Dumps of the problem during normal periods, for comparison
- DB or IMS Monitor reports during the performance problem period
- DB or IMS Monitor reports during normal operations, for comparison
- Copy of the IMS log during the performance problem period
- Copy of the IMS log during the normal period, for comparison

If a coordinator controller (CCTL) application program experiences a performance problem in a Database Control (DBCTL) environment, you might need the following documentation in addition to that listed above:

- Any CCTL traces or monitor reports
- A dump of the CCTL subsystem during the period of performance degradation

### Related reference

[“Component identification keyword procedure” on page 34](#)

Use a component identification number with at least one other keyword to search the IBM software support database.

## MSG procedure

If, after analyzing the message, you believe that the message should not have been issued or describes an error condition, use the MSGxxxxxxx keyword.

### Keyword: MSGxxxxxxx

Replace the xxxxxxxx part of keyword MSGxxxxxxx with the actual message identifier (for example, the keyword for message DFS0861 is MSGDFS0861).

### Search argument examples

If, for example, you receive message DFS3401I RACF NOT AVAILABLE, and you determine that RACF is indeed available in your system, the search argument to use is:

```
5655J3800 MSGDFS3401I
```

For a structured database search, use this search argument:

```
PIDS/5655J3800 MS/DFS3401I
```

## INCORROUT procedure

INCORROUT is when either output is missing, or output is incorrect.

### Keyword: INCORROUT

Use this procedure to determine the appropriate search argument.

Always use the keyword INCORROUT for problems related to incorrect or missing output.

### Keyword: utility module name

If the incorrect or missing output is associated with a utility, use the utility module name as a keyword. For example, if output from the File Select and Formatting Print utility (DFSERA10) is incorrect, use DFSERA10 as a keyword.

### Keyword: command

If the output from a command is missing or incorrect, use the first three letters of the command as a keyword. Also, you should use the **OR** operator in the search argument with CMDxxx, where xxx is replaced by the first three letters of the command.

If, for example, the DISPLAY command provides incorrect output, use the following search argument:

```
5655J3800 INCORROUT DIS | CMDDIS
```

For a structured database search, use this search argument:

```
PIDS/5655J3800 INCORROUT PCSS/DIS
```

If applicable, you can add the output column or heading as a keyword in the search argument.

### Keywords: columns, headings, fields

Whenever possible, you can add additional keywords to narrow the field of search results. If a particular heading, field name, or column is incorrect, use it as a keyword. For example, if the deadlock event

summary section of the IMS Monitor report (DFSUTR20) is incorrect for the DMB NAME column, use the following search argument:

```
5655J3800 INCORROUT DFSUTR20 DEADLOCK | DMB
```

For a structured database search, use this search argument:

```
PIDS/5655J3800 INCORROUT RIDS/DFSUTR20  
PCSS/DEADLOCK PCSS/DMB
```

If you receive too many search results, remove the **OR** operator (|) to focus the selection.

## Keyword: database type or call

If the incorrect output is a database record, use the database type (such as VSAM, HDAM, or HIDAM) and possibly the call (such as GU, ISRT, or DELETE).

## Additional diagnostics

This section does not apply to a Database Control (DBCTL) environment.

If the output is a transaction message produced as output from an application program, perform the steps below. (The message can be directed either to a terminal or to another application program. This is called a program switch.)

1. If the output is missing, continue with this step; otherwise, go to step 2.

a. When the output is missing, determine if the transaction is being scheduled.

- Issue the /DIS ACTIVE command to make sure the transaction is not stopped.
- Then issue the /DIS TRAN command to find out if the transaction is scheduled.

QCT should decrease by at least one each time the transaction is scheduled and terminates normally.

If the transaction is not being scheduled, go to step 1f.

b. Determine if the message is being enqueued to the proper output destination by issuing one of the following commands:

- Issue the /DIS TRAN command (for program switch). ENQCT should increase.
- Issue the /DIS LTERM command (for output to terminal). ENQCT should increase.

If the message is not being enqueued to the proper output destination, go to step 1e.

c. If the output destination is another application program, it should be scheduled as a result of the message enqueue.

If the transaction is scheduled but there is no input, the problem is probably within the SYS function.

If the application program is not scheduled, go to step 1f.

d. If the output destination is a terminal, verify that I/O errors did not prevent the message from being sent. Take both of the following actions.

- Review the console log for I/O error messages.
- Issue the /DIS LTERM command for operational status.

If you detected valid I/O errors, stop here and correct the hardware problem. Otherwise, the problem is probably within the TM function. Stop here and build your search argument.

e. Determine if the application program is using the proper PCB for the ISRT call.

- Force a dump in the application program at the time of the ISRT call.



If the proper PCB is being used, the problem is probably within the SYS function. Stop here and build your search argument. Otherwise, stop here and correct the application program.

f. Determine if the resources necessary to schedule the application program are available.

- Issue the /DIS ACTIVE command for the active region.
- Issue the /DIS SUBSYS ALL command for all external subsystems connected to or in the process of being connected to IMS.
- Issue the /DIS TRAN command to make sure the transaction is not stopped.
- Issue the /DIS DATABASE command to determine if the necessary databases are available.

If a resource is not available, stop here and make it available. Otherwise, force a console dump. Use the PST ANALYSIS step in procedure [“WAIT/LOOP procedure” on page 44](#) to determine the reason the transaction is not being scheduled. Stop here and build your search argument using that information.

2. If the incorrect data is input to an application, perform this step, otherwise go to step 3.

a. Verify the text data in the X'01' log record to determine if the data reached IMS properly.

If the data did not reach IMS properly, go to step 2c.

b. Force a dump in the application program immediately after the application program GU call, in order to determine if the data reached the I/O area correctly.

If the data did not reach the I/O area correctly, the problem is probably within the SYS function. Stop here and report the problem. Otherwise, the application program received the data correctly. Stop here.

c. Start the line or node trace and verify the data in the X'6701' log record to determine if the data reached the input TP buffer correctly.

If the data reached the input TP buffer correctly, the problem is probably within the DC function. Stop here and report the problem. Otherwise, if the data did not reach the input TP buffer correctly, the problem is probably a hardware or an operating system failure. Stop here and correct the hardware or operating system problem.

3. Determine if the message data is actually incorrect rather than merely formatted incorrectly.

- Compare received data with expected data.
  - Check MFS blocks for correct format definition.
- a. Force a dump in the application program just before the ISRT call to determine whether the data is correct in the I/O area at the time of the ISRT.

If the data in the I/O area is incorrect, the problem is probably in the application program. Stop here and correct the application program. Otherwise, continue. Verify the text in the X'03' log record to determine whether the data reached the message queue correctly.

If the message did not reach the message queue correctly, the problem is probably within the SYS function. Stop here and build your search argument. Otherwise, continue.

b. Start the line or node trace and verify the data in the X'6701' log records, in order to determine if the data reached the output TP buffer correctly.

If the data did not reach the output TP buffer correctly, the problem is probably within the DC function. Stop here and build your search argument. Otherwise, if the data is correct in the output TP buffer, but not at the terminal, the problem is probably a hardware or operating system failure. Stop here and correct the hardware or operating system problem.

## IRLM problems

Incorrect output from the IRLM can be divided into the following three areas:

- Incorrect information on a display status command
- Locks granted when locks should not be granted

- Locks not granted when locks should be granted

For help in diagnosing these problems, call the IBM Support Center. A support representative will tell you what type of documentation to gather.

## WAIT/LOOP procedure

The procedures for the WAIT and LOOP keywords are combined because the WAIT and LOOP symptoms might not be distinguishable at first.

### Determine the type of WAIT or LOOP that is in progress

Use the following procedure to determine the type of WAIT or LOOP occurring, and to find the appropriate keywords for the problem.

Maintenance might change the offsets in these control blocks. For a current version of the control blocks, assemble DFSADSCT.

#### 1. Is IMS being shut down?

- If the operator issued a CHECKPOINT DUMPQ, PURGE, or FREEZE command before the manifestation of the wait/loop, go to [“Shutdown processing” on page 58](#).
- If IMS is not being shut down, continue with the next step.

#### 2. Determine whether IMS was in selective dispatching mode.

Find the dispatch work areas in the formatted dump. The dispatch work areas are created using the DISPATCH or ALL IMS dump formatting options. The dispatch work area eye catcher is \*\*DSP.

The selective dispatch bits are in the SFLAGS field in the DYNAMIC SAP EXT. section, where the X'xxxxxx8x' bit represents selective dispatching. To determine whether selective dispatching was entered for save area prefixes (SAPs), search the DISPATCH AREA section for the following message:

```
*** NOTE: THIS TCB IS IN SELECTIVE DISPATCHING FOR SAPS
```

If you find this message, IMS wrote an X'450F' log record to the OLDS. This log record contains information about dynamic SAPs, such as the highest number of dynamic SAPs used and the number of times IMS was in selective dispatch for dynamic SAPs.

Examine this X'450F' log record to help determine what might have led to the shortage of dynamic SAPs. Then go to the [“Determine the type of WAIT or LOOP that is in progress” on page 44](#). While performing SAP analysis, keep in mind that the dynamic SAPs are labeled DYNAMIC SAP, and that the CURRENT TCB= indicates the associated task control block (TCB).

If IMS is not in selective dispatching mode, continue with the next step.

#### 3. Can the operator communicate with IMS through the z/OS system console by using the IMS outstanding reply to enter an IMS command, such as /DISPLAY?

- If no, or if you are not sure, go to step 5 now.
- If yes, the problem might be caused by:
  - A data communication failure.
  - The inability of a task to acquire a resource.
  - Non-completion of an event, such as I/O.

Continue with the next step.

#### 4. Can the IMS master terminal operator (MTO) communicate with IMS by issuing various IMS commands, such as /DISPLAY?

- If yes, go to [“Determine the type of WAIT or LOOP that is in progress” on page 44](#).
- If no, the problem might be data communication related. If IMS is still running, issue the following commands:

- Issue the IMS /DIS NODE *nodename* command. Save the IMS console output.
- Turn on the IMS node trace with the /TRA SET ON NODE *nodename* command.  
Data is captured in the IMS X'6701' log record. Save the IMS OLDS for execution with IMS utility programs DFSERA10 and DFSERA30.
- Consider turning the VTAM buffer trace and VTAM internal trace on to complement the IMS node trace, as follows:

```
F NET,TRACE,TYPE=BUF,ID=nodename
F NET,TRACE,TYPE=VTAM,MODE=EXT,OPT=(API,PIU,MSG)
```

GTF must be active for this option.

- Obtain a memory dump of the IMS and VTAM regions using this series of commands:

```
DUMP COMM=(dump title)
R id JOBNAME=(j1,j2,j3,j4,j5,j6,j7),SDATA=(CSA,PSA,RGN,SQA,SUM,TRT),END
```

The variables have the following meanings:

- j1** IMS CTL region job name.
- j2** VTAM region job name.
- j3** IMS DL/I region job name.
- j4** Suspicious IMS dependent region job name, if any.
- j5** Suspicious CCTL (CICS) region name, if any.
- j6** DBRC region job name.
- j7** IRLM region job name (if IRLM database locking was used).

The jobs are listed in order of importance.

**Recommendations:** A memory dump of the IMS CTL, VTAM, DL/I, and suspicious dependent region or CCTL is usually sufficient to solve wait/hang problems. Occasionally, the DBRC and IRLM (if they are used for database locking) can be a factor. Obtain a memory dump of DBRC and IRLM as well to ensure that the problem can be resolved quickly.

SYS1.DUMP data sets are often not large enough to hold all regions requested in the DUMP command. Make them large enough to hold the regions. If the z/OS SVC DUMP command fails due to lack of space, take separate memory dumps in smaller combinations to accommodate the smaller SYS1.DUMP data set size.

- Go to the [“Determine the type of WAIT or LOOP that is in progress”](#) on page 44.

#### 5. Query the IMS Dispatch Work Areas.

- Find the Dispatch Work Areas in the formatted dump. The Dispatch Work Areas are created using the DISPATCH or ALL IMS dump formatting options. The Dispatch Work Area eye catcher is \*\*DSP.
- Scan each Dispatch Work Area (STM, CTL, restart data set, and so on) except for the DRC and dependent region entries (labeled DEP, MPP, BMP, DBT, DRA, or IFP). Examine the QPOST field at offset X'1C'.

If the high-order bit of the QPOST field is off, note the address and type of Dispatch Work Area.

- If, after scanning all Dispatch Work Areas, except for the DBRC (DRC) task and dependent regions, you find that the QPOST high-order bit is always set, one of the following situations has occurred:

- IMS is in an IMS WAIT (IWAIT) state. Go to [“Determine the type of WAIT or LOOP that is in progress”](#) on page 44 now.
- If at least one Dispatch Work Area has an incorrect high-order bit, a LOOP or operating system WAIT has occurred. Continue with the next step.

6. Query the TCB/RB chain.

- Find the current ECB, address space ID (ASID), and TCB address for each Dispatch Work Area noted previously in step 5b.
  - In IDSPWRK SECTION 1, find field CECB at offset X'28'. The field CECB at offset X'28' contains the ECB of the current dispatched ECB.
  - In IDSPWRK SECTION 1, find the field ASIDS at offset X'30'. The first halfword of the field ASIDS at offset X'30' contains the ASID number for the task; the second halfword contains the CTL region ASID.
  - In IDSPWRK SECTION 1, find the field TCB at offset X'40'. The field TCB at offset X'40' contains the TCB address for the task.
- Find the formatted TCB/RB chain in the z/OS formatted dump. Use the IPCS SUMMARY FORMAT ASID(X'\_\_ ') command for the ASID/TCB found in step 6a. Use the following FIND command to locate the TCB:

```
F 'TCB: xxxxxxxx' 1 16
```

where xxxxxxxx is the 8-character TCB address, including leading zeros.

- Examine the request block (RB) structure (PRBs, SVRBs, or IRBs), focusing on the last RB in the chain for that TCB. The TCBRBP field at offset X'00' contains the address of the last RB. Use the following FIND command to locate the RB:

```
F 'RB: xxxxxxxx' 1 16
```

where xxxxxxxx is the 8-character RB address, including leading zeros.

Exception: Using the last RB in the TCBs RB chain is usually accurate. However, there are occasions when additional RBs might be appended to the end of the chain to facilitate dump processing, but they have nothing to do with the problem. X'00020033' in the WLIC field in any RB in the RB chain normally indicates dump processing. In such a case, examine the RBs prior to the RB with WLIC=X'00020033'. If the RB before the RB containing WLIC=X'00020033' contains WLIC=X'0002000C, it might be necessary to examine the RB before the RB containing WLIC=X'0002000C'.

Example:

```
PRB WLIC = X'00020006'
PRB WLIC = X'00020078'
SVRB WLIC = X'0002000C'   Examine prior RB.
SVRB WLIC = X'00020033'   <== Indicates dump processing
SVRB WLIC = X'00020078'
```

- Examine the LINK field in the RB found in step 6c. The high-order byte of the LINK field is the wait count field.
  - If the wait count is X'00', the task is probably looping. Perform the following steps:
    - Perform system loop diagnostics. Obtain the OPSW and registers from the looping RB, (located in the following RB or in the TCB, if this is the last RB (TCBRBP)) for a snapshot of the loop.
    - Obtain the PSW address from the z/OS system trace table. Use the IPCS VERBX TRACE ASID(xx) command to obtain the entries for the ASID in question. Focus on the entries for the TCB found in step 6a. You can ignore entries between any SVC and associated SVCR because they reflect necessary z/OS operating system activity indirectly involved in the loop. (The IMS TYPE2 SVC is an exception to this since it results in execution of IMS code.) Sorting the pertinent addresses by OPSW address greatly aids in laying out the loop.

- Resolve the PSW address found by using either IPCS BROWSE mode, the IPCS WHERE command, or by using an LPA or NUCLEUS MAP to obtain the name of the modules involved in the loop. The IPCS commands used to obtain the maps are LPAMAP, and VERBX NUCMAP. Calculate the offset at which the instruction appears in the modules to outline the path of the loop.
- Another source of information for the looping task can sometimes be found at the top of the IMS SAPS AND SAVEAREA section (\*\*SSA) of the IMS formatted dump. Look for the \*\*\*\* A C T I V E \*\*\*\* save area set nearest the top of the \*\*SSA with the SAPECB filed matching the CECB field obtained in step 6a. The save area flow can indicate IMS modules involved in the loop or those passing control to the looping function.
- If the wait count is not X'00' (that is, = X'01', or X'02', and so on), a system WAIT has probably occurred. Perform the following steps:
  - Obtain the address portion of the OPSW. It points to the waiting module.
  - Resolve the PSW address found by using either IPCS BROWSE mode, the IPCS WHERE command, or by using an LPA or NUCLEUS MAP to obtain the name of the waiting module. The IPCS commands used to obtain the maps are LPAMAP, and VERBX NUCMAP, respectively. Calculate the offset at which the wait occurred in the module. This information can be used for APAR searches and to assist IBM Software Support representatives.
  - Use the CECB field obtained in step 6a to find the related SAP save area by scanning for the SAPECB match in the IMS formatted memory dump \*\*SSA section.

## SAP analysis procedure

1. Find the formatted SAP AND SAVE AREA section in the IMS formatted dump.

Choose either the SAVEAREA, SYSTEM, ALL or SAVEAREA,SUM options of the IMS Offline Dump Formatter. The eye catcher of the SAP AND SAVE AREA section is \*\*SSA.

The following table defines the key fields in SAP analysis.

*Table 3. Key fields in SAP analysis*

Offset	Field name	Length	Field description
SAP+X'00'	SAPFLAG1	1	X'80' = Active SAP X'40' = Waiting SAP
SAP+X'01'	SAPDSPCD	1	IMS TCB number. This number matches the associated TCB number at offset X'3B' in the dispatch work area.
SAP+X'14'	SAPIWAIT	4	In waiting SAPs, this is the address of the last active save area. Those below this address are residual. In SAPs that are active but not waiting, this field is residual and should not be used.  Exception: SAPIWAIT might not be valid for Fast Path save area sets (DBF-prefixed modules). The active save area set usually ends with DBFXSL30, the Fast Path wait module, unless DFSIWAIT or DFSISERW appears previously in a save area set.
SAP+X'18'	SAPECB	4	Address of the ECB associated with this ITASK. If the PST is used, this field points to the beginning of the PST.

Table 3. Key fields in SAP analysis (continued)

Offset	Field name	Length	Field description
SAP+X'24'	SAPCDSP	4	Address of the current dispatch work area.
SAP+X'30'	SAPSDPNO	4	Dispatch number for the ITASK.

2. Begin SAP analysis at the end of the sorted SAPs.

Find the end of the sorted SAPs. Eye catcher **\*\*\*END OF SORTED SAP FORMATTING** marks the end of the list. SAPs are sorted by the SAPSDPNO (system dispatch number). The most recently dispatched ITASKs are at the end of the sorted SAPs. These are the ITASKs that have been waiting the longest and possibly causing the other ITASKs to wait behind them by holding a resource, such as a lock or a latch.

3. Scan backwards from the end, examining only active or waiting SAPs. Focus only on the active save area sets (that is, SAPFLAG1 has the X'00' bit turned on (X'08', X'Cx', X'Dx', X'Fx')). Active save area sets are marked with the eye catcher **\*\*\*\* W A I T I N G \*\*\*\*** or **\*\*\*\* A C T I V E \*\*\*\***. To find waiting or active SAPs, use the following find command:

```
F ' **** ' PREV
```

The SAVEAREA,SUM option of the Offline Dump Formatter produces only active save area sets. Active running SAPs are marked with the eye catcher **RUN**. The end of this formatting is marked by the eye catcher **\*\*\*\*\* END SAP SUMMARY**.

4. Skip all normal save area sets.

This step describes all normal save area sets. After you have identified all types of normal save area sets, you can disregard them because they are unrelated to the problem.

a. WAITING save area sets in which module name DFSIWAIT appears after label EP at the second-level save area are considered normal save area sets.

The following example shows a normal save area set at the second level:

```
***SAVE AREA SET***
EP DFSQMRT0-11/13/94
SA 00133BC4      WD1 8091E430   HSA 80000000   LSA 00133C0C ...

EP DFSIWAIT
SA 00133C0C      WD1 00000000   HSA 00133BC4   LSA 00133C54 ...

EP DFSFLLG0-220-PL46803
SA 00133C54      WD1 00000000   HSA 00133C0C   LSA 00133C9C ...
.....
```

b. The only normal save area sets in which the save area set contains DFSIWAIT at the third level are shown in the following example. Ensure that register 08 contains a value of X'00000003' for any of the first four save area sets, as shown in the example. Otherwise, it is abnormal and indicates an intent conflict, as described in [“Intent conflict” on page 55](#). Use the SAPSECB field to obtain the PST address for use in the intent conflict procedure.

```
EP DFSSMIC0 --> EP SMSC2      --> EP DFSIWAIT with REG08 = x'00000003'
EP DFSSMIC0 --> EP DFSSMSC2 --> EP DFSIWAIT with
REG08 = x'00000003'
EP DFSSMIC0 --> EP DFSSMSC1 --> EP DFSIWAIT with
REG08 = x'00000003'
EP DFSSMIC0 --> EP MPPENQ00 --> EP DFSIWAIT with REG08 = x'00000003'

EP DFSFXC30 --> EP DFSFXC30-WFITEST --> EP DFSIWAIT
EP DFSVTP00 --> EP VTPOWORK --> EP DFSIWAIT
EP DBFHCL00 --> EP DBFHGU10 --> DBFXSL30
```

c. The only normal save area sets in which the save area contains DFSIWAIT at the fourth level are those shown in the following example. Ensure that register 08 in the DFSIWAIT save area set contains X'00000003'. Otherwise, it is abnormal and indicates an intent conflict, as described in [“Intent conflict” on page 55](#). Use the SAPSECB field to obtain the PST address for use in the intent conflict procedure.

The following examples show normal save area sets at the fourth level:

```
DFSSMIC0 --> DFSSMSC0 --> SMSC1000 --> DFSIWAIT  REG08 = x'00000003'  
DFSFXC30 --> DFSDLA30 --> DLA32000 --> DFSIWAIT
```

d. The following active save area sets are probably normal, so you can ignore them.

- Save area sets marked ACTIVE or RUN with SAPDSPCD=X'07'. This is a DRC task SAP. This condition is usually normal for the DBRC task.
- Save area sets marked ACTIVE or RUN with SAPDSPCD=X'0F'. This is the ESI task SAP if SAPCDSP=X'00000000'.
- Dependent region save area sets marked ACTIVE with SAPDSPCD=X'03'(MPP), X'04'(BMP), X'0D'(DRA), X'12'(IFP), X'13'(DBT), X'0C'(ESS), or X'00'(RESIDUAL), in which the top save area indicates it was returned. (The last bit of the address in the field labeled RET, which is register 14, is odd or has X'FF' in the high-order byte.)
- If the SAPDSPCD=X'13'(DBT), and the first save area EPA is marked UNKNOWN with the second-level save area RET field marked returned (the last bit of the address in RET is odd), this is a normal save area set if the first save area EPA is within module DFSDASC0 or DFSDAST0.

5. Obtain abnormal save area set information.

The remaining save area sets (those that are ACTIVE or WAITING, but abnormal, as described in step 4 are involved in the wait in some way.

**Recommendation:** Concentrate on one save area set at a time, beginning with the first abnormal save area set. Remember to start from the end of the sorted SAPs.

If you find an abnormal save area set marked \*\*\*\*\* A C T I V E \*\*\*\*\* (SAPFLAG1=X'80'), the problem is associated with the TCB/RB save area set. Use the address of the current dispatch area in SAPCDSP to find the dispatch work area associated with this save area set. Go to step 6a in the WAIT/LOOP procedure. Continue from there, using the ASID/TCB obtained from the dispatch work area. If the high-order bit in QPOST is on (QPOST=X'8x'), this SAP is suspended. Record this save area set and continue to the next abnormal save area set. Discontinue step 6a because this save area set should probably be ignored. Otherwise, continue.

Record the following key fields from the abnormal save area sets flagged as:

\*\*\*\*\* W A I T I N G \*\*\*\*\*

- a. The address of the SAP.
- b. For each save area in the save area set, from the first save area down to the save area pointed to by the SAPIWAIT field, obtain the following information. (See exception for SAPIWAIT in [Table 3 on page 47](#) before proceeding.)
  - i) EP module name
  - ii) APAR level (the APAR number and last few letters of the changeID string)
  - iii) RET address (this is register 14)
  - iv) EPA address

If the module name is UNKNOWN and the module save area set begins with DFSDLA00, the EPA address can probably be resolved in the DL/I region dump by using IPCS BROWSE mode for the DL/I ASID.

c. The offset from which DFSIWAIT, DFSISERW, or DBFXSL was invoked from the calling module.

You can calculate the offset by subtracting the EPA address in the save area **before** the save area pointed to by SAPIWAIT from the RET address of the save area pointed to by SAPIWAIT.

The following table shows key data from an abnormal save area set.

Table 4. Key data from an abnormal save area set

EP module name	APAR number	Last few ChangeIDs	RET	EPA	Wait call offset
DFSCST00	PL45938	abcde	80A7BA14	00A8E110	
DFSDBDR0	PL49770	..mnopr	60A8E6D6	00A07A58	
DFSBML00	none		50A07AC2	00B5DAE0	X'10E'
DFSIWAIT	none		40B5DBEE	70A7C7F6	

6. Identify the reason for the WAIT.

To identify the reason for the WAIT, do the following:

- a. Assemble the module that issued the wait. Use the offset obtained in step 5 as an approximate displacement into the module where an IWAIT or ISERWAIT was issued. Examine the code and comments at that point. Most modules give the reason for the IWAIT in the comments above the IWAIT issue point.

The EP name might not be the actual module name, but rather a CSECT within a module. To find the actual module name, using IPCS BROWSE mode, scan backwards from the EPA address for the actual module name.

7. Repeat steps 5 and 6 for the first three abnormal save area sets you found.

You should be able to gather enough information from the first three abnormal save area sets to perform a search or determine the cause of the problem.

## Keyword: WAIT

At this point, you can be sure that you are in an IMS WAIT. Therefore, WAIT is an appropriate keyword for the search argument.

## Keyword: module name issuing IWAIT or ISERWAIT

The Module Name column in your worksheet indicates the modules that issued the IWAITs. These modules can provide useful search arguments. Use the 8-character module name for this keyword.

## Keyword: WAIT reason

The IWAIT REASON column in your worksheet indicates the reason or resource, or both, that is causing the IMS WAIT.

For example, if the reason was a WAIT for the DPST latch, the IWAIT REASON keyword is DPST LATCH.

## Keyword: additional related keywords

External events might trigger WAITs. These events might be indicated by console messages, or they might be related to a procedure that was being performed at the time the WAIT began.

You can use each of these additional keywords in the search argument when applicable.

## Search argument example

Consider this scenario:

- IMS went into a IWAIT after a WADS write error occurred.
- Multiple unusual save area sets were found from module DFSFLLG0.
- The reason for the IWAIT was found to be the LOG LATCH.

The broad search argument to use is:



```
5655J3800 WAIT LOG | LATCH | W ADS | DFSFLLG0
```

For a structured database search, use this search argument:

```
PIDS/5655J3800 WAIT PCSS/LOG | PCSS/LATCH | PCSS/WADS | RIDS/DFSFLLG0
```

With this search argument, you might receive numerous search results, which will probably contain the APAR describing your problem. You can then take various combinations of the additional keywords that were compared with the **OR** operator in the above example and use the **AND** operator on the keywords instead. You can use this technique to narrow your field of search until you find the appropriate APAR.

## PST analysis

This section deals with analyzing regions for possible problems in scheduling, intent conflicts, and so forth.

1. Determine the number of active regions.

SCDREGCT at SCD+X'C8A' is a 2-byte field that contains the number of active regions, if any.

If SCDREGCT = X'0000', no regions are active. Go back to [“Determine the type of WAIT or LOOP that is in progress” on page 44.](#)

If SCDREGCT is not equal to X'0000', go to step 2.

2. Determine if the scheduler sequence queues (SSQs) have any entries.

Obtain the address of the transaction anchor block (TAB) from the SCDTAB field in the DSECT (label TABEP in the formatted dump). The TAB, which is mapped by DSECT DFSTAB, consists of:

- TAB header
- Headers for each of the six subqueues (SSQ1 - SSQ6)
- Class vector table (CVT)
- Transaction class tables (TCTs)

If the count of partition specification tables (PSTs) waiting on any subqueue (field TABSCHQC) equals 0, no region should be waiting on any subqueue. However, you should also check each subqueue header. Calculate the address of the subqueue header for a specific subqueue (SSQ#) as follows:

- a.  $SSQ\# \times X'18' - X'8' = \text{offset of header for SSQ\#}$
- b.  $\text{Offset of header for SSQ\#} + \text{SCDTAB address} = \text{address of header for SSQ\#}$

Perform this calculation for each subqueue number. If field TABSSQ $n$ F, where  $n$  is the subqueue number, is not zero, this field contains the address of an entry on the SSQ for the specified subqueue.

- a. The SSQ consists of the following six subqueues. All subqueues are formatted in a dump.

### Subqueue 1

Reserved for future use.

### Subqueue 2

JMP region waiting for work.

### Subqueue 3

MPP region waiting for work.

### Subqueue 4

MPP/JMP region waiting for intent.

### Subqueue 5

BMP/JBP region waiting for intent.

### Subqueue 6

MPP/BMP/JMP region waiting for input.

- b. Each subqueue represents a resource. A PST enqueued on a subqueue is waiting for that resource.

- c. The TAB and SSQs are formatted after the SCD LATCH EXTENSION in an IMS formatted dump, as follows:

```

**TAB - TRANSACTION ANCHOR BLOCK**
0D1873B0          005800FF 00000000      *          .....*
0D1873C0  0000000E 00000000 00000000 00000000      *.....*
0D1873D0  00000000 00000000 00000000 00000000      *.....*
      LINES      0D1873E0-0D1873EF  SAME AS THE ABOVE
0D1873F0  00000000 00000000 0CF18544 0CF00C40      *.....1...0.*
0D187400  00000000 00000000 00003614 00000000      *.....*
0D187410  0CF18C40 0CF18C40 00000000 00000000      *.1. .1. ....*
0D187420  00003AEB 00000000 00000000 00000000      *.....*
0D187430  00000000 00000000 0000396E 00000000      *.....*
0D187440  00000000 00000000 00000000 00000000      *.....*
0D187450  000010B4 00000000 0D187858 0D1878B0      *.....*
0D187460  0D187908 0D187960 0D1879B8 0D187A10      *.....*
0D187470  0D187A68 0D187AC0 0D187B18 0D187B70      *.....*
.....
.....
.....
.....

***SCHEDULER SEQUENCE QUEUES***

DFSPSTQE  00000000      SUBQ  1          NOT ACTIVE
                        SUBQ  2          NOT ACTIVE
                        SUBQ  3          NOT ACTIVE
                        SUBQ  4          NOT ACTIVE
                        SUBQ  5          NOT ACTIVE
                        SUBQ  6          NOT ACTIVE

```

- d. If the words NOT ACTIVE follow the subqueue entry, no PSTs are enqueued on that entry.
- e. If entries are listed for subqueue 3, go to [“No work to do” on page 53](#).
- f. If no entries are listed for subqueue 3, go to step 3.
3. Are there subqueue 4 or 5 entries?
- Subqueue 4 does not apply to a DBCTL environment.
- Entries on subqueue 4 or 5 are waiting for intent conflicts to be resolved.
- a. If entries are listed for subqueue 4 or 5, go to [“Intent conflict” on page 55](#).
- b. If not, go to step 4.
4. Are there subqueue 6 entries?
- This step does not apply to a DBCTL environment. Continue with the next step.
- Entries on subqueue 6 are waiting for input.
- a. If there are entries listed for subqueue 6, go to [“WAIT for input” on page 56](#).
- b. If there are no entries, go to step 5.
5. Are all regions accounted for?
- Compare the number of regions in the SCDREGCT (SCD+X'C92') with the number of regions enqueued on the subqueues. (The SCDREGCT is 2 bytes.)
- a. If the numbers of regions are equal, go to step 6.
- b. If the numbers of regions are not equal, all regions are unaccounted for. Go to the analysis for [“PST analysis” on page 51](#).
6. Report the problem.
- This problem occurs when there are entries queued on the subqueues and no reason can be found to prevent their scheduling, but nothing schedules. Report the problem to the IBM Support Center.

## PST active

You reach this point in the analysis either when:

- The SCDREGCT field is not equal to zero, and there are no entries on the Scheduler Sequence Queues, or
  - No problem was found in analyzing the PSTs on the subqueues, and the number of PSTs on the subqueues is less than that in the SCDREGCT field.
1. Locate the PSTs.  
Find the stack of dependent region PSTs in the dump. (Two stacks of PSTs exist in the dump. System PSTs are printed separately from the dependent region PSTs.)
  2. Is the PST scheduled?
    - a. Find all the PSTs with PSTTERM (X'1BC') = X'02' (ACTIVE) and PSTCODE1 (X'B7A') = X'10' (SCHEDULED).
    - b. Ignore the PSTs without the SCHEDULED bit on.
  3. For the scheduled PSTs, do SAP analysis.
    - a. PST at offset minus X'04' (field name PTR) is usually the SAP address. (The PTR field is the last entry on the line above the X'0000' line in the dump.) If not, PST + X'5B8' (PSTSAV1) is the address of the first Save Area in a set, and WD1 in that Save Area is the address of the SAP.
    - b. Go to [“Determine the type of WAIT or LOOP that is in progress” on page 44](#). Return here after doing SAP analysis for the scheduled PSTs only.
  4. Are there any ACTIVE non WAITING SAPs?
    - a. If any of the SAPs are marked ACTIVE go to step 5.
    - b. If SAPs are found WAITING, use normal SAP analysis to report the problem. Use the search argument format [“Search argument example” on page 50](#).
  5. Is the dependent region active within an IMS save area set?
    - a. If SAP + X'08' (SAPCNTRL) = X'10', this region is in a DL/I call within IMS. Go to step 6.
    - b. Otherwise go to step 7.
  6. Analyze the region dump.  
You must analyze the region dump using the PSW address to identify the problem. Refer to [“WAIT/LOOP procedure” on page 44](#), steps 6c and 6d.
  7. Determine what the application program is doing.  
You must analyze the region dump using the PSW address to identify what the application program is doing.  
  
In a DBCTL environment, you must analyze the CCTL region dump using the PSW address to find out what the DRA, CCTL, or application program is doing. Refer to [“WAIT/LOOP procedure” on page 44](#), steps 6c and 6d.
  8. Determine the reason the latch is not freed.  
If a latch is being waited for, and the owner is not waiting for I/O, use SAP analysis to identify the reason for the WAIT.

## No work to do

This section does not apply to a DBCTL environment.

You came to this point because subqueue 3 contains PSTs.

1. Locate the PSTs on subqueue 3.

The addresses under the field name SQPSTADD are the PST addresses. In the formatted dump, the PSTs start with the eye catcher \*\*\* DB PST AREA \*\*\*. Locate the PSTs that are on subqueue 3.

2. Find the classes the PSTs can execute.

PST + X'C68' (PSTCLASS) is an 8-byte field. Each byte indicates a class transaction that the PST is allowed to process. For example, if PSTCLASS = 00010003 00050006, the PST can process classes 0001, 0003, 0005, and 0006.

3. For each PST on subqueue 3, locate the transaction class table (TCT) for each class that the PST can process. There is one TCT for each class.
  - a. Obtain the TAB address from the SCDTAB. SCD+B88 points to SCDTAB and is labeled TABEP in IMS Dump Formatter.
  - b. Take the first PSTCLASS value and subtract 1.
  - c. Multiply this result by 4.
  - d. Add this value to the TABCLASS offset value + X'A0'.
  - e.  $TCT = 4 \times (\text{first PSTCLASS value} - 1) + X'A0'$ .

When the high-order byte contains a X'80' this indicates the TCT class is not active.

4. Can any SMBs be scheduled?

$TCT + X'04' = \text{zero}$  or the address of an SMB that can be scheduled.

- a. If zero, no SMBs can be scheduled. Go to step 7.
  - b. If SMBs can be scheduled, locate the SMBs and then go to step 5.
5. Is SMB locked or stopped?
  - a. If  $SMB + X'24'$  (SMBSTATS) = X'10' (STOPPED) or X'08' (LOCKED), go to step 6.
  - b. Otherwise, go to step 9.
6. Are there any more SMBs on this class?
  - a. If  $SMB + X'04'$  (SMBQEFP) is not equal to zero, it is the address of the next SMB. Move on to the next SMB and repeat step 5.
  - b. If  $SMB + X'04'$  (SMBQEFP) = zero, there are no more SMBs. Go to step 7.
7. Are all classes accounted for?
  - a. If all classes found in PST + X'C68' (PSTCLASS) are not accounted for, repeat step 4 for each remaining class.
  - b. Otherwise, go to step 8.

8. Are all regions accounted for?

To determine whether all regions are accounted for, use SCDREGCT (SCD + X'C8A'). The SCDREGCT is 2 bytes. There is one PST for each region.

- a. If the number of PSTs on subqueue 3 is equal to the SCDREGCT and they have been examined and accounted for, there are no transactions scheduled for the regions. This is a normal WAIT, and there is no work for IMS to perform. This is not a problem.
  - b. Otherwise, go back to 3 to continue the scheduler queue analysis.
9. Locate the PSB directory (PDIR).

If the SMB is not locked or stopped, locate the PDIR:  $SMB + X'3C'$  (SMBPDIR) = address of the PDIR.

10. Can PDIR schedule?

Locate the PDIR entry. When any of the following bits are ON, the PDIR is unable to schedule.

**PDIR + X'20' (PDIRCODE) =**  
 X'40'X'10'X'08'X'02'

- a. If the PDIR cannot schedule, go back to step 6.
  - b. Otherwise, go to step 11.
11. Is PDIR marked parallel?
  - a. If the PDIR is marked scheduled but not parallel:

```
PDIR+X'20' (PDIRCODE) = X'04' (Scheduled)
and:
PDIR+X'21' (PDIROPTC) is not equal to X'04' (Not parallel)
```

If there are entries listed for subqueue 6, go to “WAIT for input” on page 56 to determine if any of the waiters on subqueue 6 are pseudo WFIs scheduled against the same PDIR. If there is a pseudo WFI scheduled against the same PDIR, report the problem to the IBM Support Center.

If there are no entries listed for subqueue 6 or none of the waiters on subqueue 6 point to the same PDIR, go back to step 6.

b. If marked parallel (PDIR +X'21' = X'04'), go to step 12.

#### 12. Are enough messages enqueued for another PST?

If the PDIR is marked parallel, check if enough messages are enqueued on the SMB to schedule another PST.

a. You do this by finding:

i) SMB+X'46' (SMBPARLM) = number of messages per region (2 bytes).

ii) SMB+X'44' (SMBRGNS) = number of message regions scheduled for the SMB (2 bytes).

iii) SMB+X'1A'(SMBENQCT) minus SMB +X'18' (SMBDEQCT) = number of messages currently enqueued. (To find the number currently enqueued, subtract the messages dequeued from those enqueued.)

b. If the number of messages currently enqueued (step 12a3) is greater than the number of messages per region (step 12a1) multiplied by the number of message regions scheduled (step 12a2), there are enough messages enqueued on the SMB to schedule another PST. Go back to step 6.

c. Otherwise, go to step 13.

#### 13. Report the problem.

At this point, regions are waiting, enqueued on subqueue 3 with transactions that can be scheduled. Report the problem to the IBM Support Center.

## Intent conflict

You reach this point by having entries on subqueue 4 or 5.

An intent problem is indicated when the PST is on the intent queue.

#### 1. Locate the PSTs that are on subqueue 4 or subqueue 5, or both.

The addresses under the field name SQPSTADD are the PST addresses. To analyze the INTENT CONFLICT fields in a PST, you must locate the PST in the unformatted section of the dump.

#### 2. Is the PSB work pool too small?

a. If PST + X'B7A' (PSTCODE1) = X'06', the PST is on the PSB WAIT queue for pool space. The PSB work pool is too small. You must increase the size of the PSBW parameter in the DFSPBxxx member.

b. Otherwise, go to step 3.

#### 3. Is the Data Management Block (DMB) pool too small?

a. If PST + X'B7A' (PSTCODE1) = X'20', the DMB pool is too small. You must increase the size of the DMB parameter in the DFSPBxxx member.

b. Otherwise, go to step 4.

#### 4. Can intent be satisfied?

a. If PST + X'B7A' (PSTCODE1) = X'40', the intent cannot be satisfied. Go to step 6.

b. Otherwise, go to step 5.

5. Is the region scheduled?

a. If any PST has the following:

- PST +X'B7A' (PSTCODE1) = X'10'(SCHEDULED)
- and:
- PST +X'1BC' (PSTTERM) = X'02'(ACTIVE)

the region is scheduled, and this a normal WAIT for subqueue 4 and subqueue 5. Usually this is not a problem. Go back to the subqueue 6 entry of [“PST analysis” on page 51](#), step 4 and continue.

b. Otherwise, go to step 7.

6. There is an intent conflict.

If you reach this point, there is an intent conflict. Usually, the intent conflict is caused by a PSB having the exclusive option. This option is defined during the PSBGEN. See the PSBGEN section of *IMS Version 15.5 System Utilities*. If the exclusive option did not cause the intent conflict, report the problem to the IBM Support Center.

7. Report the problem.

If you reach this point, the problem is that the last region to terminate should have posted the PST on subqueue 4 and subqueue 5 and did not. In a DBCTL environment, the last thread to unschedule a PSB did not post subqueue 4 or 5. Thus, there is a WAIT with a PST on subqueue 4 or subqueue 5 with no scheduled regions. Use subqueue 4 or subqueue 5 in your search argument, or report the problem to the IBM Support Center.

## WAIT for input

You can reach this point only by having entries on subqueue 6.

1. Find the PSTs on subqueue 6.

The addresses under the field name SQPSTADD are the PST addresses. The PSTs are found in the stack of PSTs.

2. Find Scheduler Message Blocks (SMBs) for the PSTs.

For each PST enqueued on subqueue 6, find the related SMB: PST +X'C4' (PSTSMB) = address of the SMB.

3. Are any of the regions on subqueue 6 pseudo WFIs?

- If SMB+X'27' (SMBFLAG3) = X'08' (WFI transaction), the region is not a pseudo WFI.
- If the region is a pseudo WFI, check if the region is holding any resources needed by transactions waiting to be processed.

4. Are any messages enqueued on SMB?

There should be no messages enqueued on the SMB.

- SMB+X'1A' (SMBENQCT) minus SMB+X'18' (SMBDEQCT) = number of messages enqueued
  - If there are messages enqueued on the SMB, go to step 6.
  - If no messages are enqueued, go to step 5.

5. Are all regions accounted for?

Compare the count of regions enqueued on the subqueues with the count in SCDREGCT (SCD + X'C92') (2 bytes).

- If the counts are equal, all regions are accounted for, and the IMS regions are in a normal scheduling environment. The problem is not with scheduling.
- If not equal, other regions are active in IMS. Go to [“PST active” on page 52](#).

6. Report the problem.

The problem is that IMS messages are enqueued on the SMB and wait-for-input (subqueue 6) is not posted. Report the problem to the IBM Support Center.

## Loop

Use standard z/OS system diagnostic procedures for loops.

Using the RB found in step 6c of [“WAIT/LOOP procedure”](#) on page 44, determine the PSW address. The PSW address is labeled OPSW. The PSW address is always the second word following the label. This PSW address belongs to one of the modules involved in the loop.

You can use the z/OS system trace to examine entries for the ASID and TCB indicated in the Dispatch Work Area at step 5 of the [“WAIT/LOOP procedure”](#) on page 44. The PSW address in the system trace entries indicates the modules involved in the loop.

Locate the PSW addresses in the storage section of the dump and scan backward through the eye catchers on the right side of the dump until you find a module identifier.

The looping module might not be an IMS module. Sometimes, the addresses are in the Link Pack Area (LPA) or the nucleus and might require an LPA or nucleus map.

## Create the search argument

You can use the following additional keywords in the search argument to narrow the search, but they might not be necessary.

### Keyword: LOOP

At this point, you can be sure that you are in a loop situation. Therefore, LOOP is an appropriate keyword for the search argument.

### Keyword: module names involved in the loop

The module names derived in the loop procedure above are also valid keywords.

### Keyword: label in module

If it is a tight loop, labels from the assembly listing of the modules involved might be useful keywords.

### Keyword: additional related keywords

External events can trigger loops. These events might be indicated by console messages or be related to a procedure that was being performed at the time the LOOP began.

## Search argument example

Consider the following scenario:

- IMS went into a loop.
- The active modules indicated in the RB chain and the z/OS system trace table were DFSCFEI0 and DFSCFE00.
- The loop began after the operator issued a /DISPLAY NODE command.

The broad search argument to use is:

```
5655J3800 LOOP DFSCFE00 | DFSCFEI0 | DISPLAY | NODE
```

For a structured database search, use this search argument:

```
PIDS/5655J3800 LOOP RIDS/DFSCFE00 | RIDS/DFSCFEI0 | PCSS/DIS | PCSS/NODE
```

With this search argument you might receive numerous hits, which will probably contain the APAR describing your problem. You can then take various combinations of the additional keywords that were compared with the **OR** operator in the above example and use the **AND** operator on them instead. You can use this technique to narrow the field of search until you find the appropriate APAR.

If the loop was not in an IMS module, do not use the IMS component ID, 5655J3800.

## System wait

Use standard z/OS systems diagnostic procedures.

If the PSW address is for a system module, include that information when reporting the problem. You can use the module name in your search along with the WAIT keyword.

## Shutdown processing

Use this analysis if the operator issued a /CHECKPOINT FREEZE, DUMPQ, or PURGE to IMS and IMS failed to come down normally. Before taking IMS out of the system, be sure to use a /DISPLAY SHUTDOWN STATUS command. Obtain the listing of the /DISPLAY command and any subsequent activity to find any unusual conditions that might have prevented an orderly termination of IMS.

You should also use this analysis if IMS shut itself down and failed to terminate normally. For example, when IMS runs low on message queue space, it shuts itself down.

Before starting this procedure, you need to obtain an IMS dump in order to examine bit settings. Be aware that if you received only the first part of the DFS994I message during shutdown processing, VTAM might be involved in the failure. (For a DBCTL environment, ignore any further instructions that refer to VTAM in this topic and in the next topic, [“Shutdown analysis \(CHE FREEZE, DUMPQ, or PURGE\)”](#) on page 59.) If you received the DFS994I xxx (FREEZE, DUMPQ, PURGE), but not DFS994I IMS SHUTDOWN COMPLETED, be sure to obtain a dump of VTAM and IMS. Here are two ways to get a dump:

- Enter the z/OS DUMP command to dump the VTAM address space and then modify IMS down with a dump.
- Enter the z/OS DUMP command to dump the VTAM, IMS control, DL/I, and CCTL address spaces, and then modify IMS down without a dump.

Be sure to include the RGN option along with the other standard SDATA defaults in the DUMP command.

In the section "Shutdown Analysis" that follows, note the following:

- Displacements and test conditions can change when maintenance is applied to a system.
- The bit settings shown are cumulative. This means that they usually combine with any bits already set in the byte. Check the bit settings as described. If a bit was not set or reset as shown, include both the module name and the cumulative bit settings in each byte in your search argument.
- SET turns the bit ON. RESET turns the bit OFF. Other bits in the byte might already be ON.
- It is essential in using the following analysis to find out if the indicated bits were SET or RESET and to use only the DUMPQ/FREEZE or PURGE sections where applicable.
- The Save Areas (SAs) might not always identify the last module to have control. In some cases, control is passed back to the initiating module (such as DFSCST00), and you can find no trace of any lower modules in the SAs.
- The main control block in shutdown problem analysis is the system contents directory (SCD). This flow of control lists most of the modules involved. When you find a field that does not have the bits SET or RESET as indicated, stop the analysis and report the problem.
- Be aware that defective code can produce results that appear to contradict this information.
- The following analysis does not list every action that is taking place in IMS shutdown processing, but only activity that causes bit setting to be changed in key SCD fields.



- Comments scattered throughout the analysis are for information only. For example, the statement, "If input or output is pending, return to DFSICIO0 with RC=C to complete", is for information. Do not look at return codes, but examine only the bit settings.

## Shutdown analysis (CHE FREEZE, DUMPQ, or PURGE)

Remember that in this analysis you will be looking at bit settings, not hexadecimal values.

These sections do not apply to DBCTL shutdown:

- PURGE
- DFSICL20
- DFSICLX0
- DFSICIO0
- DFSIPCP0
- DFSCPCP0
  - DFSICL20
    - If PURGE, then set SCDCKCTL(X'C08') = X'34' and then set SCDSTOP1(X'C02') = X'80'
    - If not PURGE, then:
      - If DUMPQ, set SCDCKCTL(X'C08') = X'1C'
      - If FREEZE, set SCDCKCTL(X'C08') = X'14'
        - Reset POLL the lines and then (not applicable to DBCTL)
        - Set SCDSTOP1(X'C02') = X'C0' (for DBCTL, set AWE to TRM1)
  - DFSICLX0
  - DFSICIO0
  - DFSIPCP0
    - If SCDCFLG1(X'AC7') = X'08', then
      - Set SCDCQFLG(X'AC8') = X'04' and
      - Set SCDCNXW4(X'ACF') = X'40'
    - If input or output is pending, return to DFSICIO0 with RC=C to complete.
    - When there is no input or output pending, or when the input or output is finished, then:
      - Set SCDCPCTL(X'AC4') = X'80'
      - Set AWE to TRM1
  - DFSCST00
  - DFSTRM00

### For PURGE

- AWE = TRM1, First phase of termination
- If SCDIDCNT +1 (X'BC8') is not equal to X'000000' and SCDCKCTL(X'C08') = X'20' (PURGE):
  - Set SCDSTOP1(X'C02') = X'10'
  - Set SCDSTOP1(X'C02') = X'02'
- If SCDFTFLG(X'290') = X'20' (Fast Path active), DBFTERM0 posts the Fast Path regions for SHUTDOWN
- DFSTRM00

**For DUMPQ or FREEZE**

- If SCDIDCNT+1(X'BC8') is not equal to X'000000' and SCDCKCTL(X'C08') is not equal to X'20' (Not PURGE)
  - Set SCDSTOP1(X'C02') = X'04'
  - Set SCDSTOP1(X'C02') = X'02'
- If SCDFTFLG(X'290') = X'20' (Fast Path Active), DBFTERM0 posts the Fast Path regions for SHUTDOWN

**For DUMPQ, PURGE, or FREEZE**

- If Fast Path was active on return from DBFTERM0, or if Fast Path was not active, and SCDREGCT(X'C8A') is not equal to X'0000' (ACTIVE REGIONS), then post the PSTs waiting in the scheduler.
- If SCDSHFL1(X'3A4') = X'80' (IRLM in system) or SCDIDCNT+1(X'BC8'), or both, is not equal to X'000000' then return to DFSCST00 to wait for regions to end, If DBCTL, notify DRA before returning to DFSCST00.
- When or if SCDIDCNT+1(X'BC8') = X'000000' (REGIONS ENDED), set SCDSTOP1(X'C02') = X'01'.

**For PURGE only**

- If SCDCKCTL(X'C08') = X'20' (PURGE)
- Set SCDSTOP1(X'C02') = X'20'
- IWAIT for all output to go.

**For DUMPQ, PURGE, or FREEZE**

When all output is done for PURGE or FREEZE or DUMPQ, then:

- If SCDFTFLG(X'290') = X'20' (Fast Path active), DBFTERM1 closes the areas.
- If SCDFTFLG(X'290') is not equal to X'20' or when Fast Path areas are closed then:
  - If SCDSMMS1(X'033') = X'02' (DLI SAS), then:
    - Tell the DL/I region to close the databases (DFSSDL40).
    - IWAIT for the databases to close.
  - If not DLI/SAS, then let DFSDLOC0 close the databases.

Then when all databases and areas are closed: Set SCDSTOP1+1(X'C02') = X'04'.

- DFSCPCP0  
Set return code (RC) = 8 to ask DFSIPCP0 if communication is still going on.
- DFSIPCP0 (DFSIPCP2)
  - If no output or no messages on Q3, set return code (RC) = 0 to inform DFSCPCP0.
  - If output or messages on Q3, set return code (RC) = 4 to inform DFSCPCP0, which causes DFSCPCP0 to IWAIT.
- DFSCPCP0
  - If output is pending (RC = 4)
    - Set SCDCPCTL(X'AC4') = X'08'
    - Set SCDSTOP1(X'C02') = X'40'
    - IWAIT for DC to finish.
  - If no output or when output finishes
    - Set off SCDCPCTL(X'AC4') = X'08' (reset the bit)
    - Set SCDSTOP1+1(X'C02') = X'08'
    - Reset Poll all lines that are candidates for the SHUTDOWN message

- Set CTBFLAG3(0D) = X'10' (for all terminals that are to receive the shutdown message)
- DFSICLX0
- DFSICIO0
- DFSIPCP0
  - If any CTBFLAG3(0D) = X'10':
    - Set CTBACTL(10) = X'20'
    - Set CTBACTL(10) = X'10'
    - RC = 8 to DFSICIO0 (send SHUTDOWN message)
  - If NO CTBFLAG3(0D) = X'10':
    - Set SCDDFLGS(X'718') = X'80'
    - Set SCDCPCTL(X'AC4') = X'20'
    - RC = 4 to DFSICIO0 (quiesce lines)
- DFSICIO0
  - If RC = 4, idle the lines
  - If RC = 8, send DFS991 - IMS SHUTDOWN message
  - The WRITE interrupt from the SHUTDOWN message results in the following:
    - Set off CTBFLAG5(0F) = X'80' (reset)
    - Set off CTBFLAG3(0D) = X'10' (the)
    - Set off CTBACTL (10) = X'30' (bits)
- DFSIPCP0
 

When all line activity is stopped
- DFSCPCP0
- DFSTRM00
  - If DBCTL set SCDSTOP =SCDSTSNT, then set SCDSTOP1+1(X'C02') = X'01'
- DFSRCRT0
- DFSRCP00
  - Send "DFS994I \*CHKPT yyddd/hhmmss\*ctype" (first part of DFS994I message)
  - Set AWE = "TRM2"
  - Set off SCDCKCTL(X'C08') = X'04' (reset the bit)
- DFSTRM00
 

Set SCDTRMFL(X'430') = X'40'
- DFSCST00
- DFSTRM00
  - If DLI/SAS SCDSMMS1(X'033') = X'02', pass AWE to DFSSDL40 to begin Normal Termination
  - If not DLI/SAS or when DFSSDL40 returns
  - If SCDRFPIN(X'C32') = X'80' (Fast Path errors):
    - Print error message
    - Set off SCDRFPIN(X'C32') = X'80' (reset the bit)
    - Close queue data sets (not applicable to DBCTL)
    - IWAIT for closing
    - Set off SCDSTOP1(X'C02') = X'08' (reset the bit)
- DFSTERM0

- Terminate DASD log
- Set off SCDRECTL(X'146') = X'80' (reset the bit)
- Terminate RDS
- Terminate IMS system type tasks
- Signoff DBRC
- Quit IRLM
- Close VTAM ACB (not applicable to DBCTL)
- If DLI/SAS, SCDSMMS1(X'033') = X'02' and the ECB at SCDRSETF(X'D1C') is not equal to X'40' (posted) :
  - IWAIT for the DL/I region to end
  - Set AWE = "TRM3"
  - Set SCDTRMFL(X'430') = X'20'
  - Send "DFS994I IMS SHUTDOWN COMPLETED" (second part of DFS994I message)
- DFSTRM00
- DFSCST00

## IRLM procedure

WAIT states can be encountered during IRLM processing in four areas:

- [“Deadlock involving non-IRLM resources” on page 62](#)
- [“Deadlock involving only IRLM resources” on page 62](#)
- [“Lock request not granted because holder did not release lock” on page 63](#)
- [“IRLM latch unavailable” on page 63](#)

## Deadlock involving non-IRLM resources

### **Failure Description**

Application programs waiting for non-IRLM resources and holding IRLM resources are waiting for other applications also holding IRLM resources. The IRLM cannot detect deadlocks involving non-IRLM resources.

### **Detection**

Use the IMS WAIT diagnostic procedures to discover the non-IRLM resources being waited for. Follow the RLB chains representing resources held or requested for each requesting work unit (WHB) to discover the IRLM resources being waited for. If the wait state occurred as a result of an IRLM error, the function/subfunction is IRLM/DEADLK.

An example of a search argument is:

```
569516401 AR101 WAIT IRLM IRLM/DEADLK
```

For a structured database search, use this search argument:

```
PIDS/569516401 LVLS/101 WAIT RIDS/IRLM RIDS/DEADLK
```

## Deadlock involving only IRLM resources

### **Failure Description**

Application programs are deadlocked for IRLM resources. If all the application programs are waiting for IRLM resources (there are no application programs running which could release the locks that the

other application programs are waiting for), this is a deadlock. The IRLM should detect this condition and post one of the waiters as unable to obtain the lock because of a deadlock.

### **Detection**

Follow the RLB chains representing resources held or requested for each requesting work unit (WHB) to discover the IRLM resources being waited for. If the wait state occurred as a result of an IRLM error, the function/subfunction is IRLM/DEADLK.

An example of a search argument is:

```
569516401 AR101 WAIT IRLM IRLM/DEADLK
```

For structured database search, use this search argument:

```
PIDS/569516401 LVLS/101 WAIT RIDS/IRLM RIDS/DEADLK
```

## **Lock request not granted because holder did not release lock**

### **Failure Description**

An application program requested a lock, but the request was not granted because the holder of the resource did not release it. This does not result in a deadlock. However, If the requester is not timed out, its task and any others waiting after it might enter a wait state.

An example of a search argument is:

```
569516401 AR101 WAIT IRLM
```

For structured database search, use this search argument:

```
PIDS/569516401 LVLS/101 WAIT RIDS/IRLM
```

## **IRLM latch unavailable**

### **Failure Description**

An error in IRLM processing can result in an IRLM latch being permanently unavailable. If this condition exists, no new IRLM requests can be processed.

If this error occurs, call the IBM Support Center for help in diagnosing the problem. The support representative will tell you what type of documentation to gather.

### **Related tasks**

[“Receive-any buffer analysis” on page 321](#)

While talking with level 1 or level 2 IBM Software Support representatives, you might need to determine if you are out of receive-any (RECANY) buffers. Use either the IMS IPCS panel interface or the manual procedure to help you determine if this is the case. As you proceed through the steps, write down the information you gather.

## **IMS keyword dictionary**

You can use free-form searches to retrieve the RETAIN records that contain all the search keywords that you specify. The IMS keyword dictionary describes the conventions for free-form search keywords.

The following table describes the IMS keywords.

*Table 5. IMS keyword dictionary*

Category/keyword	Examples	RETAIN keyword
Abends	System 0C4 User 0845	ABEND0C4 ABENDU0845

Table 5. IMS keyword dictionary (continued)

Category/keyword	Examples	RETAIN keyword
Access methods	OSAM VSAM	OSAM VSAM
Automatic Operator Interface		AOI
APARs		PL12345
Checkpoint processing	Checkpoint Extended Checkpoint	CHKPT XCHKPT
CICS interface		CICS
IMS commands (sample list)	/ASSIGN /CHECKPOINT /ERESTART /TRACE /STOP	CMDASS CMDCHE CMDERE CMDTRA CMDSTO
DBRC commands	INIT.RECON CHANGE.PRILOG	INITRECON CHANGEPRILOG
Condition code		X'CC08' (hexadecimal)
Control blocks	Data Control Block Database Descriptor	DCB DBD
Database organization		HDAM
Database pre-open		PRE-OPEN
Data-sharing environment		DATA SHARING
Devices	3270 LU TYPE1	D/T3270 SLU1
DL/I address space		DLISAS
DSECTs		IDSPWRK
Emergency restart processing		ERE
Error codes (DBRC)		EC0182062
Extended restart		XRST
Fast Path	Fast path area Second CI Main storage database Sequential dependent	FASTPATH FPAREA DMAC MSDB SDEP
Feedback code		FDBK0C (HEX)
Fields	PSTUSID	PSTUSID
Function sub-function		SYS CHKRT
Function codes		FC0291
System definition	ACB NUCLEUS	ACBGEN NUC
IRLM		IRLM
Labels	LOOPNEXT FREEMAIN	LOOPNEXT FREEMAIN

Table 5. IMS keyword dictionary (continued)

Category/keyword	Examples	RETAIN keyword
Log records	TYPE 18 TYPE 67FF	TYPE18 TYPE67FF
Macros	RWOS TERMINAL	RWOS TERMINAL
Master Terminal Operator		MTO
Messages	DFS045I IEC030I	MSGDFS045I MSGIEC030I
Modules	DFSPCC20	DFSPCC20
Online change		OLC
Online data set		OLDS
Online image copy		OLIC
Parameters	ERROPT=ACCEPT	ERROPT=ACCEPT
Processing options	PROCOPT=GO	PROCOPT=GO
Publication numbers	GC18-9717	GC189717
Reason codes Use RSN as the prefix and use the hexadecimal reason code of any length. Do not include leading zeros.		RSN08 (HEX)
Registers	General purpose registers Control registers Floating point registers	REG13 (DECIMAL) CREG10 FPREG01
Restart processing		RSTRT
Return codes: Minimum of 2 digits. If there are more than 2 digits, do not include leading zeros.	Return code 12 (X'0C')	RC0C
Sense codes	Sense 080B	SNS080B
Status codes	Status code GE Status blank BLANK	STATUSGE STATUS4040
Sub-code		SUBCODE101
SVC numbers		SVC255 (DECIMAL)
Trace entry function code		TRACEE6 (DL/I) TRACE03 (DISP)
XRF environments	XRF Takeover Alternate	IMSXRF TAKEOVER ALTERNATE

## Dependency keywords

Dependency keywords can be used with the keyword string to select only those APARs that apply to a certain environment. These keys are useful when a search yields a large number of hits and you suspect that the program failure occurs only in a specific environment.

Keyword	Environment	Keyword	Environment
D/CICS	CICS	D/TRKREC	Track Recovery

Keyword	Environment	Keyword	Environment
D/CONVPROC	Conversational Processing	D/TWX	Teletype
D/FP	Fast Path	D/UCF	Utility Control Facility
D/GSAM	GSAM	D/VSAM	VSAM
D/HDAM	HDAM	D/VTAM	VTAM
D/HIDAM	HIDAM	D/1050	1050 Device Type
D/IRLM	MS/VS Resource Lock Manager (Intersystem Communication)	D/2260	2260 Device Type
D/MFS	Message Format Service (MFS)	D/2770	2770 Device Type
D/MSC	Multiple System Coupling	D/2980	2980 Device Type
D/MVS	z/OS	D/3270	3270 Large Screen
D/None	No dependencies	D/3270L	3270 Local
D/OSAM	OSAM	D/3270R	3270 Remote
D/SB	Sequential Buffering	D/3274	3274 Device Type
D/SECINDX	Secondary Index	D/3276	3276 Device Type
D/SHISAM	Simple HISAM	D/3278	3278 Device Type
D/SLU1	VTAM Type SLU 1	D/3279	3279 Device Type
D/SLU2	VTAM Type SLU 2	D/3284	3284 Device Type
D/SLU4	VTAM Type SLU 4	D/3286	3286 Device Type
D/SLU P	VTAM Type SLU P	D/3287	3287 Device Type
D/SYSGEN	PTFs that should be applied prior to system definition	D/3350	3350 Device Type
		D/3375	3375 Device Type
		D/3380	3380 Device Type
		D/3600	3600 Device Types
		D/3790	3790 Device Types

## Searching the database

To find out if a problem like the one you experienced has already been reported, you use the keyword string you completed to search an IBM Software Support database (such as Software Support Facility), or you can use it when talking to your Level 1 support representative.

### Procedure

1. Determine the maintenance level of the IMS system by identifying the APARs and PTFs that have been applied.
  - Run the SMP PTF list program or have access to online SMP/E dialogs.

**Tip:** You can extract the current maintenance level of your IMS system (or of a specific IMS load module) by using the `/DIAGNOSE SNAP MODULE(modname)` command.
2. Search SSF, using the keyword string developed by following procedures from [“Selecting the keywords”](#) on page 34. Your search is most successful if you follow these guidelines:
  - Start with a broad search argument so you receive all problem descriptions that might match your problem.



- If you find too many Authorized Program Analysis Reports (APARs) to examine, add the logical operators **AND** or **OR** to the keyword string in various combinations gradually to reduce the number of database matches (hits). If the keywords are connected by the logical operator **AND** (a blank), a record is selected if it contains both words separated by the blank. If the keywords are connected by the logical operator **OR** (|), a record is selected if it contains either of the words separated by the character, |.
- You can use dependency keywords with the keyword string to select only those APARs that apply to a certain environment. These can be particularly useful when a search yields a large number of database matches and you are almost certain that the program failure occurred in a specific environment. For the list of dependency keywords, see [“Dependency keywords” on page 65](#).

**Recommendation:** Use dependency keywords only if you are sure the problem is limited to that dependency. If you do not get any database matches, eliminate the dependency keyword.

- If you want to narrow the search to a specific release level, you can add the logical operators **AND** or **OR** for the release level keywords to the search argument. The following list describes the release level keywords for IMS 15.6:

**AR100**

IMS Services

**AR101**

Database Manager

**AR102**

Transaction Manager

**AR103**

Extended Terminal Option

**AR104**

Recovery-level Tracking

**AR105**

Database-level Tracking

**AR106**

IMS Java™ On Demand features

**R220**

Internal Resource Lock Manager (IRLM) 2.2

**R230**

Internal Resource Lock Manager (IRLM) 2.3

For a structured database search, the release level keywords are:

**LVLS/100**

IMS Services

**LVLS/101**

Database Manager

**LVLS/102**

Transaction Manager

**LVLS/103**

Extended Terminal Option

**LVLS/104**

Recovery-level Tracking

**LVLS/105**

Database-level Tracking

**LVLS/106**

IMS Java On Demand features

**LVLS/R220**

Internal Resource Lock Manager (IRLM) 2.2

## LVLS/R230

### Internal Resource Lock Manager (IRLM) 2.3

Examples:

- For the Database Manager, type
- For a structured database search, type:

**Tip:** If you do not get any database matches, remove the release level from your search argument.

3. Eliminate the APARs that also appear in the SMP PTF list from the list of database matches. These will have already been applied.
4. Compare each remaining APAR with the current failure symptoms. Analyze trace output for your problem situation, looking for similarities in the situations described by APARs that you are reviewing. Frequently, APAR descriptions include information about the traces that were run for those problems.
5. If you find an appropriate APAR, determine if it has been closed. If it has been closed, you can correct the problem by applying the fix that is associated with the APAR. If it has not been closed, contact IBM Software Support for information about what you can do until it is closed.
6. If you do not find an appropriate APAR, verify that the problem is not caused by a user specification error.
7. If you find no user specification error, contact IBM Software Support for assistance.

## Procedures for preparing an APAR

An Authorized Program Analysis Report (APAR) might be necessary if the keyword search proves unsuccessful. Call IBM Software Support for help in determining if an APAR is necessary. Only authorized IBM personnel can generate APARs.

### About this task

The information in the following table describes the procedures for preparing an APAR.

*Table 6. Procedures for preparing an APAR*

Procedure	Action
Reporting a problem	<p>To report a problem, contact IBM Software Support. Be prepared to supply such information as:</p> <ul style="list-style-type: none"><li>• Customer number.</li><li>• Release level.</li><li>• Current maintenance level (from PTF list).</li></ul> <p><b>Tip:</b> You can extract the current maintenance level of your IMS system (or of a specific IMS load module) from your IMS system by using the /DIAGNOSE SNAP MODULE(modname) command.</p> <ul style="list-style-type: none"><li>• The keyword string or strings used to search the IBM Software Support database.</li></ul>

Table 6. Procedures for preparing an APAR (continued)

Procedure	Action
Gathering APAR documentation	<p>You might be asked to supply various types of information that describe the IMS nucleus, database, environment, or activities. Include applicable items from the following list with the APAR.</p> <ul style="list-style-type: none"> <li>• JCL listings</li> <li>• Address space storage dumps at time of failure—the entire machine-readable dump data set (normally copied to tape) and the JCL used to copy the dump to tape</li> <li>• Bind map</li> <li>• z/OS console printout. A partial console is generally in the offline formatted dump.</li> <li>• Master terminal printout</li> <li>• Local/remote terminal printout</li> <li>• IMS log data sets</li> <li>• IMSGEN listing</li> <li>• DBD listing</li> <li>• PSB listing</li> <li>• ACB generation output</li> <li>• Log trace</li> <li>• Consolidated trace output</li> <li>• Transmittal notes explaining any unusual events leading up to the problem symptoms</li> <li>• SNAPs produced before and after the failing call by DFSDDLTO</li> <li>• Type X'67FF' SNAP log records</li> <li>• Type X'6705' SNAP log records</li> <li>• DBRC RECON data set</li> <li>• LPA map</li> <li>• LOGREC (especially software diagnostic records)</li> </ul>
Submitting APAR documentation	<p>When submitting material for an APAR to IBM, carefully pack and clearly label all materials sent to IBM with the following information:</p> <ol style="list-style-type: none"> <li>1. The APAR number assigned by IBM</li> <li>2. A list of data sets on the tape, including JCL, if any</li> <li>3. A description of how the tape was made, including: <ul style="list-style-type: none"> <li>• The exact JCL listing or the list of commands used</li> <li>• The recording mode and density</li> <li>• Tape labeling</li> <li>• The record format and block size used for each data set</li> </ul> </li> </ol>



---

## Chapter 5. Data areas and record formats

This section describes the major IMS control blocks and their interrelationships. It also describes the formats of records that you need to analyze when diagnosing problems.

### Finding more information on modules, control blocks, and record formats

---

You can find the module directory, IMS control block DSECTs, and the log record formats on Service Link. Contact your systems engineer for further information on accessing Service Link.

#### Control block linkage

The IMS.ACBLIB is a partitioned data set whose members are pre-system-generated, expanded PSB and DMB control blocks. You can view the formats of these control blocks by assembling the database DSECT and CSECT control blocks macro IDLI. You can also find the layout of IMS.ACBLIB members in the ACBGEN module, DFSUACB0, and the Write-PSBs-and-DMBs-to-ACBLIB module, DFSUAMB0.

The following figure provides an overview of the linkage between the major control blocks used for diagnosis.



Table 7. Control block definitions (continued)

<b>Control block acronym</b>	<b>Mapping macro</b>	<b>Description</b>
ALDS	DBFAREA	Area list data set.
AMPB	IDLI DMBASE=0	Access method prefix block. Contains information relative to a data set belonging to a database.
BALG	DBFBALG	Balancing group control block.
BFSP	IDLIVSAM BFSP	DL/I VSAM buffer handler pool prefix.
BFUS	IDLIVSAM BFUS	Subpool statistics block.
BHDR	BHDR	MSDB header.
BLOCKHDR	DFSSPBLK	Block header used by DFSPPOOL Storage Manager.
BSPH	IDLIVSAM BSPH	Buffer subpool header block. Contains the number of buffers in this subpool.
BUFC	IDLIVSAM BUFC	Buffer control block. Contains pointers to buffers.
BUFENTRY	DFSSPBLK	Used by DFSPPOOL Storage Manager to map the buffer size entries within the pool header.
CADSECT	ICADSECT	Communication area block. Contains the main dump formatter control block.
CBT	DFSCBTS	Control block. Represents storage pools (IPAGES) defined in DFSCBT00.
CCB	ICLI CCBASE=0	Conversational control block. Controls resources for conversational tasks.
CIB	ICLI CIBBASE=0	Communication interface block. Contains information the device-dependent module needs to determine Message Format Service (MFS) operation.
CIRCA	IPST	IMS control region interregion communication area.
CLB	ICLI CLBBASE=0	Communication line block. One exists for each communication line and for each node.
CLLE	DFSCILLE	Common Latch List Element. There is one block for each IMS ITASK, which is maintained in Key 7 storage.
CNT	ICLI CNTBASE=0	Communication name table. One exists for each named logical terminal and component.
CPM	(generated)	Communication password matrix. Length varies based upon the number of passwords in the CPT.
CPT	(generated)	Communication password table. Defined by user.
CRB	ICLI CRBBASE=0	Communication restart block.
CSAB	OCO	Callable Service Anchor Block. Used by IMS callable services modules.
CSVT	DFSCSVT	Callable Services Vector Table. Used by IMS callable services modules.
CTB	ICLI CTBBASE=0	Communication terminal block. One exists for each terminal and for each subpool in the system.

Table 7. Control block definitions (continued)

Control block acronym	Mapping macro	Description
CTM	(generated)	Communication terminal matrix. Length varies based upon the number of logical terminals (CNTs).
CTT	ICLI CTTBASE=0	Communication terminal table. There is one for each different type of terminal, as well as different features.
CULE	DFSCULE	Common Use List Element. Used in latching by the IMS Use Manager.
CVB	ICLI CVBBASE=0	Communication verb block. Reflects the relationship between the command message verbs and the passwords. It also reflects logical terminals associated with those commands.
CXB	(generated)	Communication extension block. Contains information that is required for control of a particular terminal. It is a logical extension of the CTB.
DBPCB	IDLI DPCBASE=0	DL/I DB PCB.
DCB	IDCBOSD	Data communication block. Contains data pertinent to the current use of a data set.
DCB-EXT	DFSDCBEX	OSAM extension to the DCB.
DDIR	IDLI DDRBASE=0	DMB directory entry. Contains an entry for each DMB known to IMS.
DFSAVEC	DFSAVECT	Dump formatter vector table.
DFSDOPTB	DFSDOPTB	Dump option entry block. It is the dump formatter CBTE request definition block.
DFSDPBFH	DFSDPBFH	Dump buffer pool blocks. Used for buffering offline dump storage.
DFSSBWO	DFSSBWA	Work area used by sequential buffering.
DMAC	DBFDMAC	DEDB area control block.
DMB	IDLI DMBBASE=0	Data management block. There is one for each database descriptor entry described in the DDIR.
DMBSEC	IDLI DMBBASE=0	Secondary list. There is one or more entry for each logically related segment and each index relationship.
DMCB	DBFDMCB	DEDB master control block.
DMHR	DBFDMHR	The buffer header for Fast Path. Describes the status of a particular buffer. The buffer headers (and buffers) are allocated in DBFCONT0. ESCDDMHR points to the first buffer and ESCDMBFN contains the number of headers. The relationship between buffer headers and buffers is fixed during IMS control region initialization.
DSEB	DFSDSPDS	Dynamic SAP Extension Block. Used to manage dynamic SAPs.
DSG	IDLI DSGBASE=0	Data set group control block. There is typically one for each data set group referenced by the DBPCB.
DSPWRK1	IDSPWRK	Dispatcher work area. There is one for each VS task (TCB) in an IMS environment.



Table 7. Control block definitions (continued)

<b>Control block acronym</b>	<b>Mapping macro</b>	<b>Description</b>
ECB	z/OS macro	Event control block. Describes the status of an event in an IMS environment.
ECNT	DBFECNT	Extended communications name table. (Fast Path)
EDSG	DFSSBDSG	Sequential buffering extension to the DSG.
EMHB	DBFEMHB	Expedited message handler block. (Fast Path)
EIB	DFSPCA	Partition Exit Interface Block Prefix.
EPCB	DBFEPCB	Extended PCB. (Fast Path)
EPF	IEPF	ECB prefix. Used to indicate the current status of the ECB and to connect the ECB to the appropriate SAP.
EPST	DBFEPST	Extended partition specification table. (Fast Path)
EQEL	DFSEQEL	Recoverable in-doubt structure queue elements. Identifies inaccessible data due to in-doubt status.
ESCD	DBFESCD	Extended system contents directory. (Fast Path)
ESRB	DBFESRB	Extended service request block. (Fast Path)
ESRT	DBFESRT	Expedited message handling region insert buffer. This buffer is a temporary save area for a message input. ESRTs are allocated in module DBFCONT0 by IMS control region initialization with a length equal to the largest terminal buffer defined. ESCDESRT points to the first ESRT. EPSTESRT points to a related ESRT. (Fast Path)
FAQE	DFSSPBLK	Free allocated queue element. Used by the DFSISMNO Storage Manager to manage storage within a pool.
FDB	IDLI FDBBASE=0	Field descriptor block.
FDT	DBFMFDB	Field description table.
FEDB	ICLI FEDBBASE=0	Front end directory block. Stores global information about the front end switching facility.
FEIB	ICLI FEIBBASE=0	Front end interface block. Contains data to allow the front end switching user exit to communicate with the transaction manager.
FRB	DFSFRB	Fast restart block.
GB	IGLI	GSAM data set control block. Contains information concerning the data set operation and pointers to other control blocks used for accessing records.
GBCB	IGLI	GSAM buffer control block. Contains the address of a unique buffer.
GLT	IGLI	GSAM load table. Provides all addresses of the GSAM load modules necessary for initialization.
GPT	IGLI	GSAM pointer table. Provides information required by resident and nonresident GSAM routines.
GQCB	IGLI	GSAM queues control block. Contains first and last pointers for the four queues of GSAM GBCBs used by GSAM BUFFIO.

Table 7. Control block definitions (continued)

Control block acronym	Mapping macro	Description
HSSR	DBFHSSR	Holds area range information from SETR statements. HSSR is formatted in the offline dump.
HSSO	DBFHSSO	Holds image copy (IC) information from SETO statements.
HSSD	DBFHSSD	Holds information for the /DISPLAY HSSP command. HSSD is formatted in the offline dump.
HSSP	DBFHSSPS	Skeleton block. Temporarily holds HSSO/HSSR/HSSD information before scheduling.
IBFPRF	IBFPRF	Buffer prefix. There is one for each buffer described in each subpool used by the OSAM buffer manager.
IBPOOL	IBPOOL	OSAM buffer handler main buffer pool. Contains statistics and vectors to OSAM buffer subpools.
IDSC	DBFIDSC	<p>IDSC is the image copy data set control block. It represents the Image Copy data set (IDS) the same way the area data set control block (ADSC) represents the area data set (ADS). IDSC also uses the same control block structure as the ADSC. An IDSC contains a description of the Image Copy data set. There are up to two IDSCs for each DEDB area with the Image Copy option. An IDSC is built dynamically at the first call to the area that is running as HSSP with the Image Copy option requested. The IDSC is released during Image Copy termination.</p> <p>The IDSC control block is formatted in the offline dump.</p>
IEEQE	DFSIEQE	In-doubt error queue element. Contains buffers of changed data (data in the in-doubt state).
ISPL	ISUBPL	OSAM buffer subpool. Provides a base for fixed length buffers and statistics about the buffers.
ISL	DXRRLISL	IRLM identified subsystem list. Contains the name of each subsystem and its status.
JCB	IDLI JCBBASE=0	Job control block. There is one for each PCB. It contains level tables and segment blocks and a trace table of the previous calls.
LCB	LCB	Link control block. Represents the link for channel to channel, memory to memory, VTAM, and binary synchronous connections in MSC.
LCD	LCDSECT	Log contents directory. Controls the interface between the logical and physical loggers in a DB/DC environment.
LCRE	DFSLCRE	Local current recovery element. Contains the sync point, checkpoint recovery information relative to each PST.
LEV	IDLI LEVBASE=0	Level table. Consists of two parts: previous call and current call that is filled in by the call analyzer.
LIPARMS	PARMBLK	Language interface parameter block.
LLB	ICLI CLBBASE=0	Link line block.
LTB	ICLI CTBBASE=0	Link terminal block.
LXB	LXB	Link extension block.

Table 7. Control block definitions (continued)

<b>Control block acronym</b>	<b>Mapping macro</b>	<b>Description</b>
MRMB	DBFMRMB	DEDB randomizing module block.
MSNB	MSNB	Message Control/Error exit interface block. Contains the block content before and after calling Message Control/Error exit DFSCMUX0 or during the interface processing.
PAC	DFSPAC	Database Resource Adapter (DRA) control block.
PAPL	DFSPAPL	DRA architected parameter list.
PARMLIST	ICADSECT	Dump formatter bulk print interface block.
PAT	DFSPAT	DRA thread control block.
PATE	DFSPAT	DRA thread entry control block.
PCA	DFSPCA	Partition Communication Area.
PCIB	ICLI PCIBASE=0	Partition communication interface block.
PCPARMS	PARMBLK	Program control parameter block.
PCT	DFSPCT	Partition chaining table.
PDAE	DFSPSEIB	Partition Definition Area Prefix. Partition Definition Area Entry.
PDEX	DFSDDIR	Partition Directory Extension
PDIR	IDLI PDRBASE=0	Program specification block directory. Contains entries for every program known to IMS.
PDL	DFSPDL	DRA dump parameter list.
PECA	DFSPSEIB	Partition Exit Communication Area.
PNT	DFSPNT	Partition Name Table.
POOLHDR	DFSSPBLK	Storage pool header used by the DFSPPOOL storage manager to keep track of pool information.
PPRE	DFSPPRE	Standard IPAGE prefix mapping macro. Used for all IPAGEs created in IMS.
PQE	DFSPQE	DRA queuing element.
PSB	IDLI PSBBASE=0	Program specification block. Relates to the application program and contains the PCBs associated with this PSB.
PSDB	IDLI DMBASE=0	Physical segment descriptor block. Describes each segment in the database.
PST	IPST	Partition specification table. There is one for each message or batch region; it contains a DECB for this partition, I/O terminal PCB, and parameters required for this region.
PTBWA	DXRPTBWA	IRLM pass-the-buck work area.
PTE	DFSPNT	Partition Table Entry.
PTK	DFSPTK	Partition Key Index Table.
PTX	DFSPTX	Partition Entry Index Table.
PXPARMS	PARMBLK	Parameter Anchor Block.

Table 7. Control block definitions (continued)

<b>Control block acronym</b>	<b>Mapping macro</b>	<b>Description</b>
QCB	IAPS SMBBASE=0	Queue control block.
QEL	IAPS SMBBASE=0	Queue Element.
QMBA	DFSQMGR	Queue Manager Buffer Area.
RCPARMS	IDL PSTBASE=0	Region control parameter block.
RCTE	DBFRCTE	Routing code table entry.
RDLWA	DXRRDLWA	IRLM deadlock process work area. Contains information that must be communicated between the deadlock process modules.
RHB	DXRRHB	IRLM resource header block. Represents a resource.
RHT	DXRRHT	IRLM resource hash table. Provides a series of anchors for resource chains.
RLB	DXRRLB	IRLM resource lock block. Represents a request for a lock or a lock held on a resource.
RLCBT	DXRRLCBT	IRLM private area control block and table. Contains addresses of IRLM entry points.
RLMCB	DXRRLMCB	IRLM master control block. Contains branch entry addresses for all RLMREQ as well as queue anchors.
RLPL	DXRRPL	IRLM request parameter list. This is the parameter list for all functional requests for the resource lock manager.
RLQD	DXRRQLD	IRLM query mapping macro. Maps IRLM control blocks/structures returned to the IMS invoker of QUERY.
RPL	IDLIVSAM	Request parameter list. Contains parameters passed to VSAM from IMS and the status returned to IMS from VSAM.
RPST	DFSRPST	Restart PST. Contains identifying information and characteristics of units of recovery.
RRE	DFSRRE	Residual recovery element. Contains sync point actions, such as Commit and Abort, relative to eachDb2 for z/OS connection out of a dependent region and is used for BMP restart processing, in-doubt processing, and restartable backout processing.
SAP	ISAP	Save area prefix. Relates to a save area set.
SBHE	DFSSBHE	Sequential buffering hash entry. Used to hash or anchor SDCB control blocks and to serialize the sequential buffer SDCB and SDSG control block subsystem chains. The SBHEs are part of the SBSCD.
SBPARMS	DFSSBPAR	Sequential buffering extension to PXPparms.
SBPSS	DFSSBPSS	Small section of the SBPST that needs to be in CSA.
SBPST	DFSSBPST	Sequential buffering extension to the PST.
SBSCD	DFSSBSCD	Sequential buffering extension to the SCD. This extension contains the SBHE hash entries.
SBUE	DFSSBBUF	Sequential buffering buffer extension. There is one SBUE for each SBUF.

Table 7. Control block definitions (continued)

Control block acronym	Mapping macro	Description
SBUF	IBFPRF SBEXT=YES	Sequential buffering buffer. One SBUF control block is used by sequential buffering to control each SB buffer. The SBUF control blocks of one SB buffer pool are contiguous in storage and are formatted as one entity.
SCAR	DFSSBCAR	Control block containing the interpreted data of one SBPARM control statement in the //DFSCTL file.
SCA1	DFSSBCAR	Control block containing the uninterpreted data of one SBPARM control statement in the //DFSCTL file.
SCD	ISCD	System contents directory. Produced at system definition time, it contains major entry points for all facilities and system control information.
SDB	IDLI SDBBASE=0	Segment descriptor block. Contains a logical description of the segment.
SDCB	DFSSBDCB	Sequential buffering extension to the DCB. Is for those DB data sets that are buffered by sequential buffering.
SDSG	DFSSBDSG	Sequential buffering extension to the DSG. Describes one I/O process. There is typically one SDSG control block for each data set group control block (DSG) that might potentially be buffered by sequential buffering.
SDWA	IHASDWA	System diagnostic work area.
SGT	DFSPRSGT	Segment table. Describes the segments used by the partial reorganization process. It is built during the DBD analysis phase. Its address is held in the common area field (COMASGT). The segment extension table (SGX) holds additional information about the segments.
SIDB	DXRSIDB	IRLM subsystem identification block. Used to identify each subsystem that relates to IRLM.
SIDX	DFSSSIE	Subsystem index entry.
SMB	IAPS	Scheduler message block. Related to a transaction.
SPQB	ICLI SPQBASE=0	Subpool queue block. The SPQB represents the dynamic user for an ETO terminal and represents a set of static queues (CNTs) for a static ISC parallel session terminal.
SQPST	ISQPST	PST queue. Associated with the scheduler sequence queue.
SRAN	DFSSBRAN	Sequential range. Used in sequential buffering to describe a recently referenced set of consecutive DB blocks. Sequential buffering allocates one Sequential SRAN control block for each buffer set of each buffer pool. SB also allocates Random SRAN control blocks to each buffer pool. The Sequential SRANs and Random SRANs of one SB buffer pool are contiguous in storage and are formatted as one entity.
SSIB	IEFJSSIB	Subsystem identification block. Identifies the subsystem that requested services.

Table 7. Control block definitions (continued)

Control block acronym	Mapping macro	Description
SSOB	IEFJSSOB	Subsystem options block. Used to request a particular function from the z/OS subsystem.
SSVP	DFSSSVPL	System Services Parameter List. Used by IMS System Macros for parameter lists for mailing out of line calls. There is one SSVP per ITASK, anchored off of the SAP.
TAB	DFSTAB	Transaction anchor block.
TCT	DFSTAB	Transaction class table. Used for queuing of messages in a priority sequence within a specified class.
TPPCB	IDL I TPCBASE=0	Program communication block. There is one for each logical database being referenced by the application program.
UEHB	UEHB	User exit header block. Used for automated operator exit interface processing.
UXDT	DFSUSRX	User Exit Definition Table. Contains control information and user exit addresses for user exits managed by IMS standard user exit service.
UXRB	DBRUXRB	<p>A unit of work (UOW) is represented by a UOW exclusive resource control block (UXRB), similar to the XCRB representing the CI. The UXR B contains information about the UOW (for example, Area, RBA) and is used for resolving potential UOW resource contention among dependent regions. Other UXR B fields include the lock token, number of associated XCRBs, the owning EPST, the update intent flag, and the PCB.</p> <p>The UXR B control block is formatted in the offline dump.</p>
VSI	IDLIVSAM VSI	VSAM sharing information control block. Controls VSAM sharing between subsystems.
WHB	DXRWHB	IRLM work unit block. Contains the anchor for all requests associated with that owner.
XCRB	DBFXCRB	Exclusive control resource block.
XMCA	DFSXMC	Cross-Memory Control-Address Spaces. There is one block for each IMS subsystem, which is maintained in Key 0 storage.
XMCI	DFSXMC	Cross Memory Control-ITASKs. There is one block for each IMS ITASK, which is maintained in Key 7 storage.
ZIB	IZIB	Zone initialization block. Used by the DFSISMNO Storage Manager to keep track of a buffer obtained using ICREATE.

#### Related reference

[“Contents formatted for FMTIMS options” on page 531](#)

FMTIMS options can be specified on the FMTIMS statement in any order.

## IMS Control Block Table (CBT) Pools

---

The IMS CBT storage manager (also known as the DFSBCB storage manager) manages pools of fixed-length control blocks.

All CBT pools can be displayed by issuing the **/DISPLAY POOL CBT**. Issuing the **/DISPLAY POOL** command with other keywords, such as DBB, DCC, DEP, DISP, FP, GEN, OSAM, SUM, and others, limits the display of the CBT pools to those related to the functional area of IMS that the keyword represents.

All of the CBT pools are briefly described in the following alphabetical list.

**AAB**

Obsolete CBT entry

**ADSC**

Fast Path area data set control block

**AESL**

Fast Path DEDB ADS list

**AHDR**

ETO autologon LU header/ autologon hash table synonym

**APST**

Partition specification table - above the line

**ATPW**

Attach catalog PSB work area block

**AWE**

Asynchronous work element

**BCPT**

Checkpoint id table

**BQEL**

ISAM/OSAM/VSAM buffer queue element

**BXQE**

BCB queue elements for use by DFSBCBxx modules only

**CATR**

IMS catalog request block

**CBLK**

Used in place of message queue to keep track of CPI-C scheduling requests

**CCB**

Conversational control block

**CFEZ**

Block used to store local parms for DC trace re-entrancy

**CLLE**

Common latch list element

**CM24**

Common work unit, 24-bit

**CMWU**

Common work unit

**CRSP**

OM command response block

**CSAG**

Callable services anchor block (CSAB) - global storage

**CSAL**

Callable services anchor block (CSAB) - local storage

**CSWA**  
Common system work area - global storage

**CTBW**  
Communication terminal block work area

**CULE**  
Common use list element

**D1WA**  
Dispatcher work area 1

**DACT**  
DL/I call accounting block

**DBBD**  
Database (DDIR) descriptor

**DBPB**  
Data base purge block

**DBRC**  
DBRC work area

**DCM**  
Obsolete CBT entry

**DDIR**  
DMB directory entry

**DDRE**  
DMB directory extension

**DESC**  
LU 6.2 descriptor block

**DG2W**  
Global dispatcher area

**DL2W**  
Local dispatcher area

**DLWA**  
DL/I private work area

**DMAC**  
Fast Path DEDB area control block for dynamic areas

**DMHR**  
Fast Path VSO dynamic DHMRs for write staging area

**DPST**  
Dependent region PST

**DPXB**  
Dispatcher extension block

**DSME**  
Fast Path data space mapping entry

**DSML**  
Fast Path data space mapping list

**EMAC**  
Obsolete CBT entry

**EPST**  
Fast Path extended PST

**EQEL**  
Resolve indoubt structure queue element



**EZS**

External connection status element

**FEIB**

Front end message switch interface block

**FMMC**

Fast Path control block to contain VSAM extents

**FNCB**

Fast Path notify control block

**FPB6**

Fast Path DBFBPND6 block for FP 64-bit buffer pools

**FPCP**

Fast Path command parameter list

**FSRB**

Fast Path SRB

**GDS**

Obsolete CBT entry

**GESE**

Global external subsystem entry

**GIOB**

Global IOSB

**GOWA**

Global OSPA

**GPNT**

Fast Path global name table

**GQMW**

Global queue manager work area

**GSAV**

Global save area

**HSHE**

Hash table element header

**HSHS**

Hash table slot header

**HTEB**

BPE LFS hash table services: hash table element block, CTL private

**HTED**

BPE LFS hash table services: hash table element block, DLI private

**IAFP**

IAFP dataset control block

**IBKD**

Interface Block for user exit services in DL/I region private storage. The IBKC control block is used for working storage and a parameter list when invoking a user exit through IMS DFSUSRX user exit service.

**IBKP**

Interface Block for user exit services in control region private storage. The IBKP control block is used for working storage and a parameter list when invoking a user exit through IMS DFSUSRX user exit service.

**IDT**

Identify table entry

**IEQE**

Inflight/indoubt data buffers

**IOSB**

I/O supervisor block

**IPST**

System PST in DLI/SAS private, 31-bit storage

**IRLM**

IRLM parameter area

**KLSD**

LSO dependent control block

**L56X**

Fast Path DBCTL log record

**LCLL**

Local common latch list element (local storage)

**LCRE**

Local current recovery entry

**LG24**

An LSAV below the 16m line

**LGND**

Logon descriptor control block sets

**LGWA**

Log work area

**LGWX**

Log work area extension (private)

**LPNT**

Fast Path local name table

**LPST**

Local PST block

**LQB**

Local queue block

**LQMW**

Local queue manager work area

**LRA**

Obsolete CBT entry

**LS24**

24-bit quick block, local save area / AWE

**LSAV**

Local save area

**LSWA**

Local system work area

**LUB**

LUB pool below 16 MB line for LU 6.2

**LXB**

Dynamic link extension block for VTAM and TCP/IP links. The LXB control block is mapped by the LXB copy.

**MMCL**

OSAM Media Manager Interface module parameter list.

**MMPL**

Media Manager Services parameter list.

**MMRQ**

Media Manager Request Element prefix plus Media Manager Request Element.

**MPB**

Obsolete CBT entry

**MPCB**

PCB block for MSC bandwidth. The MPCB control block is mapped by the DFSMSPCB macro.

**MSCL**

MSC logical link control blocks: LLB, two LTBs, and a CRB. The MSCL control block is mapped by the DFSMSCL macro.

**MSCP**

MSC physical link control blocks: LCB and CTT. The MSCP control block is mapped by the DFSMSCP macro.

**MSEB**

Work area for the TM and MSC Message Routing and Control User exit routine (DFSMSCE0). The MSEB work area is mapped by the DFSMSCEB macro.

**MSGP**

Message pool

**MUTE**

OTMA MCB user table entry

**OCMD**

OM command instance block

**OFB**

Obsolete CBT entry

**OLRK**

OLR KSDS update entry

**OSSA**

OSAM save area block.

**OSWA**

OSAM work area (IOMA)

**OTDD**

OTMA destination descriptors

**OTMD**

OTMA member descriptors

**PCIB**

Partition CIB

**PDEX**

Partition directory extension

**PDIR**

PSB directory entry

**PF62**

Message prefix block for LU 6.2

**PFQE**

CQSPUT failure queue element

**PGMD**

Program (PDIR) descriptor

**PST**

Partition specification table

**QAB**

QAB pool for LU 6.2

**QLST**

Work area to hold queue manager parameter list

**QLTC**

DEDB and area quiesce latch blocks for DEDB Alter

**QMBA**

Queue manager buffer area (large message)

**QMBS**

Queue manager buffer area (small message)

**QQSN**

Queue space notification area

**QS24**

Temporary save area set with parm areas - 24-bit storage

**QSAV**

Temporary save area set with parm areas - 31-bit storage

**RACW**

RACF work area used for racinit, fracheck, etc.

**RBAT**

Fast Path VSO RBA update table entries

**RCNT**

Remote communication name table. The RCNT control block is mapped by the RCNT copy.

**RCTD**

Fast Path routing code (RCTE) descriptor

**RCTE**

Fast Path routing code table entry

**RECA**

VTAM receive any buffer

**RPST**

Restart partition specification table

**RRE**

Residual recovery entry

**RSCX**

Resource extension block

**SAA**

Obsolete CBT entry

**SAP**

Save area prefix

**SIDX**

Subsystem index entry

**SLOG**

DC monitor work area

**SMB**

Scheduler message block

**SOPB**

Sign-on parameter block

**SPQX**

SPQB extension block

**SQOF**

Shared queues overflow hash table element

**SRA**

Obsolete CBT entry

**SRBC**  
Common SRB pool

**STAT**  
DBCTL and DRA statistics area

**STB**  
Obsolete CBT entry

**STTR**  
DL/I trace stacks

**SVPG**  
System service parameter lists - global

**SVPL**  
System service parameter lists - local

**TCBT**  
TCB table

**TDBC**  
Obsolete CBT entry

**TDCB**  
Obsolete CBT entry

**TIB**  
TIB pool for LU 6.2

**TLS**  
Transaction level statistics area

**TPIP**  
TPIPE pool for OTMA

**TRND**  
Transaction (SMB) descriptor

**TT24**  
Trace table (24-bit storage)

**TT3P**  
Trace table (31-bit, private storage)

**TTAB**  
Trace table (31-bit storage)

**UOWE**  
Unit of work table entry

**USRD**  
User descriptor control blocks

**USTB**  
Obsolete CBT entry

**UXIC**  
User exit instance (common)

**UXIP**  
User exit instance (private)

**UXSC**  
User exit static area (common)

**UXSP**  
User exit static area (private)

**VRPL**  
VSAM RPL/save area stack

**VTCTB**

VTAM session control block

**VWA**

Volatile work area

**WLMB**

Work area of parameter list for the Workload Manager classification service.

**X124**

DL/I pool below 16m line

**XMCI**

Cross memory itask block

**XMCL**

Local XMCI block (ctl private)

**XPST**

Extended PST area (dependent region)

**XT62**

LU 6.2 and OTMA PST extension

**YQAB**

OTMA queue anchor block

## Control block interrelationship diagrams

---

These diagrams show the interrelationships between major control blocks in an IMS environment.

**Note:** This information is intended for users with experience in obtaining and reading IMS system dumps. The training course *IMS Diagnostic Approaches* provides an introduction to this information. Further information about this training course is available on the Information Management Training and Certification web site.

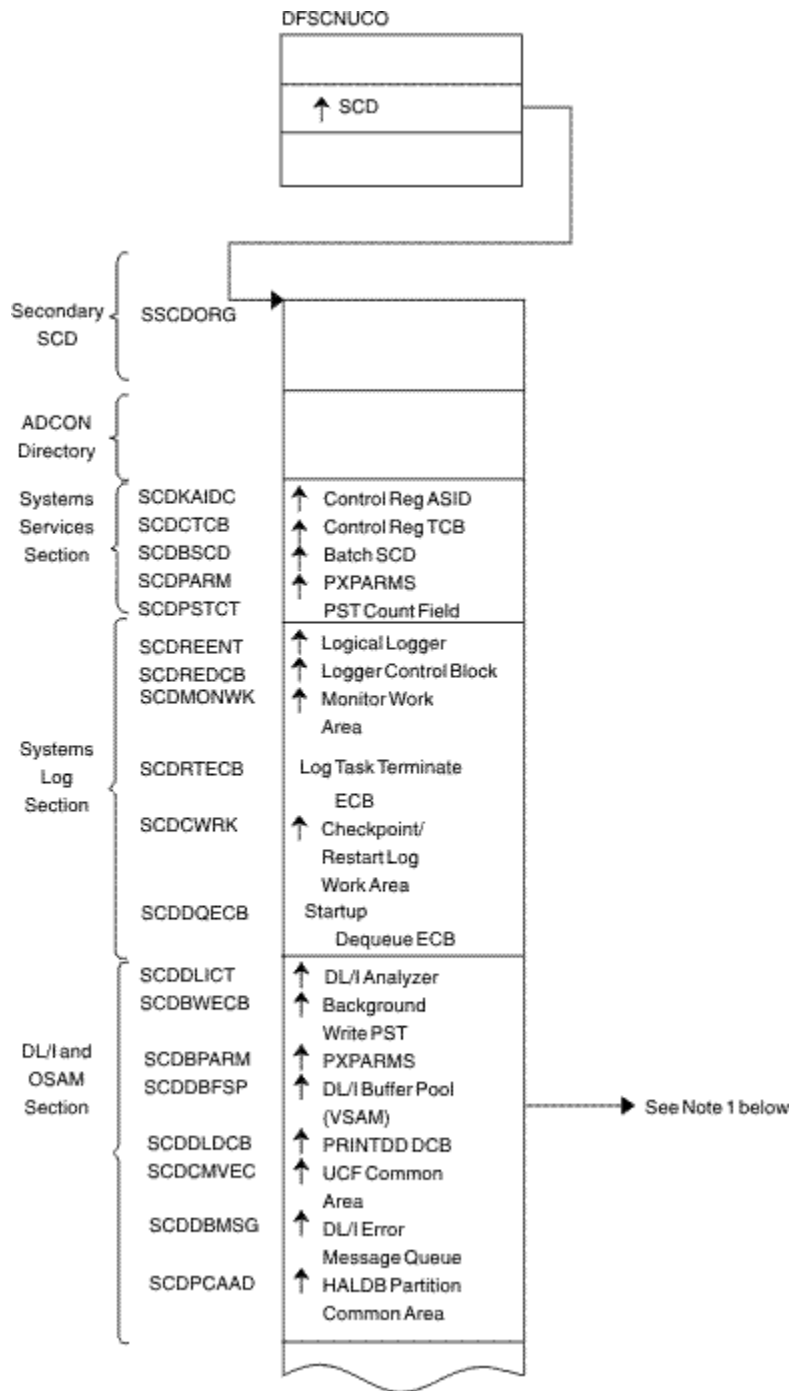
Subsections:

- [“Online system contents directory \(SCD\)” on page 89](#)
- [“DFSPPX0–parameter blocks” on page 95](#)
- [“DL/I OSAM buffer pool” on page 96](#)
- [“Sequential buffering control blocks” on page 97](#)
- [“Buffer handler pool \(VSAM\)” on page 99](#)
- [“OSAM DECB with IOB in use” on page 100](#)
- [“OSAM IOB pool showing available IOBs” on page 100](#)
- [“Storage management control block relationships created for the MAIN pool” on page 101](#)
- [“Storage management control block relationships for preallocated storage blocks” on page 102](#)
- [“Storage management control block relationships \(DFSPOOL pools\)” on page 104](#)
- [“Storage management control block relationships \(DFSCBT00 pools\)” on page 105](#)
- [“Database Manager control blocks for a representative database” on page 106](#)
- [“Database control blocks” on page 108](#)
- [“Diagram of a Data Management Block \(DMB\)” on page 110](#)
- [“Overview of Fast Path control blocks” on page 110](#)
- [“Relationships between buffer control blocks for Fast Path databases” on page 111](#)
- [“GSAM control block overview” on page 112](#)
- [“GSAM control blocks” on page 113](#)
- [“DL/I control block relationships” on page 114](#)
- [“IMS Transaction Manager control blocks” on page 116](#)

- [“Intersystem communication control block structure” on page 116](#)
- [“VTCB load module” on page 117](#)
- [“Multiple Systems Coupling \(MSC\) control block overview” on page 119](#)
- [“Multiple Systems Coupling \(MSC\) main storage-to-main storage control block overview” on page 119](#)
- [“z/OS storage map showing IMS-to-IRLM interrelationships” on page 120](#)
- [“IRLM overall control block structure” on page 121](#)
- [“IRLM storage manager pools” on page 122](#)
- [“IRLM lock request examples” on page 123](#)
- [“Control block overview of Database Recovery Control \(DBRC\)” on page 123](#)
- [“Organization and basic linkages: DOF \(Device Output Format\) and MOD \(Message Output Descriptor\)” on page 124](#)
- [“Organization and basic linkages: DIF \(Device Input Format\) and MID \(Message Input Descriptor\)” on page 125](#)

### **Online system contents directory (SCD)**

The following graphics show the online system contents directory.



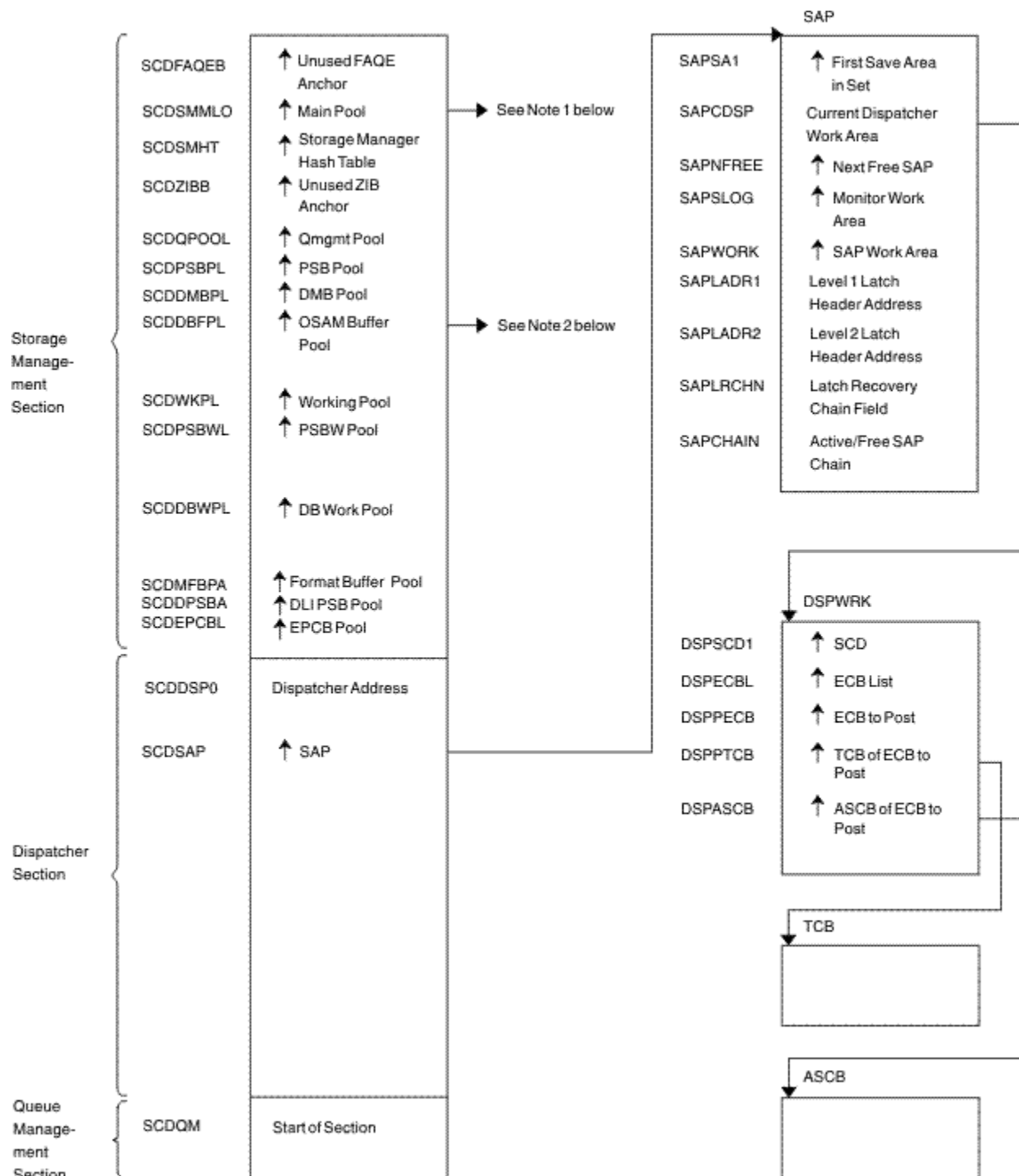
- Note 1: See Figure 11 on page 97 and Figure 13 on page 99.

Figure 4. Online system contents directory (SCD) Part 1 of 6



Sequential Buffering Section	SCDSBPTR	▲ SB SCD
Data Sharing Section	SCDIRPM	▲ IRLM Parns
	SCDRDSH0	▲ DFSRDSH0 (ASYNCR Data Sharing Routine)
Common Services Section	SCDPCCC0	▲ DFSPCCC0 (IRLM/DBRC Handler)
	SCDQHDSR	▲ Queue Header Table Address
	SCDCIR00	▲ Create ITASK Module
STAE/ESTAE Section	SCDFMOD0	▲ Entry Point of Attach ITASK
	SCDXSTA0	A(ESTAE)
Latch/Lock Section	SCDLRSAP	▲ Latch Recovery ITASK SAP
	SCDLMGRA	▲ Latch Manager Address
Formatted Dump Section	SCDDSDWA	▲ SDWA at Dump Time
Timer Services Section	SCDCKVAL	Clock Value
	SCDTIMEP	▲ Timer Services Module (DFSFTIM0)
Trace Services Section	SCDTRBLK	▲ Trace Control Block
External Subsystem Section	SCDPITME	PITRACE Buffer
	SCDESETP	▲ ESET Prefix
Dynamic Control Block Builder Section	SCDCBTA	▲ Control Block Extension Address
	SCDBCBO0	▲ Address of Control Block Build

Figure 5. Online system contents directory (SCD) Part 2 of 6



- Note 1: See [Figure 16](#) on page 102.
- Note 2: See [Figure 11](#) on page 97.

Figure 6. Online system contents directory (SCD) Part 3 of 6

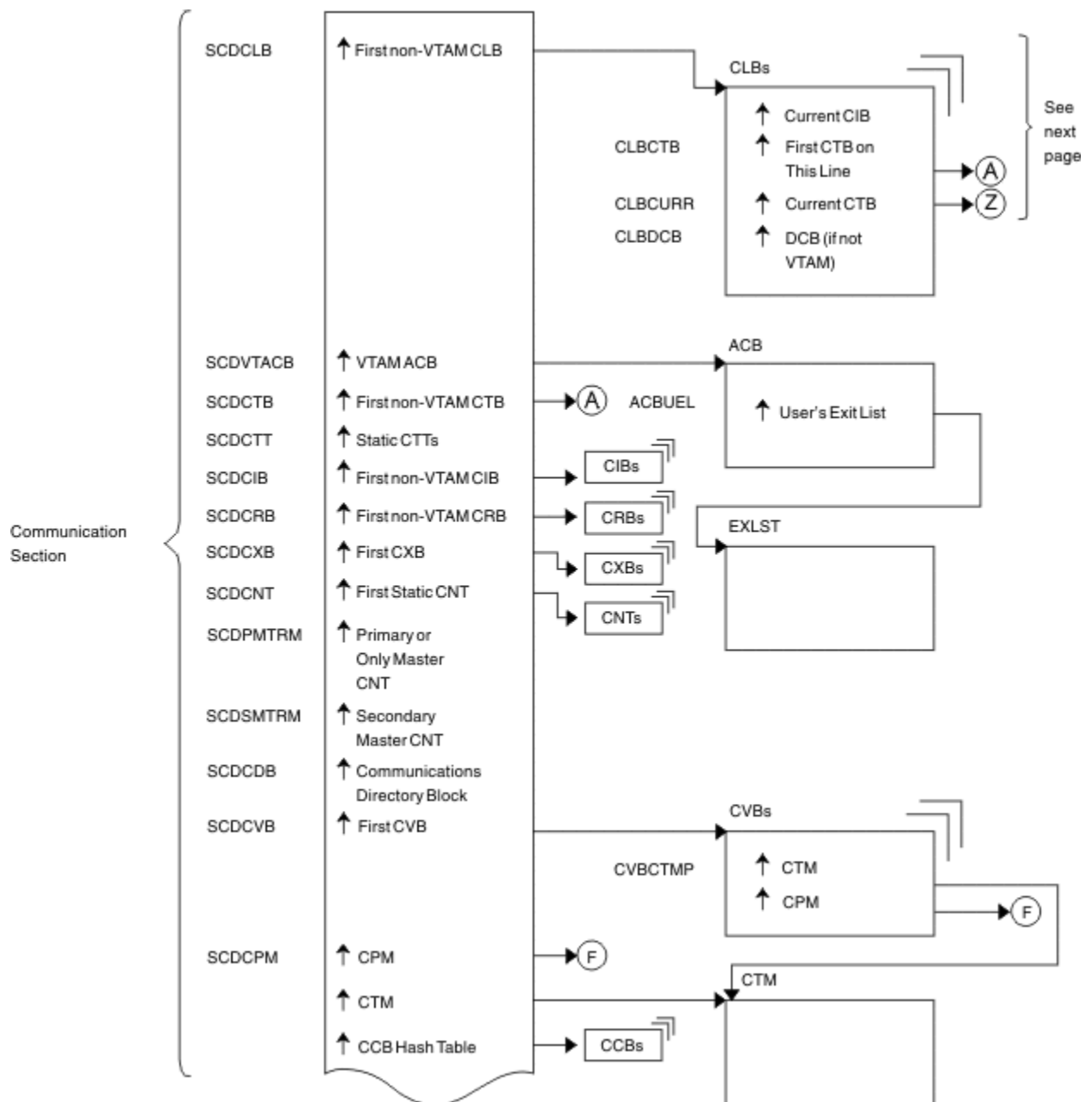


Figure 7. Online system contents directory (SCD) Part 4 of 6

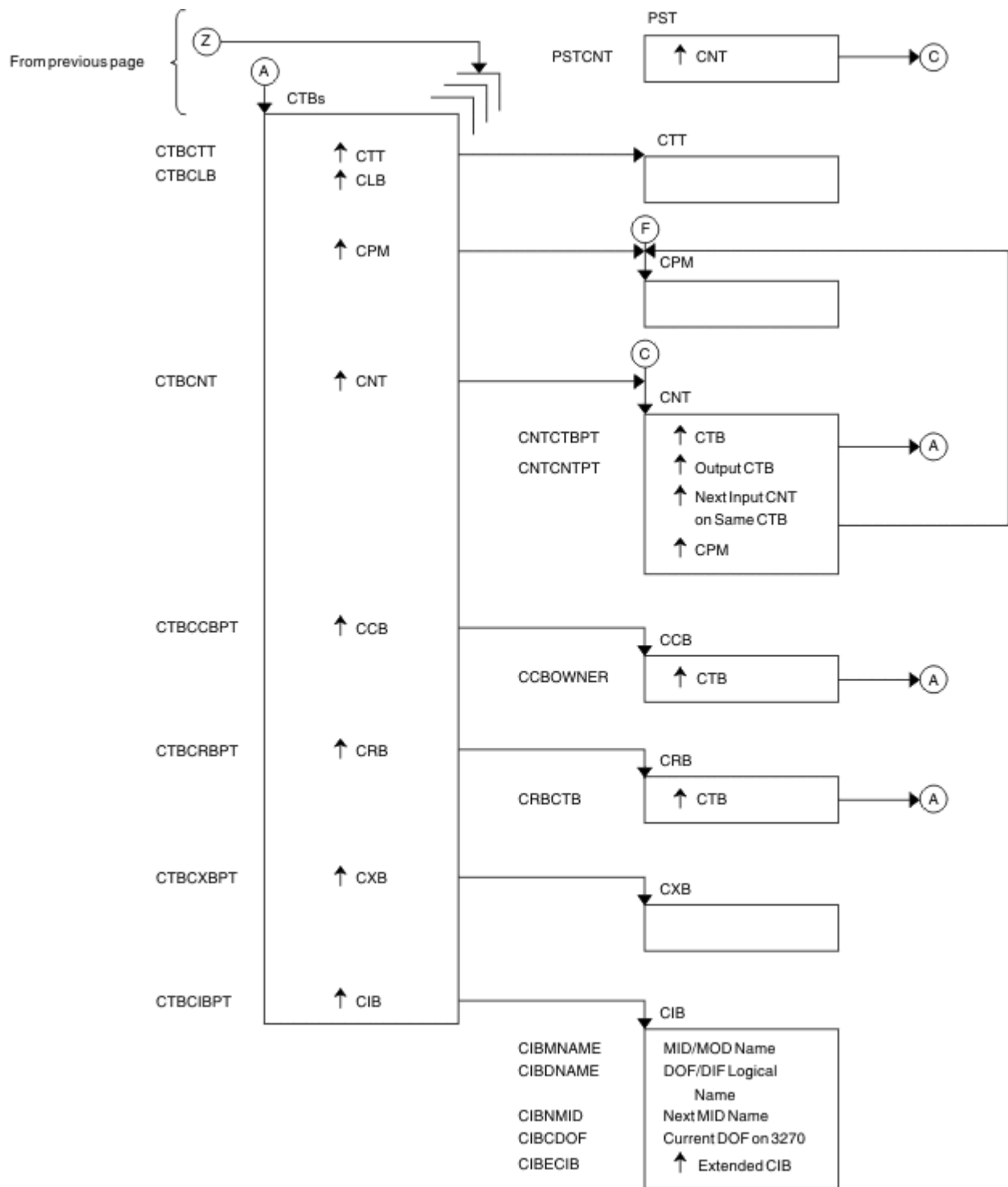


Figure 8. Online system contents directory (SCD) Part 5 of 6

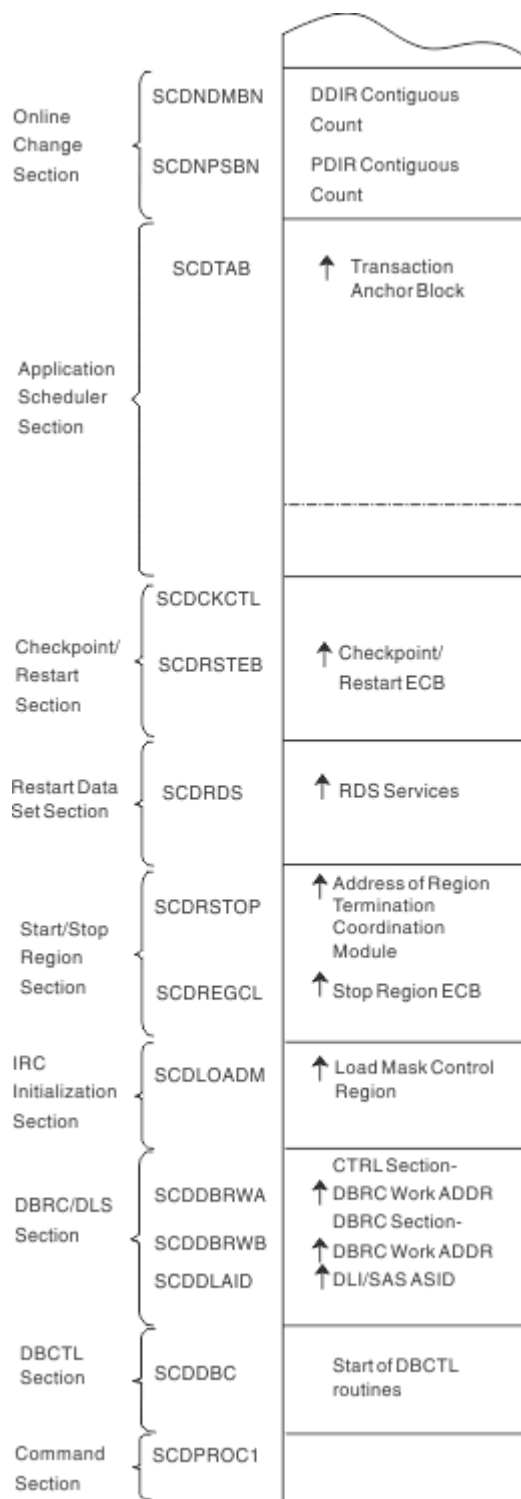
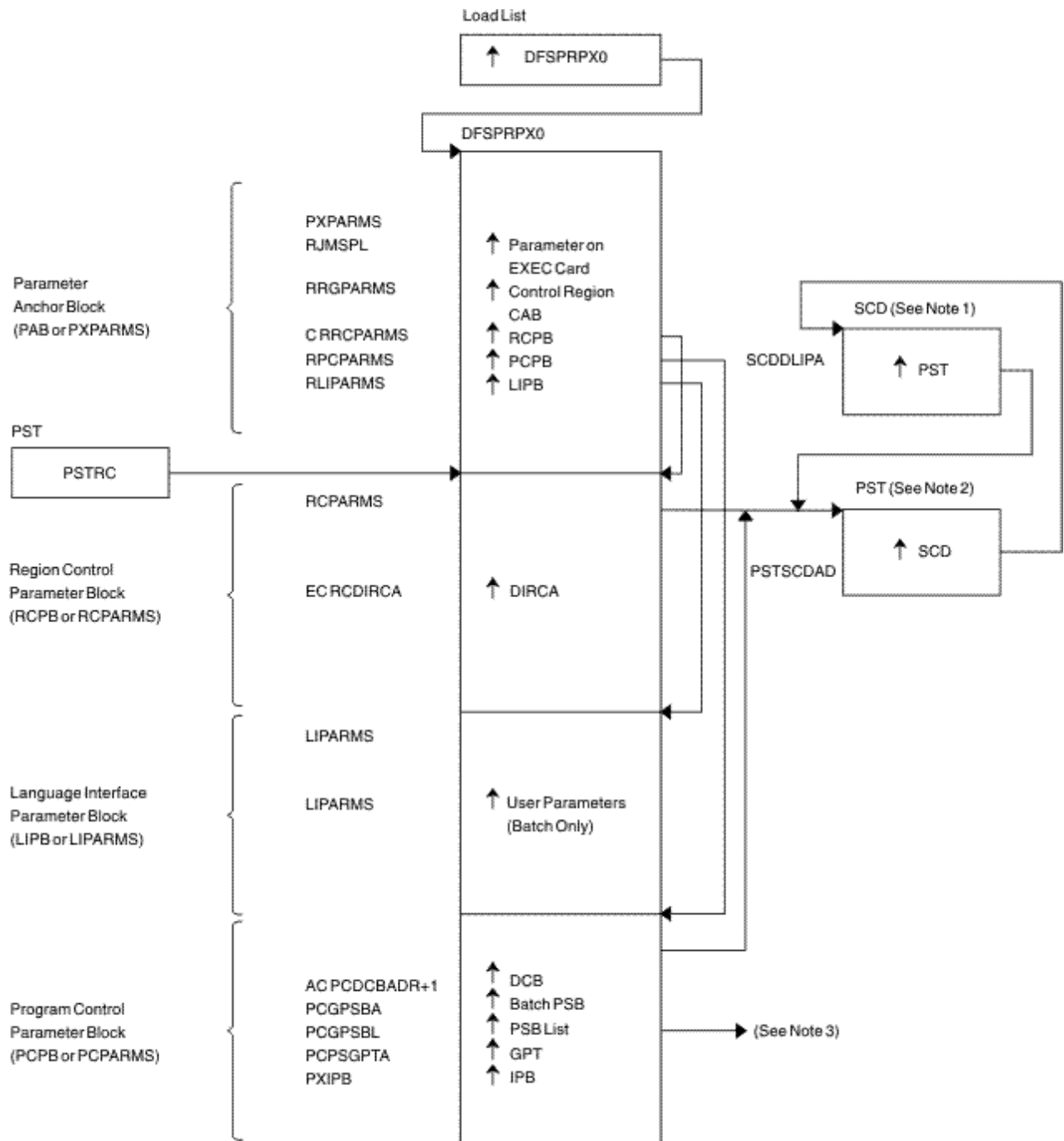


Figure 9. Online system contents directory (SCD) Part 6 of 6

### DFSPRPX0-parameter blocks

The following figure shows the parameter blocks for DFSPRPX0.



- Note 1: See [Figure 4](#) on page 90
- Note 2: See [Figure 21](#) on page 108
- Note 3: See [Figure 26](#) on page 113

Figure 10. DFSPRPX0-parameter blocks

### DL/I OSAM buffer pool

The following figure shows the control blocks for DL/I OSAM buffer pool.

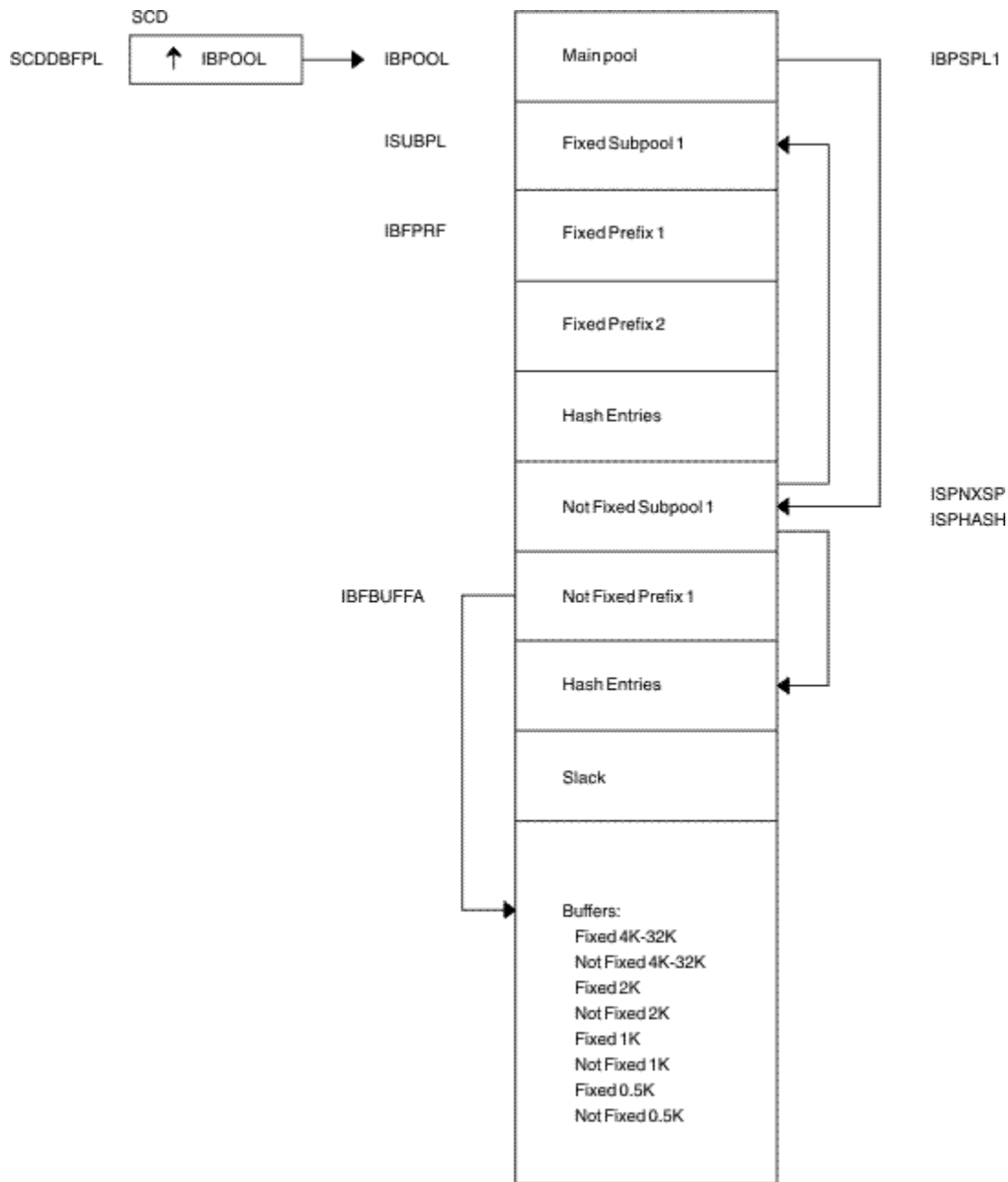


Figure 11. DL/I OSAM buffer pool

### Sequential buffering control blocks

The following figure shows the sequential buffering control blocks.

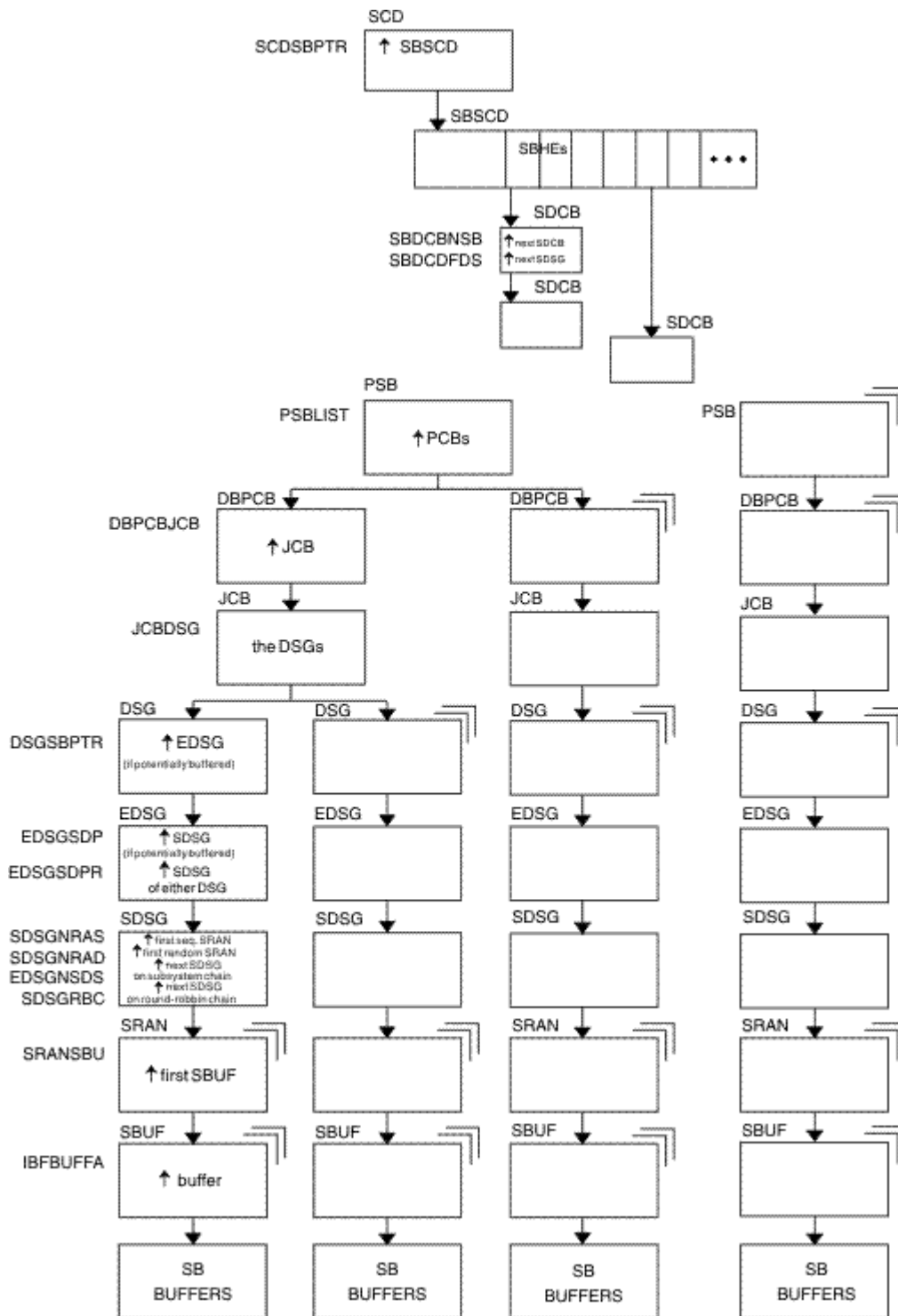


Figure 12. Sequential buffering control blocks

**Notes to Figure 12 on page 98:**

1. SCD is the IMS systems content directory.
2. SBSCD is a sequential buffering extension to the SCD.
3. SBHEs are sequential buffering hash entries located within the SBSCD (sequential buffering extension to the systems content directory). IMS uses SBHEs to:
  - Anchor the sequential buffering extension to the DCB (SDCB)



- Serialize the SDCB and SDSG subsystem chains (defined in notes “4” on page 99 and “8” on page 99).
- SDCB is a sequential buffering extension to the data communication block. There is one SDCB for each data set that is actively being sequentially buffered. There must be a separate SDCB for each SBPST that references a HALDB partition, because information in the SDSG will change as the DL/I calls go from partition to partition. As a result, multiple SBPSTs cannot share an SDCB, as is possible for non-HALDB databases. For HALDB, there is one SDCB for each partition used by a PST. IMS uses each SDCB to anchor any sequential buffering SDSGs that have buffer pools allocated to them.
  - The chains of SDCBs and SDSGs anchored in the SBHEs are called the SDCB and SDSG subsystem chains.
  - The program specification blocks, DBPCBs, job control blocks, and the data set group control blocks in the figure are DL/I control blocks.
  - EDSG is a sequential buffering extension to the DSG. The field EDSGSDP points to the SDSG if the data set group control block is potentially buffered by SB. If the DSG is not potentially buffered (but another DSG for the same data set and same application is), then the field EDSGSDPR points to one of the SDSGs of these "other" DSGs.
  - SDSG is a sequential buffering extension to the data set group control block. The SDSG is present if the user wants to have the DSG sequentially buffered. The SDSG is the control block that controls one sequential buffering buffer pool.
  - SRAN is a sequential buffering control block that describes references in one set of recently referenced consecutive data set blocks.
  - SBUF is a sequential buffering control block that describes one individual buffer.

### Buffer handler pool (VSAM)

The following figure shows the buffer handler pool (VSAM).

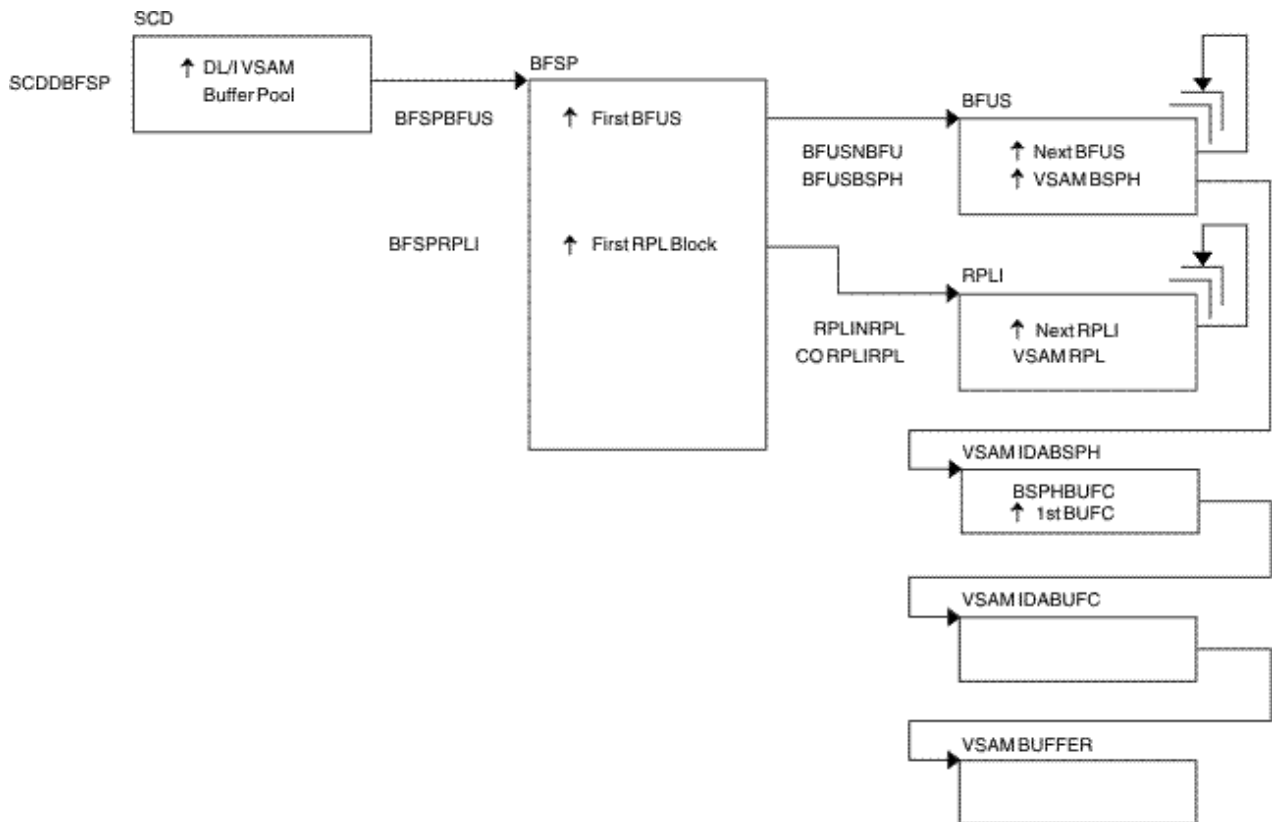


Figure 13. Buffer handler pool (VSAM)

### OSAM DECB with IOB in use

The following figure shows the OSAM DECB with IOB in use.

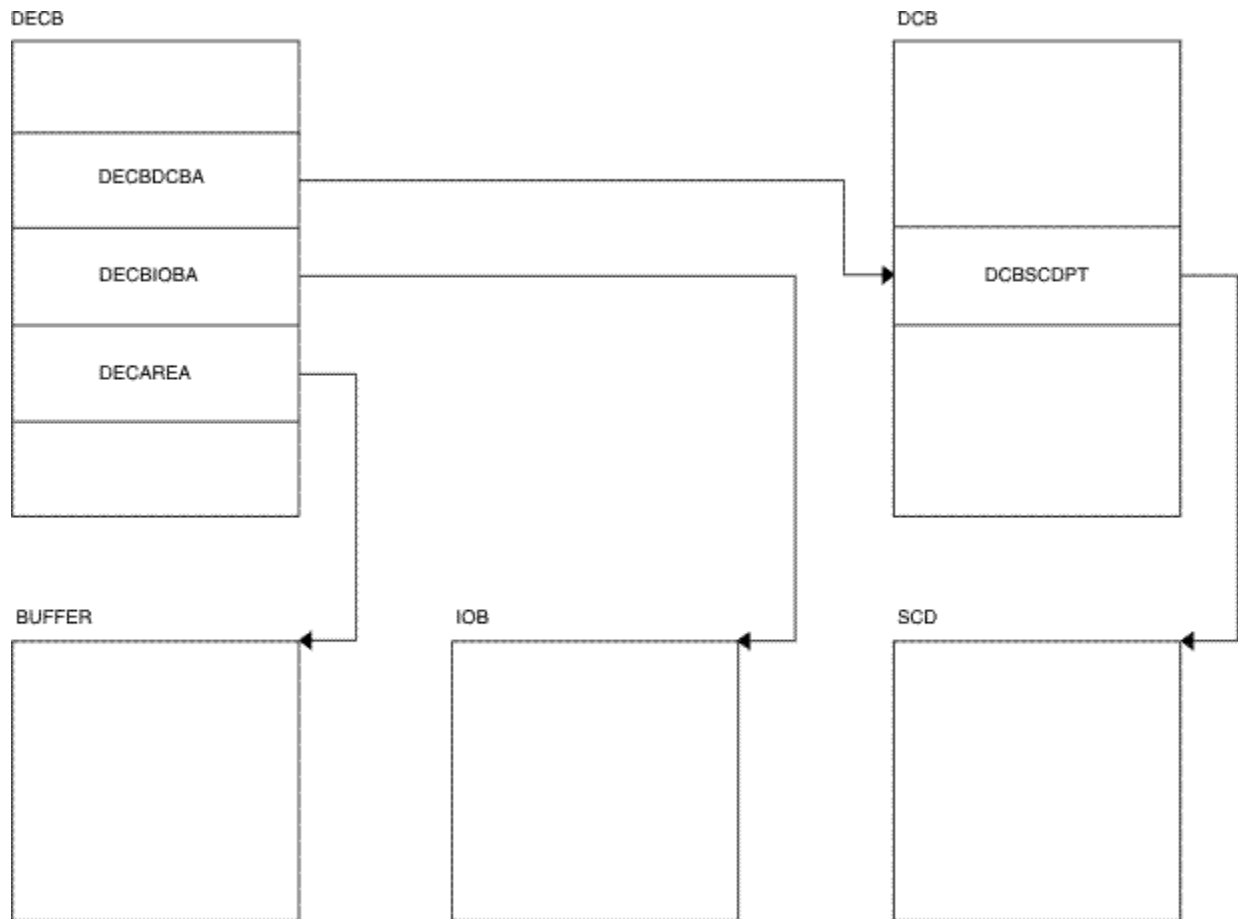


Figure 14. OSAM DECB with IOB in use

### OSAM IOB pool showing available IOBs

The following figure shows OSAM IOB pool showing available IOBs.

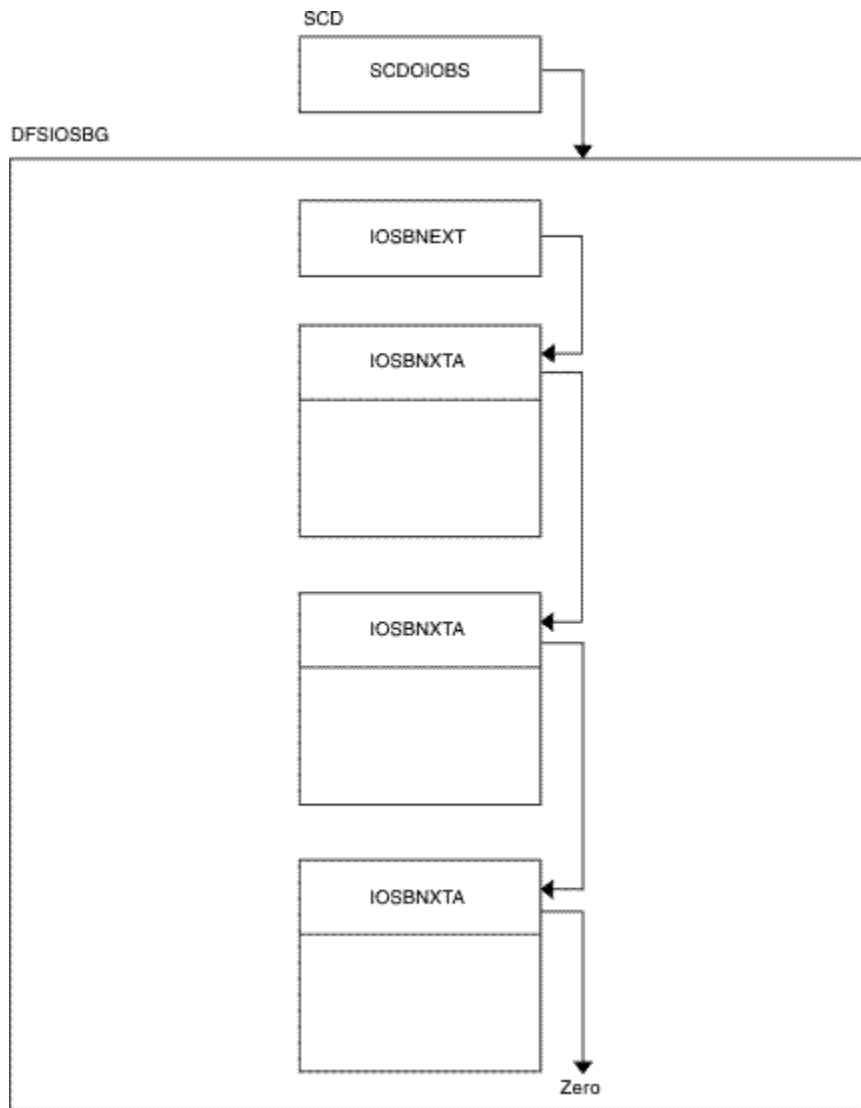


Figure 15. OSAM IOB pool showing available IOBs

Storage allocated using the ICREATE/IDESTROY macros is obtained from the MAIN (WKAP) pool. The control block relationship for the MAIN pool is shown in [Figure 16 on page 102](#).

#### Storage management control block relationships created for the MAIN pool

The following figure shows storage management control block relationships created for the MAIN pool.

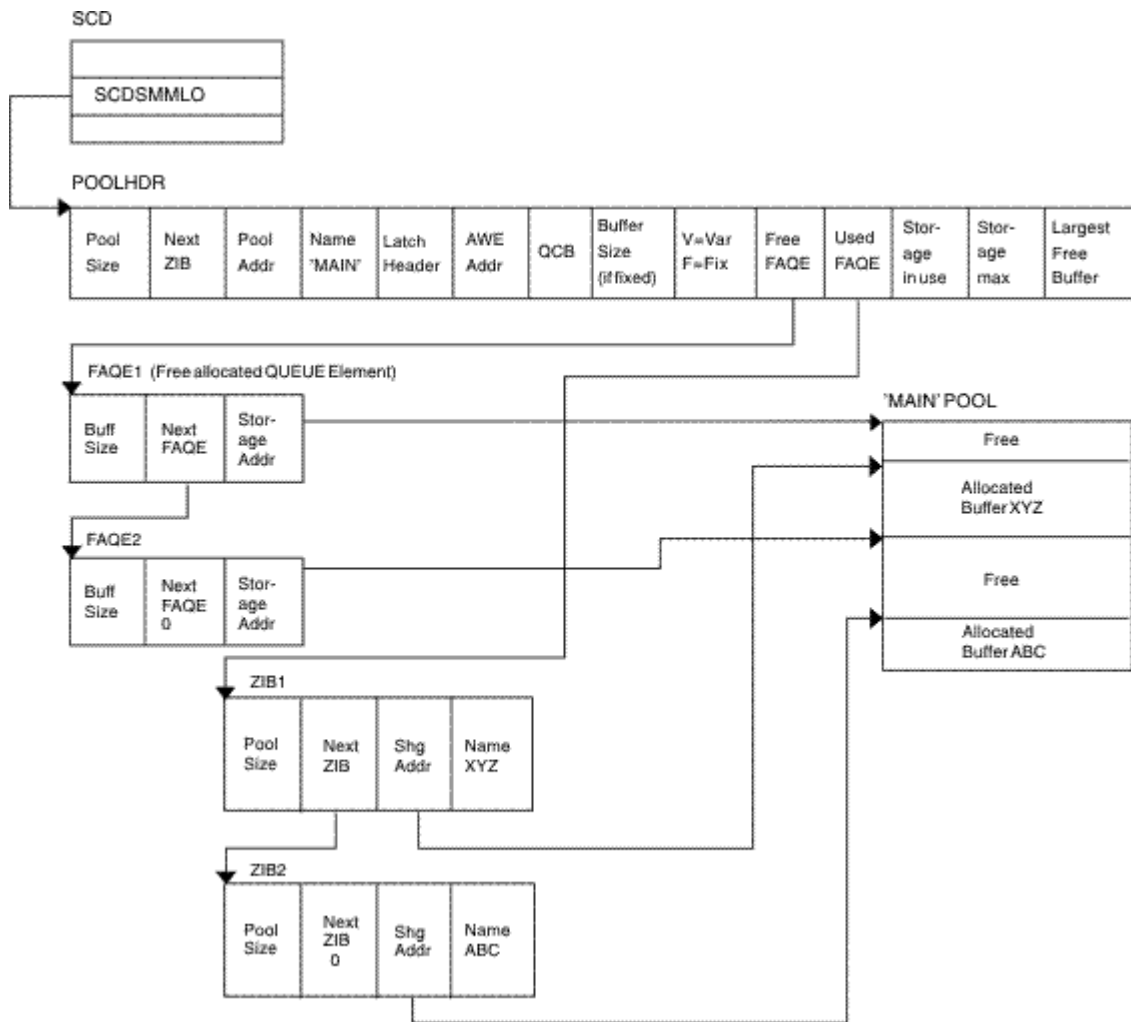


Figure 16. Storage management control block relationships created for the MAIN pool

### Storage management control block relationships for preallocated storage blocks

The following diagram shows the control block relationships for those pools managed by the DFSISMN0 Storage Manager.



be compressed, serialization logic is not required when allocating or releasing a buffer from one of these blocks.

If the primary block is not obtained until the first GET request, it along with any secondary blocks are placed on the compressible block chain anchored off the pool header. Serialization logic must be used when scanning the blocks on the compressible chains.

An 8-byte prefix and an 8-byte suffix is added to each buffer. The prefix and suffix are used by the Storage Manager exclusively. The size of the prefix and suffix is included in the current pool size.

The buffer size used to satisfy an incoming request is determined on a best fit basis. Unless the size of the buffer requested is the same size as the actual buffer, there is some unused storage between what the caller views as the end of the buffer and the actual end of the buffer. The buffer the user receives appears to be of the size requested. Any unused space is transparent.

The following pools are defined with user overlay detection: AOIP, CIOP, CMDP, DYNP, EMHB, HIOP, LUMC, LUMP, and SPAP. If a pool is defined with user overlay detection, an 8-byte constant is added to the user portion of the buffer. As far as the caller is concerned, the length of the buffer received is the length requested, followed by an 8-byte constant. For example, if a caller requests a 100-byte buffer from a pool with a user overlay detection, and the smallest buffer size available to satisfy the request is 128 bytes, the user overlay detection constant is placed at an offset of 100 bytes into the buffer. Bytes 107 through 127 are unused.

The user overlay detection constant is used by IMS modules. The Storage Manager does not look at the user overlay detection constant.

#### **Storage management control block relationships (DFSPool pools)**

The following figure shows storage management control block relationships (DFSPool pools).

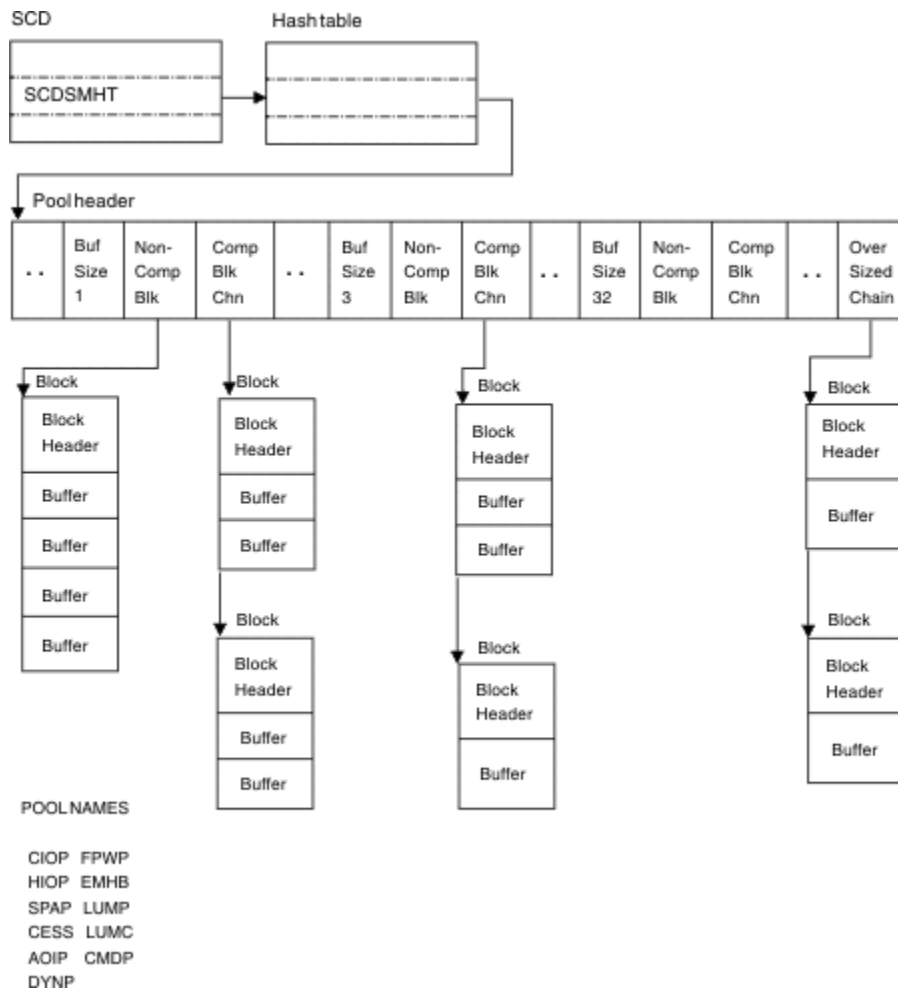


Figure 18. Storage management control block relationships (DFSPPOOL pools)

### Storage management control block relationships (DFSCBT00 pools)

The following figure shows the Storage Management (DFSCBT00 Pools) control blocks relationships.

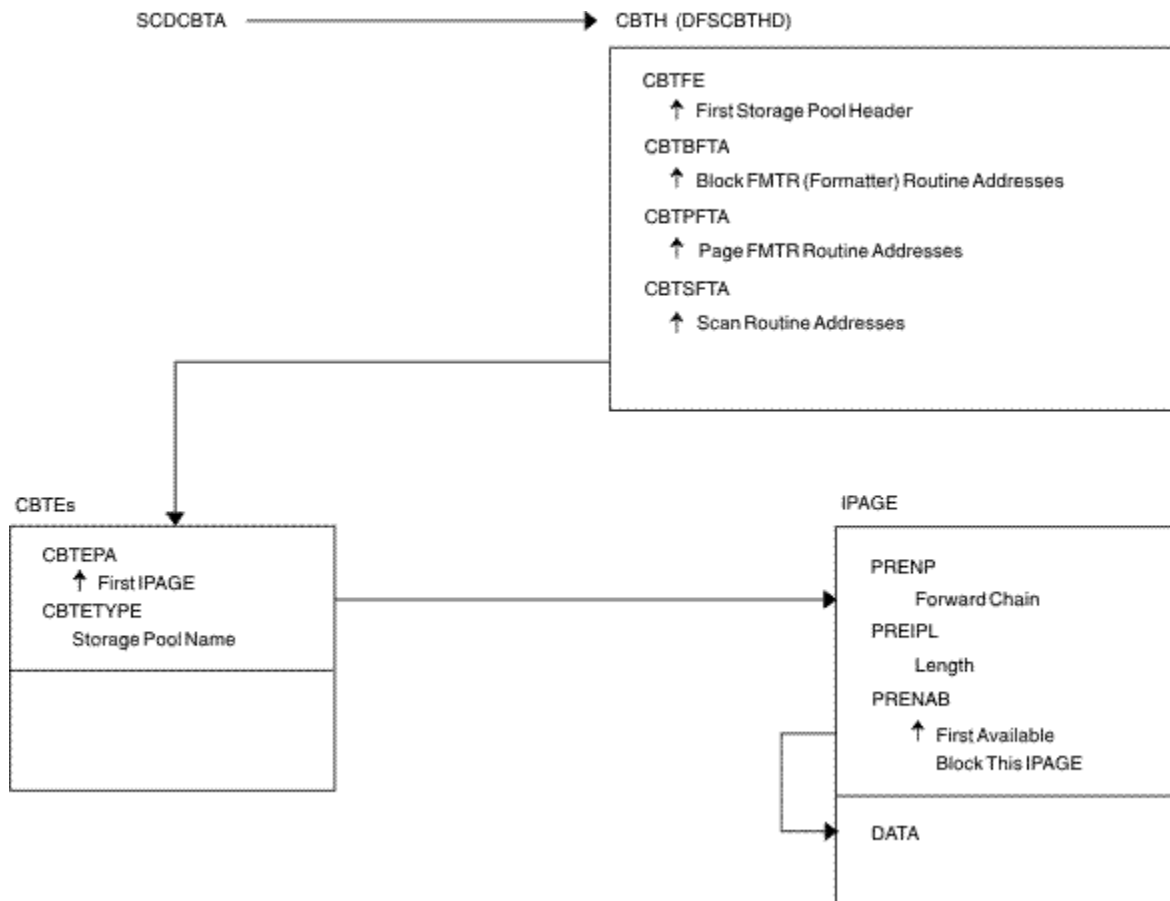


Figure 19. Storage management control block relationships (DFSCBT00 pools)

### Database Manager control blocks for a representative database

The following figure shows the Database Manager control blocks for a representative database.



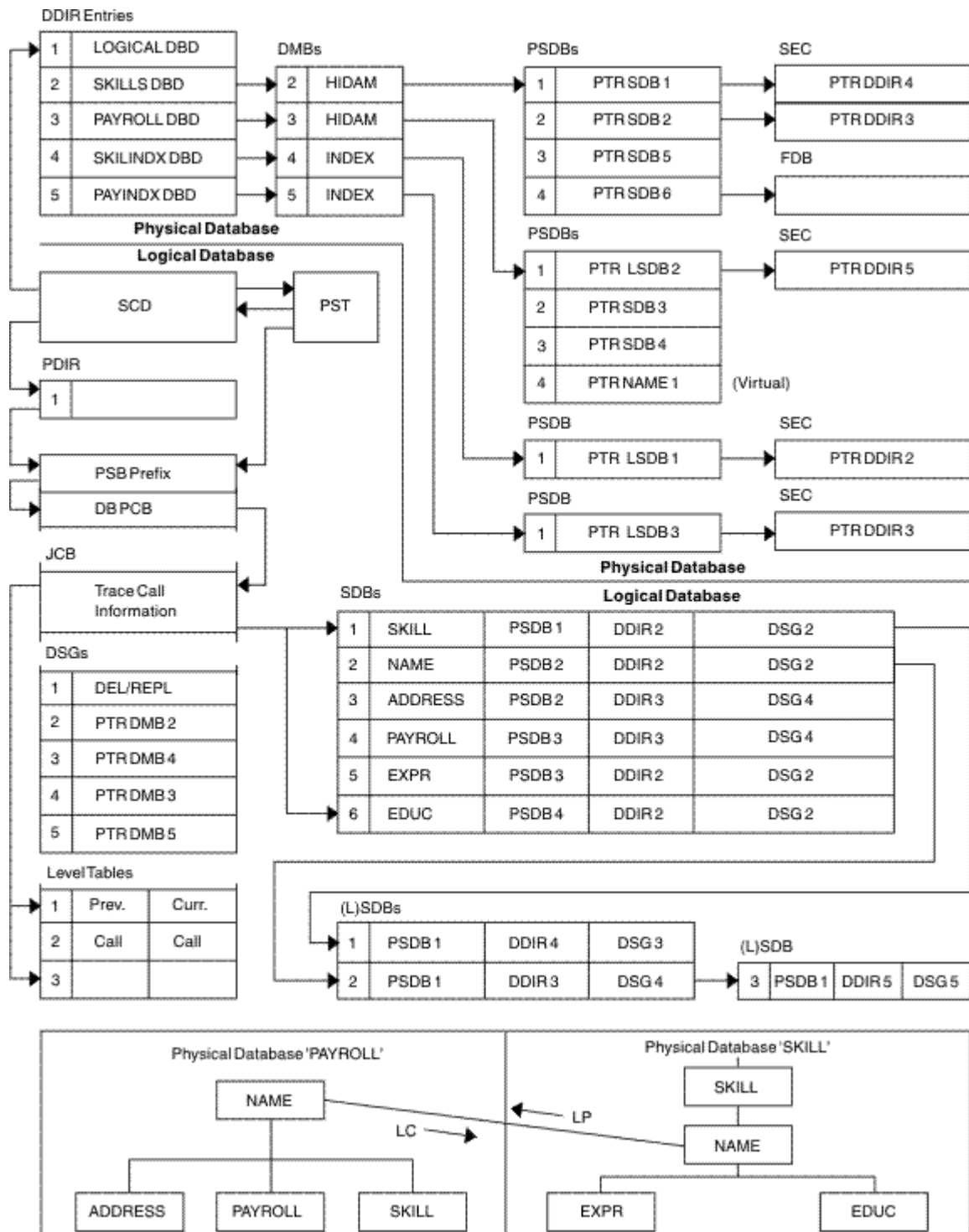


Figure 20. Database Manager control blocks for a representative database

**Note the following HALDB differences for Figure 20 on page 107:**

- The SDBs pointer to the DDIR always points to the HALDB Master's DDIR.
- The PSDBs are under the HALDB master DMB in the DMB pool. The partition DMBs do not contain PSDBs.
- There is no separately defined DDIR or DMB for the primary INDEX database of a PHIDAM. Instead there is an additional AMP in the partition DMB for the primary index.
- There is an ILE DSG for the ILDS which follows the Delete/Replace DSG.

The following figure shows the relationships between database control blocks.

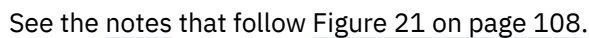


Figure 21. Database control blocks: Part 1 of 2

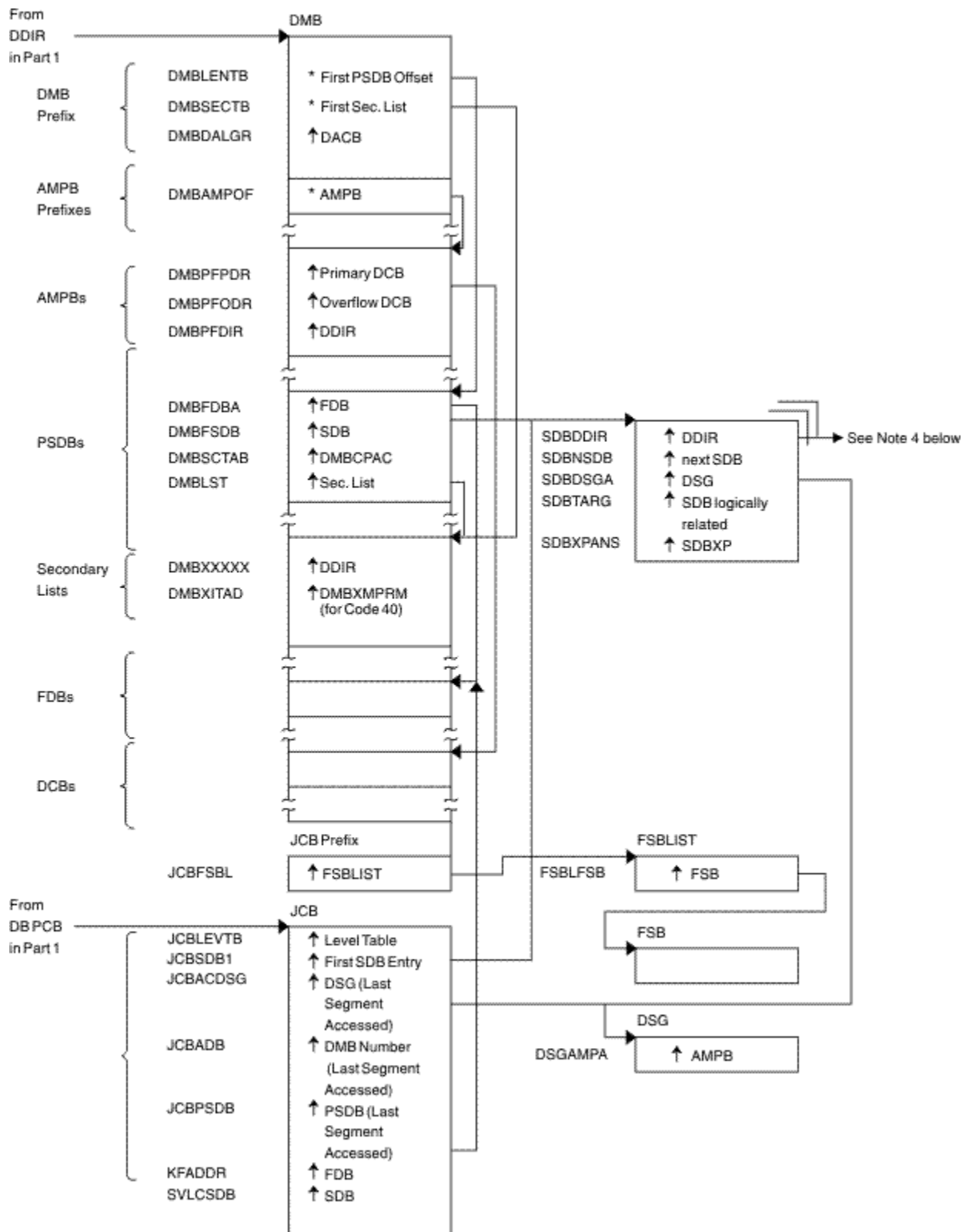


Figure 22. Database control blocks: Part 2 of 2

**Notes to Figure 21 on page 108:**

1. See Figure 4 on page 90.
2. See Figure 10 on page 96.

3. This is a unique HALDB control block. This control block points the partition DDIR to each other and points the partition DDIR to the master DDIR.
4. For HALDB, the SDB points to the Master DDIR.

### Diagram of a Data Management Block (DMB)

The following figure shows a diagram of a Data Management Block (DMB).

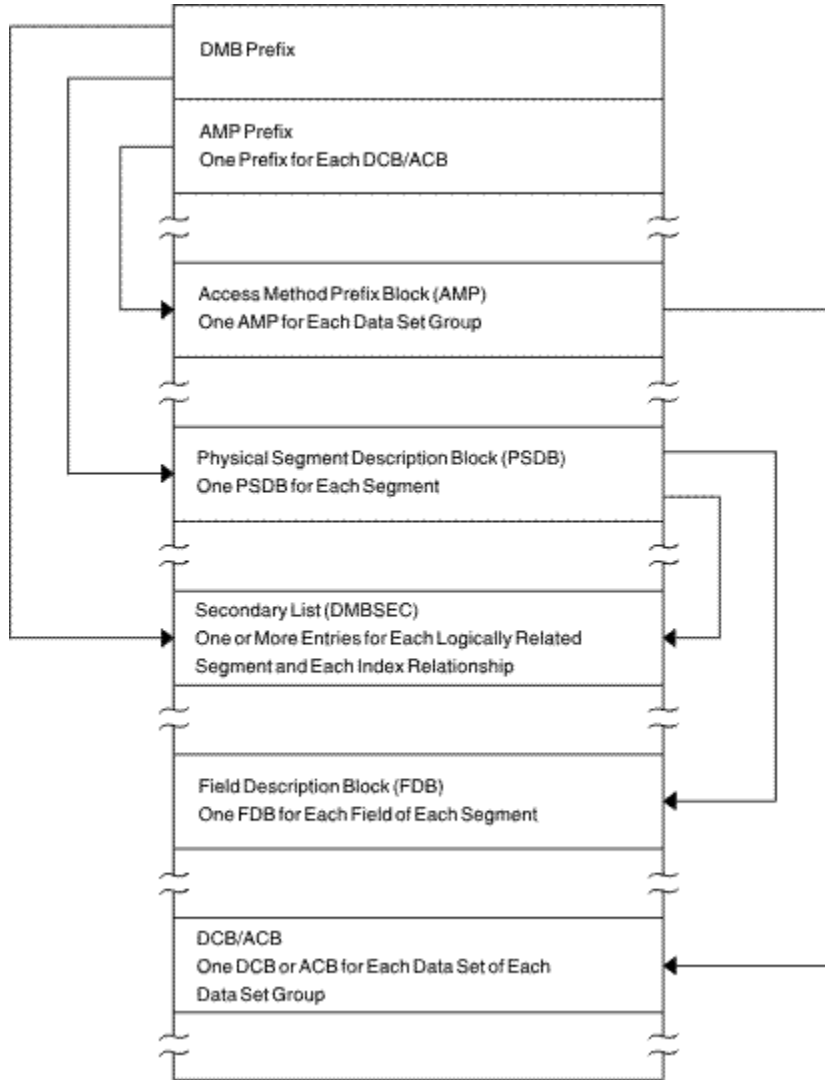


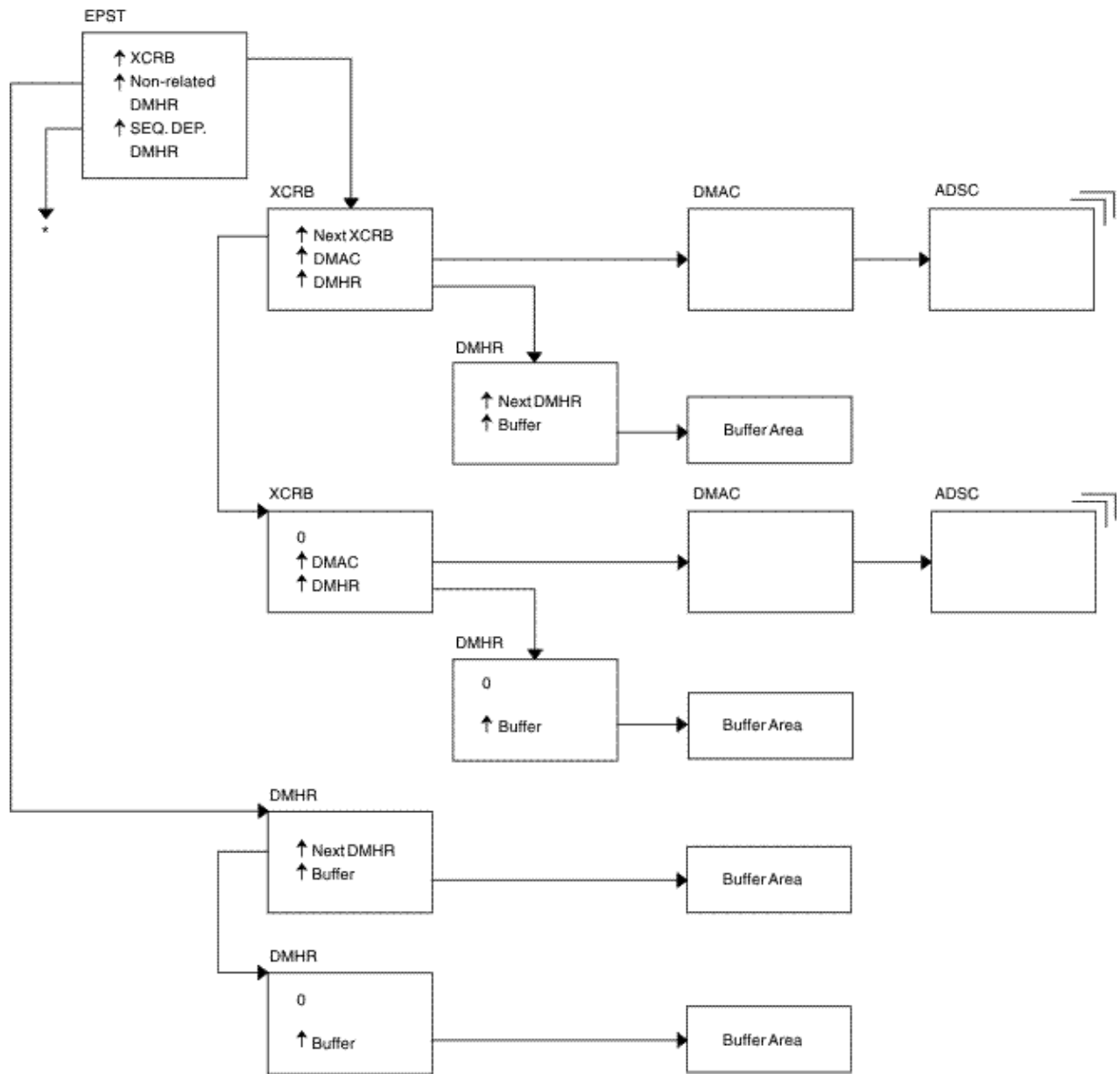
Figure 23. Diagram of a Data Management Block (DMB)

**Note to Figure 23 on page 110:** For a HALDB, dual DMBs exist in storage. When HALDB Online Reorganization is not in progress, one DMB is active and the other inactive. When HALDB Online Reorganization is in progress, both DMBs are active, with one DMB representing the input data sets, and one DMB representing the output data sets.

### Overview of Fast Path control blocks

The following figure shows an overview of Fast Path Control Blocks.





\* EPSTSDBH (This chain is identical to non-related DMHR chain.)

Figure 25. Relationships between buffer control blocks for Fast Path databases

### GSAM control block overview

The following figure shows a GSAM control block overview.

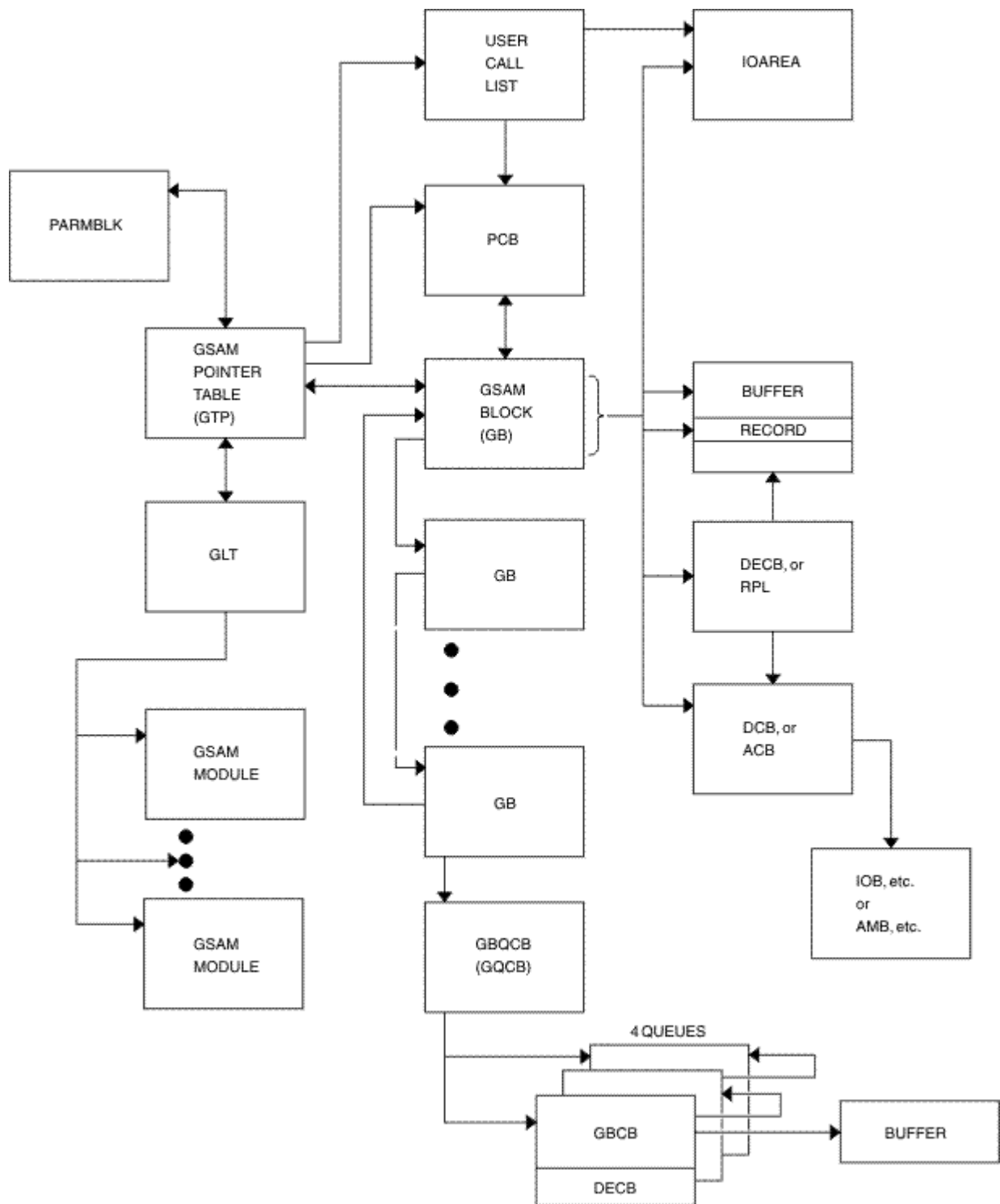


Figure 26. GSAM control block overview

### GSAM control blocks

The following figure shows the GSAM control blocks.

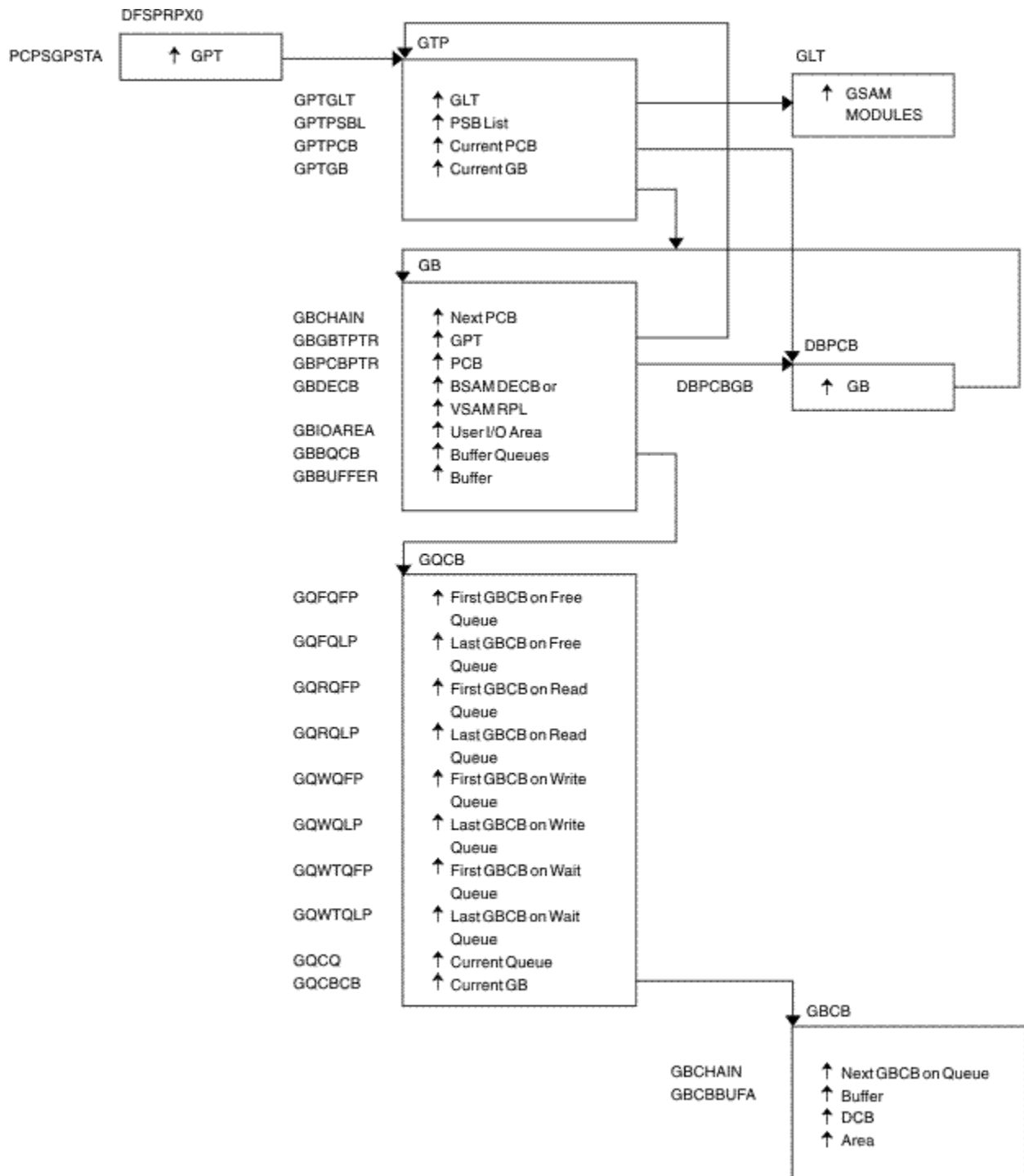


Figure 27. GSAM control blocks

### DL/I control block relationships

The following figure shows the DL/I control block relationships.



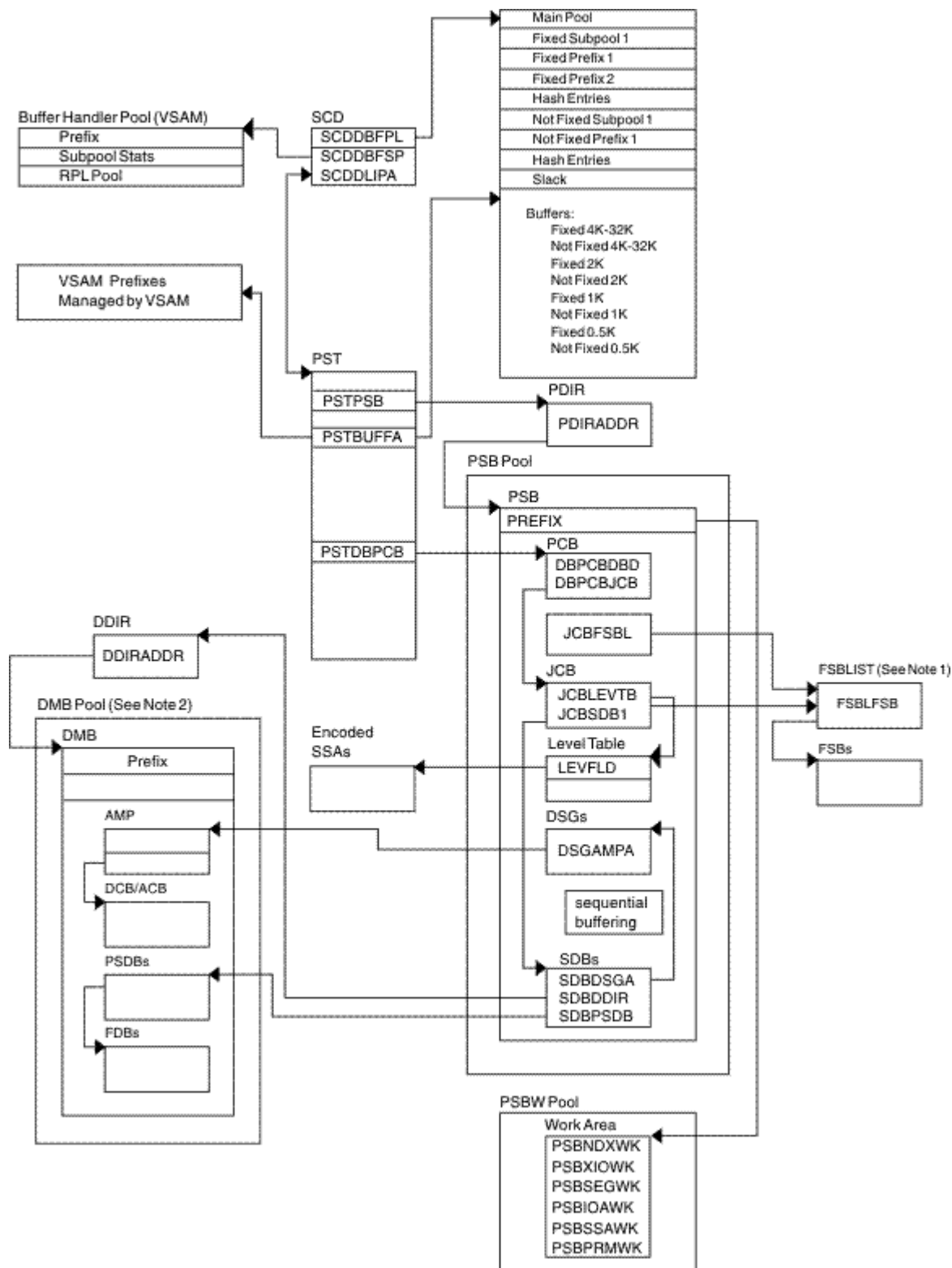


Figure 28. DL/I control block relationships

**Notes to Figure 28 on page 115:**

1. The FSBLIST contains pointers to the Field Sensitivity Block (FSB). The FSB describes this user's logical use of the sensitive field.
2. A partition HALDB DMB is not in the DMB pool. For HALDB, only the Master DMB is in the DMB pool.

## IMS Transaction Manager control blocks

The following figure shows the IMS Transaction Manager control blocks.

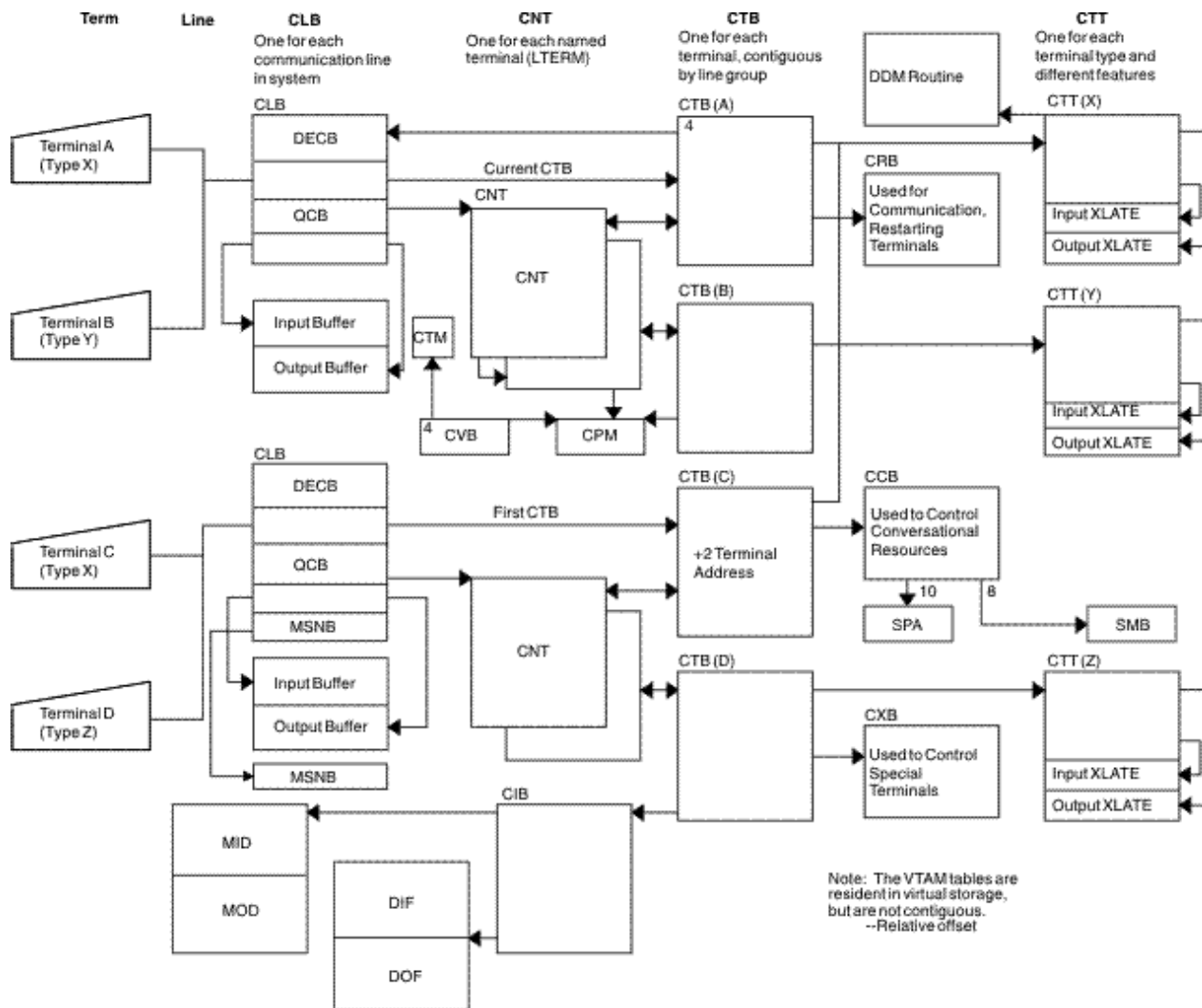
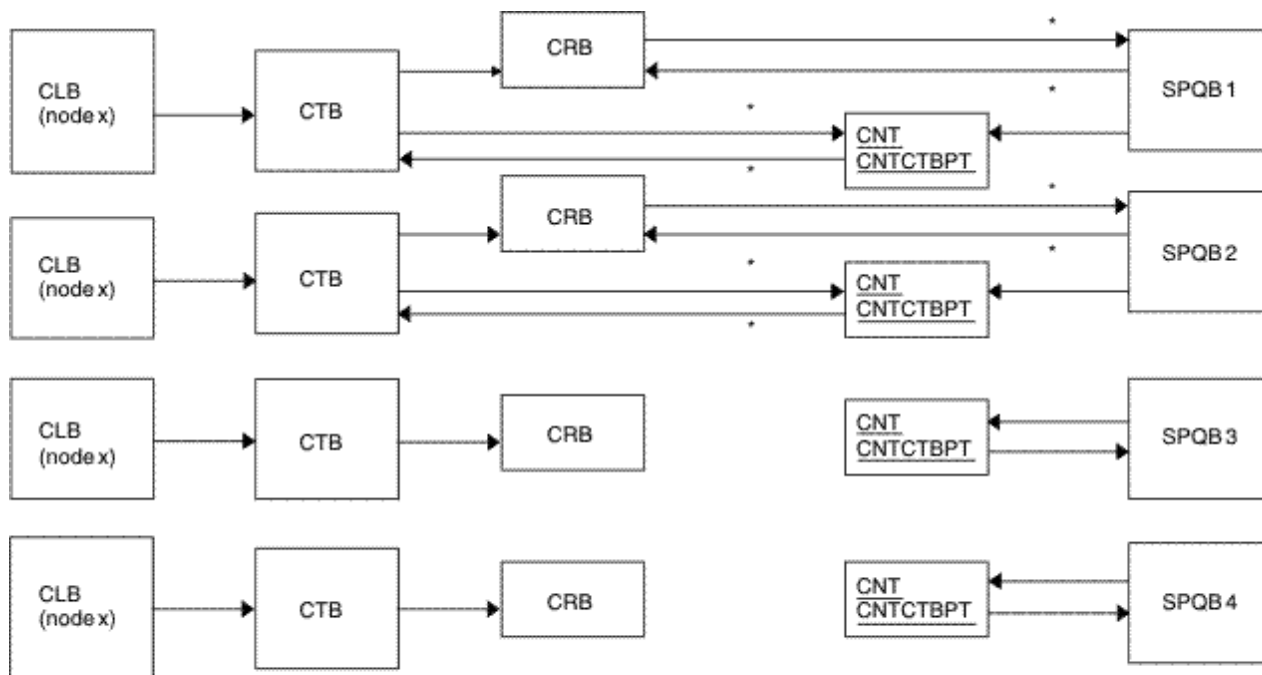


Figure 29. IMS Transaction Manager control blocks

## Intersystem communication control block structure

The following figure shows the intersystem communication control block structure.



**Note**

Subpool Queue Blocks (SPQB1 and SPQB2) are allocated for sessions. SPQB3 and SPQB4 are not. One SPQB is required for each parallel session.

\* Asterisks indicate that these pointers are set when blocks are allocated.

Figure 30. Intersystem communication control block structure

**VTCB load module**

The following figure shows the VTCB Load Module.

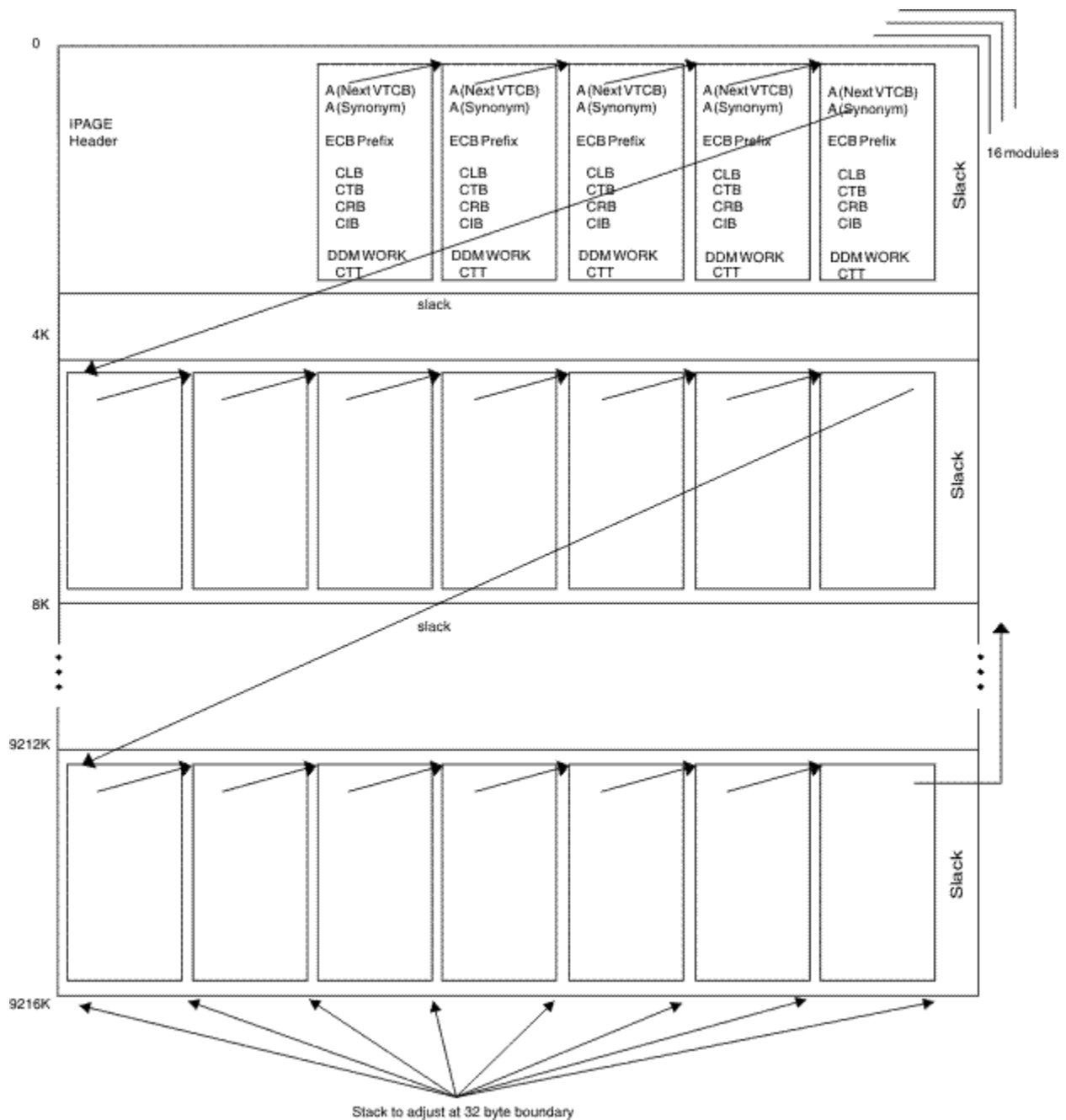


Figure 31. VTCB load module

As illustrated in the following figure, IMS maintains a VTAM terminal control block (VTCB) for each VTAM terminal except MSC VTAM terminals. A VTCB can contain a:

- Communication line block (CLB)
- Communication terminal block (CTB)
- Communication restart block (CRB)
- Communication interface block (CIB)
- Device-dependent module (DDM) work area
- Communication terminal table (CTT) (used only for ETO terminals)

The system of pointers between blocks within a VTCB is the same as the system of pointers used for VTAM terminals.

Some terminals do not require all six blocks. For example, static VTAM blocks use a statically created CTT.

You can find the VTCB for a terminal through the terminal's node name. To do so, you use the DFSCBTS macro interface.

### Multiple Systems Coupling (MSC) control block overview

The following figure shows the Multiple Systems Coupling (MSC) control block overview.

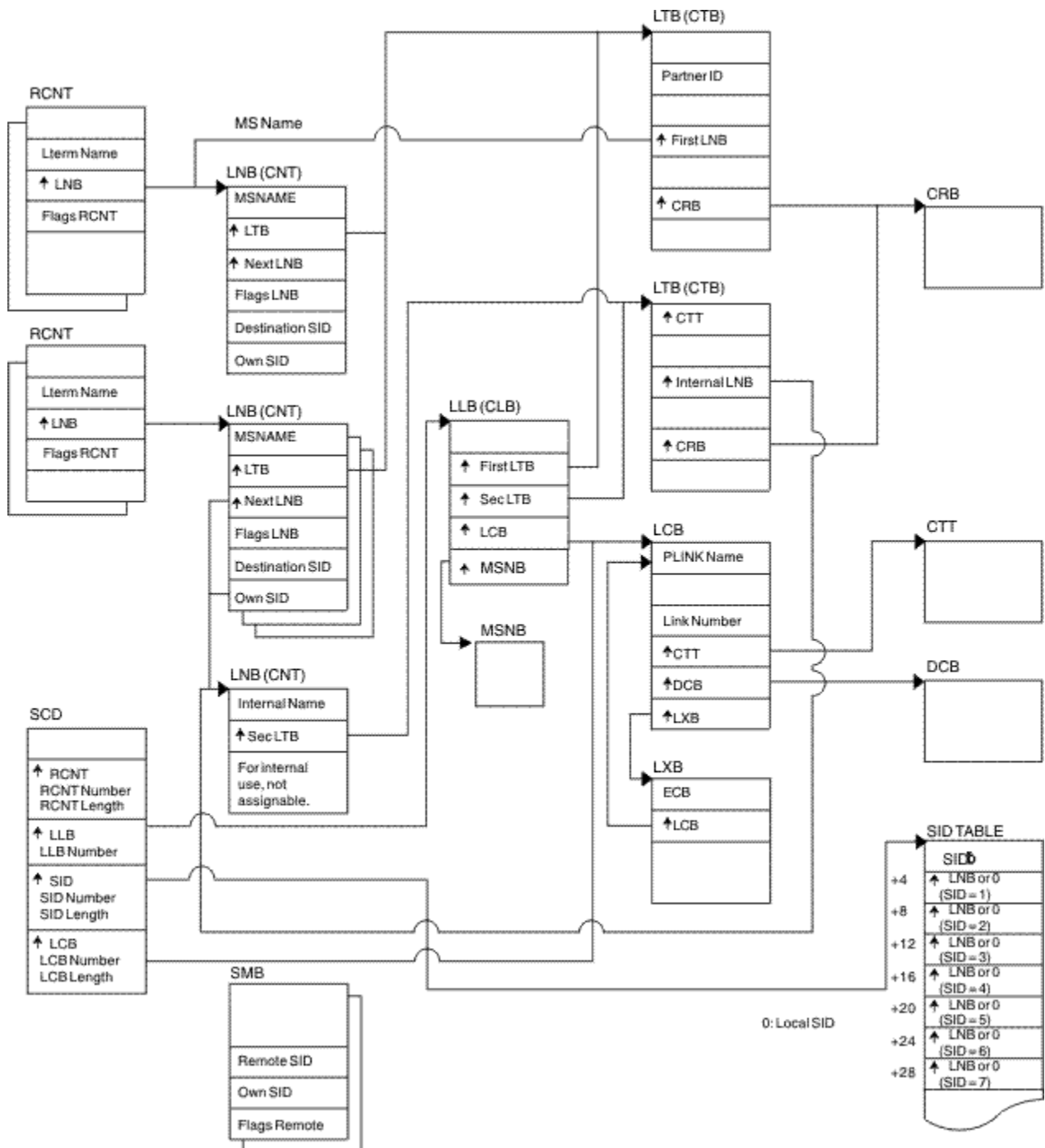


Figure 32. Multiple Systems Coupling (MSC) control block overview

### Multiple Systems Coupling (MSC) main storage-to-main storage control block overview

The following figure shows the Multiple Systems Coupling (MSC) Main Storage-to-Main Storage control block overview.

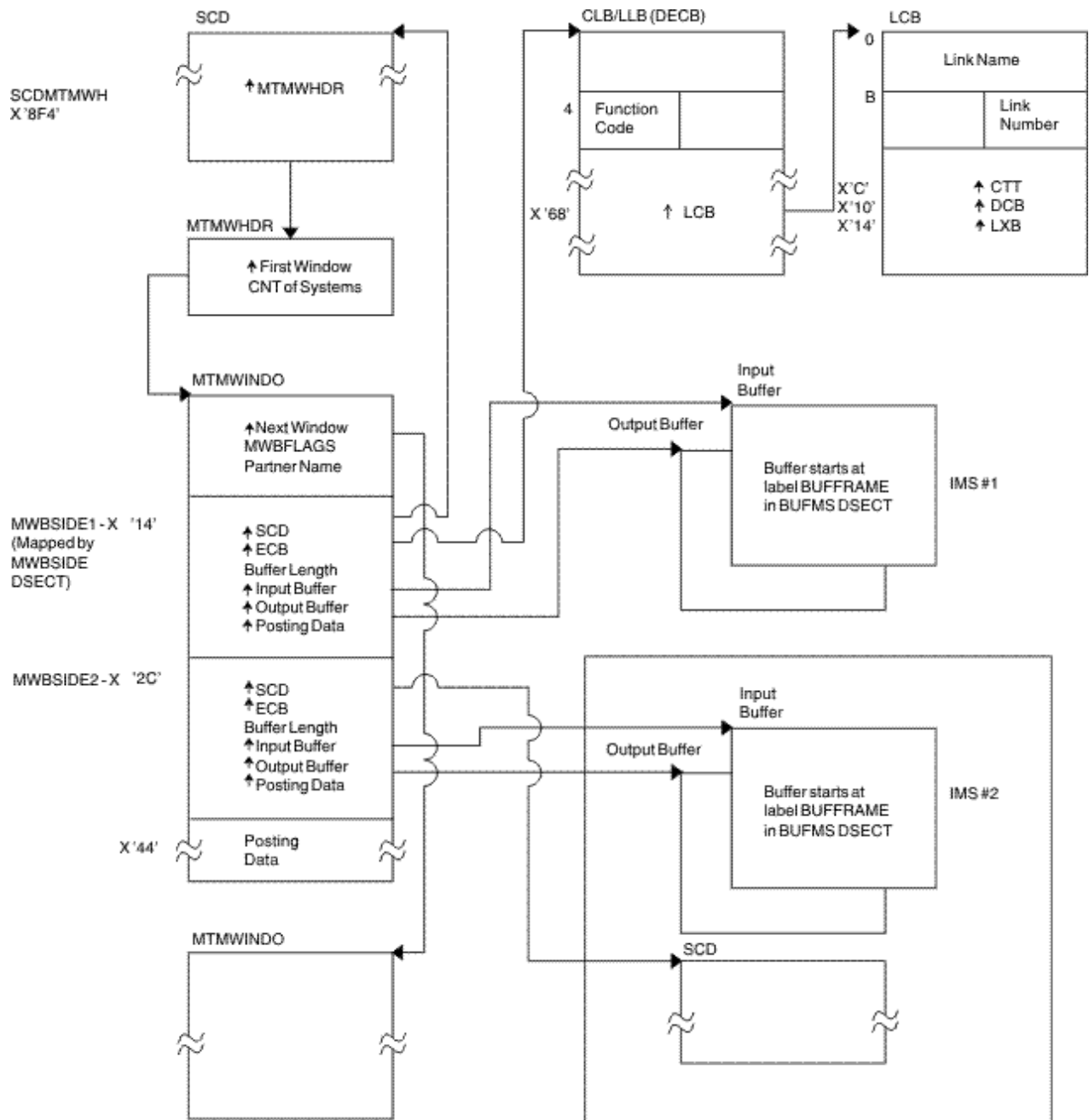


Figure 33. Multiple Systems Coupling (MSC) main storage-to-main storage control block overview

### z/OS storage map showing IMS-to-IRLM interrelationships

The following figure shows a z/OS storage map displaying IMS-to-IRLM interrelationships.

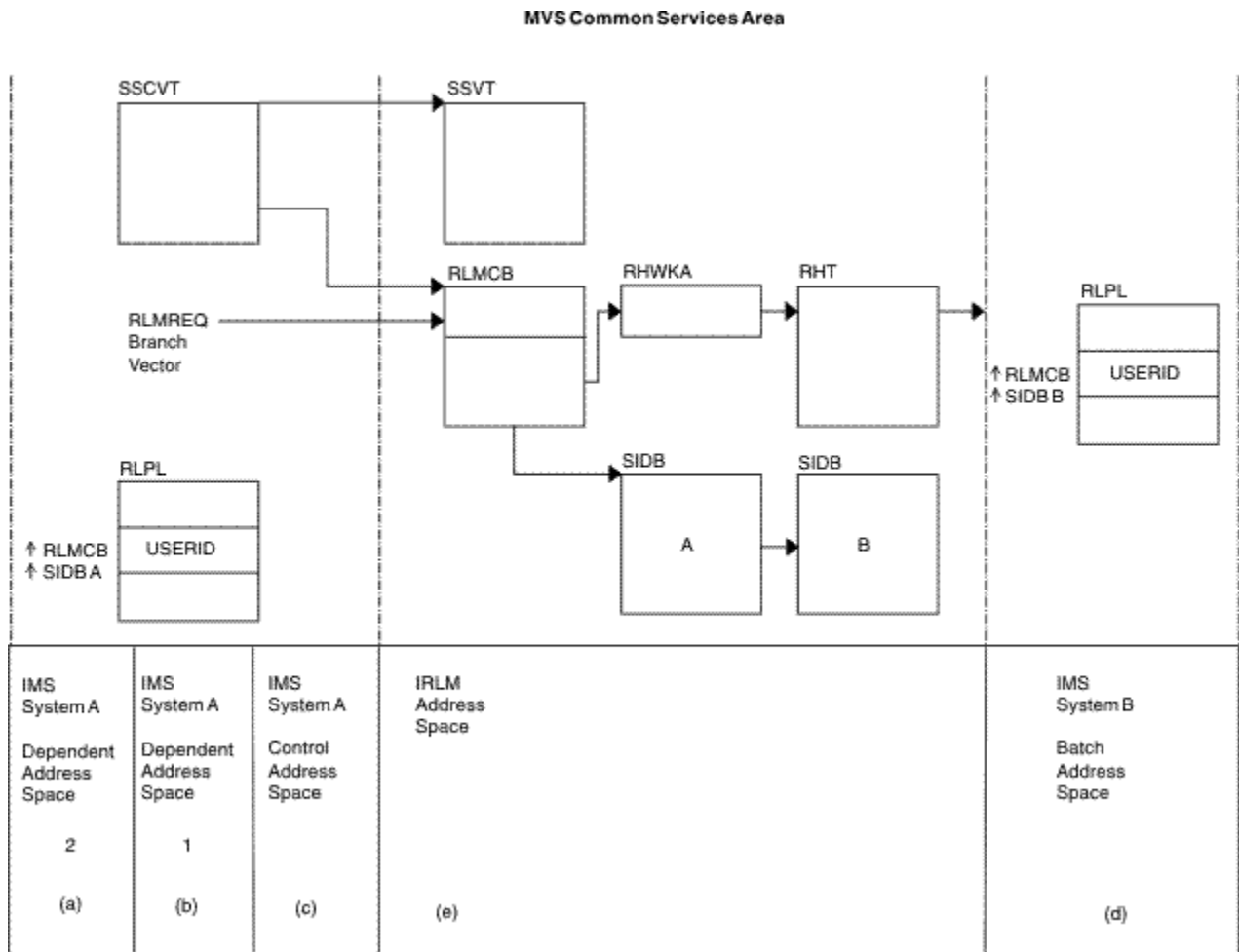


Figure 34. z/OS storage map showing IMS-to-IRLM interrelationships

**Notes to Figure 34 on page 121:**

1. (a), (b), and (c) are z/OS address spaces that make up one online IMS subsystem.
2. (d) is a z/OS address space containing an IMS batch subsystem.
3. (e) is an IRLM address space to which the two IMS subsystems are connected.
4. The RLPLs used by both IMS subsystems reside in the z/OS common services area (CSA).
5. To obtain and release global locks, the IMS subsystems branch to the IRLM code (The subsystems enter the IRLM code through the RLMREQ branch vector within the RLMCB that resides in the CSA.)
6. The IRLM control block structure that controls the global locks resides in the CSA.
7. When PC=YES is in effect, the RHT is in a private address space.

**IRLM overall control block structure**

The following figure shows the overall control block structure of IRLM.

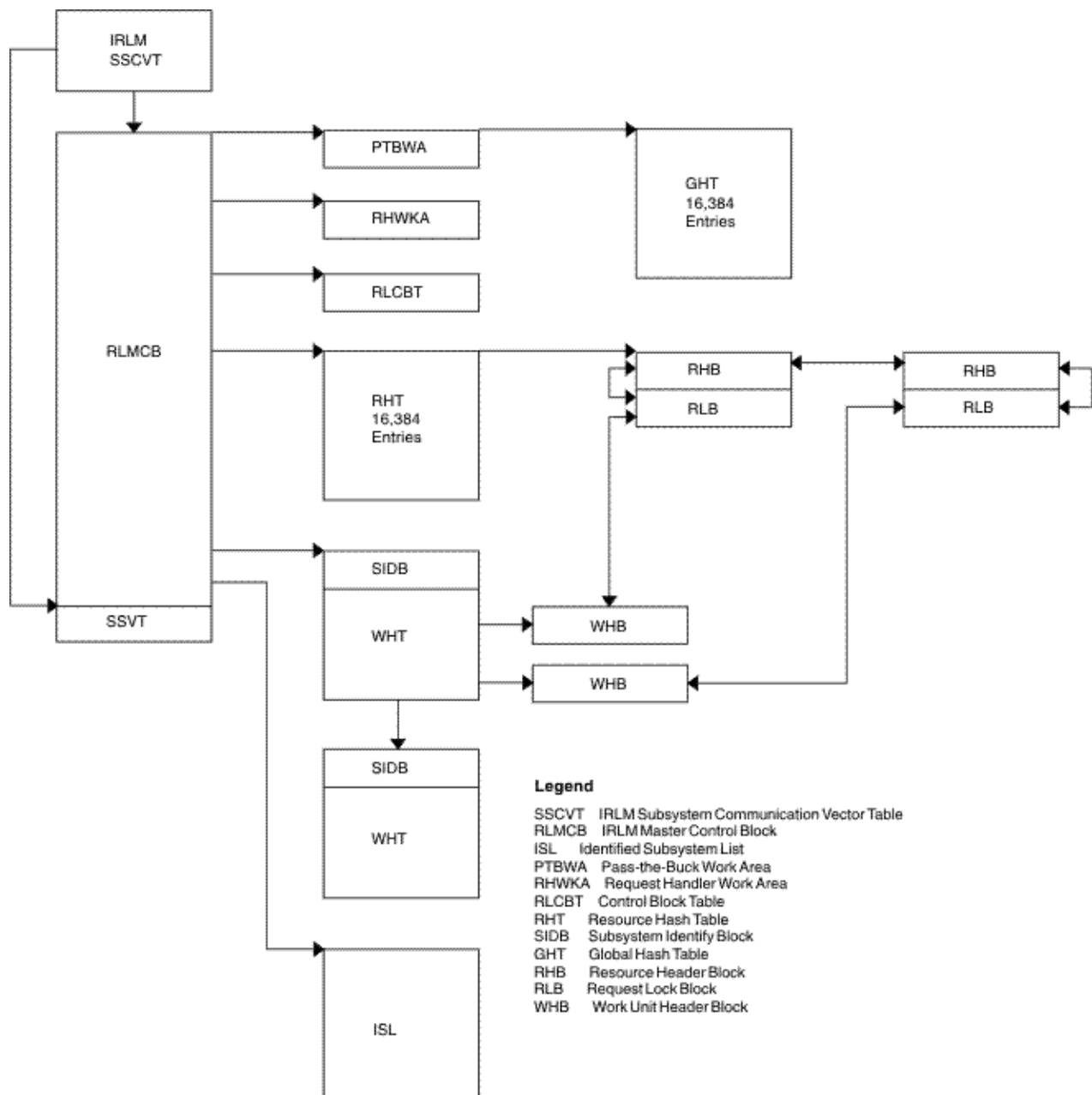


Figure 35. IRLM overall control block structure

### IRLM storage manager pools

The following figure shows the IRLM Storage Manager pools.



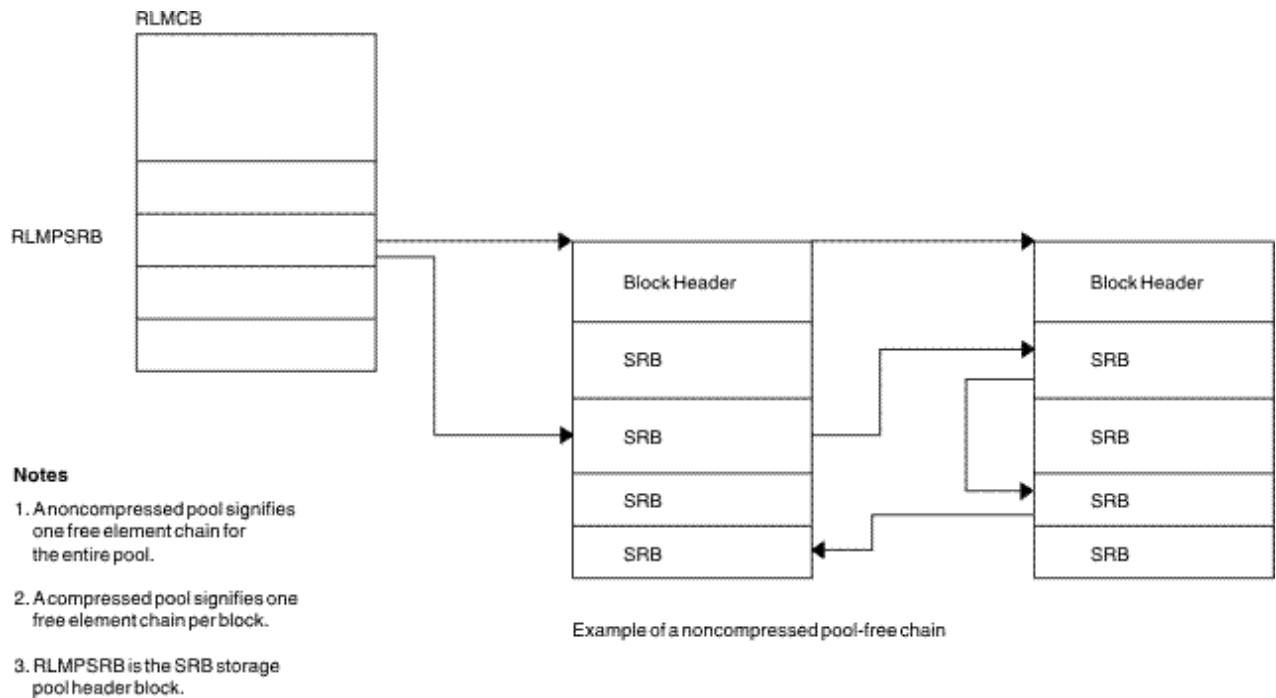


Figure 36. IRLM storage manager pools

### IRLM lock request examples

The following figure shows examples of IRLM lock requests.

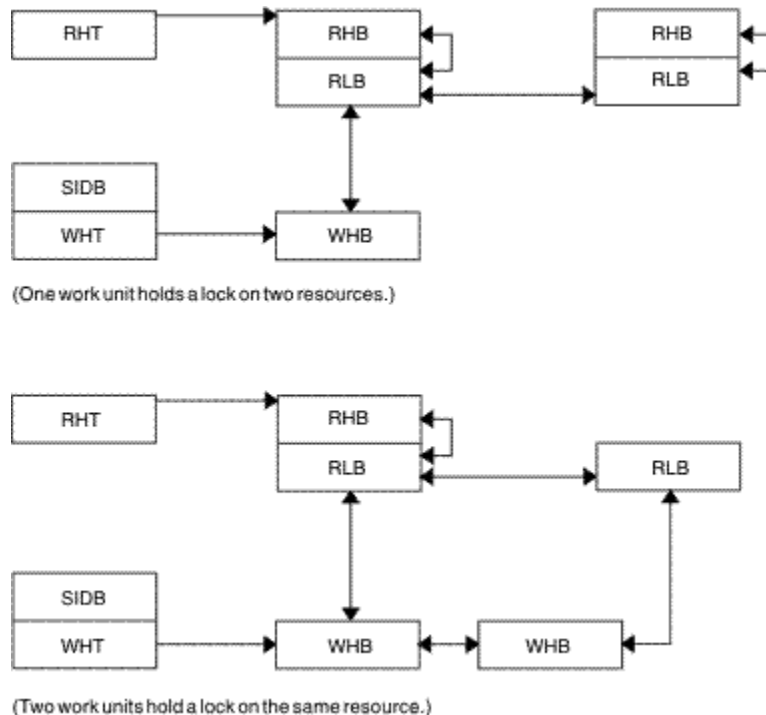


Figure 37. IRLM lock request examples

### Control block overview of Database Recovery Control (DBRC)

The following figure shows an overview of the Database Recovery Control (DBRC) control blocks.

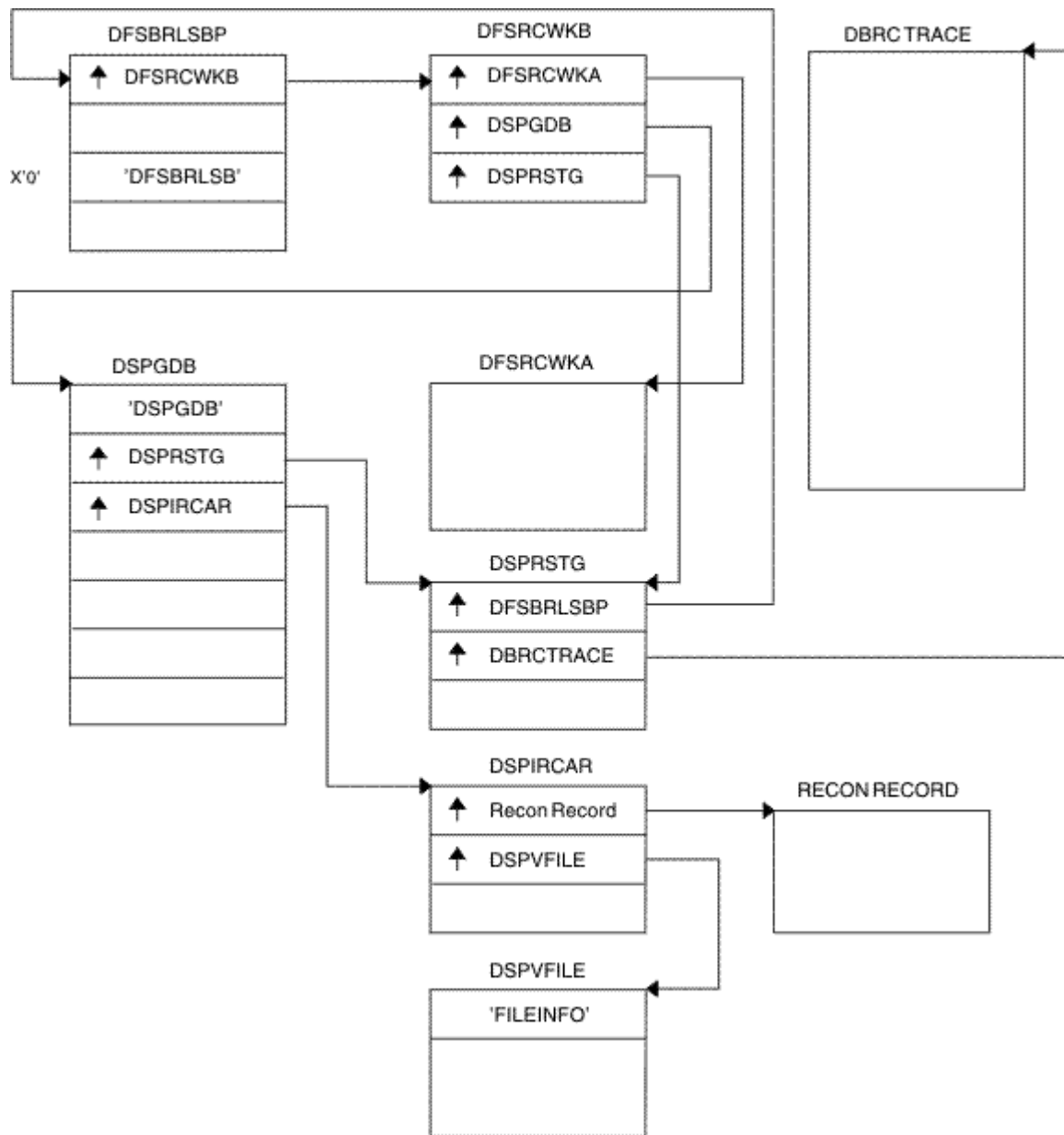


Figure 38. Control block overview of Database Recovery Control (DBRC)

### Organization and basic linkages: DOF (Device Output Format) and MOD (Message Output Descriptor)

The following figure shows the organization and basic linkages of Description Output Format (DOF) and Message Output Descriptor (MOD).

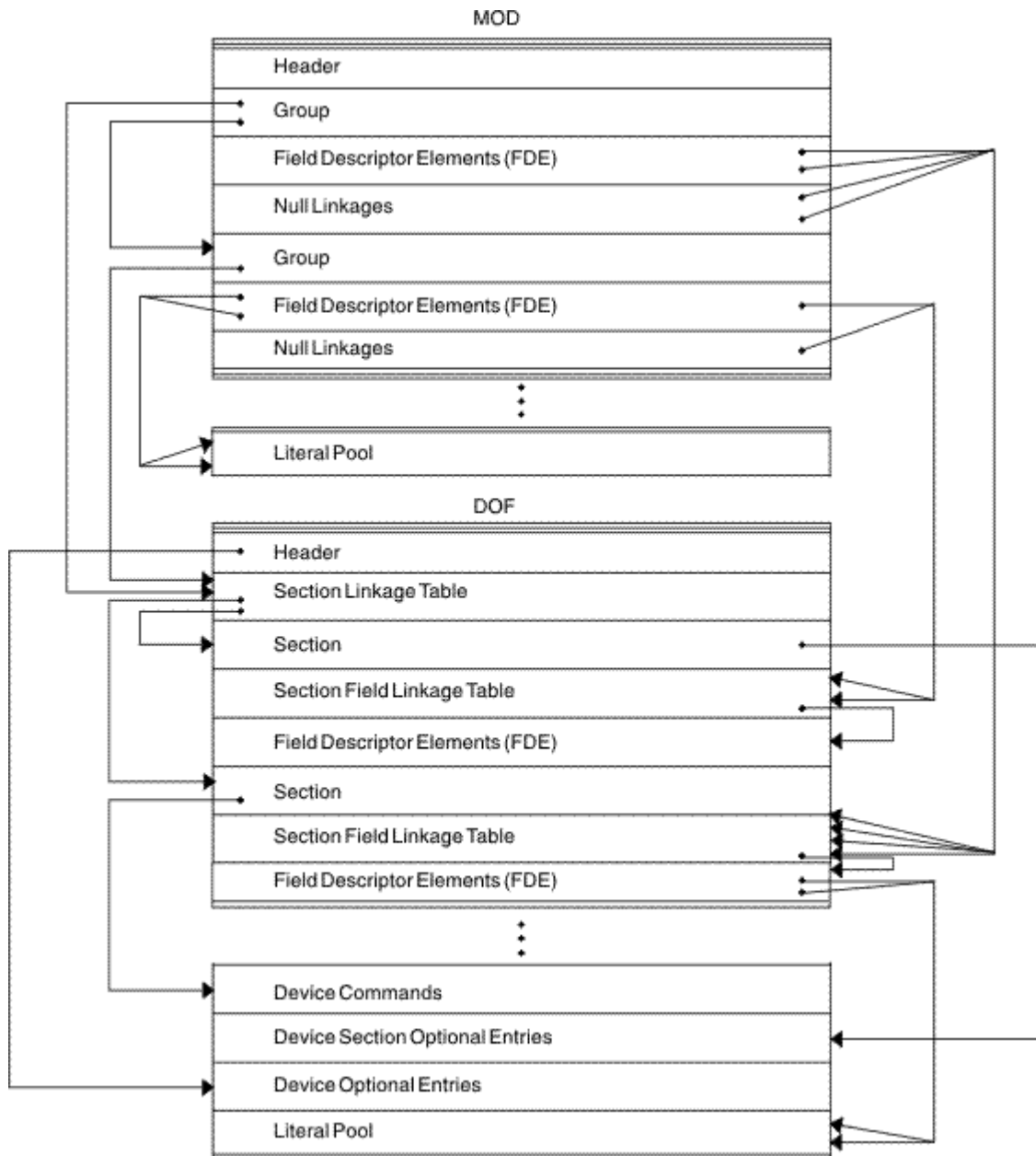


Figure 39. Organization and basic linkages: DOF (Device Output Format) and MOD (Message Output Descriptor)

### Organization and basic linkages: DIF (Device Input Format) and MID (Message Input Descriptor)

The following figure shows the organization and basic linkages between Device Input Format (DIT) and Message Input Descriptor (MID).

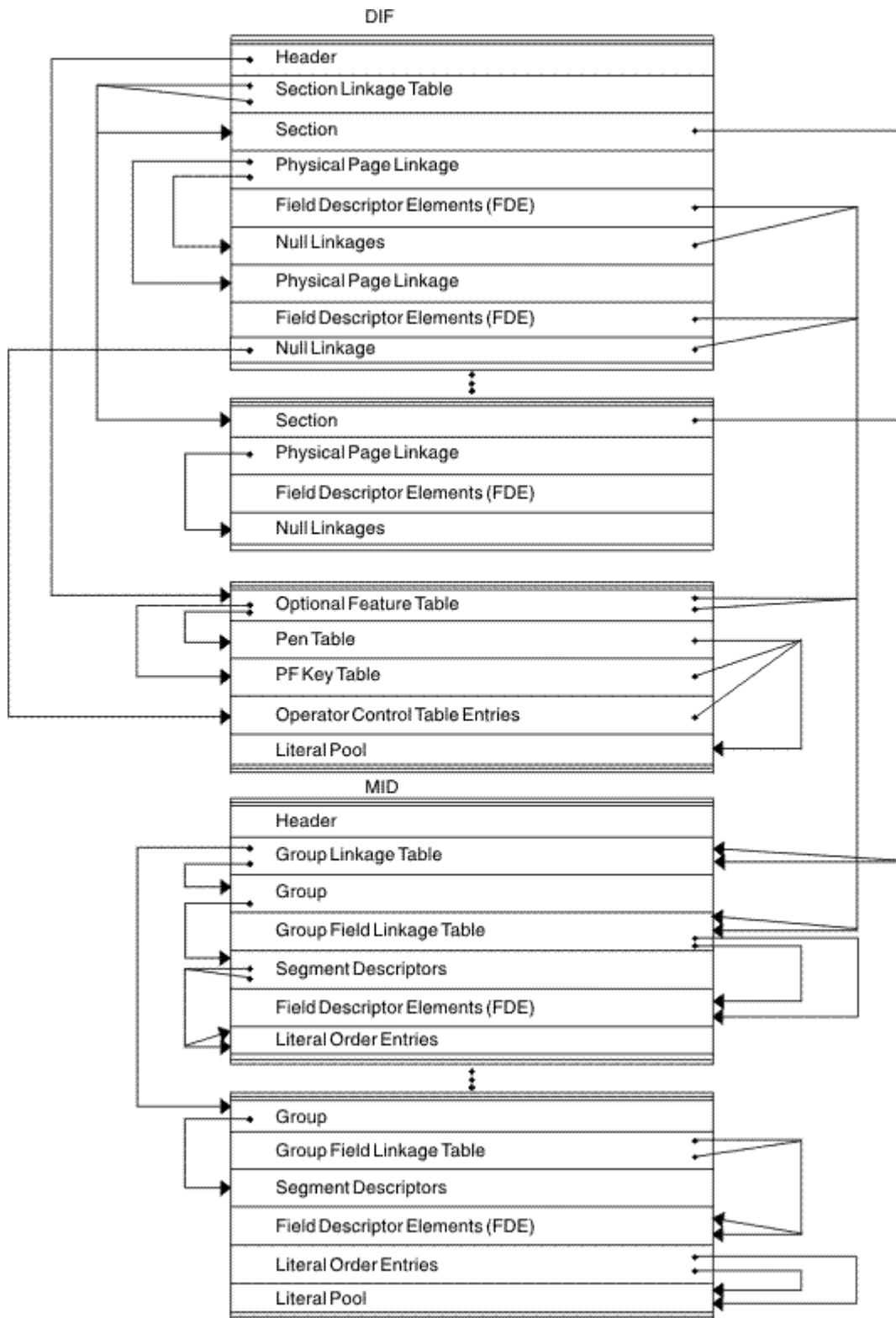


Figure 40. Organization and basic linkages: DIF (Device Input Format) and MID (Message Input Descriptor)

# DL/I record formats

The DL/I address space has seven distinct record formats.

## HSAM and SHSAM database

HSAM and SHSAM databases share the following record formats.

### Segment Formats

The following figure shows the DL/I data record formats for HSAM and SHSAM databases.

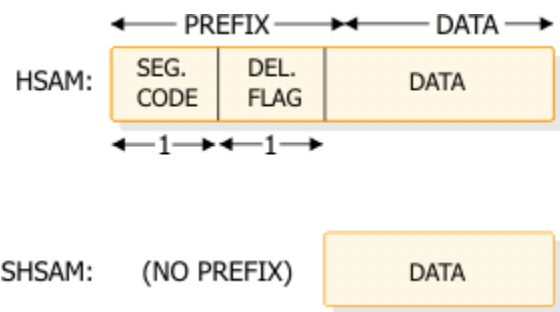


Figure 41. HSAM and SHSAM segment format

### Delete Byte (Flag) Format

The following figure shows the delete byte (flag) format.

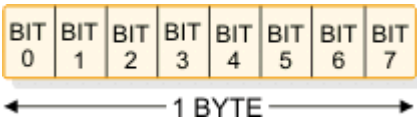


Figure 42. Delete byte (flag) format

#### Bit

#### Description

- |   |  |
|---|--|
| 0 | Segment deleted (HSAM or index).                                       |
| 1 | DB record deleted (HSAM or index).                                     |
| 2 | Segment processed by DELETE.   |
| 3 | Reserved.  |
| 4 | Data and prefix are separated in storage.                              |
| 5 | Segment has been deleted on its physical path.                         |
| 6 | Segment has been deleted on its logical path.                          |
| 7 | Segment space available to be freed; bits 5 and 6 must also be set on. |

### Block Format for HSAM and SHSAM

There are no dependent segments in a SHSAM block. Block size must be a multiple of segment size. The following figure shows the block format for HSAM and SHSAM.

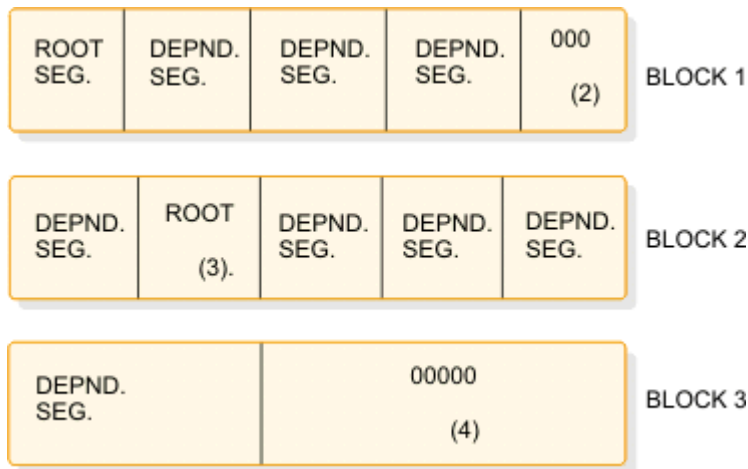


Figure 43. Block format for HSAM and SHSAM

**Notes:**

1. Pad with zeros if no room for next segment.
2. Next database record starts immediately.
3. Pad with zeros in last block, after last segment.

## HISAM and SHISAM database

HISAM and SHISAM databases share the following record formats.

### Segment Format

Figure 44 on page 128 and Figure 45 on page 128 show the segment format of HISAM and SHISAM.

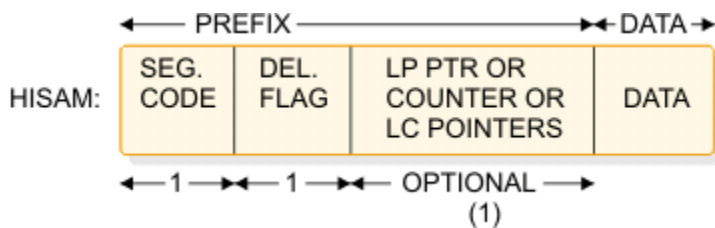


Figure 44. HISAM segment format

**Note:**

1. This field can be omitted, or it can be used to hold:
  - A 4-byte LP pointer (if this segment is an LC).
  - A 4-byte counter (if this segment is an LP).
  - One or more 4-byte LC pointers (if this segment is an LP).



Figure 45. SHISAM segment format

**Note:** This diagram is for a root-only database.

## LRECL Format

The following figure shows the LRECL format.

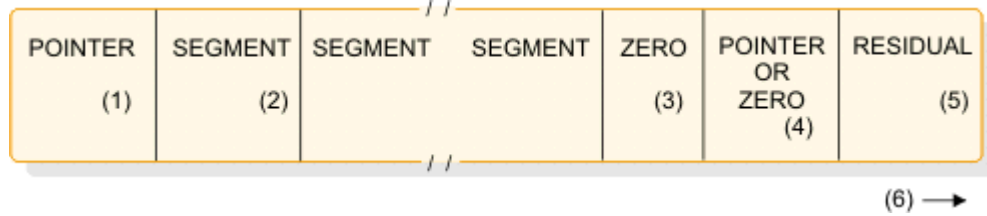


Figure 46. LRECL format

### Notes:

1. 4-byte RBA of ESDS record containing additional dependent segments for this root occurrence.  
SHISAM: This field is omitted.
2. HISAM: Segment includes prefix and data.  
SHISAM: Segment includes only data (no prefix). (See [Figure 45 on page 128](#))
3. 1-byte of zeros indicates the end of segments in this LRECL.
4. This field is omitted.
5. Space not used.
6. VSAM LRECLs must have an even length.

## VSAM Block Formats

The following figure shows the VSAM block formats.



Figure 47. VSAM block formats

### Notes:

1. LRECL length might change between KSDS and ESDS, depending on user definition.
2. 10 bytes in a blocked data set; 7 bytes in an unblocked data set.

## HDAM, HIDAM, PHDAM, or PHIDAM database

HDAM, HIDAM, PHDAM, and PHIDAM databases share the following record format.

### Segment Format

The following figure shows the segment format of HDAM, HIDAM, PHDAM, and PHIDAM databases.

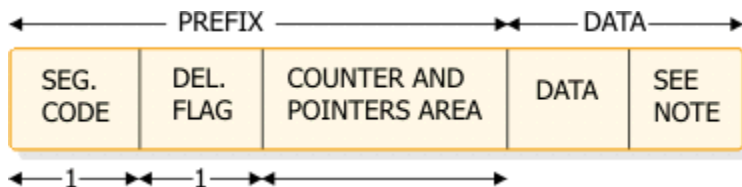


Figure 48. HDAM, HIDAM, PHDAM, or PHIDAM segment format

In order for all segments to be half-word-aligned, a slack byte is added to the end of any segment whose length is an odd number.

### Prefix of a Segment

The following figure maps the prefix of a segment.

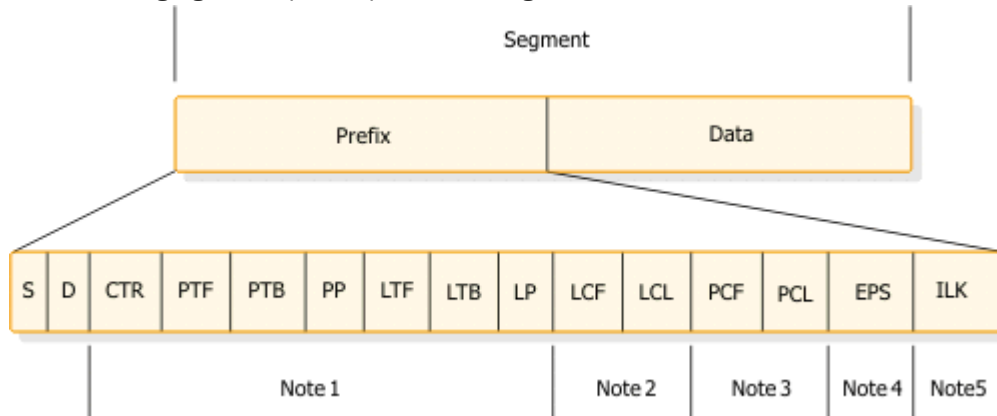


Figure 49. Mapping the prefix of a segment

### Notes to Figure 49 on page 130:

1.

#### Prefix Flag

##### Prefix Flag Description

Segment code (S)

Delete flag (D)

The pointers that exist in this section of the prefix are identified in the PSDB field DMBPTR, as shown in the following list:

**X'80'**

Counter (CTR) for logical relationships

**X'40'**

Physical twin forward (PTF)

**X'20'**

Physical twin backward (PTB)

**X'10'**

Physical parent (PP)

**X'08'**

Logical twin forward (LTF)

**X'04'**

Logical twin backward (LTB)

**X'02'**

Logical parent (LP)



## **X'01'**

Hierarchical direct pointing (For twin-type pointing, this bit is off)

### **2.**

How to locate all logical children: logical child first (LCF); logical child last (LCL)

#### **a.**

At DMBFLAG, if flag DMBLCX (X'20') is on, then DMBLST points to a secondary list for this segment. Secondary lists are used for information concerning indexes, logical children, or the logical parents.

#### **b.**

Secondary list entries whose field DMBSCDE (SEC+0) has flag DMBSLC (X'02') on are descriptions of logical children for a logical parent. Within these secondary lists, the field DMBSLCFL (X'02') has the number of the first and last logical child pointers in the prefix of the logical parent.

#### **c.**

A logical parent can have multiple types of logical children; thus, there can be more than one logical child secondary list entry for a logical parent. The last secondary list for each segment has the DMBSND flag (X'80') set on in the field DMBSCDE (SEC+0).

### **3.**

How to locate all physical children: physical child first (PCF); physical child last (PCL)

#### **a.**

Physical child pointers are only present if this segment uses twin-type pointing rather than hierarchic-type pointing. The PSDB entries for the children of the segment being mapped indicate the number of the pointer in the prefix of the parent segment which points to the first and last occurrence of the child segments.

#### **b.**

The PSDB fields DMBPPFD and DMBPPBK are used for these numbers. The PSDB entries for the children of the segment being mapped can be found by scanning the PSDBs for those segments with a parent segment code (PSDB+1) that matches the segment code (PSDB+0) of the segment being mapped.

### **4**

An EPS (extended pointer set) that is 28 bytes in length is present in the prefix of an LC segment prefix of a HALDB.

### **5**

An ILK (indirect list entry key) that is 8 bytes in length is present in each segment of a PHIDAM or PHDAM.

## **Related concepts**

[“Dump analysis introduction” on page 161](#)

In a pseudoabend supervisor call (SVC) dump that is generated by module DFSERA20, you can find the failing program specification table (PST) by searching the save areas for the caller of module DFSERA20. In the save area flow, DFSERA20 is called INTERA20 and register 1 contains the failing PST address.

## **OSAM and VSAM ESDS block format**

OSAM and VSAM ESDS blocks share the following format.

### **Block format**

The following figure shows the OSAM and VSAM ESDS block format.

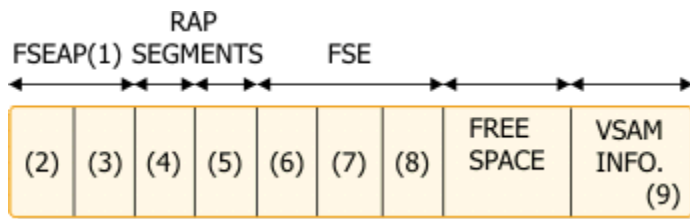


Figure 50. OSAM and VSAM ESDS block format

**Notes:**

1. Free space element anchor point.
2. 2-byte offset to first free space element; contains zeros in a bit map block.
3. 2-byte length (see 7); value is zero.
4. 4-byte root anchor point (RAP). The number per block is specified in DBDGEN, except if HIDAM with TF (and not TB) is pointing at root level, one anchor point per block is provided and it heads a LIFO chain of roots inserted in that block. If HIDAM or PHIDAM with TB or NT is pointing at the root level, there are no anchor points provided.
5. User database segments (prefix and data). In a bit map block, the bit map starts here and extends to the end of the block or to the VSAM control information.
6. 2-byte offset to next free space element (FSE) from start of block.
7. 2-byte length of free space, including 8-byte FSE.
8. 4-byte identification of the task that freed this space.
9. 7 bytes of VSAM control data; omitted for OSAM.

This format applies at the conclusion of initial load. The subsequent deletion of segments can result in free space elements that alternate with user database segments.

## VSAM LRECL for a primary index

The format of a VSAM LCRECL for a primary index depends on whether it is currently on a storage device, in the buffer pool, or being returned by the buffer handler.

### Format on a storage device or in the buffer pool

The following figure shows the format on a storage device or in the buffer pool.



Figure 51. LRECL format on storage device and in buffer pool

**Note:**

1. Four-byte RBA pointer to VSAM database root segment whose key value is the same as the value in the next field of this segment.

### Format as returned by the buffer handler

The following figure shows the VSAM LRECL format as returned by buffer handler (1).



Figure 52. VSAM LRECL format as returned by buffer handler

#### Notes:

1. Same as buffer pool format, except for pointer and segment code in front.
2. Four-byte pointer with value of zero.
3. The segment code value is 01.
4. Four-byte RBA pointer to VSAM database root segment whose key value is the same as the value in the next field of this segment.

### VSAM block format on a storage device or in the buffer pool

The following figure shows the VSAM block format on device and in buffer pool.



Figure 53. VSAM block format on device and in buffer pool

## Secondary index or PSINDEX database (VSAM only)

The LRECL Format on Device and in Buffer Pool are described.

### LRECL Format on Device and in Buffer Pool

One segment per LRECL. The following figure shows the LRECL Format on Device and in Buffer Pool.

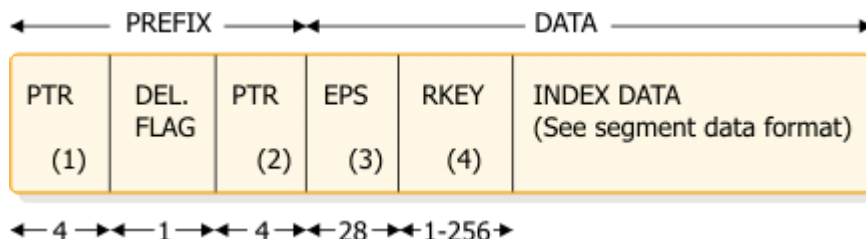


Figure 54. LRECL format on device and in buffer pool

#### Notes:

1. Nonunique keys: This points to ESDS LRECL with the same key value. Unique keys or PSINDEX: This field is omitted.

2. Direct pointer to index target segment. Omit this field if symbolic pointing is used or if this is a HALDB PSINDEX.
- 3 The EPS is present only if this is a HALDB PSINDEX. The 4-byte pointer to the target segment is included in the EPS.
- 4 RKEY means root key. The RKEY field is present only if this is a HALDB PSINDEX. This is the key value for the root of the target segment and its length can be from 1 to 256 bytes.

#### LRECL as Returned by Buffer Handler

The following figure shows LRECL as returned by buffer handler.

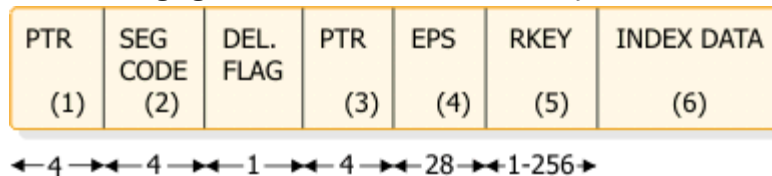


Figure 55. LRECL as returned by buffer handler

#### Notes:

1. Four-byte pointer contains zeros.
2. Code value is 01.
3. Direct pointer to index target segment. Omit this field if symbolic pointing is used or if this is a HALDB PSINDEX.
- 4 The EPS is present only if this is a HALDB PSINDEX. The 4-byte pointer to the target segment is included in the EPS.
- 5 The RKEY field is present only if this is a HALDB PSINDEX. This is the key value for the root of the target segment and its length can be from 1 to 256 bytes.
- 6 Sequential segment data format.

#### Block Format on Device and in Buffer Pool

The following figure shows the block format on device and in buffer pool.



Figure 56. VSAM block format on device and in buffer pool

#### Segment Data Format

The following figure shows the segment data format.

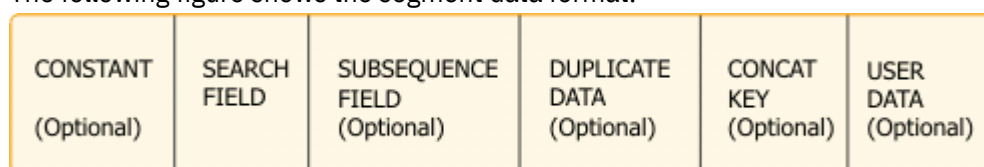


Figure 57. Segment data format

# Variable-length segments

The HISAM, HDAM, HIDAM, PHDAM, and PHIDAM Segment Format are described.

## HISAM, HDAM, HIDAM, PHDAM, and PHIDAM Segment Format

The following figure shows the HISAM, HDAM, HIDAM, PHDAM, and PHIDAM Segment Format.

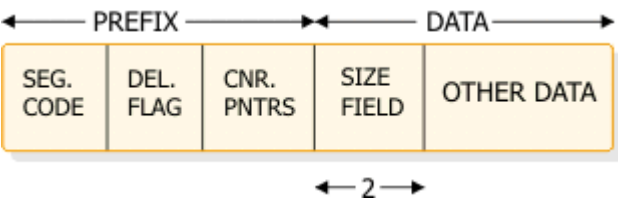


Figure 58. HISAM, HDAM, HIDAM, PHDAM, and PHIDAM segment format

**Note:** Variable-length segment must have a 2-byte length field at the front of the DATA portion.

## HDAM, HIDAM, PHDAM, and PHIDAM

When prefix and data are separated. The following figure shows HDAM, HIDAM, PHDAM, and PHIDAM.

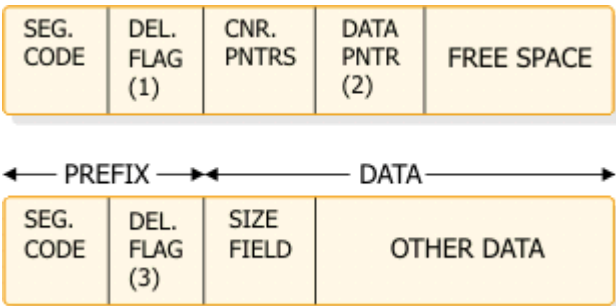


Figure 59. HDAM, HIDAM, PHDAM, and PHIDAM

### Notes:

1. DEL FLAG containing X'08' indicates that the data has been separated from the prefix.
2. DATA PNTR is a direct pointer to the segment containing the "other data".
3. The flag value is X'FF'.



---

## Chapter 6. CQS - Common Queue Server service aids

Use trace records, log records, and utilities to analyze problems in Common Queue Server (CQS).

### About this task

## Diagnosing a CQS related problem

---

CQS produces SDUMPs for internal errors. The CQS dumps are in the SYS1.DUMP data sets. CQS can also produce LOGREC data set entries for errors.

### CQS related problems

For a CQS environment, related problems might include:

- IMS WAIT problems
- CQS WAIT/HANG problems
- CQS checkpoint problems
- CQS restart problems
- CQS structure rebuild problems

Implement normal operating procedures to preserve the following documentation close to the time the error occurred:

- Additional manual dump intervention
- z/OS log stream (for problems related to IMS shared queues)
- Most recent SRDS (structure recovery data set) for each dumped structure

For a CQS WAIT/HANG problem, obtain dumps and syslogs of CQS address spaces in the sysplex such as:

- One dump and syslog from the master CQS
- One dump and syslog from the non-master CQS
- One dump and syslog from the error CQS
- One dump and syslog from the normal CQS

A sysplex contains only one master CQS and, most likely, one error CQS in a sysplex. Thus, the maximum number of CQS dumps and syslogs taken is three. If the sysplex contains fewer than three CQS address spaces, then dumps and syslogs of all CQS address spaces are needed.

Before obtaining the syslog, issue the following z/OS DISPLAY commands to write the sysplex information to the syslog:

```
D CF,CFNAME=cfname
D XCF,CF
D XCF,STRUCTURE,STRNAME=strname
```

For a CQS loop problem, obtain two dumps. To obtain the two dumps:

1. Obtain a z/OS SVC dump of the CQS and its associated IMS control region address space by issuing the following command:

```
DUMP COMM=(dump title)
R id,JOBNAME=(j1,j2),
SDATA=(CSA,PSA,RGN,SQA,SUM,TRT),END
```

In this command, *j1* is the CQS job name, and *j2* is the IMS control region job name.

2. Save the IMS log data sets that are created during the error period.

3. Save the current z/OS log data sets that are created. You can copy the current z/OS log data sets for the CQS log stream by using the IEBGENER utility. No z/OS log data sets are logged (unlike the IMS logger, which does have log archive capability through SLDS).

If an isolated event type within CQS encounters an error, IBM Software Support might request additional trace level settings for the various trace types. If a structure rebuild or structure checkpoint related problem occurs, you will also need to dump the CQS address spaces for any CQS associated with the given structure, and save the associated SRDS (structure recovery data set) for the CQS structure checkpoints and CQS system checkpoints.

## CQS additional manual intervention for dump creation

Additional considerations for dump creation are presented.

### Additional dump considerations

CQS environment additional dump considerations include:

- Structure dumps
- CQS regions and other CQS clients with their related CQS regions
- CQS regions and other CQS clients with their related CQS regions from other IMSplex members
- z/OS Logger

## CQS structure dump contents

CQS structure dumps should include the primary structures, the overflow structures, and the associated lock entries.

### CQS structure dump example

The following example shows the STRLIST for a dump:

```
DUMP COMM=(MSGQ STRUCTURE DUMP)
R nn,
  STRLIST=(STRNAME=imsmsgq01, LOCKE, (LISTNUM=ALL, ADJ=CAPTURE, EDATA=UNSER),
  STRNAME=imsmsgq01oflw, LOCKE, (LISTNUM=ALL, ADJ=CAPTURE, EDATA=UNSER)), END
```

Where *imsmsgq01* is the main structure name, and *imsmsgq01oflw* is the overflow structure name.

When an IMS structure dump is necessary, the z/OS Logger function might be involved with the problem. The following example dumps the z/OS Logger address spaces, the logger structure, and the IMS CF structure.

```
DUMP COMM=(CQS/LOGR STRUCTURE DUMP)
R xx, STRLIST=(STRNAME=imsmsgq01, LOCKE,
  (LISTNUM=ALL, ADJ=CAPTURE, EDATA=UNSER), CONT
R xx, STRNAME=imsmsgq01oflw, LOCKE,
  (LISTNUM=ALL, ADJ=CAPTURE, EDATA=UNSER), CONT
R xx, STRNAME=mvslogqmsg01, LOCKE, ACC=NOLIM,
  (LISTNUM=ALL, EDATA=UNSER, ADJ=CAPTURE)), CONT
R xx, JOBNAM=(IXGLOGR), DSPNAME=('IXGLOGR'.SYSLOGR0), CONT
R xx, SDATA=(COUPLE, ALLNUC, LPA, LSQA, PSA, RGN, SQA, TRT, CSA, GRSQ, XESDATA), END
```

Where:

#### ***imsmsgq01***

The main structure name.

#### ***imsmsgq01oflw***

The overflow structure name.

#### ***mvslogqmsg01***

The associated logger structure.



## CQS - IEADMCxx example with structures

Create three SYS1.PARMLIB members named IEADMCIA, IEADMCIB, and IEADMCIC by issuing the following command:

```
JOBNAME=(j1,j2,j3,j4,j5),SDATA=(CSA,PSA,RGN,SQA,SUM,TRT,GRSQ),  
REMOTE=(SYSLIST=(*('j1','j2','j3','j4','j5'),SDATA))
```

Where:

**j1**

IMS Control Region Jobname.

**j2**

IMS DLI Region Jobname.

**j3**

DBRC Region Jobname.

**j4**

IRLM Region Jobname.

**j5**

IMS CQS Region.

```
JOBNAME=(j6,j7,j8,j9,j10),SDATA=(CSA,PSA,RGN,SQA,SUM,TRT,XESDATA),  
REMOTE=(SYSLIST=(*('j6','j7','j8','j9','j10'),SDATA))
```

Where:

**j6**

APPC Region.

**j7**

APPC Scheduler.

**j8**

VTAM.

**j9**

Other CQS Client Region.

**j10**

Other CQS Region.

```
JOBNAME=(IXGLOGR),DSPNAME=('IXGLOGR'.SYSLOGR0),  
SDATA=(COUPLE,ALLNUC,LPA,PSA,RGN,SQA,TRT,CSA,GRSQ,XESDATA),  
STRLIST=(STRNAME=imsmsgq01,LOCKE,(LISTNUM=ALL,ADJ=CAPTURE,EDATA=UNSER),  
STRNAME=imsmsgq01oflw,LOCKE,(LISTNUM=ALL,ADJ=CAPTURE,EDATA=UNSER),  
STRNAME=mvslogqmsg01,LOCKE,ACC=NOLIM,(LISTNUM=ALL,EDATA=UNSER,ADJ=CAPTURE))
```

Where:

**imsmsgq01**

The main structure name.

**imsmsgq01oflw**

The overflow structure name.

**mvslogqmsg01**

The associated logger structure.

## CQS - IEADMCxx dump activation

To create a dump from the IEADMCIA, IEADDMCIB and IEADMCIC parmlib members, issue the following z/OS command: DUMP TITLE=(DUMP OF IMSplex and Partners),PARMLIB=(IA,IB,IC).

### About this task

Three dump data sets are created on the z/OS image from which the command is issued. Two dump data sets are created on each image in the sysplex that matches the REMOTE specifications for the JOBNAMES.

**Recommendation:** Provide the z/OS logger address space from the system experiencing problems to z/OS logger support.

## CQS - z/OS log stream example

The merged z/OS log stream can be used to examine CQS log records. IEBGENER can be used along with the default log stream subsystem exit routine, IXGSEXIT, to copy the log records at time of failure for later analysis.

### About this task

```
//CQSCPYLG JOB USERID,USERID,MSGLEVEL=1,CLASS=K
//*****
//* This job copies a CQS log stream to a dataset (max 32K / record) *
//*
//* - Replace the DSN on the SYSUT1 card with your CQS logstream *
//* name. *
//*
//* - Replace the DSN on the SYSUT2 card with your desired output *
//* dataset name. You may also need to adjust the space *
//* allocations, depending on the size of your logstream. *
//*****
//STEP1 EXEC PGM=IEBGENER,REGION=1024K
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT1 DD DSN=SYSLOG.MSGQ01.LOG,
// SUBSYS=(LOGR,IXGSEXIT),
// DCB=(BLKSIZE=32760)
//SYSUT2 DD DSN=CQS.LOG.COPY,
// DISP=(NEW,KEEP,DELETE),
// VOL=SER=USER05,
// SPACE=(CYL,(2,10)),
// UNIT=SYSDA
```

## CQS structure recovery data set

Save the most recent CQS SRDS (structure recovery data set) for each structure that is dumped. Use the IDCAMS REPRO command if the LRECL is acceptable (less than 32 761).

### About this task

## CQS checkpoint problems

There are two types of CQS checkpoints: system checkpoint and structure checkpoints.

### CQS checkpoint messages

Most problems are of structure checkpoint type because it is a sysplex-wide operation with shared resources (SRDS data set, structures on the CF), and it needs cooperation through z/OS IXLUSYNC between all CQs within the sysplex. Sometimes, another CQS process (initialization, termination, rebuild, overflow threshold, or overflow scan) can interfere with the checkpoint process and cause it to fail.

## CQS System Checkpoint Messages

1. CQS0030I for a successful CQS system checkpoint.
2. CQS0035E for a failed CQS system checkpoint. If the CQS system checkpoint failed with CQS0035E, refer to "CQS messages" in *IMS Version 15.5 Messages and Codes, Volume 2: Non-DFS Messages* for the details of the failure and the recommended system programmer action.

## CQS Structure Checkpoint Messages

The syslog of a successful CQS structure checkpoint will contain five CQS messages in the following order:

```
CQS0220I CQS cqsname START STRUCTURE CHECKPOINT FOR STRUCTURE strname
CQS0200I STRUCTURE strname QUIESCED FOR STRUCTURE CHECKPOINT
CQS0201I STRUCTURE strname RESUME AFTER STRUCTURE CHECKPOINT
CQS0030I SYSTEM CHECKPOINT COMPLETE, STRUCTURE strname LOGTOKEN logtoken
CQS0221I CQS COMPLETE STRUCTURE CHECKPOINT FOR STRUCTURE strname
```

If the CQS structure checkpoint failed with a CQS0222E message, refer to the "CQS messages" section in *IMS Version 15.5 Messages and Codes, Volume 2: Non-DFS Messages* for the details of the failure and the recommended system programmer action.

If message CQS0222E is not displayed and all five of the normal structure checkpoint messages did not appear on the console, CQS probably has encountered a serious WAIT/HANG problem.

The CQS log records, the structure dump of the related structure, and the SRDSs (structure recovery data sets) are helpful in diagnosing the problem. After collecting all the documents, you can stop the CQS in error and restart it to resolve the WAIT/HANG problem.

## CQS structure rebuild problems

The most common structure rebuild problem is a rebuild failure. Some environmental situations can occur that cause rebuild to fail.

### Rebuild failures

Other types of rebuild problems are much more rare, such as rebuild hanging, rebuild not being initiated when required, work hanging after a successful rebuild, rebuild losing data objects, and rebuild duplicating data objects. Follow these general steps to address any rebuild failure you encounter:

#### • Collect SYSLOGs

Collect the syslog for each LPAR that is running a CQS that is sharing queues. Evaluate each syslog for the following information:

- How the rebuild was initiated (operator command, structure failure, CF failure, link failure).
- How the rebuild was stopped (operator command or CQS).
- Rebuild master (CQS0240I message).
- Rebuild type (COPY or RECOVERY in CQS0240I message).
- Structure quiesced or resumed messages:
  - CQS0200I STRUCTURE *strname* QUIESCED FOR *reason*
  - CQS0201I STRUCTURE *strname* RESUMED AFTER *reason*
- Structure status change messages (CQS0202I).
- Structure rebuild messages:
  - CQS0240I CQS *cqsname* STARTED STRUCTURE *copy/recovery* FOR STRUCTURE *strname*
  - CQS0241I CQS *cqsname* COMPLETED STRUCTURE *copy/recovery* FOR STRUCTURE *strname*
  - CQS0242E CQS FAILED STRUCTURE *copy/recovery/rebuild* FOR STRUCTURE *strname*
  - CQS0243E CQS *cqsname* UNABLE TO PARTICIPATE IN REBUILD FOR STRUCTURE *strname*

- CQS0244E STRUCTURE RECOVERY REQUIRED AFTER RECOVERY FAILURE FOR STRUCTURE *strname*
- CQS0245E STRUCTURE *strname* REBUILD ERROR

- **Consult the CQS Restart and Rebuild Error Reason Codes table**

- **Check rebuild status**

Check the rebuild status by issuing the following command on every LPAR where a CQS participating in the rebuild resides:

```
D XCF,STRUCTURE,STRNAME=strname
```

If the output indicates that rebuild is waiting for a particular event, a CQS might not be responding to a rebuild event because it is hung or in a loop, which hangs the rebuild. Consider dumping the CQS address space and canceling the CQS that is not responding to the rebuild event, to see if that enables the rebuild to continue.

- **Analyze if structure still viable**

If a structure copy initiated by an operator failed, no action needs to be taken to restore access to the structure. The structure is still viable and you still have access. Analyze why the structure copy failed, to determine whether you need to take action to prevent a subsequent rebuild failure.

- **Restore link, if applicable**

If a structure rebuild was initiated because of a link failure and the structure rebuild failed, try to restore the link to restore access to the structure. The structure is still viable. Analyze why the structure rebuild failed, to determine whether you need to take action to prevent a subsequent rebuild failure.

- **Contact IBM**

If you are unable to resolve the problem, take the following actions:

- Copy the SYSLOG, including the D XCF,STRUCTURE,STRNAME=*strname* output from every LPAR.
- Dump all the CQS address spaces, including the rebuild master CQS address space. Message CQS0240I indicates the rebuild master name.
- Retain the CQS log records. The CQS log might contain important log records pertaining to data objects put on the structure, moved on the structure, or deleted from the structure. The CQS log might also contain important log records pertaining to rebuild, such as:
  - Rebuild begin log record (4301).
  - Rebuild end log record (4302).
  - Rebuild failed log record (4303).
  - Rebuild lost UOW list log record (4304).
  - Request log records (03xx, 07xx, 08xx, 0Bxx, 0Dxx).
- Retain the IMS log records.
- Create a structure dump if you suspect a rebuild hang. The structure dump might contain important information about structure locks.
- Call the IBM Software Support for help.

## CQS trace records

Analyze CQS trace records in a formatted dump to determine which function encountered an error, and whether a problem is environmental or internal.

### Trace tables

Trace record eye catchers in a formatted dump provide clues about which functions resulted in errors. You might be able to correct environmental problems immediately. Refer internal IBM problems to IBM with appropriate documentation, such as system console logs and dumps.

CQS trace records are written to one or more of the trace tables shown in the following table.

*Table 8. Trace tables that contain CQS trace records*

Table name	Number of tables	Table description
ERR	1	Errors
CQS	1	CQS activity, including errors
INTF	1	CQS interface events
OFLW	1 per structure (EMHQ, MSGQ)	Structure overflow events
SEVT	1 per structure (EMHQ, MSGQ)	Structure event activity
STR	1 per structure (EMHQ, MSGQ)	Client activity for this structure, including errors

Each CQS trace record is 32 bytes long, except records in the SEVT and OFLW tables. Those tables use an expanded 64-byte format. In a standard 32-byte trace entry, the first byte is the trace code and the second byte is the trace subcode. Many trace records contain a structure ID that identifies which structure the trace record applies to: the MSGQ primary structure, the MSGQ overflow structure, the EMHQ primary structure, or the EMHQ overflow structure. Trace records that apply to a client request contain a client ID that identifies the client that issued the request. The last 8 bytes are the STCK time stamp of when the trace record was written. The mapping of the rest of the bytes in the trace record is unique to the trace code and subcode.

The expanded 64-byte format used by trace records in the SEVT and OFLW tables contains 16 words of trace data in the following format:

**Word 0**

Byte 0 contains the trace code of the event.

Byte 1 contains the trace subcode of the event.

Byte 2 contains data dependent on the type of event, typically the structure ID.

Byte 3 contains data dependent on the type of event, typically the client ID.

**Word 1 - word 12**

These 48 bytes contain trace data generated by the event. The exact content is dependent on the type of event.

**Word 13**

This word contains the ID of the event control block (ECB) of the task that wrote the trace record in the table.

**Word 14 - word 15**

These 8 bytes contain the time stamp of the event in STCK format.

CQS trace records are mapped by macros that use the naming convention CQSTRxxx, where xxx represents the function that is being traced. For example, CQSTRPUT maps trace records associated with the CQSPUT request. Trace record mapping is based upon the trace code and the trace subcode.

Find the CQS trace code in the following table to locate the CQS macro that maps the trace record. The following table shows the CQS trace codes, the macro that maps the trace code, and a description of the trace macro.

*Table 9. CQS trace codes and mapping macros*

Trace code	Macro	Description
3	CQSTRCON	CQSCONN request
4	CQSTRDSC	CQSDISC request
5	CQSTRSY	CQSRSYNC request

Table 9. CQS trace codes and mapping macros (continued)

Trace code	Macro	Description
6	CQSTRINF	CQSINFRM request
7	CQSTRPUT	CQSPUT request
8	CQSTRRD	CQSREAD request
9	CQSTRBRW	CQSBRWSE request
0A	CQSTRUNL	CQSUNLCK request
0B	CQSTRMOV	CQSMOVE request
0C	CQSTRRCV	CQSRECVR request
0D	CQSTRDEL	CQSDDEL request
E	CQSTRQRY	CQSQUERY request
0F	CQSTRCHK	CQSCHKPT request
10	CQSTRSHT	CQSSHUT request
11	CQSTRUPD	CQSUPD request
30	CQSTRICQ	CQS initialization
31	CQSTRTCQ	CQS termination
32	CQSTRYCH	System checkpoint
40	CQSTRIST	Structure initialization
41	CQSTRSTS	Structure service
42	CQSTRTCH	Structure checkpoint
43	CQSTRRBL	Rebuild
44	CQSTROFL	Overflow
45	CQSTRSTE	Structure event
50	CQSTRLOG	Log services
51	CQSTRTBL	Table services
52	CQSTRDYA	Dynamic allocation services
53	CQSTRDSS	Data set services
54	CQSTRDSP	Data space services
55	CQSTRLRR	Log record router
56	CQSTRXCF	z/OS cross-system coupling facility interface
57	CQSTRCMD	Command
60	CQSTRSTT	Statistics
70	CQSTRINT	CQS client interface

Trace codes for CQS requests are defined in the CQSRQTYP macro. Trace codes for other CQS functions are defined in the CQSCODES macro. CQS trace records in a formatted dump might contain eye catchers

that provide clues about which function encountered an error, such as "overflow," "rblld," "str chkpt," and "duplex."

CQS request trace records sometimes contain a return code, reason code, and completion code from the request. CQS request return codes, reason codes, and completion codes are mapped by macros that use the naming convention CQSRRxxx, where xxx represents the function that is being traced. For example, the macro CQSRRPUT maps return codes, reason codes, and completion codes that are associated with the CQSPUT request. The following table shows the macros that define the return codes, reason codes, and completion codes for CQS requests.

*Table 10. CQS mapping macros and request trace records*

<b>Macro</b>	<b>CQS request macro for return codes, reason codes, and completion codes</b>
CQSRRCON	CQSCONN
CQSRRDSC	CQSDISC
CQSRRRSY	CQSRSYNC
CQSRRINF	CQSINFRM
CQSRRPUT	CQSPUT
CQSRRRD	CQSREAD
CQSRRBRW	CQSBRWSE
CQSRRUNL	CQSUNLCK
CQSRRMOV	CQSMOVE
CQSRRRCV	CQSRECVR
CQSRRDEL	CQSDDEL
CQSRRQRY	CQSQUERY
CQSRRCHK	CQSCHKPT
CQSRRSHT	CQSSHUT
CQSRRUPD	CQSUPD

CQS trace records in formatted dumps contain eye catchers that identify the trace code and the trace subcode.

The following example shows a CQS trace record with eye catchers:

```
INFRM: INF DONE FOR Q      06090101 05E3F3F2 F7F0D3C1 40404040
                           40404040 05541160 AF975E81 59426906
```

The trace code is in the first byte (X'06'), which the CQSRQTYP macro documents as the CQSINFRM request. The eye catcher is INFRM. The CQSTRINF macro maps the trace records for trace code X'06'.

The trace subcode is in the second byte (X'09'), which the CQSTRINF macro documents as "inform done for queue." The eye catcher is INF DONE FOR Q.

The CQSTRINF macro documents byte 3 for trace code X'06' as containing the structure ID (X'01'). Structure ID X'01' indicates the primary MSGQ structure.

The CQSTRINF macro documents byte 4 for trace subcode X'06' as containing the client ID (X'01'). Client ID X'01' represents the client that issued the CQSINFRM request. The CQSTRINF macro documents words 2, 3, 4, and 5 for trace subcode X'06' as containing the name of the queue for which the inform was done. This queue name is for queue type 05 (the IMS transaction queue). The queue name is T3270LA (X'E3F3F2F7F0D3C1').

The CQSTRINF macro documents word 6 for trace subcode X'06' as the ECB of the task that wrote this trace record (X'05541160').

The CQSTRINF macro documents words 7 and 8 as the STCK time of when the trace record was written.

## CQS log records

You can use Common Queue Server (CQS) log records to diagnose problems related to the CQS address space and generate various reports, such as statistics about the number of requests.

### About this task

CQS writes records to the z/OS log stream that contains all CQS log records from all CQs that are connected to a structure pair. You can use the log records to:

- Diagnose problems related to the CQS address space.

For CQS internal errors, the IBM support representative will direct you to print the appropriate log records.

You can sometimes use information in the log records to set up a keyword string to search APAR descriptions and compare them to your own problem.

- Generate various reports related to the CQS address space, such as statistics about the number of requests.

By knowing the content and format of the log records, you can set up a DFSERA10 job to format and print the specific log records you want.

Each CQS log record contains a log record prefix, followed by data that is unique to the record. Macro CQSLGPFX maps the log record prefix.

You can view the CQS log record formats by assembling mapping macro CQSLGREG with TYPE=ALL.

Individual log record DSECTs can be obtained by assembling the ILOGREG macro while including a format statement for the desired log record.

Table 11. CQS log records

Type	Subtype	Mapping macro	Conditions for writing the log record
X'03'	X'01'	CQSLGCON	CQSCONN request: The client connect to a structure completed.
X'04'	X'01'	CQSLGDSC	CQSDISC request: The client disconnect from a structure completed.
X'07'	X'01'	CQSLGPUT	CQSPUT OBJECT request completed.
	X'02'		CQSPUT COMMIT request completed.
	X'03'		CQSPUT START request completed.
	X'04'		CQSPUT FORGET request completed.
	X'05'		CQSPUT ABORT request completed.
	X'06'		CQSPUT request failed.
	X'07'		CQSPUT system checkpoint record was written.
X'08'	X'08'	CQSLGRD	CQSPUT FORGET request completed. This is a batched log record.
	X'01'		CQSREAD request completed.
	X'02'		CQSREAD request failed.
	X'03'		CQSREAD system checkpoint record was written.



Table 11. CQS log records (continued)

Type	Subtype	Mapping macro	Conditions for writing the log record
		CQSLGCHD	This system checkpoint header record is not a complete log record, but it is used in CQSLGPUT and CQSLGRD system checkpoint log records.
X'0B'	X'01'	CQSLGMOV	CQSMOVE or CQSUNLCK request completed.
	X'02'		CQSMOVE or CQSUNLCK request failed.
	X'03'		CQSMOVE or CQSUNLCK request moved an object between the primary and overflow structure.
X'0D'	X'01'	CQSLGDEL	CQSDDEL request: Delete-type 1 (delete by token) completed.
	X'02'		CQSDDEL request: Delete-type 2 (delete by queue name) completed.
	X'03'		CQSDDEL request: Delete-type 3 (delete by queue name and UOW) completed.
	X'04'		CQSDDEL request: Delete-type 1 (delete by token) completed. This is a batched log record.
		CQSLGBHD	This batched log record header record is not a complete log record, but is used in CQSLGPUT and CQSLGDEL batched log records.
X'10'	X'01'	CQSLGSHT	CQSSHUT request completed.
X'32'	X'01'	CQSLGYCH	System checkpoint started.
	X'02'		System checkpoint ended.
	X'03'		System checkpoint failed.
X'40'	X'01'	CQSLGIST	Beginning of log stream.
X'42'	X'01'	CQSLGTCH	Structure checkpoint started.
	X'02'		Structure checkpoint ended.
	X'03'		Structure checkpoint failed.
X'43'	X'01'	CQSLGRBL	Structure rebuild started. Statistics about the old structure, the rebuild structure, and rebuild failure are mapped by CQSSSTT6.
	X'02'		Structure rebuild ended. Statistics about the old structure, the rebuild structure, and rebuild failure are mapped by CQSSSTT6.
	X'03'		Structure rebuild failed. Statistics about the old structure, the rebuild structure, and rebuild failure are mapped by CQSSSTT6.
	X'04'		Structure rebuild resulted in a lost UOW list. This record lists the lost UOWs.
X'44'	X'01'	CQSLGOFL	Overflow threshold began.
	X'02'		Overflow threshold ended.
	X'03'		Overflow threshold failed.

Table 11. CQS log records (continued)

Type	Subtype	Mapping macro	Conditions for writing the log record
	X'04'		Overflow mode ended.
	X'05'		Overflow status change.
	X'06'		Qnames were moved to overflow.
	X'07'		Qnames were removed from overflow.
	X'08'		CQSOVERFLOWQNMR, a control list entry containing the list of queue names deleted from overflow, was deleted.
	X'09'		Overflow Scan Begin.
	X'0A'		Overflow Scan End.
	X'0B'		Private Queue Scan Begin.
	X'0C'		Structure to be deleted.
X'60'	X'01'	CQSLGSTT	Structure statistics were written at the end of system checkpoint. Individual statistics areas are mapped by CQSSSTT1, CQSSSTT2, CQSSSTT3, CQSSSTT4, and CQSSSTT5.
	X'00'		Internal BPE service statistics were written at the end of system checkpoint. BPE statistics areas are mapped by macro BPESSTA.

## Printing CQS log records

To print the CQS log records from the z/OS system log, use the IMS File Select and Formatting Print utility (DFSERA10) with exit routine CQSERA30.

### Example JCL to print CQS log records

The following example shows the required JCL to print the log records from a z/OS system log. This JCL causes the z/OS logger to invoke the default log stream subsystem exit routine, IXGSEXIT, to copy the log records. The exit routine returns a maximum of 32 760 bytes of data for each log record even though CQS supports larger log records. You can specify the name of a different exit routine, if necessary.

Use the following JCL to print the CQS log records:

```
//CQSERA10 JOB MSGLEVEL=1,MSGCLASS=A,CLASS=K
//STEP1 EXEC PGM=DFSERA10
//STEPLIB DD DISP=SHR,DSN=IMS.SDFSRESL
//SYSPRINT DD SYSOUT=A
//TRPUNCH DD SYSOUT=A,DCB=BLKSIZE=80
//SYSUT1 DD DSN=SYSLOG.MSGQ01.LOG,
// SUBSYS=(LOGR,IXGSEXIT),
// DCB=(BLKSIZE=32760)
//SYSIN DD *
CONTROL CNTL H=EOF
OPTION PRINT EXITR=CQSERA30
END
//
```

#### DD statements

##### STEPLIB

DSN= points to IMS.SDFSRESL, which contains the IMS File Select and Formatting Print utility (DFSERA10).

##### SYSUT1

DSN= points to the CQS log stream name that was specified in the LOGNAME= parameter in the CQSSGxxx PROCLIB member.

## Control Statements

### H=

Specifies the number of log records to print. H=EOF prints all log records.

### EXITR=CQSERA30

The CQS log record routine that is called to format each log record. This routine prints the record type and time-stamp information for each record, and dumps the contents of the record (up to a maximum of 32 760 bytes (X'7FF8')).

## Limiting Log Data to a Specified Time Range

You can limit the log records you print to those in a particular interval of time by using the FROM and TO parameters on the SUBSYS statement. For example, the following DD card:

```
//SYSUT1 DD DSN=SYSLOG.MSGQ01.LOG,
//          SUBSYS=(LOGR,IXGSEXIT,
//          'FROM=(2001/042,11:00:00),TO=(2001/042,12:00:00)',
//          DCB=(BLKSIZE=32760)
```

would pass log records only from 11:00 to 12:00 on day 42 of the year 2001 to the DFSERA10 program. Dates and times specified in this manner are in GMT, and the seconds field of the time values is optional. If you want to use local dates and times, add the LOCAL keyword to the statement:

```
//SYSUT1 DD DSN=SYSLOG.MSGQ01.LOG,
//          SUBSYS=(LOGR,IXGSEXIT,
//          'FROM=(2001/042,11:00:00),TO=(2001/042,12:00:00),LOCAL',
//          DCB=(BLKSIZE=32760)
```

## Copying CQS log records for diagnostics

IBM Software Support sometimes requires a copy of a range of CQS log records for problem determination. You can use the IEBGENER utility program to copy some or all of the CQS log for a structure to a BSAM data set for sending to IBM Software Support.

### IEBGENER utility program example

The copy made by the IEBGENER utility is a binary image of the log records. The following JCL is a job that copies CQS log records between 15:10 and 15:30 local time on day 89 of 2001 to a data set named CQS.LOG.COPY:

```
//CQSCPYLG JOB MSGLEVEL=1,CLASS=K
//*****
//* THIS JOB COPIES A CQS LOG STREAM TO A DATASET (MAX 32K / RECORD) *
//*****
//STEP1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT1 DD DSN=SYSLOG.MSGQ01.LOG,
//          SUBSYS=(LOGR,IXGSEXIT,
//          'FROM=(2001/089,15:10),TO=(2001/089,15:30),LOCAL',
//          DCB=(BLKSIZE=32760)
//SYSUT2 DD DSN=CQS.LOG.COPY,
//          DISP=(NEW,KEEP,DELETE),
//          VOL=SER=EDSDMP,
//          SPACE=(CYL,(10,10)),
//          UNIT=SYSDA
```

If you copy CQS log records using the IEBGENER utility, the following guidelines apply:

- The copied records cannot be used by CQS in any way (such as restart or recovery). They are for diagnostic purposes only.
- CQS log records that are greater than 32 KB in length are truncated. The SUBSYS exit supports a maximum of a 32 KB record size.



# Chapter 7. CSL - Common Service Layer service aids

Common Service Layer (CSL) and Resource Manager (RM) trace records can help you analyze problems in CSL.

## Related tasks

[“Collecting data about CSL-related problems” on page 23](#)

The Common Service Layer address spaces, Open Database Manager (ODBM), Operations Manager, Structured Call Interface, and Resource Manager, produce SDUMPs for internal errors. The CSL dumps are in the SYS1.DUMP data sets.

## CSL trace records

You can analyze CSL address space trace records (for example OM, RM, or SCI) in a formatted dump to determine whether a problem is environmental or internal.

### Trace records

Trace record eye catchers in a formatted dump provide clues about which function resulted in an error. You might be able to take action to correct environmental problems immediately. Refer internal problems to IBM with appropriate documentation, such as system console logs and dumps.

ODBM trace records are written to one or more of the trace tables shown in the following table.

*Table 12. Trace tables for ODBM trace records*

Table name	Number of tables	Table description
CSL	1	CSL activity, including errors
ERR	1	Errors
ODBM	1	ODBM activity
PLEX	1 per IMSplex	IMSplex activity, including errors

OM trace records are written to one or more of the trace tables shown in the following table.

*Table 13. Trace tables for OM trace records*

Table name	Number of tables	Table description
CSL	1	CSL activity, including errors
ERR	1	Errors
OM	1	OM activity
PLEX	1 per IMSplex	IMSplex activity, including errors

RM trace records are written to one or more of the trace tables shown in the following table.

*Table 14. Trace tables for RM trace records*

Table name	Number of tables	Table description
CSL	1	CSL activity, including errors
ERR	1	Errors
PLEX	1 per IMSplex	IMSplex activity, including errors

Table 14. Trace tables for RM trace records (continued)

Table name	Number of tables	Table description
REPO	1	IMSRSC repository activity, including errors
RM	1	RM activity, including errors

SCI trace records are written to one or more of the trace tables shown in the following table.

Table 15. Trace tables for SCI trace records

Table name	Number of tables	Table description
CSL	1	CSL activity, including errors
ERPL	1	Parmlist errors
ERR	1	Errors
INTF	1	SCI interface activity
INTP	1	Interface parmlist
PLEX	1 per IMSplex	IMSplex activity, including errors
SCI	1	SCI activity, including errors

Each CSL trace record contains 32 bytes (the ERPL and INTP tables in the SCI address space contain records that are 256 bytes). The first byte is the trace code, which indicates the function that wrote the trace record. Examples of trace code functions include address space initialization, address space termination, the CSLMCMC request, the CSLRMUPD request, and the CSLSCRQS request. The second byte is the trace subcode, which indicates the category of the trace record. Examples of trace subcode categories include begin request, end request, CQS error, and SCI error. Most trace records include a 2-byte module identifier of the module that wrote the trace record. The last 8 bytes are the STCK time stamp of when the trace record was written. Trace record mapping of the rest of the fields is unique to the trace subcode.

CSL address space trace codes and other common codes used in trace records are mapped by a macro following the naming convention of CSLxCODE macro, where x represents the CSL address space as shown in the following table.

Table 16. CSL address space trace code mapping macros

Codes macro name	Description
CSLDCODE	ODBM codes
CSLOCODE	OM codes
CSLRCODE	RM codes
CSLSCODE	SCI codes
CSLZCODE	CSL codes common to multiple CSL address spaces

CSL address space trace records are mapped by a macro following the naming convention of CSLxTRC macro, where x represents the CSL address space as shown in the following table.

Table 17. CSL address space trace record mapping macros

Trace macro name	Description
CSLDTRC	ODBM trace records
CSLOTTRC	OM trace records

Table 17. CSL address space trace record mapping macros (continued)

Trace macro name	Description
CSLRTRC	RM trace records
CSLSTRC	SCI trace records
CSLZTRC	CSL trace records common to multiple CSL address spaces

Trace record mapping is based upon the trace subcode, which identifies the category of trace record. One particular trace subcode can apply to many trace codes. Each trace record mapping also includes a pictorial representation in a comment block. Use the trace subcode to locate the trace record mapping in the CSLxTRC macro. Some trace codes are unique to a particular CSL address space, others are common to more than one CSL address space.

The CSLxCODE macro includes 2-byte module identifier codes that are used in trace records and messages when it is necessary to identify a CSL module. The module identifier represents the module that wrote the trace record. Module identifier codes are defined as follows:

**CSL address spaces**

X'0001'-X'6FFF'

**CSLZ modules**

X'7000'-X'77FF'

**BPE modules**

X'7800'-X'7FFF'

**Not used**

X'8000'

**Reserved for BPE tracing**

X'8001'-X'FFFF'

CSL request trace records sometimes contain a return code, reason code, and completion code from the request. CSL request return codes, reason codes, and completion codes are mapped by macros following the naming convention CSLxRR, where x represents the CSL address space as shown in the following table. These macros are in the IMS.SDFSMAC data set.

Table 18. CSL request return, reason, and completion codes mapping macros

Macro	Description
CSLDRR	ODBM return codes, reason codes, and completion codes
CSLORR	OM return codes, reason codes, and completion codes
CSLRRR	RM return codes, reason codes, and completion codes
CSLSRR	SCI return codes, reason codes, and completion codes

CSL trace records in formatted dumps contain eye catchers that identify the trace code, the trace subcode, and the module that wrote the trace record. The EPL and INTF trace tables in SCI contain parameter lists, tables, and other data areas that cross the SCI interface. These data areas are formatted in the SCI dump.

**Related concepts**

[Overview of the IMSRSC repository \(System Definition\)](#)

**Related reference**

[BPE configuration parameter member of the IMS PROCLIB data set \(System Definition\)](#)

## RM trace record example

---

An example of an RM trace record with eye catchers is described.

### RM trace record with eye catchers

```
CEVTX:*CQS SERVICE ERR  RCQE 60110000 03000042 0000000C 00000304 00000008 0BC60C20 B6B1AF09 07C68F08
```

The trace code is in the first byte (X'60'), which the CSLRCODE macro documents as CQS Event Exit. The eye catcher for this is CEVTX. The trace subcode is in the second byte (X'11'), which the CSLRTRC macro documents as a miscellaneous CQS service error. The eye catcher for this is \*CQS SERVICE ERR. The asterisk at the beginning of the eye catcher indicates an error.

The CSLRTRC macro documents byte 5 for trace subcode X'11' as containing the service request X'03', which is the CQSCONN request.

The CSLRTRC macro documents byte 8 for trace subcode X'11' as containing the module ID X'0042', which the CSLRCODE macro defines as module CSLRCQE0. The module name is included in the eye catcher on RCQE. The CSLRTRC macro documents word 3 for trace subcode X'11' as containing the CQSCONN return code X'0000000C'. The CQSRRCON macro defines the CQSCONN return codes, reason codes, and completion codes. The CQSRRCON macro defines return code X'0000000C' as a list error.

The CSLRTRC macro documents word 4 for trace subcode X'11' as containing the CQSCONN reason code X'00000304'. The CQSRRCON macro defines reason code X'00000304' as no requests successful.

The CSLRTRC macro documents word 5 for trace subcode X'11' as containing the CQSCONN completion code X'00000008'. CQSRRCON macro defines completion code X'00000008' as no resource structure is defined. RM was unable to connect to the resource structure because it is not defined. This is probably an environmental problem where the resource structure was not correctly defined to CQS.

The CSLRTRC macro documents word 6 for trace subcode X'11' as containing the ECB address (X'0BC60C20').



---

# Chapter 8. DB - Database service aids

You can use service aids, such as the IMS test program (DFSDDLTO), as well as various diagnostic techniques, to analyze problems with IMS databases.

## About this task

### Job control block trace

---

The job control block (JCB) trace is one of most useful diagnosis tools for any application problem that might occur. The trace is an easy way to determine the last five calls that were issued, and what their return codes were.

#### Analyzing the JCB trace

Analyzing the JCB trace is a good way to identify application problems. For example, sometimes the application programmer forgets to handle a certain status code, even though it identifies an error situation. Seeing the call and its return code draws attention to this application error and makes it easier to resolve.

The JCB trace is always on (you do not need to do anything to turn it on), and is included in every IMS dump. The job control block portion of the dump is formatted under the heading, JCB. The JCB trace is a wrap-around area that consists of six 2-byte entries. The first entry begins at the offset of JCBTRACE in the JCB portion of the dump and is followed immediately by the remaining five entries. As the entries are inserted into the trace area, previous entries are shifted left.

In the first through fifth entries, the first byte identifies the DL/I call. The second byte in these entries contains the second character of the DL/I I/O status code (return code). The sixth entry contains information about the call that immediately preceded the call that was being processed when the abend occurred; this information can be useful in determining what occurred prior to the failure. The function of that prior call is identified in field JCBPREVF of the JCB, and the status code of the prior call is in field JCBPREVR.

If one of the 2-byte fields in the JCB trace contains X'0000', no call was made.

#### Example JCB trace

The JCB trace might contain the following six fields:

```
0000 0000 0205 0305 0140 0140
```

This trace indicates that only four calls were made, the most recent of which was a get-unique call (either GU or GHU), as indicated by the first-byte code of X'01'. The status code for the most recent call was X'40'.

#### Related concepts

[“Dump analysis introduction” on page 161](#)

In a pseudoabend supervisor call (SVC) dump that is generated by module DFSERA20, you can find the failing program specification table (PST) by searching the save areas for the caller of module DFSERA20. In the save area flow, DFSERA20 is called INTERA20 and register 1 contains the failing PST address.

### Sample JCB trace

A sample JCB trace is shown.

#### JCB dump example

The following table shows an example of a JCB dump.

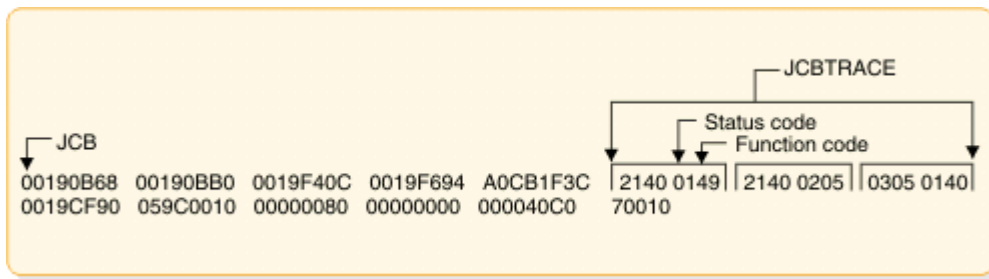


Figure 60. Example of a JCB dump

## JCB trace call function codes

DL/I user calls are listed, including codes and calls.

### DL/I user call encoded functions

The following table shows the DL/I user call encoded functions, which are contained in DFSDLA00, at label FUNCSTRT.

Table 19. DL/I user call encoded functions

Code	Call	Code	Call
00	GB	65	LOG
00	GBT	70	RELOAD
00	GHB	80	OPEN
00	GHBT	81	CLOSE
00	GHP	82	STOP
00	GL	83	CHANGE
00	GND	84	SNAP
00	GNX	85	CHECK POINT
00	GP	86	STATISTICS REQUEST
01	GHU	87	CMD
01	GU	88	GCMD
03	GHN	89	ROLB
03	GN	90	PURGE
04	GHNP	A0	UNLD
04	GNP	A1	GSCD
20	DLET or REPL	A2	MOVE
21	REPL	B0	SPND
22	DLET	F1	XSET
23	DLET or REPL	F2	XRUN
40	ISRT	F3	XFIN
41	ISRT	F4	XSCD
42	ASRT	F5	XOFF

Table 19. DL/I user call encoded functions (continued)

Code	Call	Code	Call
60	DEQ		

## DL/I test program - DFSDDLTO

The DL/I test program is an IMS application that issues calls to DL/I based on control statement information. For diagnostic purposes, this allows you a means of separating the application logic from DL/I logic to resolve problems.

Optionally, the DL/I test program compares the results of the calls with expected results provided in control statements. If the returned results do not match the expected results, the program can provide a SNAP of any combination of DL/I blocks, I/O buffer pool, subpools 0-127, and the entire region. The test program can also invoke the IMS SNAP call, by means of its control statements, during normal execution to provide diagnostic information on the DL/I calls that are executing correctly.

## COMPARE statement SNAPs

When a DL/I call does not produce the results you expect, you can use the COMPARE statement to compare the actual results of a call with the expected results. The normal output of this statement usually provides enough information to determine what is causing the problem.

When the output from a COMPARE statement does not provide enough information, you can use the SNAP option of the COMPARE statement to obtain additional diagnostic information. Specifically, the I/O buffer pool and the DL/I blocks are dumped. You can use the generated diagnostic output, in conjunction with *IMS Version 15.5 Messages and Codes, Volume 3: IMSAbend Codes* to determine the cause of the user abend you are diagnosing.



**Attention:** The COMPARE SNAP statement is a call to DL/I. Therefore, when a SNAP option is issued, some data in the captured area might be changed as a result. To prevent inadvertent change to data that is not involved in the problem, use a COMPARE SNAP statement only for the specific data that is involved in the problem.

Some control blocks are always dumped. Others are dumped only when you request them in the SNAP options.

The following control blocks are always dumped:

- The SCD
- The PST (save areas related to the current DL/I task are a part of the PST)
- The Retrieve trace area

The following SNAP option requests dump the control blocks or buffers listed:

- A request for the buffer pool dumps:
  - OSAM buffer pool prefix and buffer pool, if present
  - VSAM subpool prefix and buffer prefix and subpools
  - Header for the DL/I, dispatcher, scheduler, and latch trace tables
  - The DL/I trace table
  - The dispatcher trace table
  - The scheduler trace table
  - The latch trace table
  - Hierarchical direct (HD) trace table, if present
  - Sequential buffering control blocks and buffer pools, if present
- A request for the current DB PCB or all PSB-related control block dumps:

- Delete/replace work areas, when allocated
- ENQ/DEQ trace table, if present
- PSB and PSB work areas
- PCB information, including JCB, DSGs, and level table
- The block of SDBs, SDB expansion blocks, and generated SDBs
- DMB directories
- DMBs for the current PSB
- PNTs associated with partition DMBs

If you also requested buffers, a request for the current DB PCB or all PSB-related control block dumps:

- Any HISAM/QSAM buffers
- Any VSAM LRECs for each qualifying DSG
- A request for the entire region, or subpools 0-127, dumps the entire region or the subpools.

A SNAP of the entire region or subpools is sent to a SNAP data set.

If the SNAP destination is the IMS log, the request is changed to a SNAP of all control blocks, regardless of other option specifications.

A region or subpool SNAP, when requested, appears before any additional SNAPs that were requested.

If the destination of the SNAP is the IMS log, you can select and format these records (type X'67FD') from the log by using the File Select and Formatting Print utility (DFSERA10) with formatting exit routine DFSERA30.

## SNAPs on exceptional conditions

---

IMS produces SNAPs of DL/I control blocks on the IMS log (or the CICS system log).

The SNAP call facility identifies calling routines that generate snap dumps. Supervisor call (SVC) dumps are generated only for the intended abend codes or status codes, and for unknown calling routines.

IMS produces SNAPs of DL/I control blocks on the IMS log (or the CICS system log) in the following exceptional situations:

- A pseudoabend condition is encountered in a DL/I module.
- A system or user abend occurs for either a message region or a batch message region.

Control block SNAPs are produced in the same format as those produced by a DL/I SNAP call specifying ALL or YYY as SNAP options.

The SNAP IMS log records are record type X'67', subrecord type X'FF'. You can select these log records from the IMS log with the File Select and Formatting Print utility (DFSERA10). You can format output selected from the log with the formatting edit routine DFSERA30.

Internal IMS functions can request the snapping of specific virtual storage areas by issuing a SNAP Specific call to DFSERA20.

The following IMS functions request or use the SNAP Specific facility:

- SBSNAP option, on completion of calls from IMS modules to the Sequential Buffering buffer handler
- SBESNAP option, during SB evaluation
- SB COMPARE option, when detecting a mismatch between the buffer content that the SB buffer handler was returning to the OSAM buffer handler and the content of the database block as it is stored on DASD

For IMS online regions and CICS, these SNAPs are written to the IMS log. For IMS batch regions, these SNAPs can be written to either the log or to a data set specified on another DD statement.

When written to the log, the IMS log records have a record type X'67' and a subrecord type X'E'. The value of the low-order half-byte of the subrecord type depends on the IMS function that requests the SNAP. The subrecord types are:

**X'ED'**

SBESNAP option

**X'EE'**

SBSNAP option

**X'EF'**

SB COMPARE option

The formatting edit routine DFSERA30 can format output selected from the log.

**Related reference**

[“Printing log records and trace table entries” on page 511](#)

You can use the File Select and Formatting Print utility (DFSERA10) to print both log records from the IMS log data set and the externalized trace table entries that are recorded in the DFSTRAXx data set.

[“Sequential buffering service aids” on page 227](#)

When you receive a message or abend that indicates a problem with sequential buffering (SB), several diagnostic tools are available. Some of these tools are useful for diagnosing other IMS database-related problems.

## DL/I call image capture

---

DL/I call image capture (module DFSDLTR0) enables you to trace and record all DL/I calls issued by an application program. The trace output is in a format acceptable as input to the DL/I test program DFSDDLT0.

### About this task

DL/I call image capture is a useful debugging tool because it allows you to rerun an application program and generate the DL/I calls necessary to duplicate the condition that caused the program failure. This run provides you with documentation to assist you in problem determination.

You can run the trace in either a batch or DB/DC environment.

### Related tasks

[“DL/I call image capture program” on page 221](#)

The DFSDLTR0 program, which operates independently, traces and records all DL/I calls issued by an application or multiple applications. The output is in a format acceptable as input to the DL/I test program DFSDDLT0.

## Batch environment

In a batch environment, you start DL/I call image capture using the DLITRACE control statement in the DFSVSAMP DD data set.

### About this task

The control statement allows you to trace either all DL/I calls issued by an application program or a range of calls. The traced information can be put in a sequential data set, the IMS log data set, or into both concurrently.

## Online environment

In a DB/DC, DCCTL, or DBCTL environment, you start and terminate DL/I call image capture by issuing the /TRACE command from the master terminal (DB/DC and DCCTL only) or from the system console.

### About this task

For example, to trace full-function database calls for a named PSB and send the output to an external data set, issue the following command:

```
/TRACE SET ON PSB psbname OPTION LOG
```

## How to retrieve DL/I call image capture data from the log data set

If trace data is sent to the IMS log data set, you can retrieve it using the File Select and Formatting Print utility (DFSERA10) and the DL/I call image capture exit DFSERA50.

### About this task

To use DFSERA50, you need to insert a DD statement defining the output data set in the DFSERA10 input stream. The default ddname for this DD statement is TRCPUNCH. The statement must specify LRECL=80.

## DL/I analysis

---

Debugging suggestions for a batch environment, including DL/I and DBB regions are described.

These debugging suggestions are useful in a batch environment. The information is valid for DL/I or DBB regions.

Before diagnosing abends in a batch region, review the external conditions. Verify that your environment is correct by asking the following questions:

- Are the JOBLIB/STEPLIB DD statements pointing to the correct libraries?
- Are the PSBLIBs and DBDLIBs at the same level as the JOBLIB/STEPLIB modules?
- If running with an ACBLIB, was the ACBGEN run under the same level of IMS you are currently running on?
- Were the databases correctly allocated and intact before starting the current run?

### Related tasks

[“Locating database-related traces” on page 164](#)

Locating Retrieve trace, JCBTRACE, DL/I trace and LOG data set are shown.

## IMS abends

User abends and system abends are discussed.

In general, abend dumps have two causes:

- An abend issued by an IMS module (user abend)
- A program check within an IMS module (system abend)

All IMS abends are issued with the dump option.

### User abends

There are two methods by which an IMS module can issue an abend when an error condition is detected.

- The first method is the standard ABEND macro issued by the code at the point of error detection. With this method, the PSW, at entry to the abend, points at the code within the module that both detected the error and issued the abend.

- With the second method, the module that detects the error does not issue the abend, but instead passes the error indication back to the program request handler, which then issues a real abend. The PSW, at entry to the abend, now points to the program request handler rather than to the module that detected the error. The pseudoabend method is used by DL/I modules that abend an application program in a dependent region but do not abend the IMS control region in a DB/DC environment.

When the DL/I test program is used as the application program, the pseudoabend is passed back to the test program rather than to the program request handler. This allows the test program to request a formatted SNAP rather than only an abend dump.

## Dump analysis introduction

In a pseudoabend supervisor call (SVC) dump that is generated by module DFSERA20, you can find the failing program specification table (PST) by searching the save areas for the caller of module DFSERA20. In the save area flow, DFSERA20 is called INTERA20 and register 1 contains the failing PST address.

The SNAP call facility identifies calling routines that generate snap dumps. SVC dumps are generated only for the intended abend codes or status codes, and for unknown calling routines.

The following list provides general considerations for dump analysis:

- The first request block (RB) on the RB chain represents the IMS batch region controller (DFSRR00). The second RB on the RB chain represents the batch program controller (DFSPCC30). Module DFSPCC30 always links to the application program that is named in the parameter field of the EXEC statement. Therefore, the application program must be represented by the third RB. However, if the application program uses an IMS service, and that service abends, the third RB points to the offending IMS routine.
- The last two SVRBs represent ABEND and ABDUMP. The register contents at the time of abend are usually found in the first abend SVRB. The IMS STAE work area (DFSFSWA0) and the RTM work area in z/OS are also used to hold the register contents at abend time.
- There are two PSTs in a batch environment. One is used for all application calls and the second is used for background write whenever it is activated.
- Each PST has a 20-level save area set as part of the PST. When the abend occurs, ABDUMP prints the save areas that are associated with the active PST.
- At abend time, the IMS STAE routine gets control to flush the database buffers and close the log data set. It builds six additional save areas and chains them to the last save area in the active PST. The IMS STAE routine is partially contained within module DFSPCC30 and has an entry ID starting with the characters PCE.
- Most IMS modules use register 12 as a base register.

### Related concepts

[“Job control block trace” on page 155](#)

The job control block (JCB) trace is one of most useful diagnosis tools for any application problem that might occur. The trace is an easy way to determine the last five calls that were issued, and what their return codes were.

[“DL/I trace formats” on page 167](#)

The figures in this section show the formats of the most commonly used DL/I trace entries. The figures are included to help you understand the DL/I trace entries in order to communicate more effectively with IBM Software Support representatives and to build a valid search argument.

### Related reference

[“HDAM, HIDAM, PHDAM, or PHIDAM database” on page 129](#)

HDAM, HIDAM, PHDAM, and PHIDAM databases share the following record format.

## Dump analysis - detailed

To thoroughly analyze a dump, you need to understand the save area, DL/I call sequence, and the buffer handler request sequence. This section discusses each of these elements.

### Save areas

A DL/I call passes from the application program to the DL/I language interface (DFSLI000), to the program request handler (DFSPR000), to the batch nucleus (DFSBNUC0), and then to the DL/I call analyzer (DFSDLA00).

If everything works properly, the save area trace shows the contents of the registers at entry to the application program, the program request handler, and the DL/I analyzer. The DL/I analyzer passes the first save area in the PST to a DL/I module. This PST save area is the first save area below the save area that holds the contents of the registers at entry to the DL/I analyzer.

The contents of register 1 at entry to the DL/I analyzer is a pointer to the PST. This is the only register passed to the analyzer (the user call list pointer is passed to the analyzer in PSTIQPRM).

If the abend is a program check or an inline abend, the save area trace always gives a true indication of the flow of control between DL/I modules and the current depth of save area set usage. Most DL/I modules or X'01' with the low-order byte of register 14 on return to a higher-level module.

If the abend is a pseudoabend, the save areas below the analyzer might have been reused and therefore would not reflect the conditions at the time the abend condition was detected; for example, the DB Monitor might have been called by the analyzer.

**Note:** When pseudoabends are detected by some modules, the registers 14 to 12 at error, are stored at PSTSAVL+12. The high order byte in PSTSAVL+12 will contain a one-byte code for the module detecting the error. Here are the modules which will save registers and their corresponding codes in PSTSAVL+12:

**X'AA'**  
DFSDLR00  
**X'BB'**  
DFSDDL00  
**X'CC'**  
DFSDDL00  
**X'DD'**  
DFSDXMT0  
**X'EE'**  
DFSURGU0  
**X'FF'**  
DFSRCHB0

Here is an example from the formatted PST of an abend U0853:

```
WD1 00000000 HSA 202C6BC8 LSA 2CD73B08 RET AA049128 EPA 30B02F40 R0 30000355
R1 212AD040 R2 2CD78790 R3 2FB6F5B4 R4 8004911E R5 2FB6FA8C R6 01410254
R7 21748060 R8 2FB6F82C R9 00000002 R10 30B053C0 R11 000401E0 R12 00047DC0
```

Since "RET" (PSTSAVL+12) contains X'AA', module DFSDLR00 detected this pseudoabend.

### DL/I call sequence

You can determine the current DL/I call and the sequence of calls leading up to the failure by scanning the DL/I trace table. Find the last entry made in the trace table by using the current entry pointer and then scanning backward in the table for the last entry made by the DL/I analyzer (entry code AA). This entry represents the current DL/I call.



You can determine the call sequence by continuing the backward scan, noting each entry made by the analyzer. Along with the call function, the analyzer also records the PCB address that was passed in the user's call list.

## Buffer handler request sequence

The buffer handler router traces each request to the buffer handler from a DL/I module. When the router receives the request, it passes the request to the OSAM buffer handler or the VSAM interface module. When the call is complete, control returns to the router. The router obtains the next available trace table entry and stores information describing the input and output for the buffer handler call.

By looking at all buffer handler entries between two DL/I analyzer DFSDLA00 entries (two specific DL/I calls), you can determine all requests made to the buffer handler to satisfy any specific DL/I call. A typical request to the buffer handler is a GET by relative byte address from the retrieve module. The entry made for this GET by relative byte address has a function code of E2, the RBA requested, and, if the request was satisfied (return code 0), the address of the segment read into the buffer pool.

## Generalized DL/I problem analysis

One method of problem analysis is discussed. Not all DL/I abends can be diagnosed using this sequence, but you can use the steps as a guide to DL/I debugging. All numbers are in hexadecimal.

### Procedure

1. The following approaches are valid if the IMS dependent region subtask appears in the dump.
  - Look at the user's call list for the current or last call. PSTIQPRM points to the call list. For all dependent region types, if the reentrant DL/I language interface, DFSLI000, is used, the user's call list address can be found in the contents of register 1 in the save area set at entry point to DFSPROX0-115 from the save area trace.
  - To find the last call parameters in a MPP or BMP dump, locate module DFSFSWA0 in the dump. Scan this module for ECP. At offset X'104' from ECP is a pointer to the parameters that made the last call to DL/I.
  - To find the PCBs in an MPP or BMP dump, find DIRCA in module DFSFSWA0. The word immediately following DIRCA contains the address of an area of storage obtained by the GETMAIN macro instruction. This area contains the PCB list and all non-GSAM PCBs. The format of this area is:
    - At offset X'14' is the beginning of the PCB list passed to the program.
    - Immediately following the end of the PCB list is a copy of the I/O PCB, if one exists.
    - The next PCB (and subsequent PCBs) follow the end of the I/O PCB.
  - Because they exist elsewhere in the dump, GSAM PCBs are not copied here. The pointers to the GSAM PCBs can be found in the PCB list at offset X'14.'
2. If the abend occurred after the DL/I analyzer received the call, but before the application program got control back, the last call entry (code AA) in the DL/I trace table matches the current call. Use the technique described in "DL/I call sequence" on page 162 to determine the call sequence as far back as possible, noting the PCB address associated with each call.
3. Compare the contents of PSTDBPCB to the PCB address in the last call entry in the trace table. If they are different, index maintenance is probably in control using its PCB within the PSB. Check the save area trace to verify this.
4. Find the current PCB from the address in the trace table, and then find the JCB. Starting at label JCBTRACE in the JCB are six 2-byte trace entries for the last six calls issued against this PCB. The oldest entry is at the beginning and the newest entry is at the end of JCBTRACE. The first byte of an entry is the encoded call function and the second byte is the last half of the status code for that call. For example, an 0140 is an entry for a GET UNIQUE call that resulted in a blank status code. This trace is maintained by the DL/I analyzer at the completion of the call.
5. Look at the contents of JCBLEV1C. If the call is a get or an insert, the retrieve module zeros this word at entry and then stores a pointer to each level table entry when it completes the call for that particular

level. If the word is zero, retrieve is still trying to satisfy the call at the root level. Generally, JDBLEV1C reflects the lowest level satisfied during the current or last get or insert call.

6. Check each level table entry to see if it holds a valid current position. Valid position is indicated by the absence of the empty bit in FLAG1 (LEVEMPTY in LEVF1, bit 1 byte 1). If this bit is off (valid position), LEVSDB points to the SDB currently in use or the last one used for this level. At the same time, LEVTTR, which contains either a relative byte address (RBA) or a relative record number (RRN), should match the current position saved in the SDB (SDBPOSC). In addition, if the database is HISAM, LEVSEGOFF matches SDBPOSN. This is the offset into the current relative record number.
7. Look at the key feedback area—level table position. The key feedback area contains the fully concatenated key of the segment currently positioned on. If a level table entry contains a valid position, the contents of the key feedback area for that level is the key (if any) of the segment whose SDB is pointed to by LEVSDB and whose database position is contained within LEVTTR and LEVSEGOFF. The contents of the key feedback area are never cleared or blanked out. Therefore, unless the level table entry indicates it has a valid position, the residue in the key feedback area might not be meaningful.
8. Map the database structure involved in the failure. Starting with the root SDB, which you can find with a pointer in the JCB (JCBSDB1), take each SDB in the sequence it is found in the dump and examine the field SDBPARA. This field is a pointer to the parent SDB (the root SDB points at the PCB). Map the structure according to SDBPARA; the result should match the logical structure defined at PSBGEN time. When mapping the structure, note the contents of SDBTARG. If this field is nonzero, the segment is involved in either logical relationships or indexing. The code in the high-order byte indicates which is the case.
9. Use the DL/I trace table to analyze the sequence of buffer handler calls. The buffer handler trace is the most useful debugging tool for DL/I. The trace is available in both batch and DB/DC environments, and the entries are identical
  - Get calls are the most common, so this section uses a get call as an example. In an attempt to satisfy a get call, the retrieve module must examine a segment or a series of segments to see if it meets the call requirements. All segments must be requested from the buffer handler and the request must be in the form of an RBA, RRN, or a specific key request.
  - The most common request from retrieve to the buffer handler is a byte locate. The parameters passed to the buffer handler are the function (byte locate), the RBA requested, and the data set in which the RBA exists. At exit to the buffer handler router, the next available trace entry is obtained and the code of the function requested is stored in the first byte. The buffer handler function codes are listed in the PST DSECT under PSTFNCTN. The byte locate function code is E2. The second byte of the trace entry is the relative PST number responsible for the request, which in batch is always an 01.
  - Along with the function code, the DSG and RBA are placed into the entry. When the call to the buffer handler (OSAM or VSAM) is completed, the results are traced, again by the buffer handler router. The return code is stored in the third byte. The return codes are listed in the PST DSECT under PSTRTCDE. If the call is successful, the address of the segment within the buffer pool is stored at displacement C. This trace now shows each segment (RBA) requested by retrieve; by examining the buffer pools the contents of the segments and their prefixes can be seen. RBAs found in the trace table can be compared to position fields in the SDB and level table to accurately re-create the get call.

## Locating database-related traces

---

Locating Retrieve trace, JCBTRACE, DL/I trace and LOG data set are shown.

### About this task

The following figure shows how to locate the following traces:

- Retrieve trace—records the flow through the retrieve module subroutines.
- JCBTRACE—traces the status of the prior six calls.

- DL/I trace—shows calls made to the call analyzer, buffer handler, and hierarchic direct space management, as well as information on Delete/Replace.
- LOG data set—records database changes, before and after images.

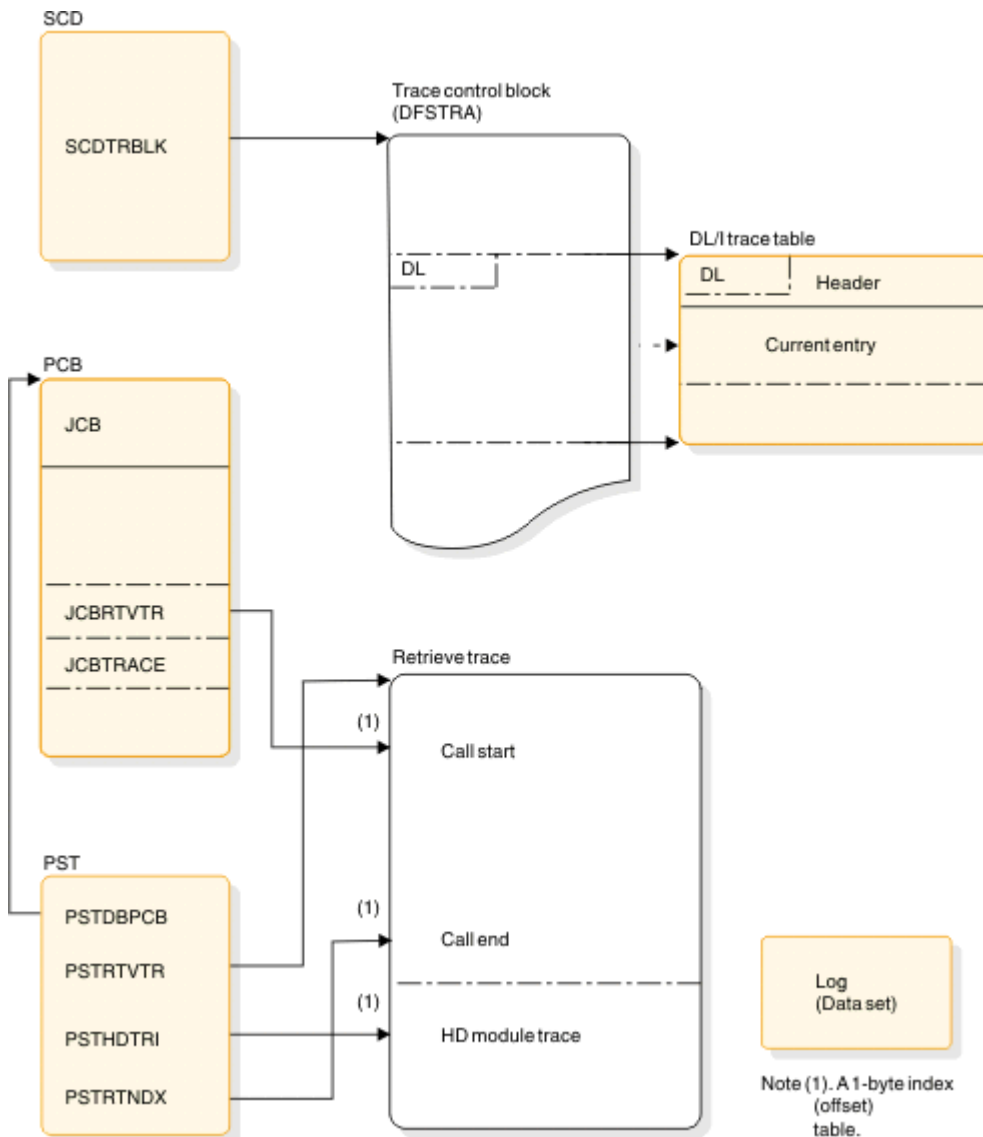


Figure 61. How to locate the database traces

### Related concepts

“DL/I analysis” on page 160

Debugging suggestions for a batch environment, including DL/I and DBB regions are described.

## DL/I trace

The DL/I trace table is a combined trace consisting of entries from DL/I calls, the DL/I buffer handler, DL/I OPEN/CLOSE, HD space management, lock activity (using PI or IRLM), OSAM, DFP interface, HALDB OLR trace, and ABENDU0427.

IMS always sets the DL/I and lock trace on at initialization except for batch, for which the default for traces is off. The trace level is set to high and it is written in memory.

The DL/I trace and the DL/I Call Image trace are different traces. The DLITRACE statement in IMS.PROCLIB member DFSVSMxx sets the DL/I Call Image trace on, the DL/I trace does not.

If the trace was written to the log, you can use the File Select and Formatting Print utility (DFSERA10) with an exit routine (DFSERA40 or DFSERA60).

- The Knowledge-Based Log Analysis (KBLA) panel interface.
- The File Select and Formatting Print utility (DFSERA10) with an exit routine (DFSERA40 or DFSERA60).

#### **Related reference**

[“Type-1 trace table interface” on page 574](#)

For each trace, you can learn about the trace identifier, the events that are traced, and, if the trace is documented in this information, the topic where you can find more information. You use the trace identifier as an eye catcher to locate a trace in a dump.

[“Sequential buffering service aids” on page 227](#)

When you receive a message or abend that indicates a problem with sequential buffering (SB), several diagnostic tools are available. Some of these tools are useful for diagnosing other IMS database-related problems.

## **Using the DL/I trace facility**

The DL/I trace facility is an important diagnostic tool that can help you determine the cause of a problem. Frequently, a problem occurs as a result of the interaction between two separate tasks. Interpreting the DL/I trace entries can be the best way of determining what each task was doing, and when.

### **Example for using the DL/I trace facility**

An IMS Fast Path application receives an abend 1027, and the user reports the problem to the support staff. Some of the steps the diagnostician might take are:

1. Find the abend code in *IMS Version 15.5 Messages and Codes, Volume 3: IMSAbend Codes*. This information indicates that the return code is in register 15.
2. Register 15 in the dump contains a value of X'0D.'

*IMS Version 15.5 Messages and Codes, Volume 3: IMSAbend Codes* indicates that this return code indicates that an enqueue or dequeue call was issued by module DBFBENQ0, and the return code from DFSLRH00 was X'12,' which indicates an invalid call.

3. Look at the DL/I trace to determine what resource was involved (if the DL/I trace was on at the time of the abend). If the DL/I trace was not on, it might be necessary to create the problem again with the DL/I trace facility on.

The list of trace entry IDs in [“DL/I trace formats” on page 167](#) indicates that one of the trace entries is "Exclusive control ENQ/DEQ PI trace entry" ([“The X'CA' trace entry for Fast Path calls” on page 181](#)). This is a good place to start the DL/I trace analysis.

What you learn from the DL/I trace might help you to:

- Identify and resolve an application error.
- Review APAR descriptions to see if this problem has occurred previously.
- Report the problem to IBM Software Support.

#### **Related concepts**

[“DL/I trace formats” on page 167](#)

The figures in this section show the formats of the most commonly used DL/I trace entries. The figures are included to help you understand the DL/I trace entries in order to communicate more effectively with IBM Software Support representatives and to build a valid search argument.

#### **Related reference**

[“X'C4' trace entry” on page 177](#)

The X'C4' trace entry is shown.

## DL/I trace formats

The figures in this section show the formats of the most commonly used DL/I trace entries. The figures are included to help you understand the DL/I trace entries in order to communicate more effectively with IBM Software Support representatives and to build a valid search argument.

**Exception:** Not every trace entry is shown. You can obtain the entries that are not described by assembling IDLIVSAM TRACENT from IMS.SDFSMAAC.

### Related concepts

[“Dump analysis introduction” on page 161](#)

In a pseudoabend supervisor call (SVC) dump that is generated by module DFSERA20, you can find the failing program specification table (PST) by searching the save areas for the caller of module DFSERA20. In the save area flow, DFSERA20 is called INTERA20 and register 1 contains the failing PST address.

[“Using the DL/I trace facility” on page 166](#)

The DL/I trace facility is an important diagnostic tool that can help you determine the cause of a problem. Frequently, a problem occurs as a result of the interaction between two separate tasks. Interpreting the DL/I trace entries can be the best way of determining what each task was doing, and when.

### Related tasks

[“Program isolation trace” on page 220](#)

The program isolation (PI) trace traces all calls to the IMS enqueue/dequeue module (DFSFXC10) and writes the trace entries to the system log as type X'67FA' records.

## X'0C' trace entry

The X'0C' trace entry is shown.

### TRACE ID = X'0C'

```
word 0 – byte 1 - X'0C' - DL/I OPEN/CLOSE trace entry for each
           data set. This entry shows a successful OPEN/CLOSE.
           For an error during OPEN/CLOSE, the data in ENTRY6 and
           ENTRY7, X'18' and X'1C' respectively, is shown in the
           "error condition"
           byte 2 - PST number
           bytes 3-4 - Trace sequence number description.
word 1 – byte 1 - PSTFNCTN (See note below)
           bytes 2-3 - DMB number
           byte 4 - DCB number
word 2 – DCB address
word 3 – DD name
word 4 – DD name
word 5 – PSTDBPCB - database PCB address
word 6 – DMB address (Error condition) - Offset in DFSDL0C0 where error was
           detected.
word 7 – bytes 1-3 - PSTPSB-PSB address - database PCB address (Error
           condition) - Word "LKER" or reason codes described in
           message DFS0730I
           byte 4 - Not used
```

## X'31', X'32', X'34', X'B1', and X'B2' trace entries

The X'31', X'32', X'34', X'B1', and X'B2' trace entries are shown.

### Example of X'31', X'32', X'34', X'B1', and X'B2' trace entries

#### TRACE ID = X'31', X'32', X'34', X'B1', X'B2'

```
word 0 – byte 1 - X'31', X'32', X'34', X'B1', or X'B2' - Function code for
           HD space management (see note 1 below)
           byte 2 - PST number
           bytes 3-4 - Trace sequence number
word 1 – bytes 1-2 - Length of request (see note 3 below)
```

```

        bytes 3-4 - Offset (requested or returned)
word 2 - byte 1 - PSTTRNID (ID of module calling space management)
        byte 2 - PSTTRMSC (subcode of module calling the buffer handler -
        see note 4 below)
        byte 3 - Not used
        byte 4 - PSTRTCDE (return code from space management)
word 3 - byte 1 - Flag byte (X'80' - entry already in use)
        bytes 2-4 - PSTDATA (core address - see note 5 below)
word 4 - PSTBYTNM (RBA or RRN - see note 6 below)
word 5 - RBA of space given to caller
word 6 - bytes 1-2 - DMB number
        byte 3 - DCB number
        byte 4 - Reserved
word 7 - MSG/ABEND feedback

```

#### Notes:

1. You need the X'32' entries to resolve this problem.
2. Numbers 3 and 4 are very important. In most cases, the segment was deleted by another task (see PST number), and this task (see PST number) tried to enqueue on the segment that waited while the other PST finished its processing. During the attempt, an FSE was found and abend U0832 resulted. An IMS internal error usually causes this problem.
3. The length of the segment that was freed.
4. The real storage address of the segment during the time of deletion.
5. The PSTBYTNM is the key field in the trace table. Look for a X'32' entry with the PSTBYTNM field equal to the PSTBYTNM field found in the buffer trace.

## X'60' trace entry

The X'60' trace entry is shown.

#### The X'60' trace entry

```

TRACE ID   = X'60'

word 0      - byte 1 - X'60' OSAM START I/O
              byte 2 - Zero (no PST number)
              bytes 3-4 - Trace sequence number
word 1      - IOSB address
word 2      - START I/O - DECB address
word 3      - START I/O - RBN/M
word 4      - Buffer address
word 5      - Buffer data (offset X'40' into buffer)
word 6      - byte 1 - Request type
              - byte 2 - not used
              - byte 3 - 'M' DCBFDAD seek value (MBBCCCHHR)
              - byte 4 - 'C' DCBFDAD seek value (MBBCCCHHR)
word 7      - CHHR DCBFDAD seek value (MBBCCCHHR)

```

## X'61' trace entry

The X'61' trace entry is shown.

#### The X'61' trace entry

```

TRACE ID   = X'61'

word 0      - byte 1 - X'61' OSAM POST
              byte 2 - Zero (no PST number)
              bytes 3-4 - Trace sequence number
word 1      - IOSB address
word 2      - DCB address
word 3      - byte 1 - DFSA0S70 INTERNAL TRACE BYTE
              byte 2 - TOTAL I/O COUNT FOR THIS STARTIO
              bytes 3-4 - Sequence number of associated X'60' entry

```

```

word 4 - Buffer address
word 5 - Buffer data (offset X'40' into buffer)
word 6 - byte 1 - POST code
        byte 2 - not used
        byte 3 - 'M' DCBFDAD seek value (MBBCHHR)
        byte 4 - 'C' DCBFDAD seek value (MBBCHHR)
word 7 - CHHR DCBFDAD seek value (MBBCHHR)

```

## X'62' trace entry

The X'62' trace entry is shown.

### The X'62' trace entry

```

TRACE ID = X'62'

word 0 - byte 1 - X'62' OSAM trace entry for OPEN/CLOSE/EOV
        - byte 2 - Zero (no PST number)
        - bytes 3-4 - Trace sequence number
word 1 - Not used
word 2 - DCB address
word 3 - DCBRELA
word 4 - byte 1 - Not used
        byte 2 - R15 return code
        bytes 3-4 - Not used
word 5 - OPEN/CLOSE/EOV error code (same as in message DFS0730I)
word 6 - byte 1 - Caller's function
        - byte 2 - Not used
        - byte 3 - 'M' value from DCBFDAD
word 7 - byte 1 - 'CHHR' from DCBFDAD

```

## X'63' trace entry

The X'63' trace entry is shown.

### X'63' trace entry

```

TRACE ID = X'63'

word 0 - byte 1 - X'63' OSAM MM I/O START
        byte 2 - MM function code
        bytes 3-4 - Trace sequence number
word 1 - MMRE address
word 2 - DECB address
word 3 - MMP address
word 4 - Buffer address
word 5 - Media Manager return code
word 6 - CI number
word 7 - not used

```

## X'64' trace entry

The X'64' trace entry is shown.

```

TRACE ID = X'64'

word 0 - byte 1 - X'64' OSAM MM I/O POST
        byte 2 - MM function code
        bytes 3-4 - Trace sequence number
word 1 - MMRE address
word 2 - DECB address
word 3 - MMP address
word 4 - Buffer address
word 5 - not used
word 6&7 - byte 1 - post code
        - bytes 2-3 - status code
        - bytes 4-5 - sense code
        - bytes 6-8 - not used

```

## X'65' trace entry

The X'65' trace entry is shown.

```
TRACE ID    = X'65'
word 0      - byte 1 - X'65' OSAM MM trace entry for OPEN/CLOSE/EOV
              byte 2 - MM function code
              bytes 3-4 - Trace sequence number
word 1      - MMIB address
word 2      - DECB address
word 3      - HURBA value
word 4      - HABRA value
word 5      - Media Manager return code
word 6&7    - Media Manager feedback code
```

## X'69' trace entry

The X'69' trace entry is shown.

### The X'69' trace entry

```
TRACE ID    = X'69'

word 0 - byte 1 - X'69' - Sequential Buffering buffer invalidation
              trace entry
              byte 2 - PST number
              bytes 3-4 - Trace sequence number
word 1 - bytes 1-2 - DMB number
              - byte 3 - DCB number
              - byte 4 - Function code at entry to DFSSBCI0 (see note 1 below)
word 2 - bytes 1-2 - Number of processed DCBs
              - bytes 3-4 - Number of invalidated SBH buffers
word 3 - DSG address of owner of the last invalidated SBH buffer
              or zero
word 4 - byte 1 - SBPSTTGS - Global serialization trace (see
              note 2 below)
              byte 2-4 - Not used
word 5 - Not used
word 6 - SBH buffer CB address of last invalidated SBH buffer or zero
word 7 - Block number in call or zero
```

### Notes:

- X'00'**  
Sequential Buffering buffer invalidation trace entry.
- X'01'**  
Sequential Buffering buffer invalidation trace entry.
- X'02'**  
Invalidate specific according to OSAM buffer prefix.
- X'80'**  
Global serialization entered (SBH search started).
- X'40'**  
Waiting for PST to be posted.

## X'6A' trace entry

The X'6A' trace entry is shown.

### The X'6A' trace entry

```
TRACE ID    = X'6A'

word 0 - byte 1 - X'6A' - Sequential Buffering buffer
              evaluation trace entry
              byte 2 - PST number
              bytes 3-4 - Trace sequence number
```



```

word 1 - bytes 1-2 - DMB number
        - byte 3 - DCB number
        - byte 4 - Not used
word 2 - byte 1 - Type of evaluation (see note 1)
        - byte 2 - Not used
        - byte 3 - Result of evaluation of sequentially
                  (see note 2)
        - byte 4 - Result of evaluation of I/O rate (see note 2)
word 3 - DSG address
word 4 - SBPSTCNB (=SBH CALL NUMBER THIS PST)
word 5 - byte 1 - Not used
        - bytes 2-4 - Threshold cost for SB logic
word 6 - - byte 1 - Not used
        - bytes 2-4 - Current cost of SB logic
word 7 - bytes 1-2 - Threshold value for I/O rate
        - bytes 3-4 - Current value of I/O rate

```

#### Notes:

##### 1. C'P'

Periodical evaluation

##### C'E'

Early evaluation

##### 2. C'P'

Evaluation is positive

##### C'N'

Evaluation is negative

## X'6B' trace entry

The X'6B' trace entry is shown.

### The X'6B' trace entry

#### TRACE ID = X'6B'

```

word 0 - byte 1 - X'6B' - Indicates why SB was or was not used
        byte 2 - PST number
        bytes 3-4 - Trace sequence number
word 1 - C'TERM'
word 2 - byte 1 - SCDSBFL - Sequential buffering flag (see note 1)
        - byte 2 - Resource allocation failure (see note 2)
        - byte 3 - Info from user exit routine (see note 3)
        - byte 4 - SBPSTITR - Termination trace flag (see note 4)
word 3 - Not used
words 4-5 - Job name
words 6-7 - PSB name

```

#### Notes:

##### 1. X'80'

...SCDSBNSB: DON'T LOAD SB MODULES

##### X'20'

...SCDSBLER: ERROR WHILE LOADING SB MODULES SB

##### X'10'

...SCDSBOER: OTHER SB ERRORS

##### 2. X'80'

...SBPSTGM1: GM ERROR FOR CB OR WORKAREA

##### X'40'

...SBPSTGM2: GM ERROR FOR SBH BUFFERS

##### X'20'

...SBPSTGM3: MAXSB= LIMIT EXHAUSTED

##### X'10'

...SBPSTGM4: MAX NBR OF IOSB EXHAUSTED

**X'08'**  
 ...SBPSTGM5: GETIOSB FAILURE

**X'04'**  
 ...SBPSTGM6: PAGE-FIX ERROR

**X'02'**  
 ...SBPSTGM7: I/O-ITASK INIT FAILURE

**X'01'**  
 ...SBPSTGM8: GM ERROR FOR CB OR WORKAREA

3. **X'80'**  
 ...SBPRMPDI: DISALLOW USAGE OF SB
- X'40'**  
 ...SBPRMPAD: CONDITIONAL ACTIV BY DEFAULT
4. **X'80'**  
 ...SBPSTITP: USER PROVIDED SB= KEYW IN PSBGEN
- X'40'**  
 ...SBPSTITC: SBPARM CARD PROCESSED
- X'01'**  
 ...SBPSTITS: /STOP SB ISSUED BY MTO

## X'6C' trace entry

The X'6C' trace entry is shown.

### The X'6C' trace entry

```
TRACE ID    = X'6C'

word 0 - byte 1 - X'6C' - Indicates if Sequential buffering was
                        used trace entry
                        byte 2 - PST number
                        bytes 3-4 - Trace sequence number
word 1 - bytes 1-2 - DMB number
        - byte 3 - DCB number
        - byte 4 - Not used
word 2 - bytes 1-2 - Number of refreshed SBH buffers
        - bytes 3-4 - Number of invalidated SBH buffers
word 3 - DSG address of owner of the last touched SBH
        buffer or zero
word 4 - byte 1 - SBPSTTGS - Global serialization
        trace (see note below)
        - bytes 2-4 - Not used
word 5 - OSAM BH prefix address
word 6 - SBH buffer CB address of last touched buffer or zero
word 7 - Block number
```

#### Notes:

**X'80'**  
 Global serialization entered (SBH search started).

**X'40'**  
 Waiting PST was posted.

## X'6F' trace entry

The X'6F' trace entry is shown.

### The X'6F' trace entry

```
TRACE ID    = X'6F'

word 0 - byte 1 - X'6F' - Sequential Buffering search by RBA issued
```

```

                                by OSAM BH trace entry
                                byte 2 - PST number
                                bytes 3-4 - Trace sequence number
word 1 - bytes 1-2 - DMB number
        - byte 3 - DCB number
        - byte 4 - Last byte of return code from OSAM BH
word 2 - First trace word within SDSG
word 3 - DSG address
word 4 - Second trace word within SDSG
word 5 - OSAM BH prefix address
word 6 - SBH buffer CB address
word 7 - Block number

```

## X'80', X'81', X'82' trace entries

The 'X'80', X'81', and X'82' trace entries are shown.

### The X'80', X'81', X'82' trace entry

```

TRACE ID   = X'80', X'81', X'82'

word 0 - byte 1 - X'80', X'81', X'82' - Database authorization,
        change-authorization, and re-authorization request
        byte 2 - PST number
        bytes 3-4 - Trace sequence number
words 1-2 - AURDBDNM - database name
word 3 - byte 1 - AURACC - database access
        - byte 2 - AURECD - authorized encoded state
        - byte 3 - AURSLV - database share level
        - byte 4 - AURWRKC - authorization work field
word 4 - bytes 1-2 - AURDMBNO - Global DMB number
        - bytes 3-4 - AURERRCD - DBRC error reason code
word 5 - AURSYSID - IMS online subsystem id
word 6 - AURDDIRA - DDIR address
word 7 - AURDSGCH - DSG address of last in DSG chain
        or
word 7 - TCB number for restart authorization by DFSRDA00

```

## X'AA' trace entry

The X'AA' trace entry is shown.

### The X'AA' trace entry

```

TRACE ID   = X'AA'

word 0 - byte 1 - X'AA' - Analyzer entry - This entry is created
        for each call passed to DFSDLA00. All entries
        are the internal activities in IMS that take place
        as a result of the user call. Be sure to use only the
        entries with the same PST number as the one identified
        as the failing PST.
        byte 2 - PST number (see note 1 below)
        bytes 3-4 - Trace sequence number
word 1 - Address of user parameter list (this list consists of all
        entries up to and including the entry with a X'80' in the
        high-order byte of a word.
word 2 - Call function for current call (GU, GN and so on -
        see note 2 below)
word 3 - PCB address for current call
words 4-5 - If DB PBC, LEVLEV thru LEVSEGOFF (first 10 bytes of level
        table for level of segment returned on prior call) IF
        TP PCB, character string is TP CALL
word 6 - bytes 1-2 - If DB PBC, LEVLEV thru LEVSEGOFF (first 10 bytes
        of level table for level of segment returned on prior
        call) IF TP PCB, character string is TP CALL
        bytes 3-4 - Status code in PCB from prior call (see
        note 3 below)
word 7 - LEVSDB - SDB address for level of segment returned on prior call

```

### Notes:

1. Use only the trace entries for the PST that had the failure.
2. Determine the current call.
3. Shows how the prior call for this PCB completed.

## X'AB' trace entry

The X'AB' trace entry is shown, with data set information, caller information, VSAM request option 1, and VSAM request option 2 listed.

### The X'AB' trace entry

#### TRACE ID = X'AB'

word 0 – byte 1 - X'AB' - ABEND U0427 trace entry  
          byte 2 - PST number  
          bytes 3-4 - X'0427'

word 1 – byte 1 - PSTFNCTN - Buffer handler function code  
          - byte 2 - RPLREQ  
          - bytes 3-4 - Trace sequence number

word 2 – bytes 1-2 - Offset to abend within DFSDVSM0  
          - byte 3 - DSGINDA - data set information (see note 1)  
          - byte 4 - DSGINDB - caller information (see note 2)

word 3 - RPLI address (Register 8)

word 4 - RPLARG - VSAM argument

word 5 - RPLAREA - VSAM area pointer

word 6 – byte 1 - RPLERREG - VSAM return code  
          - byte 2 - RPLERRCD - VSAM error code  
          - byte 3 - RPLOPT1 - VSAM request option (see note 3)  
          - byte 4 - RPLOPT2 - VSAM request option (see note 4)

word 7 - AMP address

### Notes:

1. See the following table for data set information.

Table 20. Data set information

Code (hex)	DSGINDA	Definition
X'80'	DSGDSOLS	This is the last DSG in JCB.
X'44'	DSGDSORI	Data set group is root in index.
X'20'	DSGDSOHD	Data set group is HDAM.
X'10'	DSGDSOHI	Data set group is HDAM.
X'08'	DSGDSOH2	Data set group is HISAM case 2.
X'04'	DSGDSOH1	Data set group is HISAM.
X'02'	DSGDSOHS	Data set group is HSAM or SSAM.
X'01'	DSGVSAM	Data set group is VSAM.

2. See the following table for caller information.

Table 21. Caller information

Code (hex)	DSGINDB	Description
X'80'	DSGSETLR	From SETL routine for SYNAD routine.
X'40'	DSGGETR	From GET routine for SYNAD routine.
X'20'	DSGBATIS	Record returned is batch, DSGIRECA is actual address.

Table 21. Caller information (continued)

Code (hex)	DSGINDB	Description
X'10'	DSGNXTIS	Next sequential root is current keyed record.
X'08'	DSGSETL2	Second SETL has been issued.
	DSGSETK2	Move key to DSG high key area.
X'04'	DSGGETGT	A GET in BISAM being done using a SETL GT.
X'02'	DSGKEYSR	Buffer pool has been searched for key.
X'01'	DSGSTLIS	This is STL for INSERT.

3. See the following table for VSAM request option 1.

Table 22. VSAM request option

Code (hex)	RPLOPT1	Description
X'80'	RPLLOC	Locate mode.
X'40'	RPLDIR	Direct processing.
X'20'	XRPLSEQ	Sequential.
X'10'	RPLSKP	Skip SEQ access.
X'08'	RPLASY	Asynchronous.
X'04'	RPLKGE	Search key GT/EQ.
X'02'	RPLGEN	Generic key request.
X'01'	RPLECBSW	External ECB.

4. See the following table for VSAM request option 2.

Table 23. VSAM request option 2

Code (hex)	RPLOPT2	Description
X'80'	RPLKEY	Keyed access.
X'40'	RPLADR	Addressed access.
	RPLADD	Addressed access.
X'20'	RPLCNV	CINV access (by RBA).
X'10'	RPLBWD	FWD=0/BWD=1
X'08'	RPLLRD	ARD=0/LRD=1
X'04'	RPLWAITX	SYN processing wait exit.
X'02'	RPLUPD	Update.
X'01'	RPLNSP	Note string position.

## X'AC' trace entry

The X'AC' trace entry is shown.

### The X'AC' trace entry

TRACE ID = X'AC'

```

word 0 - byte 1 - X'AC' - Database call analyzer entry (only present in a
              DBCTL environment)
              byte 2 - PST number
              bytes 3-4 - Trace sequence number
word 1 - Eye -catcher RTKN
word 2 - Not used
word 3 - Not used
words 4-7 - This 16-byte CCTL recovery token is used to correlate
              DL/I activity on other subsystems

```

## X'AD' trace entry

The X'AD' trace entry is shown.

### The X'AD' trace entry

#### TRACE ID = X'AD'

```

Word 0 - byte 1 - TRACE_AD X'AD' DB DL/I GUR (Get Unique Record) driver trace
              byte 2 - Relative PST number
              bytes 3-4 - Trace sequence number

Word 1 - byte 1 - TRCAD_FUNC    Trace function
              TRCAD_FLOW      EQU X'01'  Flow trace
              TRCAD_BUFF      EQU X'02'  Buffer trace
              TRCAD_IOCOPY    EQU X'03'  I/O area copy trace
              TRCAD_SSAMOVE    EQU X'04'  SSA modification trace
              TRCAD_GUCALL     EQU X'05'  GU call trace
              TRCAD_GNPCALL    EQU X'06'  GNP call trace
              TRCAD_ADD264     EQU X'07'  Add to 64 bit cache
              TRCAD_FIND64     EQU X'08'  Find in 64 bit cache
              byte 2 - TRCAD_SUBRTE Trace subroutine
              byte 3 - TRCAD_ID    Trace identifier
                  TRCAD_START     EQU X'01'  Trace start for flow
                  TRCAD_END       EQU X'02'  Trace start for flow
                  TRCAD_FIND      EQU X'03'  Trace find for buffer
                  TRCAD_WORK      EQU X'04'  Trace work for misc
              byte 4 - TRCAD_ID    Trace identifier
                  TRCAD_START     EQU X'01'  Trace start for flow
                  TRCAD_END       EQU X'02'  Trace start for flow
                  TRCAD_FIND      EQU X'03'  Trace find for buffer
                  TRCAD_WORK      EQU X'04'  Trace work for misc

Word 2 - bytes 1-4 and Word 3 - TRCAD_TOKEN    Returned token
      or
      bytes 1-4 and Word 3 - TRCAD_RSCNAME Resource name
      or
      bytes 1-4 and Word 3 - TRCAD_BUFFPTR Cache buffer pointer
      or
      bytes 1-4 - TRCAD_PCB PCB address

Word 3 - bytes 1-2 - TRCAD_PCBSTC DB PCB status code
              bytes 3-4 - TRCAD_ABTRM PST abend code

Word 4 - bytes 1-4 - TRCAD_GURWA GUR work area pointer
      or
      bytes 1-4 - TRCAD_GURLEN Buffer length of GUR output

Word 5 - bytes 1-4 - TRCAD_SSA SSA pointer
      or
      bytes 1-4 - TRCAD_IOAREA I/O area pointer

Word 6 - bytes 1-2 - TRCAD_IOALEN Total I/O area length
              bytes 3-4 - TRCAD_IOALEFT I/O area length remaining
      or
      bytes 1-4 - TRCAD_AIBPTR AIB area pointer

Word 7 - bytes 1-2 - TRCAD_RC Return code
              bytes 3-4 - TRCAD_RSN Reason code

```

Macro: IDLIVSAM

## X'C4' trace entry

The X'C4' trace entry is shown.

### The X'C4' trace entry

#### TRACE ID = X'C4'

word 0 – byte 1 - X'C4' - DELETE/REPLACE used to provide diagnosis information for error conditions. This entry is written when an error is detected.  
          byte 2 - PST number  
          bytes 3-4 - Trace sequence number  
word 1 – byte 1 - ID invoking subroutine (see note 2 below; see note 3 below)  
          byte 2 - ID of originating subroutine (see note 3 below)  
          byte 3 - Subcode (set by originating subroutine - see note 3 below)  
          byte 4 - Internal code for status code or pseudoabend (see note 3 below)  
word 2 – SDB for replace operation. DLTWS for delete operation.  
          Register value 7.  
word 3 – Level table for replace operation. DLTWA address for delete operation.  
          Register value 8.  
word 4 – Usually the PSDB address for segment. Register value 6.  
word 5 – byte 2 - DELETE/REPLACE return code  
          byte 2 - Return offset from caller's CSECT  
word 6 – PSTDSGA - DSG address  
word 7 – Information local to the subroutine that might be useful in problem resolution

#### Notes:

1. Use only the entries for the PST that abended.
2. When a DELETE/REPLACE failure occurs, you need the X'C4' entries to solve the problem. You can usually find several X'C4' entries in a row in the trace table. Scan up the trace table to the first (lowest trace sequence number) entry. This entry is usually the key to why the failure occurred. Level 2 needs this information to resolve the problem.
3. These 4 bytes, in word 2, in a DELETE/RELEASE error are documented in the *IMS Version 15.5 Messages and Codes, Volume 3: IMSAbend Codes* for the various abends. This is ENTRY1 field referred to in the DELETE/REPLACE module.

#### Related concepts

[“Using the DL/I trace facility” on page 166](#)

The DL/I trace facility is an important diagnostic tool that can help you determine the cause of a problem. Frequently, a problem occurs as a result of the interaction between two separate tasks. Interpreting the DL/I trace entries can be the best way of determining what each task was doing, and when.

## X'C6' trace entry

The X'C6' trace entry is shown.

### The X'C6' trace entry

#### TRACE ID = X'C6'

word 0 – byte 1 - X'C6' - Special promote lock trace entry  
          byte 2 - PST number (see note 1 below)  
          bytes 3-4 - Trace sequence number  
word 1 – byte 1-2 - Not used  
          - byte 3 - Special lock/unlock call (see note 1)  
          - byte 4 - Level of this lock  
words 2-3 - C'PROMOTE '  
word 4 - REQ address  
word 5 - QCB address  
words 6-7 - Resource id (see note 2)

### Notes:

1. See the following table for special lock or unlock calls.

*Table 24. Special lock or unlock calls*

Code (hex)	Special lock or unlock call	Description
X'08'	PROENQ	Lock call
X'10'	PRODEQ	Unlock call

2. Resource id is an 8-byte field:

```
bytes 1-4 - Complement of original RBA
bytes 5-6 - DMB number
byte 7    - DCB number
byte 8    - C'Z' id suffix
```

## X'C7' trace entry not using IRLM

The X'C7' trace entry is shown when not using the IRLM.

### The X'C7' trace entry (when not using the IRLM)

**TRACE ID** = X'C7'

```
word 0 - byte 1 - X'C7' - Exclusive control deadlock detection trace
           entry (Written only when a conflict causes an abend.).
           byte 2 - PST number (see note 1 below)
           bytes 3-4 - Trace sequence number
word 1 - byte 1 - PST number (see note 1 below)
           bytes 2-4 - Address of PST to be backed out (gets
           ABENDU0777 - see note 3 below)
words 2-3 (see note 2 below) - byte 1 - PST number
           bytes 2-4 - Conflicting PST address
words 4-5 (see note 4 below) - PSB name
words 6-7 (see note 4 below) - DMB name
```

### Notes:

1. The entry for the PST number that received abend U0777.
2. The addresses of the two conflicting PSTs.
3. The address of the PST that got received abend U0777.
4. The PSB and DMB name of the cause for the contention.

## X'C7' trace entry using IRLM

The X'C7' trace entry is shown when using the IRLM.

### The X'C7' trace entry (when using the IRLM)

**TRACE ID** = X'C7'

```
word 0 - byte 1 - X'C7'
           byte 2 - 00
           bytes 3-4 - Trace sequence number
word 1 - Not used
words 2-5 (see note 1 below) - byte 1 - PST number
           bytes 2-4 - PST address
words 6-7 (see note 2 below) - Resource ID
```

### Notes:

1. PST number and address of PSTs in deadlock net. If number of PSTs in deadlock net is greater than 4, only 4 are shown.



2. Resource ID that is the cause of the deadlock.

## X'C8' trace entry

The following figure shows the X'C8' trace entry.

### The X'C8' trace entry

```
TRACE ID    = X'C8'

word 0 - byte 1 - X'C8' - Lock request manager entry (DFSLMGR0)
        byte 2 - PST number
        bytes 3-4 - Trace sequence number
word 1 - byte 1 - Function - See macro DFSLMD for mapping of
        each byte in this word
        byte 2 - State (see note below)
        byte 3 - Class - the class is the relative PST number
        byte 4 - Flags
word 2 - byte 1 - Return code from IRLM
        bytes 2-4 - Can be PST, CLB, or SRB address
word 3 - Can be resource name address, token, or altered
        buffer mask
word 4 - bytes 1-2 - Lock manager subcode (2 bytes). These bytes
        along with the return code from IRLM define the
        problem.
        bytes 3-4 - This is a feedback area from the RLPL and is used
        primarily by the IBM Support Center, if needed.
words 5-7 - This is a feedback area from the RLPL and is used primarily
        by the IBM Support Center, if needed.
```

**Notes:** The possible state settings and their meaning:

#### X'00'

Unconditional release

#### X'02'

Read

#### X'04'

Share

#### X'06'

Update

#### X'08'

Exclusive

## X'C9' trace entry

The X'C9' trace entry is shown.

### The X'C9' trace entry

```
TRACE ID    = X'C9'

word 0 - byte 1 - X'C9' - Lock request manager entry (DFSLMGR0) exit
        byte 2 - PST number
        bytes 3-4 - Trace sequence number
word 1 - byte 1 - Function - See macro DFSLMD for mapping of
        each byte in this word.
        byte 2 - State (see note below)
        byte 3 - Class - the class is the relative PST number
        byte 4 - Flags
word 2 - byte 1 - Return code from IRLM
        bytes 2-4 - Can be PST, CLB, or SRB address
word 3 - Can be resource name address, token, or altered
        buffer mask
word 4 - bytes 1-2 - Lock manager subcode (2 bytes). These bytes
        along with the return code from IRLM define the
        problem.
        bytes 3-4 - This is a feedback area from the RLPL and is used
        primarily by the IBM Support Center, if needed.
```

words 5-7 - This is a feedback area from the RLPL and is used primarily by the IBM Support Center, if needed.

**Notes:** The possible state settings and their meanings:

**X'00'**

Unconditional release

**X'02'**

Read

**X'04'**

Share

**X'06'**

Update

**X'08'**

Exclusive

## **X'CA' trace entry**

The X'CA' trace entry is shown.

### **The X'CA' trace entry**

#### **TRACE ID = X'CA'**

word 0 - byte 1 - X'CA' - Exclusive control ENQ/DEQ (PI - Program Isolation) trace entry  
byte 2 - PST number (see note 1 below)  
bytes 3-4 - Trace sequence number  
word 1 - byte 1 - Record type (see note 8 below)  
byte 2 - Class for Q command operation  
byte 3 - Requested function (Use PRM DSECT (PRMFUNCTN) - see note 2 below)  
byte 4 - PRMLEVEL - Level of control requested  
(1 =Read only, 2=Share, 3=Update, 4=Exclusive - see note 3 below)  
word 2 - bytes 1-2 - Wait count (how many times this task had to wait - see note 7 below)  
bytes 2-4 - Waited for count (number of tasks waiting for this resource)  
word 3 - PITIME relative to 00:00:00 on PIDATE (SCDPITIME)  
word 4 - bytes 1-2 - Feedback from DFSFXC10 (Use PRM DSECT, PRMFBK field. See note 5 below)  
byte 3 - Return code from DFSFXC10 (see note 6 below)  
byte 4 - PSFUNCT (function codes DSECT)  
word 5 - Token from DFSFXC10 (pointer to control block enqueued resource)  
word 6 - RBA or RBN (see note 4 below)  
word 7 - bytes 1-2 - DMB number  
byte 3 - DCB number  
byte 4 - Not used

#### **Notes:**

1. Use the entries for the PST in question. If you are checking a PI problem, you might have to find this entry and then scan up the trace table using the field in note 4 (below) as a search field to find the other PST that is using the resources.
2. The requested PI function.
3. The level at which the resource was requested.
4. The RBA or RBN of the resource requested by PI (relates to X'04' in the X'CC' trace entry).
5. The 2 bytes of feedback from DFSFXC10 (X'0C' and X'0D' in PRM DSECT).
6. The return code. DFSFXC10 RETURN CODES: 0 - Successful 4 - Wait required - usually has CB trace related to it 8 - Pseudoabend, either lost deadlock (U0777) or out of ENQ/DEQ space (U0775) C - Invalid call

7. If a resource (RBA or RBN) is currently owned and the task (PST) must wait, the "wait count" (2 bytes) is incremented in a X'CA' trace entry for the task (PST) that owns the resource. The "waited for count" (2 bytes) is incremented to show that another task is waiting for the resource. This wait should also cause a X'CA', X'CB' pair of trace entries to show the wait occurred. (See the X'CB' trace entry for more details on PI waits.)
8. This shows the type of X'CA' record this is. (X'CA-08' trace entry follows.)

**X'00'**

Standard trace PI record

**X'01'**

Timing ACT/ENQ wait - may have CB trace entry associated with it

**X'04'**

Lock MGR trace record

**X'08'**

DL/I call record - see X'CA' - X'08' trace entry

## X'CA' trace entry, subtype X'08'

The X'CA' trace entry, subtype X'08' is shown.

**TRACE ID = X'CA'-X'08'**

```
word 0 - byte 1 - X'CA'-X'08' - PI-DL/I call trace entry
          byte 2 - PST number
          bytes 3-4 - Trace sequence number
word 1 - byte 1 - X'08' = DL/I call record
          bytes 2-4 - Not used
word 2 - bytes 1-2 - Wait count (how many times this task
          had to wait)
          bytes 2-4 - Waited for count (number of tasks waiting
          for this resource)
word 2 (alternate) - bytes 1-4 - name of module writing the trace record
word 3 - PI time
word 4 - PST account field for function (count of
          the time of calls)
word 5 - DL/I call (GNP, ISRT, and so on)
words 6-7 - Not used
```

The content of word 2 depends on the module that wrote the trace record. Word 2 either contains DLA0 or the wait counts for the task.

## X'C4' trace entry

The X'CA' trace entry for Fast Path calls is shown.

### The X'CA' trace entry for Fast Path calls

**TRACE ID = X'CA'**

```
word 0 - byte 1 - X'CA' - Exclusive control ENQ/DEQ (PI -
          Program Isolation) trace entry
          byte 2 - PST number (1)
          bytes 3-4 - Trace sequence number
word 1 - IRC1, indicating a Fast Path call
word 2 - Call Function (GU, GN, and so on)
word 3 - PROCOPT
word 4 - PI time (also in reg 5 in Fast Path trace,
          if active)
word 5 - A(PBC)
word 6 - A(EPCB)
word 7 - Not used
```

## X'CB' trace entry

The X'CB' trace entry is shown.

### The X'CB' trace entry

#### TRACE ID = X'CB'

word 0 - byte 1 - X'CB' - PI - (Program Isolation) trace lock  
                  elapsed time  
                  byte 2 - PST number  
                  bytes 3-4 - Trace sequence number  
words 1-2 - DMB Name for which the wait was performed  
word 3 - Same as PITIME IN X'CA' record  
word 4 - byte 1 - First byte of feedback from enqueue  
                  request  
                  byte 2 - PST owning resource at the time of wait  
                  bytes 3-4 - Trace sequence number on X'CA' record  
word 5 - Elapsed time for enqueue wait  
word 6 bytes 1-4 - word 8 bytes 1-3 - 7 bytes of resource ID  
word 7 - byte 4 - Post code

## X'CC' trace entry

The X'CC' trace entry is shown.

### The X'CC' trace entry

#### TRACE ID = X'CC'

word 0 - byte 1 - X'CC' - Lock request handler  
                  (DFSLRH00) entry  
                  byte 2 - PST number (see note 1 below)  
                  bytes 3-4 - Trace sequence number  
word 1 - Block number on RBA (see note 2 below)  
word 2 - PSTTOKEN - The object of the request  
word 3 - PSTLRPRM - These bytes are described in the  
                  PSTLRPRM chart below. The first byte equates to byte  
                  0, the second to byte 1, and so on (see note 3 below).  
word 4 - bytes 1-2 - Subcode from lock manager (IRLM) or PRMFBK  
                  feedback for DFSFXC10.  
                  byte 3 - Register 15 return code  
                  byte 4 - Return code from lock manager or DFSFXC10  
                          (Use DFSFXC10 return codes from the X'CA'  
                          trace entry, note 6) (See note 5 below)  
word 5 - byte 1 - PSTLRSUB-DFSLRH00 abend subcode (see note 7 below)  
                  bytes 2-4 - PSTABTRM - System abend code (see note 6 below)  
word 6 - PSTDSGA - Address of the DSG used by this PST  
word 7 - byte 1 - Return register  
                  bytes 2-4 - Address within module where DFSLRH00 was called

#### Notes:

1. The PST number for the task (PST).
2. The RBA or RBN of the resource for which a request was issued in a X'CA' trace entry. When some of the problem types occur, you can find the same field or the beginning RBA of the block in the traces for a different PST number.
3. Shows what the request was.
4. For PI, these 2 bytes are in the PRM DSECT at X'0C' and X'0D'.
5. For PI, follow the above. The DFSFXC10 return code is usually also placed in the register 15 return code field.
6. A key field when DFSLRH00 issues an abend (such as U0855, U03301, U03302). The abend is in hexadecimal, not in decimal (for example, 855= X'0357', 3302= X'0CE6'). Ignore the field if an abend was not issued from DFSLRH00.
7. For abends issued by DFSLRH00, this field contains the Lock Request Handler abend subcode.

You might need the X'CC' trace entry for several problem types including:

- Task was allowed to process even though a wait was requested.
- DFSLRH00 abends (such as U0855, U03302).
- Request not satisfied. These problems might indicate an internal IMS error.

The following table shows the PSTLRPRM chart (bytes 0 through 3).

<i>Table 25. PSTLRPRM chart (bytes 0 through 3)</i>	
<b>Byte 0 (hex)</b>	<b>Meaning</b>
11	Get local segment lock
12	Get local data set busy lock
13	Get local buffer update lock
14	Get local Q command lock
15	Get local catalog lock
22	Get global buffer update lock
23	Get global data set busy lock
24	Get global data set extend lock
25	Get global data set reference lock
26	Get global command lock
27	Get global command lock (CLB)
30	Get local and global root locks
31	Get local segment and global buffer update locks
32	Get local-global data set busy locks
33	Get local-global buffer update locks
34	Get local Q command and global buffer update locks
37	Get global catalog lock
41	Release local segment lock
42	Release local data set busy lock
43	Release local buffer update lock
44	Release local Q command lock
45	Release local catalog lock
52	Release global buffer update lock
53	Release global data set busy lock
54	Release global data set extend lock
55	Release global data set reference lock
56	Release global command lock
57	Release global command lock (CLB)
60	Release local and global root locks

Table 25. PSTLRPRM chart (bytes 0 through 3) (continued)

Byte 0 (hex)	Meaning
61	Release local and global data set busy locks
62	Release local and global buffer update locks
63	Release local segment and global buffer update locks
67	Release global catalog lock
70	Test local lock share or update state
71	Test global lock share or update state
72	Test local and global lock share or update
73	Test feedback for local lock
74	Test feedback for global lock
75	Test feedback for local and global locks
80	LRHGIRDX new root, LRHRRIDX old root
81	Release alternate local and global root locks
82	Get local segment and local and global buffer update locks
83	Release all subsystem global busy locks
84	Release all subsystem locks
86	Transfer locks
87	Change locks into retained status
90	Get Fast Path lock
91	Release Fast Path lock
92	Change ownership of Fast Path lock
93	Force known locks for Fast Path
94	Change locks to retain locks for Fast Path
95	Change ownership of Fast Path UOW lock from release lock ITASK to PST dependent region (HSSP only)
96	Change locks to retain locks for DL/I
97	Invalid call if function is equal to or greater than 97
Byte 1 (hex)	Meaning
80	MODE=COND
40	MODE=UNCOND
10	Owning WU given on RRIDX
00	Mode not applicable
Byte 2 (Hex)	Meaning
01	STATE=READ

Table 25. PSTLRPRM chart (bytes 0 through 3) (continued)

Byte 0 (hex)	Meaning
02	STATE=SHARE
03	STATE=UPDATE
04	STATE=EXCL
F0	STATE PRESET (Fast Path)
00	STATE not applicable
Byte 3 (hex)	Meaning
80	CLB call if LRHPRMFL=X'80'
C0	Fast Path request
68	Root lock request
40	'Single' request
20	'Local' request
10	'Get' request
08	'P-Lock' request
07	'Combined' request if <= X'07'
01	LRHTTLKX, LRHTIBDX
02	LRHGRIDU, LRHRRIDW
03	LRHGSEGX, LRHRSEGX
04	LRHGBIDX, -RBIDX, -GBIDA
05	LRHGZIDX, LRHRZIDX
06	LRHGQCMX
00	LRHRZIDA, LRHRALLX

## X'CF' trace entry

The X'CF' trace entry is shown.

### The X'CF' trace entry

**TRACE ID = X'CF'**

```

word 0 - byte 1 - x'CF' - I/O toleration (DFSTOPR0)
                                trace entry
        byte 2 - PST number
        bytes 3-4 - Trace sequence number
word 1 - byte 1 - I/O toleration return code
        - byte 2 - TORFUNC - I/O toleration function
                        code (see note 1)
        - byte 3 - TORFLG1 - I/O toleration flag 1 (see
                        note 2)
        - byte 4 - TORFLG2 - I/O toleration flag 2 (see
                        note 3)
words 2-3 - EEQEFLCS - EEQE flags
word 4 - DDIR or DMAC address
word 5 - RBA or RBN
word 6 - bytes 1-2 - DMB number

```

- byte 3 - DCB number  
- byte 4 - TORWORK+2 when DBRC change of EEQE  
word 7 - EEQE address

#### Notes:

1. See the following table for I/O toleration function codes.

*Table 26. I/O toleration function codes*

Code (hex)	TORFUNC	Description
X'01'	TORCEQE	Create EEQE.
X'02'	TORDEQE	Delete EEQE.
X'04'	TORFEQE	Find I/O toleration EEQE.
X'08'	TORMEQE	Copy/Move to I/O toleration buffer.
X'10'	TORNEQE	Send notifies on I/O toleration EEQE's.
X'20'	TORPURG	Close I/O toleration mode.
X'40'	TORDUI	Process DBRC DUI call EEQE list.
X'80'	TORDBCL	DB close I/O error write retry.
X'C0'	TORCHKPT	Do system checkpoint logging.

2. See the following table for I/O toleration flag 1.

*Table 27. I/O toleration flag 1*

Code (hex)	TORFLG1	Description
X'80'	TOR1FP	<ul style="list-style-type: none"> <li>• If on, Fast Path.</li> <li>• If off, DL/I.</li> </ul>
X'40'	TOR1NOT	Creator is notify.
X'20'	TOR1PST	<ul style="list-style-type: none"> <li>• If on, then R0 has PST address.</li> <li>• If off, then R0 has SCD address.</li> </ul>
X'10'	TOR1RST	Caller is restart log read.
X'01'	TOR1FPIR	DBFMIOS0: FP IDT resolution.
X'90'	TOR1FPRS	Caller is FP restart log read.

3. See the following table for I/O toleration flag 2.

*Table 28. I/O toleration flag 2*

Code (hex)	TORFLG2	Description
X'80'	TOR2IOT	Creator is I/O toleration.
X'40'	TOR2RD	Creator is read error.
X'20'	TOR2WRT	Creator is write error.
X'10'	TOR2USER	Creator is DBRC command.
X'08'	TOR2PERM	Creator is permanent.
X'04'	TOR2IDT	Creator is indoubt process.
X'01'	TOR2NDX	EEQEFLG2:EEQENDX KSDS INDEX CI



## X'D0' trace entry

The X'D0' trace entry is shown.

### The X'D0' trace entry

Words 0 and 1 are common to all X'D0' trace entries. The format of words 2 through 7 depends on the content of the trace entry.

#### Word 0

##### Byte 1

X'D0' - IRLM notify sent trace entry

##### Byte 2

PST number

##### Bytes 3 and 4

Trace sequence number

#### Word 1

##### Byte 1

Sub-route code:

**4**

VSAM buffer invalidation

**8**

VSAM write I/O error

**12 or 40**

VSAM data set extension

**16**

OSAM buffer invalidation

**20**

OSAM write I/O error

**24 or 44**

OSAM data set extension

**28**

Force snap

**32 or 36**

Extended error queue element add or delete

##### Bytes 2 and 3

DMB number or PID for partition

##### Byte 4

DCB number

Format for buffer invalidation or write error notify: Words 2 - 7 use the following format in a trace record for a buffer invalidation or write error notification:

#### Word 2

RBN/RBA of buffer

#### Word 3

Not used

#### Word 4

Not used

#### Word 5

Not used

**Word 6****Bytes 1 to 3**

Not used

**Byte 4**

NCBFLAG

**Word 7**

Not used

Format for OSAM data set extend: Words 2 - 7 use the following format in a trace record for an OSAM data set extension:

**Word 2**

DCBHIBLK

**Word 3**

DCBRLBLK

**Word 4**

DCBRBASN

**Word 5**

Volume serial number

**Word 6****Bytes 1 and 2**

Volume serial number (continued)

**Byte 3**

Not used

**Byte 4**

NCBFLAG

**Byte 7**

Not used

Words 2 - 7 use the following format in a trace record for a VSAM data set extension:

**Word 2**

VSILVL - VSI level number (current)

**Word 3**

VSICRBA - Highest-used relative byte address (current)

**Word 4**

VSICRBA - Highest-allocated relative byte address (current)

**Word 5**

VSILVL - VSI level number (extent)

**Word 6**

VSICRBA - Highest-used relative byte address (extent)

**Word 7**

VSICRBA - Highest-allocated relative byte address (extent)

**X'D1' trace entry**

The X'D1' trace entry contains information about buffer invalidations or write errors.

Words 0 and 1 are common to all X'D1' trace entries. The format of words 2 through 7 depends on the content of the trace entry.

**Word 0****Byte 1**

Trace entry code X'D1'

**Byte 2**

Not used (no PST number)

**Bytes 3 and 4**

Trace sequence number

**Word 1****Byte 1**

Sub-route code from the DFSNCB macro

**Bytes 2 and 3**

DMB number or PID for partition

**Byte 4**

DCB number

Words 2 - 7 for a buffer invalidation or write error notification record use the following format:

**Word 2**

RBN/RBA of buffer

**Word 3**

Buffer prefix address

**Word 4****Byte 1**

SB global serialization trace field:

**X'80'**

Global serialization entered (SBH search started)

**X'40'**

Waiting PST was posted.

**Byte 2**

Not used

**Bytes 3 and 4**

Number of invalidated buffers

**Word 5**

Last invalidated buffer address

**Word 6****Bytes 1, 2, and 3**

Not used

**Byte 4**

NCBFLAG

**Word 7**

Subsystem ID

Words 2 - 7 for an OSAM data set extent record use the following format:

**Word 2**

DCBHIBLK

**Word 3**

DCBRLBLK

**Word 4**

DCBRBASN

**Word 5**

Volume serial number

## Word 6

### Bytes 1 and 2

Volume serial number

### Byte 3

Not used

### Byte 4

NCBFLAG

## Word 7

Subsystem ID

Words 2 - 7 for a VSAM data set extent record use the following format:

## Word 2

VSILVL - VSI level number (current)

## Word 3

VSIMRBA - Highest relative byte address (current)

## Word 4

VSILVL - VSI level number (extent)

## Word 5

VSIMRBA - Highest relative byte address (extent)

## Word 6

AMP address

## Word 7

Subsystem ID

## X'D5' trace entry

The X'D5' trace entry contains information about coupling facility call requests.

X'D5' traces the completion or failure of each call request. If it finds an error, it logs error-related data and abnormally terminates with a dump. The module return and reason codes are also set in R0 and R15.

### TRACE ID = X'D5'

```
word 0 - bytes 1-4 - X'D5' - The module name 'DMAW.'
```

```
word 1 - bytes 1-4 - Last service used:
```

```
                  "CACH" - IXLCACHE,
```

```
                  "CONN" - IXLCONN,
```

```
                  "DISC" - IXLDISC,
```

```
                  "FUNC" - invalid MAWP1 function,
```

```
                  "VRP " - IDAMDVRP
```

```
word 2 - bytes 1-4 - Type of request:
```

```
                  "DATA" - for IDAMDVRP,TYPE=DATA,
```

```
                  "INDE" - for IDAMDVRP,TYPE=INDEX,
```

```
                  "OSAM" - for IXLCONN, IXLDISC,
```

```
                  "RDRG" - for IXLCACHE,
```

```
                  "VSAM" - for IXLCONN, IXLDISC,
```

```
                  "XI  " - for IXLCACHE
```

```
word 3 - Return code from service (PSTIXLRF).
```

```
word 4 - Reason code from service (PSTIXLR0).
```

```
word 5 - Processing flags for DFSDMAW0 (MAWBFLGS).
```

```
ENTRY1  - IXCYENFFUNCTION FUNCTION CODE
```

```
ENTRY2-3 - Value DFSDENF0
```

```
ENTRY4-7 - Value of the STRUCTURE NAME
```

## X'D9' trace entry

The X'D9' trace entry is shown.

### The X'D9' trace entry - words 0 through 2

Most X'D9' trace entries have the following information in the first three words, except for OLR command processing. The following figure shows words 0 through 2 of the X'D9' trace entry.

```
TRACE ID    = X'D9'

word 0 - byte 1 - x'D9' - Online Reorganization (OLR) trace
          entry.
          byte 2 - PST number.
          bytes 3-4 - Trace sequence number.
word 1 - byte 1 - Module ID.
          byte 2 - Module subcode.
          bytes 3-4 - Local DMB number.
word 2 - bytes 1-2 - Global DMB number.
          bytes 3-4 - Partition ID.
```

## X'D9' trace entry: OLR output data set validation or creation and inactive data set deletion

The following figure shows words that are specific to the OLR output data set validation or creation and inactive data set deletion.

```
TRACE ID    = X'D9'

word 3 - bytes 1-2 - Error message number as four packed
          decimal digits or as binary 0 if there
          is no error.
          byte 3 - Reserved, 0.
          byte 4 - DCB number for the data set involved.
          The x'80' bit is on if the data set is
          one of the M through V and Y data sets
          (see notes 1 and 2 below).
word 4 - bytes 1-4 - DDIR address.
```

### Notes:

1. When no error has occurred, the error message number in word 3 has a value of binary zero, and there is no further information in the trace entry beyond word 4.
2. For the following error message numbers, there is information that is specific to the particular error:

Unexpected error from system macro instruction:

```
words 5-6 - Macro name.
word 7 - bytes 1-2 - Return code.
          bytes 3-4 - Reason Code.
```

2991 - Output data set validation error:

```
word 5 - Reason code from DFS2991I message text.
```

2992 - Unexpected error from CSI or catalog management, form 1:

```
word 5 - Reason area from CSI or catalog management
word 6 - byte 1 - Reason area type:
          'C' catalog error
          'D' data set error
          'I' CSI call
```

2992 - Unexpected return code from CSI, form 2:

```

word 5 - Return code from CSI call.
word 6 - byte 1 - X'00'
         bytes 2-4 - Reason code from CSI call.

```

#### 2993 - Unexpected device class:

```

word 5 - byte 1 - UCB device class.

```

#### 2994 - Unexpected IDCAMS return code creating a data set:

```

word 5 - Return code from IDCAMS.

```

#### 2995 - Unexpected IDCAMS return code deleting a data set:

```

word 5 - Return code from IDCAMS.

```

#### 2996 - Insufficient DASD space to create a data set:

```

word 5 - bytes 1-2 - SVC 99 error reason
           code.
           bytes 3-4 - Reserved.
word 6 - SMS error reason code.
word 7 - Number of blocks wanted.

```

#### 2998 - Miscellaneous SVC 99 errors creating a data set:

```

word 5 - SVC 99 error reason
           code.
           bytes 3-4 - Reserved.
word 6 - SMS error reason code.
word 7 - Number of blocks wanted.

```

The following table shows the module ID and module subcode values for the X'D9' trace entries that represent the OLR output data set validation or creation process and the inactive data set deletion process.

*Table 29. Module and subcode ID for X'D9'*

Module ID	Module	Subcode	Meaning
A	DFSORA00	X'10'	Data set creation successful
A	DFSORA00	X'11'	Data set creation successful
A	DFSORA00	X'12'	Data set creation successful
A	DFSORA00	X'13'	Data set creation successful
A	DFSORA00	X'14'	Data set validation successful
A	DFSORA00	X'15'	Data set validation successful
A	DFSORA00	X'16'	Data set validation successful
A	DFSORA00	X'20'	Primary index was not a VSAM KSDS
A	DFSORA00	X'21'	VSAM data set did not have REUSE attribute
A	DFSORA00	X'22'	VSAM record length did not match input
A	DFSORA00	X'23'	VSAM control interval size did not match input
A	DFSORA00	X'24'	KSDS key offset or length size did not match input

Table 29. Module and subcode ID for X'D9' (continued)

Module ID	Module	Subcode	Meaning
A	DFSORA00	X'25'	Miscellaneous errors; another trace entry precedes this one
A	DFSORA00	X'C1'	Internal error: invalid DFSORA00 call
A	DFSORA00	X'C2'	Internal error: No data set in X'2930' log record
A	DFSORA00	X'C3'	Invalid input data set
A	DFSORA00	X'C4'	Multi-volume input, but no output data set
A	DFSORA00	X'C5'	Non-DASD data set
A	DFSORA00	X'C6'	Multi-volume data set to be recovered
A	DFSORA00	X'C7'	Non-DASD data set
A	DFSORA00	X'C8'	Data set not usable for OSAM
A	DFSORA00	X'C9'	Data set is a PDS or PDSE
A	DFSORA00	X'D1'	Data set is not VSAM
A	DFSORA00	X'D2'	Data set is not a VSAM KSDS
A	DFSORA00	X'D3'	VSAM data set did not have REUSE attribute
A	DFSORA00	X'D4'	VSAM record length did not match input
A	DFSORA00	X'D5'	VSAM control interval size did not match input
A	DFSORA00	X'D6'	KSDS key offset or length size did not match input
A	DFSORA00	X'D7'	Data set not usable for OSAM
A	DFSORA00	X'D8'	Data set is a PDS or PDSE
A	DFSORA00	X'D9'	Data set is not VSAM
A	DFSORA00	X'E2'	Data set is not a VSAM KSDS
A	DFSORA00	X'E3'	VSAM data set did not have REUSE attribute
A	DFSORA00	X'E4'	VSAM record length did not match input
A	DFSORA00	X'E5'	VSAM control interval size did not match input
A	DFSORA00	X'E6'	KSDS key offset or length size did not match input
A	DFSORA00	X'E7'	Data set not usable for OSAM
A	DFSORA00	X'E8'	Data set is a PDS or PDSE
A	DFSORA00	X'E9'	Data set is not VSAM

Table 29. Module and subcode ID for X'D9' (continued)

Module ID	Module	Subcode	Meaning
B	DFSORA10	X'C1'	Data set error reported by CSI
B	DFSORA10	X'C2'	No error information available from CSI
B	DFSORA10	X'C3'	Catalog error reported by CSI
B	DFSORA10	X'C4'	Unexpected return code 4 from CSI
B	DFSORA10	X'C5'	Unexpected return code 4 from CSI
B	DFSORA10	X'C6'	Unexpected return code from CSI
B	DFSORA10	X'C7'	Unexpected return code from DEVTYPE
B	DFSORA10	X'C8'	Data set not on volume
B	DFSORA10	X'C9'	Unexpected return code from OBTAIN
B	DFSORA10	X'D1'	Unexpected return code from OBTAIN
B	DFSORA10	X'D2'	Unexpected return code from TRKCALC
B	DFSORA10	X'D3'	Unexpected return code 12 from GETDSAB
B	DFSORA10	X'D4'	Unexpected return code from GETDSAB
B	DFSORA10	X'D5'	Unexpected return code from SWAREQ
B	DFSORA10	X'D6'	Invalid data set name
D	DFSORA20	X'C1'	SVC 99 information reason returned
D	DFSORA20	X'C2'	Insufficient space on volume
D	DFSORA20	X'C3'	Data set in use
D	DFSORA20	X'C4'	Insufficient space, SMS
D	DFSORA20	X'C5'	SVC 99 error and SMS reason returned
D	DFSORA20	X'C6'	SVC 99 error code returned
D	DFSORA20	X'C7'	SVC 99 error code returned
D	DFSORA20	X'C8'	Unexpected return code from SVC 99
D	DFSORA20	X'C9'	SVC 99 information reason returned
D	DFSORA20	X'D1'	SVC 99 error code
D	DFSORA20	X'D2'	Unexpected return code from SVC 99
D	DFSORA20	X'D3'	Unexpected return code from IDCAMS



Table 29. Module and subcode ID for X'D9' (continued)

Module ID	Module	Subcode	Meaning
E	DFSORA30	X'C1'	Unexpected return code from IDCAMS
G	DFSORA40	X'C1'	GETMAIN failure

### X'D9' trace entry: fence value before an OLR IPOST/IWAIT

The following figure shows the remaining words of the X'D9' trace entries that are specific to the fence value before an OLR IPOST/IWAIT:

```
TRACE ID    = X'D9'

      word 3 - Can contain the address of the PST
                to be posted.
      words 4-5 - Contains DMBORFEN.
      words 6-7 - Contains DMBAMFEN.
```

The following table shows the module ID and module subcode values for the X'D9' trace entries that represent the fence value before an OLR IPOST/IWAIT.

Table 30. Module and subcode ID for X'D9': fence value before an OLR IPOST/IWAIT

Module ID	Module	Subcode	Meaning
J	DFSORP70	X'01'	IPOST for the OLR I/O fence
J	DFSORP70	X'02'	IWAIT for the OLR action module fence
J	DFSORP70	X'03'	IPOST for the OLR I/O fence
J	DFSORP70	X'04'	IWAIT for the OLR action module fence
L	DFSORP40	X'01'	IWAIT for the OLR action module fence
L	DFSORP40	X'02'	IPOST for the OLR action module fence
M	DFSPCSH0	X'01'	IWAIT for the OLR action module fence
M	DFSPCSH0	X'02'	IPOST for the OLR action module fence
M	DFSPCSH0	X'03'	IWAIT for the OLR action module fence
M	DFSPCSH0	X'04'	IPOST for the OLR action module fence
O	DFSDLOC0	X'01'	IPOST for the OLR I/O fence
O	DFSDLOC0	X'02'	IWAIT for the OLR I/O fence
R	DFSDLR00	X'01'	IPOST for the OLR action module fence
R	DFSDLR00	X'02'	IPOST for the OLR action module fence

Table 30. Module and subcode ID for X'D9': fence value before an OLR IPOST/WAIT (continued)

Module ID	Module	Subcode	Meaning
R	DFSDLR00	X'03'	IWAIT for the OLR action module fence
R	DFSDLR00	X'04'	IPOST for the OLR action module fence
R	DFSDLR00	X'05'	IPOST for the OLR action module fence
R	DFSDLR00	X'06'	IWAIT for the OLR action module fence
R	DFSDLR00	X'07'	IWAIT for the OLR action module fence
R	DFSDLR00	X'08'	IWAIT for the OLR action module fence
S	DFSDVBH0	X'01'	IWAIT for the OLR I/O fence
S	DFSDVBH0	X'02'	IPOST for the OLR I/O fence
V	DFSDVSM0	X'01'	IPOST for the OLR I/O fence
V	DFSDVSM0	X'02'	IPOST for the OLR I/O fence

### X'D9' trace entry : next UOR determination

The following figure shows the remaining words of the X'D9' trace entries that are specific to the next UOR determination.

```
TRACE ID    = X'D9'

word 3 - The total number of UORs performed.
word 4 - The execution span for this UOR.
word 5 - The proposed size for the next UOR.
word 6 - The total bytes moved during this UOR.
word 7 - The total locks held during this UOR.
```

### X'D9' trace entry: OLR command processing

The following figure shows the X'D9' trace entry definitions used by the Online Reorganization (OLR) command processing:

```
TRACE ID    = X'D9'

word 0 - byte 1 - X'D9' Online Reorganization (OLR)
           trace entry.
           byte 2 - Zero - not used.
           bytes 3-4 - Trace sequence number.
word 1 - byte 1 - Module ID.
           byte 2 - Module subcode.
           byte 3 - Module function.
           byte 4 - FREESTOR error return code.
word 2 - Last 4 bytes of the IMS ID (SCDIMSNM+4)
           processing the command.
words 3-4 - Command VERB (INIT, UPD, TERM, and QRY
           if an type-2 command)
words 5-6 - Operation Manager name ('NONOMCMD' if OLR
           type-1 command.
word 7 - Address of storage not freed if FREESTOR
           failure.
```

For all X'D9' trace entries, the module ID, and usually the module subcode as well, indicate both the meaning of the trace entry and the format of the rest of the trace entry.

The following table shows the module ID values in X'D9' trace entries that represent OLR command processing.

*Table 31. Module and subcode ID for X'D9': OLR command processing*

Module ID	Module	Subcode	Meaning
C	DFSORC00	X'00'	OLR type-2 command issued
C	DFSORC00	X'01'	FREESTOR error during INIT error processing
C	DFSORC00	X'02'	FREESTOR error after sending command response
P	DFSORC10	X'00'	OLR type-1 command issued
P	DFSORC10	X'01'	FREESTOR error during INIT processing cleanup

### X'D9' trace entry: OLR start

The following figure shows the remaining words of the X'D9' trace entries that are specific to the OLR start.

X'9D' trace entry - words specific to OLR start

```
TRACE ID    = X'D9'
word 3 - The RBA of the cursor in the second CI
          or block.
word 4 - Unused.
word 5 - Unused.
word 6 - Unused.
word 7 - Unused.
```

### X'D9' trace entry: start of a UOR

The following figure shows the remaining words of the X'D9' trace entries that are specific to the start of a UOR.

X'9D' trace entry - words specific to start of UOR

```
TRACE ID    = X'D9'
word 3 - The first four bytes of the last committed
          cursor.
word 4 - The start time of this UOR.
word 5 - The execution span for this UOR.
word 6 - The time that was waited before this
          UOR started.
word 7 - Unused.
```

### X'D9' trace entry: UOR wait for timer

The following figure shows the X'D9' trace entries that are specific to the UOR wait for timer.

X'9D' trace entry - words specific to UOR wait for timer

```
TRACE ID    = X'D9'
word 3 - Unused
word 4 - The start time of this UOR.
word 5 - The execution span for this UOR.
word 6 - The time that will be waited before
```

the next UOR starts.  
word 7 - Unused.

## X'D9' Trace Entry: OLR Full-Block Logging

TRACE ID = X'D9'

Word 3 - Not used  
Word 4 - AMP  
Word 5 - Buffer prefix (IBFPRF for OSAM or IDABUFC for VSAM)  
Word 6 - Block number for OSAM or RBA for VSAM

The following table shows the module ID and module subcode values for the X'D9' trace entries that represent the OLR logging of a full-block of database changes in a single X'5050' log record rather than the logging of individual database changes in separate log records.

Table 32. Module and subcode ID for X'D9': OLR full-block logging

Module ID	Module	Subcode	Meaning
F	DFSDBH20	X'00'	After full-block logging during OSAM buffer steal
H	DFSDBH30	X'00'	After full-block logging during OSAM buffer purge
S	DFSDBVH0	X'03'	After full-block logging by DL/I buffer handler router before purge
V	DFSDVSM0	X'03'	Before full-block logging during VSAM buffer purge
V	DFSDVSM0	X'04'	After full-block logging during VSAM buffer purge

## X'D9' Trace Entry: OLRK PNDX Store

TRACE ID = X'D9'

Word 3 - OLRK entry address  
Word 4 - Root RBA of the primary index entry.  
Word 5 byte 1 - x'80' entry for primary index entry or VSAM)  
byte 2 - x'80' queued onto chain by OLR

## X'D9' Trace entry: OLRK ILE Store

TRACE ID = X'D9'

Word 3 - OLRK entry address  
Words 4,5 - Key of ILDS record  
Word 6 byte 1 - segment code  
byte 2 - x'40' entry for ILDS entry  
byte 3 - x'80' queued onto chain by OLR

## X'D9' Trace entry: OLRK ILE Write

TRACE ID = X'D9'

Word 3 - OLRK entry address  
Word 4,5 - Key of ILDS record  
Word 6 byte 1 - segment code  
byte 2 - x'40' entry for ILDS entry  
byte 3 - x'40' written to KSDS by VSAM interface  
x'20' deleted off chain by OLR

## X'FB' Trace entry: PSTSTLOR.

INSERT LOGICAL RECORDS (KSDS) to the primary index database or INSERT LOGICAL RECORDS (KSDS) to the ILDS database for HALDB integrated online reorganization. See the previous OLRK ILE/ILDS Store/Write trace entries that are associated with this trace entry.

```
TRACE ID    = X'FB'

Word 3 - Unused
Word 4 - Unused
Word 5 - Unused
Word 6 - Unused
Word 7 - Unused
```

## X'DA' trace entry

The X'DA' trace entry is shown.

### X'DA' trace entry

```
TRACE ID    = X'DA'

word 0 - byte 1 - X'DA' - VSAM JRNAD or UPAD exit
        byte 2 - PST number
        bytes 3-4 - Trace sequence number
word 1 - Word 3 of JRNAD or UPAD parameter list
word 2 - Word 4 of JRNAD or UPAD parameter list
word 3 - Word 5 of JRNAD or UPAD parameter list
word 4 - byte 1 - JRNAD or UPAD code (For an explanation of
        these codes, see note 3 below)
word 4 bytes 2-4 - AMB address
word 5 - PLH STACK or VSAM FOOTSTEPS
        (see notes 1 - 4 below)
word 6 - PLH STACK or VSAM FOOTSTEPS
        (see notes 1 - 4 below)
word 7 - PLH STACK or VSAM FOOTSTEPS
        (see notes 1 - 4 below)
```

### Notes:

1. If the trace contains the PLH stack entries, these entries are the module addresses of the last five VSAM record management modules that had control. If the trace contains the VSAM footsteps, each VSAM footstep is a 1 byte entry which contains VSAM diagnostic information.
2. The PLH footsteps contain 1 byte of VSAM diagnostic data and is passed back to IMS in order from last to first footstep.
3. This information might be valuable to the VSAM support representatives if you need their assistance.
4. For an explanation of these codes, see the following table.

Table 33. JRNAD and UPAD codes for X'DA' trace entry

Code	Code (hexadecimal)	Meaning
JRNAD	0C	Logical records to be shifted in a KSDS
JRNAD	10	Cannot occur
JRNAD	14	Cannot occur
JRNAD	20	Control area split starting in a KSDS
JRNAD	24	Control interval read error
JRNAD	28	Control interval write error
JRNAD	2C	Control interval to be written
JRNAD	30	Control interval to be read and marked exclusive

Table 33. JRNAD and UPAD codes for X'DA' trace entry (continued)

Code	Code (hexadecimal)	Meaning
JRNAD	34	Control interval ownership to be established
JRNAD	38	Control interval to be marked exclusive
JRNAD	3C	Create a new control interval
JRNAD	40	Release exclusive use of control interval
JRNAD	44	Mark control interval prefix invalid
JRNAD	48	Control interval read completed
JRNAD	4C	Control interval write completed
JRNAD	50	CI or CA split
JRNAD	54	Control area reclaim start
JRNAD	58	Control area reclaim end
JRNAD	5C	Control area reclaim interrupted
JRNAD	60	Control area reclaim recovery start
JRNAD	64	Control area reclaim recovery end
JRNAD	68	Start of reuse reclaimed control area
JRNAD	6C	End of reuse reclaimed control area
UPAD	00	Wait requested on I/O or defer
UPAD	04	Post ECB (XMEM only)

## X'DB' through X'FA' trace entry

The X'DB' through X'FA' trace entries are shown.

### The X'DB' through X'FA' trace entries

**TRACE ID = X'DB' - X'FA'**

```

word 0 - byte 1 - X'DB' through X'FA' PSTFNCTN - Buffer handler trace -
              This is the function
              from X'DB' thru X'FA' for which the trace
              was written (see note 1 below).
              byte 2 - PST number (see note 2 below)
              bytes 3-4 - Trace sequence number
word 1 - bytes 1-2 - PSTDMBNM - DMB number. This field
              indicates which DMB is being used.
              The DMB directory (DDIR) gives the
              first DMB.
              byte 3 - PSTDCBNM - DCB number
              byte 4 - PSTRTCDE - Usually
              indicates an error if nonzero. If an error,
              PSTDATA may contain residual data from the
              last call (see note 3 below)
word 2 - byte 1 - PSTTRMID - ID of the module calling the buffer
              handler
              byte 2 - PSTTRMSC - Subcode of the module calling the
              buffer handler
              byte 3 - PSTBHFLG - DL/I buffer handler flags
              byte 4 - PSTSUBCD - Buffer handler internal work byte
word 3 - PSTDSGA - Address of the DSG
word 4 - PSTDATA - Address in real storage of the requested data.
              May point to the last retrieved data address in a call
              (failed abend) (see note 4 below).
word 5 - PSTBUFFA - Address of buffer header. OSAM uses IBFPRF

```

```

DSECT. VSAM uses IDABUFC DSECT.
word 6 - PSTISAMW - Work area
word 7 - PSTBYTNM - Relative byte number of data or
          block number (see note 5 below).

```

#### Notes:

1. The IMS internal function that was being performed.
2. Use only the trace entries with the correct PST number.
3. Shows how the call completed. (X'00' means successful completion.)
4. Shows where the requested data is located in core only if the call completed successfully.
5. The RBA or block number that the call requested.

If the call failed, the PSTDATA field might contain the address of the last segment successfully retrieved.

Example: PSTRTCODE = X'04' (RBA past end of data set).

#### Related reference

“Space management and buffer handler module trace IDs” on page 204

In space management and DL/I buffer handler trace entries, a 1-byte module ID identifies the calling module. A 1-byte subcode identifies the specific call within the module.

## Database function codes

The DL/I function codes are shown.

### Buffer handler function codes

PSTFNCTN is located at PST + X'1C4'.

The following table shows the buffer handler function codes.

Table 34. Buffer handler function codes

Code (hex)	PSTFNCTN	Caller's request function
DB	PSTSRCHP	Search pool for record in range
DD	PSTRELLR	Release record ownership
DE	PSTRSTAT	Retrieve buffer pool statistics
DF	PSTVERFY	Verify VSAM data set
E0	PSTVPUT	Put record to VSAM data set
E1	PSTBKLC	Block Locate
E2	PSTBYLC	Byte Locate
E3	PSTISRCH	Not used
E4	PSTIESDS	Create new ESDS/OSAM LRECL
E5	PSTPGUSR	Write LRECLs for user (PURGE)
E6	PSTBFALT	Mark record altered
E9	PSTFBSPC	Free space in buffer pool (BFPL)
EA	PSTOWTCK	Perform background write function
EB	PSTBYALT	Byte locate and mark altered
EC	PSTBFMPT	Mark buffers empty (BFPL)

Table 34. Buffer handler function codes (continued)

Code (hex)	PSTFNCTN	Caller's request function
ED	PSTCHKPT	Checkpoint
EE	PSTSTAPG	Batch STAE purge at ABEND
EF	PSTERRPG	Purge user for I/O error check
EF	PSTFRWRT	OSAM buffer forced write
F0	PSTSTLBG	Retrieve first LRECL by key
F1	PSTERASE	Erase logical record
F2	PSTSTLEQ	Retrieve by key EQ or GT
F3	PSTSTLCI	Retrieve key EQ or GT - repair CI
F4	PSTSTLIS	Retrieve by key REC to chain from insert logical record (KSDS)
F7	PSTRSIAB	Reset invalidate all buffers trigger
F9	PSTCPYGU	Position by key for Image Copy
FA	PSTCPYGN	Get next record for Image Copy

The following table shows the space management function codes.

Table 35. Space management function codes

Code (hex)	PSTFNCTN	Caller's request function
31	PSTGTSPC	Get space for the segment
32	PSTFRSPC	Free space for the segment
34	PSTGTRAP	Get space close to root anchor PSTBYTNM. Request to turn off bit map bit. Refer to label PSTBTMPF.
35	PSTGZIDL	Get local serialization as a service to LRH00 during /ERE when IRLM as SLM is not there.
36	PSTRZIDL	Release local serialization
B1	PSTGTSPH	Request for space at BLOCK and OFFS B2-B5 are reserved for tracing PSTDATA. PSTOFFSET must point to the location requested.

The following table shows the Open/Close function codes.

Table 36. Open/Close function codes

Code (hex)	PSTFNCTN	Caller's request function
00	PSTOCCLS	This is a close call. This is the absence of PSTOCOPN (X'08') or PSTOCOPN is reset.
01	PSTOCDMB	The DDIR address is in register 2



Table 36. Open/Close function codes (continued)

Code (hex)	PSTFNCTN	Caller's request function
02	PSTOCPCB	The PCB address is loaded from PSTDBPCB to registers 1
04	PSTOCALL	OPEN/CLOSE all DMBs in the system
08	PSTOCOPN	This is an OPEN call
0C		Combine X'04' and X'08'
10	PSTOCDCB	OPEN/CLOSE DCB PSTDSGA = DSG
20	PSTOCLD	Open for load
21	PSTOCDMA	CLOSE and UNAUTHORIZE DMB address of DDIR in register 2
40	PSTOCDSG	OPEN/CLOSE DSG PSTDSGA = DSG
80	PSTOCBAD	The PSTOCBAD (X'80') is set to indicate to the caller that the requested function failed

The following table shows the index maintenance function codes.

Table 37. Index maintenance function codes

Code (hex)	PSTFNCTN	Caller's request function
A0	PSTXMDLT	Index maintenance for segment to be deleted
A1	PSTXMRPL	Index maintenance for segment to be replaced
A2	PSTXMISR	Index maintenance for segment to be inserted
A3	PSTXMUNL	Index maintenance for segment to be unloaded

The following table shows the block loader function codes.

Table 38. Block loader function codes

Code (hex)	PSTFNCTN	Caller's request function
00	PSTRSVDB	Reserve database resources
01	PSTDMBRD	Read DMB from ACBLIB
02	PSTPSBRD	Read PSB from ACBLIB
03	PSTINTRD	READ INTENT and DMB name lists from ACBLIB
04	PSTENQ	PI Processing is required
40	PSTEREFF	Free DB resources (SCHED failed)
80	PSTFREDDB	Free DB resources (termination)

## Buffer handler return codes

The buffer handler return codes, subcodes, and definitions are listed and defined.

### Buffer handler return codes

The following table describes the buffer handler return codes.

*Table 39. Buffer handler return codes*

Return code	Subcode	Definition
PSTCLOK	X'00'	Everything correct
PSTGTDS	X'04'	RBN beyond data set
PSTRDERR	X'08'	Permanent read error
PSTNOSPC	X'0C'	No more space in data set
PSTBDCAL	X'10'	Illegal call
PSTENDDA	X'14'	End of data set encountered — no record returned
PSTNDTFD	X'18'	Requested record cannot be found
PSTNWBLK	X'1C'	New block created in buffer pool
PSTNPLSP	X'20'	Insufficient space in pool.
PSTTRMNT	X'24'	User must terminate, no space in pool.
PSTDUPLR	X'28'	Logical record already in KSDS.
PSTWRERR	X'2C'	Permanent write error.
PSTBUFIN	X'30'	Buffer invalidate.
PSTBIDIN	X'34'	Unable to acquire BID lock.
PSTPDERR	X'38'	Unable to locate DDIR/PDIR entry.
PSTNOSTO	X'3C'	Storage not available.
PSTRRERR	X'40'	CF read and register error.
PSTCURER	X'44'	Space management OLR cursor error.
PSTCLSDS	X'48'	Attempt to access a closed data set.

## Space management and buffer handler module trace IDs

In space management and DL/I buffer handler trace entries, a 1-byte module ID identifies the calling module. A 1-byte subcode identifies the specific call within the module.

### Space management and DL/I buffer handler trace entries

The calling module places the module ID in field PSTTRMID and the subcode in field PSTTRMSC before making the call. The DL/I buffer handler and space management then move these PST fields to the appropriate traces.

The PSTTRMSC module subcodes are 0 through 9 and A through Z. If you need to find the point in the module where the call was made, look for the TIDSCx label that corresponds to the module subcode. Subcode 0 corresponds to label TIDSC0, subcode 1 to label TIDSC1, subcode A to TIDSCA, and so on.

The following table describes the ID, the calling module, and the module function

Table 40. Space management and DL/I buffer handler module trace IDs

<b>ID label</b>	<b>Module ID</b>	<b>Calling module</b>	<b>Module function</b>
TIDDLA00	A	DFSDLA00	Call analyzer
TIDDLAS0	A	DFSDLAS0	Call analyzer SSA
TIDORA00	A	DFSORA00	OLR data set creation/deletion
TIDZDC00	A	DFSZDC00	GSAM Controller
TIDORA10	B	DFSORA10	OLR data set information
TIDZDI00	B	DFSZDI00	GSAM Initialization
TIDORC00	C	DFSORC00	OLR OM command processor
TIDZDI20	C	DFSZDI20	GSAM Initialize GB
TIDDLDC0	D	DFSDLDC0	DELETE/REPLACE
TIDORA20	D	DFSORA20	Create data sets for OLR
TIDZDI30	D	DFSZDI30	GSAM Buffering Initialization
TIDFLST0	E	DFSFLST0	Batch STAE exit
TIDORA30	E	DFSORA30	Delete data sets for OLR
TIDZD110	E	DFSZD110	GSAM BSAM OPEN / CLOSE
TIDL RH00	F	DFSLRH00	LOCK request handler
TIDZD150	F	DFSZD150	GSAM VSAM OPEN / CLOSE
TIDORA40	G	DFSORA40	Performs OLR IDCAMS
TIDSDLB0	G	DFSSDLB0	IRLM status routine
TIDZD210	G	DFSZD210	GSAM BSAM I/O
TIDFXC50	H	DFSFXC50	DB SYNC point
TIDZD250	H	DFSZD250	GSAM VSAM I/O
TIDORP60	I	DFSORP60	OLR interfaces to DBRC
TIDZD310	I	DFSZD310	GSAM Buffer I/O
TIDDDLE1	K	DFSDDLE0	LOAD INSERT function
TIDZSR00	K	DFSZSR00	GSAM Extended checkpoint
TIDDDLE0	L	DFSDDLE0	LOAD INSERT function
TIDORP40	L	DFSORP40	OLR termination and cleanup
TIDZSR10	L	DFSZSR10	GSAM Restart positioned
TIDPCSH0	M	DFSPCSH0	Partitioning Common Services Handler
TIDORP20	N	DFSORP20	OLR cursor and commit manager
TIDDL0C0	O	DFSDLOC0	OPEN/CLOSE
TIDDL0V0	O	DFSDLOV0	LOGICAL/VIRTUAL OPEN
TIDDCAP0	P	DFSDCAP0	Full-Function Data capture
TIDORC10	P	DFSORC10	OLR type-1 command processor

Table 40. Space management and DL/I buffer handler module trace IDs (continued)

ID label	Module ID	Calling module	Module function
TIDDDUI0	Q	DFSDUI0	DUI processor
TIDDLR00	R	DFSDLR00	RETRIEVE function
TIDDHD00	S	DFSDHD00	Space Manager (INIT procedure)
TIDDVBH0	S	DFSDVBH0	Buffer handler router
TIDFRSP0	S	DFSFRSP0	Space Manager (free space)
TIDGGSP0	S	DFSGGSP0	Space Manager (GET space)
TIDMMUD0	S	DFSMMUD0	Space Manager (bit map update)
TIDRCHB0	S	DFSRCHB0	Space Manager (SEARCH block)
TIDRRHM0	S	DFSRRHM0	Space Manager (SEARCH bit map)
TIDRRHP0	S	DFSRRHP0	Space Manager (buffer pool)
TIDTOBH0	T	DFSTOBH0	I/O toleration buffer handler caller
TIDTOCL0	T	DFSTOCL0	I/O toleration DB close
TIDDPSB0	U	DFSDPSB0	PSB generator module
TIDURDB0	U	DFSURDB0	Database Recovery utility
TIDURGP0	U	DFSURGP0	Database Prefix Update utility
TIDURGS0	U	DFSURGS0	Database Scan utility
TIDBACK0	V	DFSBB000	BATCH backout utility
TIDDVSM0	V	DFSDVSM0	VSAM interface
TIDURRL0	V	DFSURRL0	HISAM Reorganization Reload utility
TIDURUL0	V	DFSURUL0	HISAM Reorganization Unload utility
TIDUCPD0	W	DFSUCPD0	UCF DB ZAP processor utility
TIDUCPE0	W	DFSUCPE0	UCF subroutines utility
TIDUICC0	W	DFSUICP0	Online Image Copy utility
TIDDXMT0	X	DFSDXMT0	Index maintenance
TIDRBOI0	Y	DFSRBOI0	Backout RESTART/DYN/BATCH
TIDRDBC0	Z	DFSRDBC0	Database backout control
TIDPSEL0	M	DFSDBH20 DFSDBH30 DFSDVSM0	Partition Selection Handler

### Example of a DL/I trace

The following example shows a DL/I trace. The trace entries show two GHU calls. All calls use Program Specification Table (PST) 01. When activities for different PSTs are intermixed in the trace table, you need to examine only the entries for the PST that you are interested in.

FUNCTION	WORD 0	WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD
7		PAGE 0001						

```

* DL1 TRACE TABLE - DATE 89039 TIME 17450600 SKIP 0000 TOTAL SKIP 00000000 RECORD NUMBER 00000007
ANALYZE CALL AA01008A 00008DE0 GHU 0A0D60 03080800 00004892 00004000
0008F200 .....GHU .....K.....2.
VSAM EXIT DA01008B 0272FA60 06000000 00002400 34B95982 B96E24B9 BCE6BA6E
50B9AE68 .....B.>...W.>&...
PSTBYLCT E201008C 00040100 D2014400 000A101C 0273720C 0272FA60 0274E45E 0000260C
S.....K.....U;...
VSAM EXIT DA01008D 0272FAB0 06000000 00004800 34B95982 B96E24B9 BCE6BA6E
50B9AE68 .....B.>...W.>&...
PSTBYLCT E201008E 00030100 D2014400 000A205C 02739092 0272FAB0 0274E45E 00004892
S.....K.....*...K.....U;...K
VSAM EXIT DA01008F 0272FB50 06000000 00002400 34B95982 B96E24B9 BCE6BA6E
50B9AE68 .....B.>...W.>&...
PSTBYLCT E2010090 00030100 D2014400 000A205C 0273D354 0272FB50 0274E45E 00002754
S.....K.....*...L.....&...U;...
PSTBYLCT E2010091 00030100 D2014400 000A205C 0273D11C 0272FB50 0274E45E 0000251C
S...J....K.....*...J.....&...U;...
PSTBYLCT E2010092 00030100 D2014400 000A205C 0273D354 0272FB50 0274E45E 00002754
S...K....K.....*...L.....&...U;...
PSTBYLCT E2010093 00030100 D2014400 000A205C 0273D11C 0272FB50 0274E45E 0000251C
S...L....K.....*...J.....&...U;...
PSTBYLCT E2010094 00030100 D2014400 000A205C 0273D020 0272FB50 0274E45E 00002420
S...M....K.....*...&...U;...
VSAM EXIT DA010095 0272FAB0 06000000 00004800 34B95982 B96E24B9 BCE6BA6E
50B9AE68 ...N.....B.>...W.>&...
PSTBYLCT E2010096 00030100 D2014400 000A205C 02739092 0272FAB0 0274E45E 00004892
S...O....K.....*...K.....U;...K
VSAM EXIT DA010097 0272FB50 06000000 00002400 34B95982 B96E24B9 BCE6BA6E
50B9AE68 ...P.....B.>...W.>&...
PSTBYLCT E2010098 00030100 D2014400 000A205C 0273D354 0272FB50 0274E45E 00002754
S...Q....K.....*...L.....&...U;...
ANALYZE CALL AA010099 00008DE0 GHU 0A0D60 03280800 00004892 00004000
0008F200 ...R...GHU .....K.....2.
FUNCTION WORD 0 WORD 1 WORD 2 WORD 3 WORD 4 WORD 5 WORD 6 WORD
7 PAGE 0004
VSAM EXIT DA01009A 0272FA60 06000000 00002400 34B95982 B96E24B9 BCE6BA6E
50B9AE68 .....B.>...W.>&...
PSTBYLCT E201009B 00040100 D2014400 000A101C 0273720C 0272FA60 0274E45E 0000260C
S.....K.....U;...
VSAM EXIT DA01009C 0272FAB0 06000000 00004800 34B95982 B96E24B9 BCE6BA6E
50B9AE68 .....B.>...W.>&...
PSTBYLCT E201009D 00030100 D2014400 000A205C 02739092 0272FAB0 0274E45E 00004892
S.....K.....*...K.....U;...K
VSAM EXIT DA01009E 0272FB50 06000000 00002400 34B95982 B96E24B9 BCE6BA6E
50B9AE68 .....B.>...W.>&...
PSTBYLCT E201009F 00030100 D2014400 000A205C 0273D354 0272FB50 0274E45E 00002754
S.....K.....*...L.....&...U;...
PSTBYLCT E20100A0 00030100 D2014400 000A205C 0273D11C 0272FB50 0274E45E 0000251C
S.....K.....*...J.....&...U;...
PSTBYLCT E20100A1 00030100 D2014400 000A205C 0273D354 0272FB50 0274E45E 00002754
S.....K.....*...L.....&...U;...
PSTBYLCT E20100A2 00030100 D2014400 000A205C 0273D11C 0272FB50 0274E45E 0000251C
S...S....K.....*...J.....&...U;...
PSTBYLCT E20100A3 00030100 D2014400 000A205C 0273D020 0272FB50 0274E45E 00002420
S...T....K.....*...&...U;...
VSAM EXIT DA0100A4 0272FAB0 06000000 00004800 34B95982 B96E24B9 BCE6BA6E
50B9AE68 ...U.....B.>...W.>&...
PSTBYLCT E20100A5 00030100 D2014400 000A205C 02739092 0272FAB0 0274E45E 00004892
S...V....K.....*...K.....U;...K
VSAM EXIT DA0100A6 0272FB50 06000000 00002400 34B95982 B96E24B9 BCE6BA6E
50B9AE68 ...W...&.....B.>...W.>&...
PSTBYLCT E20100A7 00030100 D2014400 000A205C 0273D354 0272FB50 0274E45E 00002754
S...X....K.....*...L.....&...U;...

```

### Related reference

[“X'DB' through X'FA' trace entry” on page 200](#)

The X'DB' through X'FA' trace entries are shown.

## Delete/Replace - DL/I trace information

The DELETE/REPLACE module provides meaningful information when abnormal conditions arise leading directly to errors detected by Delete/Replace. This information can be found in the Delete/Replace work area (DLTWA).

Abends initiated by the Delete/Replace module (780, 796, 797, 798, 799, 802, 803, 804, 806, 807, 808, and 811) are traced in the DL/I trace table in a series of entries identified by an X'C4' in the first byte (TRACE FUNCTION CODE).

The first X'C4' entry in the series is provided by the routine that encountered the problem. Each additional entry is provided by the routine that called the routine which in turn wrote the prior entry in the table. Examining these entries in reverse sequence reveals the order in which control was passed from one routine to another.

You can obtain a complete description of the trace table entry for Delete/Replace by assembling the following lines of code:

```
DSECTS  CSECT
        DFSDLDC  FUNC=DSECTS
        END
```

The second word in the Delete/Replace trace entry (called Entry1) uniquely identifies a Delete/Replace abend, and should be used by IBM and customers when submitting APARs for better problem description. In some cases, the Entry1 word from the next trace entry along with the first Entry1 word uniquely identifies the abend. The Entry1 format is:

```
BYTE 0    ID of routine supplying this entry
      1    ID of routine that encountered error
      2    Subcode number of abend if multiples
      3    Internal code for abend
```

Each routine within the Delete/Replace module has a unique 1-byte identification number. The IDs can be obtained from the assembly listings of each of the four source modules which make up the Delete/Replace call. In general they are:

```
X'01' to X'1F'—control and common subroutines (DFSDDLDC0)
X'20' to X'3F'—delete routines (DFSDDLDD0)
X'40' to X'5F'—replace routines (DFSDDLDR0)
X'60' to X'7F'—DLTWA build routines (DFSDDLW0)
```

Use the Entry1 word (the second word in the trace entry) when relating to a Delete/Replace problem in IMS with IBM Software Support.

## Retrieve trace

When an application program executes and a problem occurs (such as damaged data or unexpected results), you can use the Retrieve trace records to see how IMS responded to various calls in the application.

### To set the Retrieve trace

To set the Retrieve trace on, use either of the following methods:

- At initialization time, IMS always turns the Retrieve trace on, except for batch.
- For DB/DC and DBCTL environments, use the **/TRACE SET ON TABLE RETR** command. If you start the DL/I trace by using the **/TRACE SET ON TABLE DLI** command, the Retrieve trace is not automatically turned on.

**Note:** The Retrieve trace cannot be turned on if the DL/I trace is not active.

To determine if the trace is in the dump, check field PSTDLR1 in the PST.

#### **X'0700'**

Indicates the trace is on.

#### **X'07FC'**

Indicates the trace is off.

Field PSTRTVTR of the PST contains the address of the trace table. The byte at PSTRTNDX contains the offset to the next entry in the table.

Every time an application issues a get or insert call, the retrieve module (DFSDLR00) is called. This module is very large and contains many subroutines. By looking at the Retrieve trace, you can see the flow of control through the various subroutines of the retrieve module. As each subroutine calls another, a 2-byte hexadecimal entry is inserted into the trace table. (Byte 1 of the trace entry is the ID of the calling subroutine; byte 2 is the ID of the subroutine that is called.)

The Retrieve trace table is filled from beginning to end. When the table becomes full, tracing starts at the beginning of the table, overlaying each old entry with the new entry.

The first entry in the trace table for a call is X'F1', which is paired with entries: X'2F' (UNQL), X'30' (ROOTISRT), or X'31' (QUAL). The presence of any of these entries indicates the beginning of a trace entry for a retrieve call.

Field JCBRTVTR in the JCB also contains Retrieve trace information. JDBRTVTR contains the offsets to the initial entries in the trace table for the previous four DL/I calls that are associated with a database. The offset to the last call is in the low-order byte, and all offsets are shifted left at the start of each new call.

## Example of Retrieve Trace

The execution of an application results in an error message that indicates damaged data. You can refer to the Retrieve trace table and interpret the entries in order to determine if the problem is caused by:

- An application error
- A database design error
- An internal IMS DB problem
- An IMS system problem related to pointers

If you determine that the problem was caused by an application or database design error, you can use the Retrieve trace to debug and resolve the problem. Otherwise, you can do a keyword search. If the search results in a large number of problems, you can reduce the number of problems by including the name of the subroutine (listed in the following table) that you found in the Retrieve trace table.

*Table 41. The subroutines of the retrieve module (DFSCLR00)*

Hex ID	Subroutine title	Subroutine description
01	BLDVKEY	Builds alternate parent's concatenated key in work area.
02	CSIIGEXT	Reads root based on SSA qualification. If found, GE at level one. If not found, GE at level 0.
03	DIVRSETU	Position (DIV) was not found at this level. Sets off EOC and sets on not posted first child and siblings.
04	ENQDQ	Handles all enqueue and dequeue for retrieve.
05	FNDLPNQ	Final physical root of LP SDB and enqueue it.
06	FORTHISL	Tries to get a segment that satisfies the call at this level or higher.
07	GEEXIT	Publishes GE status code or GB (if root SDBEOC on).
08	GETPSDB	Gets the PSDB of the segment pointed to by JCBACSC.
09	GETPRIME	Issues request for SETL to retrieve next higher root in database.
0A	STLALTPS	Processes request for data by key when an alternate processing sequence is used.
0B	ISRTMPOS	While positioning for insert, a matching segment was found; checks if permissible.
0C	ISRTPOS	Checks for LC insert to locate alternate parent, validate insert, or establish position on alternate twin chain.
0D	ISRTVER	Verifies segment in POSP points to segment in SDBPOSN for HDAM and HIDAM organizations.
0E	KDTEST	Compares value in SSA to value in segment or to key feedback for requalification.
0F	LCPTRTST	Used by CC=L processing to use PCL pointer, if any.

Table 41. The subroutines of the retrieve module (DFSDLR00) (continued)

Hex ID	Subroutine title	Subroutine description
10	LTW	Main driver for requalification to determine the acceptability of current position.
11	LTWLRTN	Used by CC=L processing to see if on last or should use PCL pointer or continue trying (HS).
12	LTWLTST	Used by CC=L processing to find the last segment.
13	MOVEKEY	Moves key from segment to PCB key feedback.
14	MVSEGUSE	Moves the requested segment from the I/O area to the user area.
15	POSTCHLD	Captures child RBNs from input SDB prefix and places in SDBPOSN of dependent SDBs.
16	POSTME	Places search starting position for segment in SDB.
17	POSTTRY	Unqualified GN has found a segment. Posts the position and key.
18	POSTCURP	Moves position from JCB work words into SDB and sets post code.
19	POSTSDBN	Stores location of next segment on chain in JCB work words.
1A	READCUR	Locates current entry in passes SDB.
1B	RDLPCONK	Locates logical parent using its key.
1C	READNXT	Locates next segment from passes SDB.
1D	RDPHYPR	Locates physical pair of segments when passed SDB address of its pair.
1E	RESETMP	Initializes for unqualified call.
1F	RESETQMP	Compares previous call position in level table to current qualification where POS=M.
20	SCDCRSCK	Not first LR crossed and concatenated segment ISRT, builds concatenated key of LC physical parent.
21	SETEOC	Sets EOC in requested SDB. If logical parent enqueues outstanding, locates each and dequeues.
22	SETL	Provides interface to buffer handler for all external data requests.
23	SETLBG	Issues request for SETL to get first root in database.
24	SETPVEOC	Sets EOC on previous SDBs in the hierarchy having the same parent as the passed SDB.
25	SSAEVAL	Examines a segment to see if it satisfies the qualification.
26	SETCHEOC	Sets on SDBEOC of dependent SDBs.
27	STECHISB	Sets SDBEOC on for input SDB and siblings having same physical parent.
28	SETLMIKY	SETL to find key equal to or greater than key determined as minimum value for SSA.
29	STNPHISB	Sets EOC (if in use) and not posted for siblings of input SDB.
2A	THISLVOK	Found one at this level that satisfies the call. Uses it and checks for more levels in call.
2B	UNQGN	Gets next sensitive segment without violating parentage.



*Table 41. The subroutines of the retrieve module (DFSDLR00) (continued)*

<b>Hex ID</b>	<b>Subroutine title</b>	<b>Subroutine description</b>
2C	VLEXP	Processes variable length segment and user data compaction.
2D	WIPEDN	Clears level table below level passed to bottom of table or below entry currently cleared.
2E	XDFTEST	Qualification is secondary index. Checks index entries to validate the position.
2F	UNQL	Master driver for calls without SSAs.
30	ROOTISRT	Routine for positioning to insert at physical root of database.
31	QUAL	Driver for qualified retrievals.
32	HSAMRTN	HSAM I/O interface routine.
33	RETRY	Retry routine for processing option GOT.
34	ISRTCHCK	Use two keys in DSG for root insert.
35	VALIDATE	Validate an EPS.
36	PARTCKRC	Check results of the validate.
37	HDTARGET	PHDAM/HDAM get a key equal or greater.
38	HDNEXT	PHDAM/HDAM get next.
39	HDTARGET	PHDAM/HDAM get a first.
3A	OLRTRACE	Trace IWAIT/IPOST for OLR fence.
F1	INIT	Initialization.

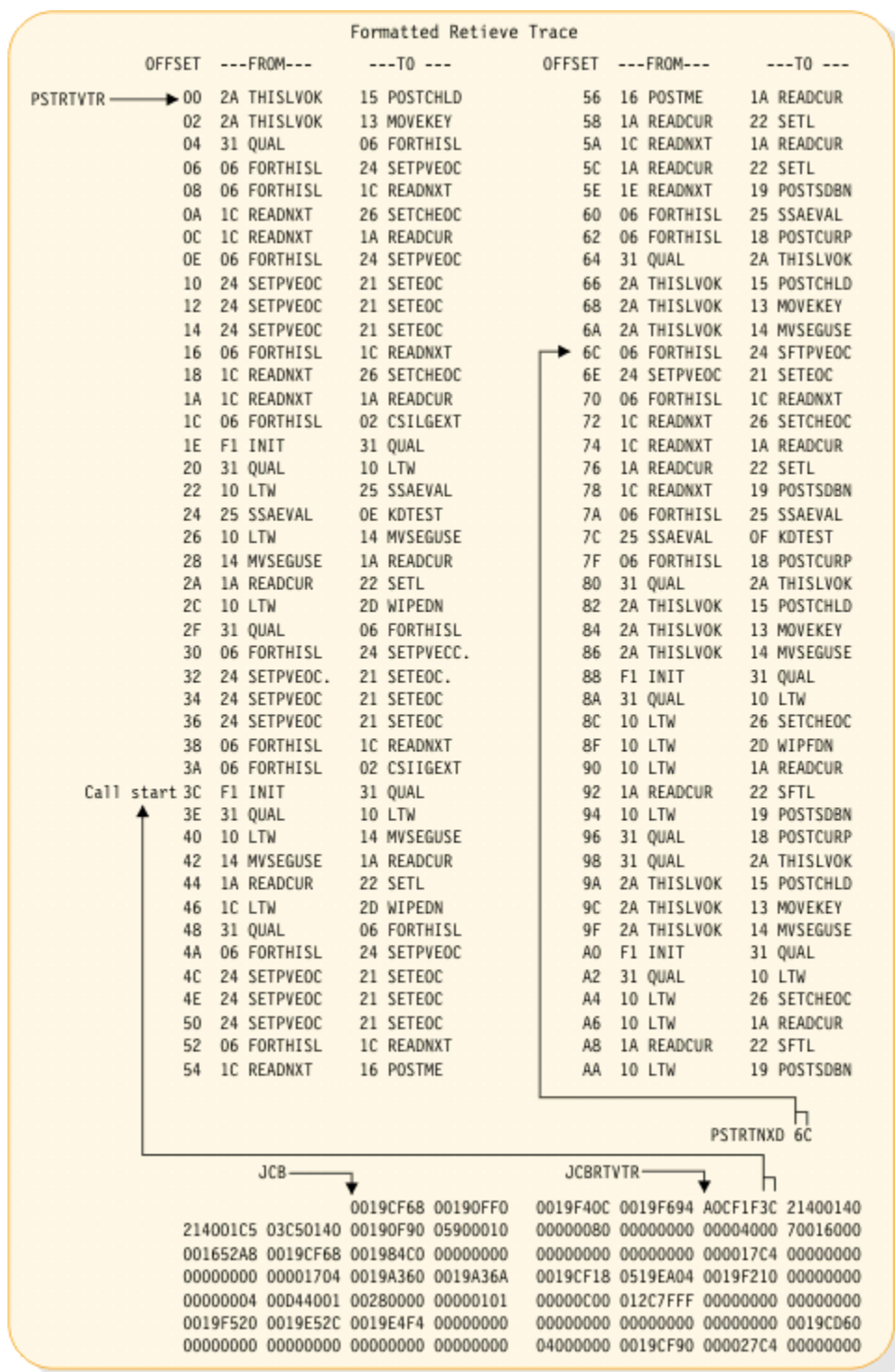


Figure 62. Example of a retrieve trace

## Online Recovery Manager trace

The Online Recovery Manager trace (ORTT) records the control flow that is related to **/RECOVER** command processing.

## Starting the Online Recovery Manager trace

The **/TRACE SET ON TABLE ORTT** command activates the trace and sends the entries to an internal table.

### About this task

You can format the table using the Offline Dump Formatter under IPCS, using either the VERBX command or the IMS Dump Formatter panels.

If a SNAP dump is taken, the table is formatted as part of the IMS dump.

If you add the OPTION LOG parameter to the **/TRACE** command, IMS sends the output to an external data set. You can use the File Select and Formatting utility (DFSERA10) with exit DFSERA60 to format the trace entries.

### Related concepts

[“Formatting IMS dumps offline” on page 512](#)

Two methods are available for formatting IMS dumps offline: interactive formatting, performed through a series of panels which provide formatting choices, and formatting by using JCL.

## Format of the Online Recovery Manager Trace

The Online Recovery Manager trace format, including record, module, explanation and trace subcode are shown.

### Trace Entry: Online Recovery Service Request

Table 42. Trace Entry: Online Recovery Service Request

Record	Module	Explanation	Trace subcode
A001	DFSRWM00 - Database Recovery Manager Master ITASK	Record cut when AWE request is received by DFSRWM00	RWM00 Request

### Trace record 3702 - create data set routine invoke DYA

The format of A001 is shown in the following table.

Table 43. Trace record 3702 - create data set routine invoke DYA

Offset	Type	Length	Description
4	Fixed	2	Requested function(awrwfunc)
6	Bit	2	RWGB_Flags
	1... ..		rwgb_startup_complete
	.1.. ..		rwgb_startup_failure
	..1. ....		rwgb_dlisas
	...1 ....		rwgb_fp_allowed
	.... 1...		*
	.... .1..		rwgb_terminating
	.... .1.		rwgb_record_pipe_alloc
	.... ...1		rwgb_drm_init_complete
	1... ..		rwgb_ORs_installed
	.1.. ....		rwgb_DRF_installed
	..11 1111		*
8	Address	4	Request AWE address

Table 43. Trace record 3702 - create data set routine invoke DYA (continued)

Offset	Type	Length	Description
12	Address	4	Next AWE address
16	Address	4	awrwcecb
20	Fixed	4	Awrwcecb->c_ecb
24	Bit	4	Rwgb_init_flags
	1... ..		rwgb_init_load_1
	.1.. ..		rwgb_init_rwsp
	..1. ....		rwgb_init_load_2
	...1 ....		*
	.... 1...		rwgb_init_ascre
	.... .1..		rwgb_init_route
	.... ..1.		rwgb_init_write
	.... ...1		rwgb_init_read
	1... ....		rwgb_init_cmd
	.1.. ....		rwgb_init_rtb
	..1. ....		rwgb_init_fp
	...1 ....		rwgb_init_dli
	.... 1111		*
	1111 1111		*
	1111 1111		*

## Trace Entry: Online Recovery Service Request Processed

Table 44. Trace Entry: Online Recovery Service Request Processed

Record	Module	Explanation	Trace subcode
A002	DFSRWM00 - Database Recovery Manager Master ITASK	Record is cut when DFSRWM00 completes processing of request	RWM00 Return

## Trace Entry: Online Recovery Service Request Processed

The format of A002 is shown in the following table.

Table 45. Trace record 3702 - create data set routine invoke DYA

Offset	Type	Length	Description
4	Fixed	2	Requested function(awrwfunc)
6	Bit	2	RWGB_Flags
	1... ..		rwgb_startup_complete
	.1.. ..		rwgb_startup_failure
	..1. ....		rwgb_dlisas
	...1 ....		rwgb_fp_allowed
	.... 1...		*
	.... .1..		rwgb_terminating
	.... ..1.		rwgb_record_pipe_alloc
	.... ...1		rwgb_drm_init_complete
	1... ....		rwgb_ORS_installed
	.1.. ....		rwgb_DRF_installed
	..11 1111		*

Table 45. Trace record 3702 - create data set routine invoke DYA (continued)

Offset	Type	Length	Description
8	Fixed	4	Request feedback (awrwfdbk)
12	Address	4	Rwgb_hold_queue
16	Address	4	Awrwcecb
20	Fixed	4	Awrwcecb->c_ecb
24	Bit	4	Rwgb_init_flags
	1... ..		rwgb_init_load_1
	.1.. ..		rwgb_init_rwsp
	..1. ....		rwgb_init_load_2
	...1 ....		*
	.... 1...		rwgb_init_ascr
	.... .1..		rwgb_init_route
	.... ..1.		rwgb_init_write
	.... ...1		rwgb_init_read
	1... ..		rwgb_init_cmd
	.1.. ..		rwgb_init_rtb
	..1. ....		rwgb_init_fp
	...1 ....		rwgb_init_dli
	.... 1111		*
	1111 1111		*
	1111 1111		

## Trace Entry: Online Recovery Service Request Processor Termination

Table 46. Trace Entry: Online Recovery Service Request Processor Termination

Record	Module	Explanation	Trace subcode
A003	DFSRWM00 - Database Recovery Manager Master ITASK	Record is cut when DFSRWM00 is terminating	RWM00 Exit

## Trace record 3702 - create data set routine invoke DYA

The format of A003 is shown in the following table.

Table 47. Trace record 3702 - create data set routine invoke DYA

Offset	Type	Length	Description
4	Fixed	2	Requested function(awrwfunc)

Table 47. Trace record 3702 - create data set routine invoke DYA (continued)

Offset	Type	Length	Description
6	Bit	2	RWGB_Flags
	1... ..		rwgb_startup_complete
	.1.. ..		rwgb_startup_failure
	..1. ....		rwgb_dlisas
	...1 ....		rwgb_fp_allowed
	.... 1...		*
	.... .1..		rwgb_terminating
	.... .1.		rwgb_record_pipe_alloc
	.... ..1		rwgb_drm_init_complete
	1... ..		rwgb_ORs_installed
	.1.. ....		rwgb_DRF_installed
	..11 1111		*
8	Fixed	4	Request feedback (awrwfdbk)
12	Address	4	Rwgb_hold_queue
16	Address	4	Awrwcecb
20	Fixed	4	Awrwcecb->c_ecb
24	Bit	4	Rwgb_init_flags
	1... ..		rwgb_init_load_1
	.1.. ....		rwgb_init_rwsp
	...1. ....		rwgb_init_load_2
	...1 ....		*
	.... 1...		rwgb_init_ascre
	.... .1..		rwgb_init_route
	.... .1.		rwgb_init_write
	.... ..1		rwgb_init_read
	1... ..		rwgb_init_cmd
	.1.. ....		rwgb_init_rtb
	..1. ....		rwgb_init_fp
	...1 ....		rwgb_init_dli
	.... 1111		*
	1111 1111		*
	1111 1111		*

## Trace Entry: Online Recovery Pipe Receive Entry

Table 48. Trace Entry: Online Recovery Pipe Receive Entry

Record	Module	Explanation	Trace subcode
A040	DFSRWM00 - Database Recovery Manager Record Receive Processor	Record is cut when DFSRWM00 is entered	RWPRO Entry

The format of A040 is shown in the following table.

Table 49. Trace record 3702 - create data set routine invoke DYA

Offset	Type	Length	Description
4	Fixed	2	Requested function(awrwfunc)
6	Bit	2	RWGB_Flags
	1... ..		rwgb_startup_complete
	.1.. ..		rwgb_startup_failure
	..1. ....		rwgb_dlisas
	...1 ....		rwgb_fp_allowed
	.... 1...		*
	.... 1..		rwgb_terminating
	.... .1.		rwgb_record_pipe_alloc
	.... ...1		rwgb_drm_init_complete
	1... ..		rwgb_ORS_installed
	.1.. ..		rwgb_DRF_installed
	..11 1111		*
8	Bit	4	Rwgb_init_flags
	1... ..		rwgb_init_load_1
	.1.. ..		rwgb_init_rwsp
	..1. ....		rwgb_init_load_2
	...1 ....		*
	.... 1...		rwgb_init_ascre
	.... 1..		rwgb_init_route
	.... .1.		rwgb_init_write
	.... ...1		rwgb_init_read
	1... ..		rwgb_init_cmd
	.1.. ..		rwgb_init_rtb
	..1. ....		rwgb_init_fp
	...1 ....		rwgb_init_dli
	.... 1111		*
	1111 1111		*
	1111 1111		*
12	Bit	4	Rwgb_read_flags
	1... ..		rwgb_read_terminating
	.1.. ..		rwgb_read_abend
	..1. ....		rwg_read_EODAD
	...1 ....		rwgb_read_open
	.... 1111		*
	1111 1111		*
	1111 1111		*
	1111 1111		*
16	Address	4	Rwgb_read_pipe
20	Address	4	Rwgb_read_buffer

**Trace Entry: Online Recovery Pipe Received Record**

Table 50. Trace Entry: Online Recovery Pipe Received Record

Record	Module	Explanation	Trace subcode
A041	DFSRWM00 - Database Recovery Manager Record Receive Processor	Record is cut when DFSRWM00 receives record from recovery product	RWPRO Record

The format of A041 is shown in the following table.

Table 51. Trace record 3702 - create data set routine invoke DYA

Offset	Type	Length	Description
4	Fixed	2	Requested function(awrwfunc)
6	Bit	2	RWGB_Flags
	1... ..		rwgb_startup_complete
	.1.. ..		rwgb_startup_failure
	..1. ....		rwgb_dlisas
	...1 ....		rwgb_fp_allowed
	.... 1...		*
	.... .1..		rwgb_terminating
	.... .1.		rwgb_record_pipe_alloc
	.... ...1		rwgb_drm_init_complete
	1... ..		rwgb_ORS_installed
	.1.. ....		rwgb_DRF_installed
	..11 1111		*
8	Bit	4	Rwgb_init_flags
	1... ..		rwgb_init_load_1
	.1.. ....		rwgb_init_rwsp
	..1. ....		rwgb_init_load_2
	...1 ....		*
	.... 1...		rwgb_init_ascre
	.... .1..		rwgb_init_route
	.... .1.		rwgb_init_write
	.... ...1		rwgb_init_read
	1... ..		rwgb_init_cmd
	.1.. ....		rwgb_init_rtb
	..1. ....		rwgb_init_fp
	...1 ....		rwgb_init_dli
	.... 1111		*
	1111 1111		*
	1111 1111		*
12	Bit	4	Rwgb_read_flags
	1... ..		rwgb_read_terminating
	.1.. ....		rwgb_read_abend
	..1. ....		rwg_read_EODAD
	...1 ....		rwgb_read_open
	.... 1111		*
	1111 1111		*
	1111 1111		*
	1111 1111		*
16	Address	4	Address of record received



Table 51. Trace record 3702 - create data set routine invoke DYA (continued)

Offset	Type	Length	Description
20	Fixed	2	Record type (logrc_type)
22	Fixed	2	Record subtype (logrc_subtype)

## Online Recovery Manager trace example

An example of the Online Recovery Manager trace output is shown.

### Online Recovery Manager trace example

```

OPTION PRINT 0=5,V=67FA,EXITR=DFSERA60
END
FUNCTION      WORD 0      WORD 1      WORD 2      WORD 3      WORD 4      WORD 5      WORD 6      WORD
7
* OR1 TRACE TABLE - DATE 2004209 TIME 212317790537 OFFSET 028D SKIP 0000 TOTAL SKIP 00000000 RECORD
NUMBER 000022B9
RWM00 Request A0019700 00A63040 0B7143F8 00000000 00000000 040C0000 00000000
09B18BAA
RWPR0 Entry   A040978A 00A13040 E7000000 00000000 00000000 00000000 40404040
09FB93AC
RWM00 Return  A00297A8 00A6B040 00000000 0B7143F8 00000000 00000000 EF000000
0A081EA6
RWPR0 Entry   A04097AD 00BDB040 EF000000 00000000 0C666B78 0CC74FD8 40404040
0A082039
RWPR0 Record  A04197AE 00BDB040 EF000000 00000000 0C666B78 0CC74FD8 40404040
0DD65C41
RWM00 Request A00197B3 00A6B040 0B7143F8 0B7143B0 00000000 040C0000 EF000000
0DD65E3C
RWM00 Return  A00297B4 00A6B040 00000000 0B7143F8 00000000 00000000 EF000000
0DD65E63
RWM00 Request A00197B5 00B4B040 0B7143B0 00000000 00000000 040C0000 EF000000
0DD65E6E
RWM00 Return  A00297B9 00B4B040 00000000 0B7143F8 00000000 040C0000 EF000000
0DD65EDB
RWM00 Request A00197BA 00A6B040 0B7143F8 0B714518 00000000 040C0000 EF000000
0DD65EE7
RWM00 Return  A00297BB 00A6B040 00000000 00000000 00000000 040C0000 EF000000
0DD65F28
RWM00 Request A00197BC 00BEB040 0B714518 00000000 00000000 040C0000 EF000000
0DD65F34
RWM00 Return  A00297BD 00BEB040 00000000 00000000 00000000 040C0000 EF000000
0DD65F40
RWPR0 Record  A04197C5 00BDB040 EF000000 00000000 0C666B78 0CC74FD8 40404040
0EAF4E7B
RWPR0 Record  A04197C9 00BDB040 EF000000 00000000 0C666B78 0CC74FD8 40404040
0EAF549C
RWPR0 Record  A0419843 00BDB040 EF000000 00000000 0C666B78 0CC74FD8 40404040
0EAF96C7
RWPR0 Record  A0419844 00BDB040 EF000000 00000000 0C666B78 0CC74FD8 40404040
0EAF9AE2
RWM00 Request A0019849 00C4B040 0B7143B0 00000000 00000000 040C0000 EF000000
0EAF1AA
RWM00 Return  A002984A 00C4B040 00000000 00000000 00000000 040C0000 EF000000
0EAF1E2
RWM00 Request A001984B 00BEB040 0B7144D0 00000000 00000000 040C0000 EF000000
0EAF1F0
RWM00 Return  A002984C 00BEB040 00000000 00000000 00000000 040C0000 EF000000
0EAF201
RWM00 Request A0019AE6 00A8B040 0B7144D0 00000000 00000000 040C0000 EF000000
20A9CABE
RWM00 Return  A0029AEA 00A8B040 00000000 00000000 00000000 040C0000 EF000000
20A9CDC0
RWPR0 Record  A0419B84 00BDB040 EF000000 00000000 0C666B78 0CC74FD8 40404040
210B71FC
RWPR0 Record  A0419B88 00BDB040 EF000000 00000000 0C666B78 0CC74FD8 40404040 210B7947

```

## Program isolation-related problem analysis

When invalid segment data is retrieved, or an unexpected user abend occurs during concurrent updates to a single database by more than one processing region under the protection of program isolation,

improper enqueue or dequeue logic has been followed in IMS. Tools are available to properly document this occurrence. Correct and adequate documentation might depend on the ability to reproduce the error condition and on the availability of IBM Software Support.

## Limiting locking resources used by an application program

In order to avoid resource problems that can be caused by runaway applications, you can limit the number of locks an application can have by using the LOCKMAX parameter.

### LOCKMAX parameter

The LOCKMAX parameter can be specified on the PSBGEN statement or at execution time. The parameter has the following format: LOCKMAX=*n*, where *n* is a number between 0 and 255. 0 is the default and specifies no maximum lock limit.

### Using the LOCKMAX parameter

The number that is specified indicates units of 1000; for example, a specification of LOCKMAX=5 means that the application cannot have more than 5000 locks at one time.

Assemble the current DSECT describing the log record for the correct field offset because the offset for a field might change. For type X'37' log records, use the DFSXFER DSECT and select records that contain X'30' in the subtype field (such as type X'3778').

**Restriction:** Although the LOCKMAX parameter allows you to limit the amount of resources that are used by an application, it cannot be used to initially specify the amount of resources to be used by an application. Use traditional methods for specifying these resources through the PSB.

### Determining a value for the LOCKMAX parameter

To decide what value to use for the LOCKMAX parameter, analyze over a period of time the X'37', X'41', and X'5937' commit log records to determine the maximum number of locks being held per unit of work by the application.

Each of these log records contains a "high water lock count" or maximum lock count, which is the maximum number of locks held by the application. The X'41' log record shows a zero for the number of locks held, except in DL/I and DBB batch cases involved in block-level data sharing.

#### Related reference

["Log records" on page 469](#)

To diagnose some problems, you need to examine the content of log records to determine what was going on in the system before the problem occurred. By knowing the layout of the log records, you can set up a DFSERA10 job that will produce the specific log records that you need to examine.

### Exceeding the LOCKMAX parameter value

When an application exceeds the value specified for the LOCKMAX parameter, a pseudoabend of type U3301 results. Modules DFSLRHOO and DBFLRHOO set this pseudoabend when the return codes and feedback from either PI or IRLM indicate that the lock request failed because granting the lock would exceed the LOCKMAX parameter value.

## Program isolation trace

The program isolation (PI) trace traces all calls to the IMS enqueue/dequeue module (DFSFXC10) and writes the trace entries to the system log as type X'67FA' records.

### About this task

Entries with IDs X'C7', X'C8', X'C9', X'CA', X'CB', and X'CC' are PI entries.

In a DB/DC environment, you start the trace by entering the /TRACE command at the master terminal operator's console. For batch or DB/DC environments, you specify LOCK=OUT on the OPTIONS statement at system initialization time.

Save the log tape and submit it as APAR documentation. If you cannot ship the log tape with the APAR, you can use the File Select and Formatting Print utility (DFSERA10) with exit DFSERA40 to select and format records related to the problem from the log tape.

You can find the layout of PI trace log record X'67FA' by assembling macro ILOGREC.

In analyzing the trace output, you see not only PI trace information but also lock manager trace information.

#### **Related concepts**

[“DL/I trace formats” on page 167](#)

The figures in this section show the formats of the most commonly used DL/I trace entries. The figures are included to help you understand the DL/I trace entries in order to communicate more effectively with IBM Software Support representatives and to build a valid search argument.

#### **Related reference**

[“Format of X'67' log record” on page 503](#)

A physical log record consists of one or more subrecords. Each subrecord is followed by its associated data.

## **DL/I call image capture program**

The DFSDLTR0 program, which operates independently, traces and records all DL/I calls issued by an application or multiple applications. The output is in a format acceptable as input to the DL/I test program DFSDDLTO.

### **About this task**

This allows you to create the scenario that might have caused the problem. By inserting compare statements requesting SNAP documentation of DL/I control blocks before and after the suspected failure, the information collected helps in diagnosing the problem.

#### **Related tasks**

[“DL/I call image capture” on page 159](#)

DL/I call image capture (module DFSDLTR0) enables you to trace and record all DL/I calls issued by an application program. The trace output is in a format acceptable as input to the DL/I test program DFSDDLTO.

## **Log analysis (database related)**

---

The IMS log is one of the most useful of all IMS service aids. Understanding log records and what information they contain can be very beneficial.

### **Log record analysis**

For all changes, write a copy of the segment before it is changed as well as a copy of the segment after it is changed, if applicable. This process not only facilitates backout and recovery, but it also is useful for diagnosis.

Analyzing log records is helpful whenever you suspect bad data or a pointer problem. Determine where the error is by referring to error messages or to the contents of the dump. When you identify the location of the problem, use the File Select and Formatting utility (DFSERA10) to print the log records for the block in error. Refer to the following table to interpret the contents of the log records. You can determine what changes to the data have been made, and in what sequence the changes were made. This information is helpful in identifying the source of the error.

Sometimes, the error is caused by an internal IMS problem; other times, the error results from incorrect data that is entered by a user or by an application.

To obtain a complete listing of all control blocks, DB, DC, and log records, assemble module DFSADSCT.

CICS puts a header on log records. To obtain the log records when running with CICS, the DD statement pointing to the CICS journal must specify DCB=RECFM=VB. This allows the File Select and Formatting utility to remove the header.

## Example of log analysis

An abend is issued against a database. You have used other diagnostic tools to analyze the call. Now you must look at the database. Follow these steps when looking at the database:

1. Analyze the buffer to identify what seems to be wrong. (See the following figure.) The first indication that something is wrong is usually found in the buffer.
2. Look at the changes to that buffer (block) on the log.
3. Determine if the bad data is actually on the database.
4. If required, determine if the image copy is propagating the bad block.

The following figure shows the general areas of database analysis: Application, Buffer, Database, Image Copy, and Log.

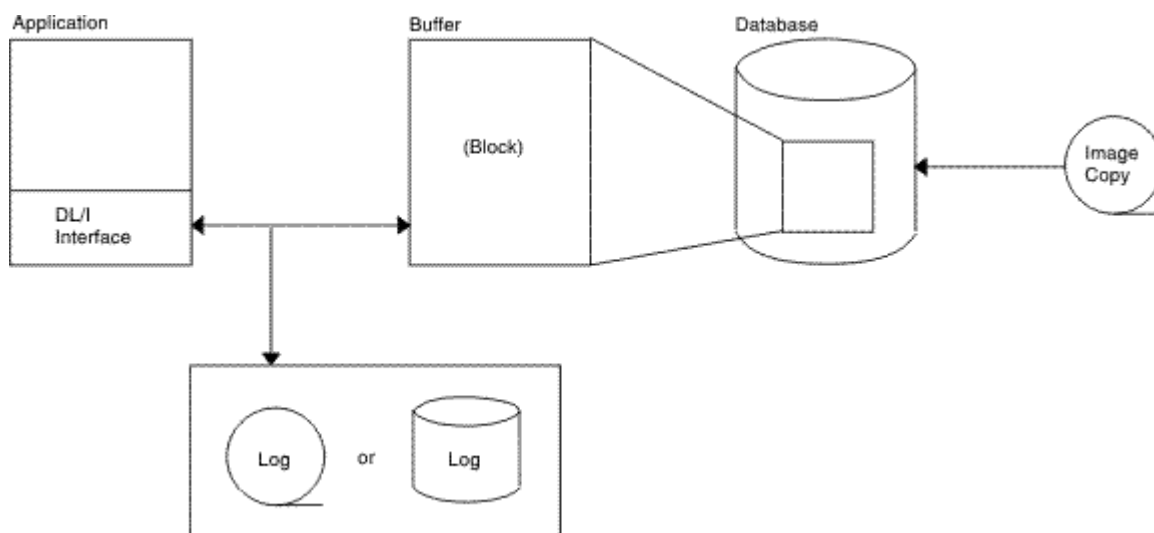


Figure 63. General areas of database (DB) analysis

## Database change log record DSECT

You can use the following table to assist you in the analysis of output from log record type X'50'.

If any differences are detected in the mapping of the DSECT, you can obtain a current copy by assembling the macro ILOGREC.

Table 52. Database change log record DSECT

Offset	Field	Length	Description
<b>DLOGB</b>	DSECT		
00	DLENGTH	2	Length of log record
02	DLOGZZ	2	Zeros for QSAM

Table 52. Database change log record DSECT (continued)

Offset	Field	Length	Description
04	DLOGCODE	1	Log record type
05	DLOGSCDE	1	Log record subrecord (X'50' X'51' X'52')
06	DLOGPSTN	2	PST number
08	DLOGRTKN	16	Recovery token
18	DLOGSTCK	8	CPU store clock (STCK)
20	DLOGVIMS	1	DLOG IMS Version/Release: X'82' Version 8 or later
28	DLOGDBF1	1	Flag 1 X'80' Record written during backout X'40' Record from DB/DC X'20' Record from batch region X'10' New date/time from DFSFTIM0 X'08' Commit each GU call (Mode=SNGL) X'04' First log record this sync interval X'02' First log record of a segment X'01' Last log record of a segment
29	DLOGDBF2	1	Flag 2 X'80' Database is nonrecoverable X'40' KSDS ERASE prohibited X'20' Bit map update for lock tracking  X'08' PHIDAM primary index; no REDO X'04' DLOGSEQ has update sequence number X'02' OLR non-backoutable; cursor not active yet X'01' OLR ITASK
2A	DLOGDBOR	1	Database organization X'70' DEDB direct organization X'40' DL/I HDAM database X'20' DL/I HIDAM database X'10' Data entry database (DEDB) X'08' Primary or secondary index database X'04' HISAM or SHISAM database
2B	DLOGDSOR	1	Data set organization X'80' VSAM access method X'40' OSAM access method X'08' Entry sequenced data set X'04' Key sequenced data set
2C	DPGMNAME	8	PSB name

Table 52. Database change log record DSECT (continued)

Offset	Field	Length	Description
34	DDBDNAME	8	Database name
3C	DDSID	1	Data set ID (DCB number) X'80' <ul style="list-style-type: none"> <li>• When this high order bit is on, then this DCB number represents one of the M-through-V or Y data sets.</li> <li>• When this high order bit is off, then this DCB number represents one of the A-through-J or X data sets.</li> </ul>
3D	DDSID2	1	For ARID
3E	DLOGSLVL	1	Database share level (for DBRC-registered databases)
3F	DLOGCALL	1	Describe DL/I call issued by application program X'80' ISRT call X'40' REPL call X'20' DLET call X'10' ROLL/ROLB/ROLS call (backout)
40	DLOGRBA	4	OSAM RBN or VSAM RBA (LRECL)
44	DLOGBLK0	2	Offset of RBA within block
48	DLOGSEQ	4	Update the sequence number when X'04' flag is on in DLOGDBF2
4C	DLOGXTOF	2	Database extension section offset (not used) <a href="#">“1” on page 226</a>
4E	DLOGDSOF	2	Data sharing section offset <a href="#">“1” on page 226</a>
50	DLOGIDOF	2	RACF userid offset <a href="#">“1” on page 226</a>
52	DLOGTKOF	2	Tracking (XRF) section offset <a href="#">“1” on page 226</a>
54	DLOGDLOF	2	DL/I call section offset (not used) <a href="#">“1” on page 226</a>
56	DLOGKYOF	2	Key data section offset <a href="#">“1” on page 226</a>
58	DLOGSPOF	2	Space management section offset <a href="#">“1” on page 226</a>
5A	DLOGUNOF	2	UNDO data offset <a href="#">“1” on page 226</a>
5C	DLOGREOF	2	REDO data offset <a href="#">“1” on page 226</a>
60	DDATE	4	Date in the format YYYYDDDF
64	DTIME	6	Time in the format HHMMSSTHMIJU
6A	DZONE	2	Offset to local time

Table 52. Database change log record DSECT (continued)

Offset	Field	Length	Description
<b>Data sharing section (DLOGDSHUR DSECT)</b>			
00	DLOGDSSN	4	Data set sequence number (DSSN)
04	DLOGLSN	6	Lock sequence number (LSN)
0A	DLOGUSID	4	Update Set ID (USID)
<b>RACF/SIGNON userid (DLOGID DSECT)</b>			
00	DLOGUSER	8	RACF userid
<b>Buffer and lock tracking for DL/I in XRF-capable systems (DLOGTRCK DSECT)</b>			
00	DLOGPOOL	2	Pool size for buffer tracking
02	DLOGBUFF	2	Buffer number for buffer tracking
04	DLOGHASH	4	Root hash value
08	DLOGLOCK	4	Lock value
0C	DLOGLFL1	1	Change logger lock flag X'80' Log record is for root segment X'40' Log record is for dependent segment X'20' Bypass reacquiring restart locks X'10' Get bid lock on DDATAID X'08' Function is erase X'04' Index maintenance X'02' Organization is SHISAM X'01' Hash is for logical parent
0D	DLOGLFL2	1	Reserved
0E	DLOGDBDN	8	DBD name
16	DLOGSKID	4	Task ID
<b>KSDS key data section (DLOGKEY DSECT)</b>			
00	DLOGKYF1	1	X'40' KSDS key X'20' Key is being erased
02	DLOGKLEN	2	Length of key
04	DLOGKDAT	variable	Key data
<b>Space management section for HD inserts and deletes (DLOGSPCE DSECT)</b>			

Table 52. Database change log record DSECT (continued)

Offset	Field	Length	Description
00	DLOGSPF1	1	Space management flags X'40' Demand space request X'20' Get free space request (ISRT) X'10' Free space request (DLET)
02	DLOGSOFF	2	Offset of space management request
04	DLOGSLEN	2	Length of space management request
<b>UNDO/REDO data section (DLOGDATA DSECT)</b>			
00	DLOGDFLG	1	X'80' Last data element in this section X'40' Data is compressed using z/OS services
01	DLOGDFUN	1	Describe physical function being logged by this request  X'80' Physical insert X'40' Physical replace X'20' Physical delete X'10' Space management create X'08' Free space element
02	DLOGDOFF	2	Offset of data in buffer
04	DLOGDLEN	2	Length of data (DLOGDDAT)
06	DLOGDDAT	variable	Variable length data
00		2 variable	Compressed data format in DLOGDDAT Expanded data length Compressed data
	DBCKCHN	6	Back chain <sup>"2"</sup> on page 226
	DBLGSEG	8	Logical logger sequence number <sup>"2"</sup> on page 226

**Notes:**

1. To find each section, add the offset to the beginning of the log record.
2. The log back chain and logical logger sequence number are at the end of the log record.



## Sequential buffering service aids

---

When you receive a message or abend that indicates a problem with sequential buffering (SB), several diagnostic tools are available. Some of these tools are useful for diagnosing other IMS database-related problems.

### Useful tools for diagnosing IMS data-related problems

- DL/I trace table entries
- Dump formatting of IMS control blocks
- SNAPs of IMS control blocks during pseudoabends

The //DFSSTAT statistics report is also a useful tool for evaluating a potential sequential buffering problem.

SB provides additional problem determination tools, which are described in this section:

- SBSNAP and SBESNAP options
- SB IMAGE CAPTURE option and the SB Test program (DFSSBHD0 utility)
- The SB COMPARE option

For most invocations of SB pseudoabend buffer handler functions, entries in the DL/I trace tables are provided. The SB trace table entries are:

#### **X'6F'**

Search/read by RBN

#### **X'6C'**

Refresh SB buffer after a write

#### **X'69'**

Invalidate SB buffers

#### **X'6A'**

Evaluate SB buffering

#### **X'6B'**

Describe why SB was or was not used for the application

In addition, the X'D1' DL/I trace table entry created by DFSNOTB0 contains some information about invalidation of SB buffers.

### Related concepts

[“DL/I trace” on page 165](#)

The DL/I trace table is a combined trace consisting of entries from DL/I calls, the DL/I buffer handler, DL/I OPEN/CLOSE, HD space management, lock activity (using PI or IRLM), OSAM, DFP interface, HALDB OLR trace, and ABENDU0427.

[“SYS - System service aids” on page 469](#)

Various tools, utilities, and traces can help you analyze IMS system problems.

[“Dump formatting options” on page 551](#)

General formatting options are available for IMS and z/OS memory dumps.

[“SNAPs on exceptional conditions” on page 158](#)

IMS produces SNAPs of DL/I control blocks on the IMS log (or the CICS system log).

## SBSNAP option

The SBSNAP option generates a SNAP of the relevant control blocks and areas involved in the calls of the OSAM buffer handler to the SB buffer handler.

### About this task

Use the SBSNAP option when you receive a message saying that either Sequential Buffering:

- Has been activated when you do not expect it to be
- Has not been activated when you expect it to be activated

IMS monitors the physical I/O being done by individual applications and then uses SB I/O reference pattern-analysis algorithms to select the most efficient method of data access. When you suspect a problem with these algorithms, the SBSNAP option provides diagnostic output you can analyze. The information that is provided in the SNAPs provides an indication of why SB chose between issuing a random read of one single block and a sequential read of multiple consecutive blocks.

As a result of analyzing SBSNAP output, you might realize you need to reorganize the database, redesign the database, or set different thresholds for the SB definition. The SBSNAP option is also useful when you are tuning your usage of SB after you've installed IMS or migrated to a new version.

To activate the SBSNAP option, provide a SBSNAP control statement in the //DFSCTL file. (See *IMS Version 15.5 System Definition* for detailed information.)

SNAPs are written to the IMS log as type X'67EE' records. You can format and print these records by using the File Select and Formatting Print utility (DFSERA10) with exit routine DFSERA30.

The SBSNAP option often creates a very large amount of SNAP output. You might therefore decide to limit the SNAP to a specific short period of the application execution. To limit the SBSNAP option to one period of the application execution, use the START and STOP keywords on the SBSNAP control statement. The syntax for these keywords is:

```
START=n STOP=
```

where *n* and *m* are the numbers of calls made to the SB buffer handler by the executing application.

To determine what values to use for *n* and *m*, look at the SPBSTCNB fields in the DL/I trace table and, if available, SNAP dumps (created by SBESNAP option). For each application, IMS maintains these call numbers in the SBPST, in the SPBSTCNB field. This field is periodically written to:

- The X'6A' DL/I trace table entry
- SNAPs that are created by the optional SBESNAP facility

Specifying START=*n* activates the SBSNAP option during the *n* the call to the SB buffer handler; specifying STOP=*m* deactivates the SBSNAP option during the *m*the call to the SB buffer handler.

## SBESNAP option

You activate the SBESNAP option by providing a SBESNAP control statement in the //DFSCTL file.

### About this task

The SBESNAP option SNAPs the control blocks that are necessary for understanding the reason the SB evaluation logic did or did not recommend use of SB.

SNAPs are written to the IMS log as type X'67FD' records. You can format and print these records by using the File Select and Formatting Print utility (DFSERA10)with exit DFSERA30.

## Testing algorithms using the SB Test utility (DFSSBHDO)

When you run the SB IMAGE CAPTURE option of the DFSSBHDO utility, the same sequential buffering (SB) handler call sequence (which is issued during the processing of a specific application) captures on the IMS log all internal IMS calls to the SB buffer handler. You can use this utility to investigate and test behaviors of various algorithms.

The DFSSBHDO utility with the SB IMAGE CAPTURE option is useful for investigating:

- The SB I/O reference pattern analysis algorithms
- The impact of changes to user-specifiable SB parameter values (the BUFSETS parameter value)

Running the same SB buffer handler call sequence multiple times is useful in the following situations:

- You need to use the SBSNAP option but do not know when to Start or Stop the SBSNAP option.
- You want to experiment with different SB algorithm parameters and observe the impact of these changes on the //DFSSTAT statistics.
- You want to test changes to the SB I/O reference pattern analysis algorithms and observe the impact of these changes on the //DFSSTAT statistics.

You activate the SB IMAGE CAPTURE option by providing a SBIC control statement in the //DFSCTL file.

## SB COMPARE option

You activate the SB COMPARE option when you suspect that the SB buffer handler returns incorrect block images into the buffers of the OSAM buffer handler. When you activate the SB COMPARE option, the SB buffer handler performs a self-check to see whether this suspicion is correct and provide problem determination information when the SB buffer handler really returns incorrect data.

### About this task

When the SB COMPARE option is active, the SB buffer handler compares each block image that is returned to the OSAM buffer handler with the corresponding block image that is stored on DASD. When the comparison detects a mismatch between the two block images, the SB buffer handler invokes the SNAP-specific function, which produces a SNAP that describes the mismatch and contains:

- Relevant buffers and control blocks of DL/I
- The OSAM buffer handler
- The SB buffer handler

Module DFSSBSN0 then issues an abend (for batch) or a pseudoabend (for DB/DC, DBCTL, and CICS).

**Exception:** In a data-sharing environment, the SB buffer handler sometimes returns a back-level block image to the OSAM buffer handler. Therefore, in data sharing, the SB COMPARE option does not issue abends or pseudoabends.

You activate the SB COMPARE option by providing a SBCO control statement in the //DFSCTL file. Refer to *IMS Version 15.5 System Definition* for more information on the SBCO control statement in the //DFSCTL file.

SNAPs are written to the IMS log as type X'67EF' records. You can format and print these records by using the File Select and Formatting Print utility (DFSERA10) with exit DFSERA30.

## GSAM control block dump - DFSZD510

When a GSAM error occurs or when a DUMP or SNAP call is issued to a GSAM PCB, a formatted dump of the GSAM control blocks is written to the file that is defined as DDNAME IMSERR or SYSPRINT. You can use this GSAM control block dump (named DFSZD510) to diagnose GSAM problems.

### Example of when to use a GSAM control block dump

Some situations in which you would use a GSAM control block dump are when you receive a message identifying a GSAM error, or when you are having problems repositioning a GSAM data set when you are trying to restart an application that previously failed.

The following control blocks are included in the dump:

- GSAM pointer table (GPT)
- GSAM load table (GLT)
- GSAM data set control block (GB)
- GSAM queue control block (GQCB)
- GSAM buffer control block (GBCB)
- IMS program communication block (PCB)
- Data event control block (DECB)
- Request parameter list (RPL)

To produce a DSECT that shows the layout of the GSAM control blocks, assemble macro IGLI.

### Example of a formatted GSAM control block dump

In this example, key eye catchers are shown (in bold text) so that those sections of the dump are easier to find. Each problem is different, but diagnosing almost all GSAM problems involve at least these key areas of the dump.

#### About this task

```

                                * * * GSAM CONTROL BLOCKS DUMP * * *
07A010 GSAM POINTER TABLE
    GPTCNTLR 800271D8  GPTERROR 00  GPTFC GHU  GPTF1 0007A220  GPTF2 0004D50C
    GPTF3 00000000  GPTF4 00000000  GPTGB 0007A0C0  GPTGLT 0007A060  GPTHSEVC 08
    GPTMAIN 00001350  GPTMODE 00  GPTPCB 0007A090  GPTPMBLK 00009C90  GPTPSBL 00005540
    GPTRS1 00009C58  GPTSAVE 00079000  GPTSZS 0800  GPTSZW 0800  GPTTRACE 00009DF0
    GPTTYPE 00  GPTWORK 00079800
07A060 GSAM LOAD TABLE
    GLTBSAM 8007B0C0  GLTBUFIO 00000000  GLTCBDM 8007CCB0  GLTCNTLR 800271D8  GLTGPT 0007A010
    GLTOPENB 80032118  GLTOPENV 00000000  GLTVSAM 00000000
07A090 IMS PGM CONTROL BLK
    DBPCBDBD DBD37877  DBPCBFLG 02  DBPCBGB 0207A0C0  DBPCBLEV 0000  DBPCBMKL 0000000C
    DBPCBNSS 0000FFFF  DBPCBPRO L  DBPCBSFD  DBPCBSTC AM  DBPCBURL 00000000
    DBPCBRRA 00000000 00000000
07A0C0 GSAM BLOCK
    GBBFPORT 0000  GBBLKLEN 0000  GBBLKOH1 0001  GBBLKOH2 FFE0  GBBLKREF 00000401
    GBBLKSI 01C2  GBBQCB 00000000  GBBUFFER 00064CA0  GBBUFFSW 08  GBBUFNO 01
    GBCDISP 0000  GBCHAIN 0007A220  GBCRTNCD 0028  GBCSEVCD 08  GBCTRS 0000
    GBDCBPTR B007A178  GBDDNAME GS378770  GBDECB 0007A1D4  GBDEVTYPE 208E  GBDSORG 81
    GBERRSW 00  GBEXLST 8607BEA2  GBGPTPTR 0007A010  GBGSAMSW 50  GBIOAREA 00093000
    GBLENLEN 0000  GBLRECL 0096  GBMAXTR BB60  GBMINRCL 0000  GBVOL 0001
    GBOPENSW D1  GBPCBPTR 0007A090  GBPRTNCD 0000  GBRECFM 90  GBRECPTR 00064D36
    GBREQC 6201  GBREQP 0020  GBREQU 6201  GBRPLPTR 0007A1D4  GBRRAPTR 00091888
    GBSERA 0000  GBSERR 0600  GBSUPVR 00  GBTRCALC BB60  GBTRECL 0096
    GBURTNCD 00  GBVLSQ 0001
07A178 DATA CONTROL BLOCK (DCB)
    DCBBFTEK 06  DCBBLKCT 04FDBEBC  DCBBLKSI 01C2  DCBBUFEB 01064C98  DCBBUFL 01C2
    DCBBUFNO 01  DCBBUF0F 00  DCBCEKSI 00C894B0  DCBCIND1 00  DCBCIND2 00
    DCBCNTLRL 00D57F48  DCBDDNAM 00  DCBDEBAD 009D1554  DCBDEN AD  DCBDEVT 2E
    DCBDSORG 4000  DCBDVTBA FDBEBC  DCBE0BR 01D57650  DCBE0BW 00D57650  DCBEODA 07BEB8
    DCBEODAD 0607BEB8  DCBEXLST 9007A110  DCBFDAD1 00000000  DCBFDAD2 05000104  DCBFUNC A0
    DCBIFLG C8  DCBIFLGS 00  DCBIOBA 410050F0  DCBIOBAD 00005088  DCBIOBL 09
    DCBKEYCN 00  DCBKEYLE 00  DCBLRECL 0096  DCBMACR 97D8  DCBMACRF 2424
    DCBMODE 00  DCBNCP 01  DCBODEB 00005088  DCBOFFSR 30  DCBOFFSW 30
    DCBOFLGS 92  DCBOPTCD 00  DCBPRTOV AD  DCBPRTSP 00  DCBREAD 92C897D8
    DCBRECFM 90  DCBREL 2EADA0  DCBRELAD 00000000  DCBRELB 002EADA0  DCBSTACK 00
    DCBSVCXL 00005088  DCBSYNA 07BF68  DCBSYNAD 0907BF68  DCBTIOT 007C  DCBTRBAL ADA0
    DCBTRTCH 00  DCBWCP 01  DCBWCP0 30  DCBWRITE 92C897D8
```

[illegible]

In this example, an unformatted GSAM control block dump is shown without key eye catchers.

Chapter 8. DB - Database service aids **231**

*.....0...0....PURG.....*	0007A040 00079000 08000800 00001350 00009C58	0007A0C0 00005180 00000000 00000000
*.....*	0007A060 800271D8 0007A010 00000000 80032118	8007B0C0 00000000 00000000 00000000
*...Q.....*	0007A080 00000000 8007CCB0 00000000 00000000	C4C2C4F3 F7F8F7F7 00004040 D3404040
*.....DBD37877.. L *	0007A0A0 0207A0C0 40404040 40404040 0000000C	0000FFFF 00000000 00000000 00000000
*.....*	0007A0C0 0007A220 00000401 00010000 00010096	00000096 01C20000 0000208E 0001FFEO
*.....B.....*	0007A0E0 40400000 00289081 06000000 02830283	12020000 5008D101 00000000 00000000
*.....J.....*	0007A100 0007A010 0007A090 B007A178 0007A1D4	8607BEA2 00093000 00091B88 00000000
*.....M.....*	0007A120 00064CA0 00064D36 BB60BB60 00000000	C7E2F3F7 F8F7F7D6 00000000 00000000
*.....GS378770.....*	0007A140 00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000
*.....*	LINE 0007A160 SAME AS ABOVE	
0007A180 00050001 04FDBEBC 002EADA0 01064C98	01C24000 00005088 0607BEBA 9007A110	
*.....B.....*	0007A1A0 007C2424 009D1554 92C897D8 00C894B0	0907BF68 000001C2 30013030 410050F0
*.....H.Q.H.....B.....0*	0007A1C0 01D57650 00D57650 00000096 00D57F48	00000000 7F000000 00200000 B007A178
*.N...N.....N.....*	0007A1E0 00064CA0 000050F8 00000000 00000000	C4C2C4F3 F7F8F7E7 00004040 C7404040
*.....8.....DBD3787X.. G *	0007A200 0207A220 40404040 40404040 0000000C	0000FFFF 00000000 00000000 00000000
*.....*	0007A220 0007A0C0 00000000 00000000 00010096	00000000 01C20000 0000208E 0001FFEO
*.....B.....*	0007A240 40400000 00009081 C2000000 02830283	00200000 0000C001 00000000 00000000
*.....B.....*	0007A260 0007A010 0007A1F0 8007A2D8 0007A334	00000000 00000000 00000000 00000000
*.....0...Q.....*	0007A280 00000000 00000000 BB60BB60 00000000	C7E2F3F7 F8F7F7D6 00000000 00000000
*.....GS378770.....*	0007A2A0 00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000
*.....*	LINE 0007A2C0 SAME AS ABOVE	
0007A2E0 00000000 00000000 00000000 00000000	01C24000 00000001 00000001 90000000	
*.....B.....*	0007A300 C7E2F3F7 F8F7F7D6 02002424 00000001	00000001 000001C2 00000000 00000001
*GS378770.....B.....*	0007A320 01000001 00000001 00000096 00000001	00000000 00000000 00800000 00000000
*.....*	0007A340 00000000 00000000 00000000 00000000	
*.....*	0007A700 84000000 18800000 000300CC FFFB26B4	00000000 00000000 00000000 00000000
*.....*	0007A720 0004D0A8 00080008 0004D0B0 00100010	0004D0B2 00020002 0004D0B4 0004D0B8
*.....*	0007A740 0004D0BC 0004D0C0 00080008 0004D0C8	00080008 00000000 40404040 40404040
*.....H.....*	0007A760 10004040 40404040 40404040 40404040	40404040 40404040 40404040 40404040
*..*	0007A780 00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000
*.....*	0007A7A0 00000000 00000000 00000000 00000000	00000000 00000000 0004D114 00009DF0
*.....J.....0*	0007A7C0 009B6020 00000000 00000000 00093000	0004D54 00000000 00000000 00029E50
*.....*	0007A7E0 00000000 00000000 00000000 00000000	C4C4D3E3 F0F1F340 D3D6C1C4 40404040
*.....DDLTO13 LOAD *	0007A800 00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000
*.....*	LINES 0007A820-0007A860 SAME AS ABOVE	
0007A880 00000000 00008500 0004D0A8 00000000	00000000 080073E8 00000000 00000000	
*.....Y.....*	0007A8A0 00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000
*.....*	0007A8C0 00000000 080073E8 0004D050 00000000	00000000 00000000 00026B70 000641D8
*.....Y.....Q*	0007A8E0 00000000 00000000 00000000 00000000	00000000 00000000 00000000 00053040
*.....*	0007A900 00000000 00000000 000300CC 00000000	00000000 00000000 00000000 00000000
*.....*	0007A920 00000000 00000000 0004D94C 00000000	00000000 00000000 00000000 00000000
*.....R.....*	0007A940 00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000
*.....*		

0007A960 00000000 00000000 00000000 00000000	00009DA0 00000000 00010C00 0004D350
*.....L.*	
0007A980 00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000
*.....*	

LINE 0007A9A0 SAME AS ABOVE	
0007A9C0 84000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000
*.....*	
0007A9E0 00000000 00000000 00000000 00000000	00000000 0088266F 11173205 02000000
*.....*	
0007AA00 0004D050 00000000 00000000 00000000	00000000 00000000 00000000 00000000
*.....*	
0007AA20 00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000
*.....*	
0007AA40 00060040 00000000 00000000 00000000	00000000 00000000 00000000 00000000
*.....*	
0007AA60 00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000
*.....*	
0007AA80 00000000 00000000 00000000 00000000	00000000 00000000 00000000 0002CEE6
*.....W*	
0007AAA0 00000000 00000000 00000000 0005713F	00057040 00000000 07FC4040 40404040
*.....*	
0007AAC0 00000000 00000000 00000000 00057340	00000000 00057140 00000000 00000000
*.....*	
0007AAE0 00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000
*.....*	
LINE 0007AB00 SAME AS ABOVE	
0007AB20 00000000 00000000 0004DD64 00000000	00000000 00000000 0004DD90 00000000
*.....*	
0007AB40 00000000 00000000 00000000 08000000	000C0000 00000000 00000000 00000000
*.....*	
0007AB60 00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000
*.....*	
LINE 0007AB80 SAME AS ABOVE	
0007ABA0 00000000 00000000 00000000 00000000	00000000 00000000 00000000 80049040
*.....*	
0007ABC0 00009DA8 0004D554 4003AAE2 0002AF6C	00009DA0 00009DF0 00009C90 00009DF0
*.....N..S.....0..0*	
0007ABE0 0008FD64 00009DF8 00093000 00000000	00093000 00090548 00000004 0004D050
*.....8.....*	
0007AC00 0003A7A0 00000000 0004D50C 0004D59C	FF02AFD2 0002BCEC 00009DA0 0004D050
*.....N...N...K.....*	
0007AC20 00009C90 0004D50C 00009E38 00009D90	00093000 00000000 80093000 080073E8
*.....N.....Y*	
0007AC40 00000004 0004D050 0002AF6C 00000000	0004D554 0004D5E4 FF02D5C6 00034100
*.....N...NU..NF....*	
0007AC60 000073E8 0004D050 0002D9D0 00009DF0	0002CCD8 00029E50 080073E8 00005500
*...Y.....R...0...Q.....Y...*	
0007AC80 00093000 00000000 0004D0A8 0004D050	0002BCD8 00000000 0004D59C 0004D62C
*.....Q.....N...O...*	
0007ACA0 FF034196 00034BD4 000073E8 0004D050	0002D9D0 00009DF0 0002CCD8 00029E50
*.....M...Y.....R...0...Q...*	
0007ACC0 080073E8 00005500 00093000 0004D050	0004D0A8 080073E8 00034100 00000000
*...Y.....Y.....*	
0007ACE0 0004D5E4 0004D674 FF034E7A 0003C8B8	00000000 0004D050 0002D9D0 00009DF0
*..NU..O.....H.....R...0*	
0007AD00 0002CCD8 00029E50 080073E8 00005500	00093000 0004D050 00000000 080073E8
*...Q.....Y.....Y*	
0007AD20 00034BD4 00000000 0004D62C 0004D6BC	00000000 00000000 00000000 00000000
*...M...O...O.....*	
0007AD40 00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000
*.....*	
0007AD60 00000000 00000000 00000000 00000000	0004D674 0004D704 00000000 00000000
*.....O...P.....*	
0007AD80 00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000
*.....*	
0007ADA0 00000000 00000000 00000000 00000000	00000000 00000000 0004D6BC 0004D74C
*.....O...P...*	
0007ADC0 00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000
*.....*	
LINE 0007ADE0 SAME AS ABOVE	
0007AE00 0004D704 0004D794 00000000 00000000	00000000 00000000 00000000 00000000
*...P...P.....*	
0007AE20 00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000
*.....*	
0007AE40 00000000 00000000 0004D74C 0004D7DC	00000000 00000000 00000000 00000000
*...P...P.....*	
0007AE60 00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000
*.....*	
0007AE80 00000000 00000000 00000000 00000000	0004D794 0004D824 00000000 00000000
*.....P...Q.....*	

```

0007AEAO 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0007AEC0 00000000 00000000 00000000 00000000 00000000 0004D7DC 0004D86C
*.....P...Q.*
0007AEE0 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
      LINE 0007AF00 SAME AS ABOVE
0007AF20 0004D824 0008BD98 00000000 00000000 00000000 00000000 00000000
*..Q.....*
0007AF40 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0007AF60 00000000 00000000 0004D86C 0004D8FC 00000000 00000000 00000000
*.....Q...Q.*
0007AF80 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0007AFA0 00000000 00000000 00000000 00000000 0004D8B4 00000000 00000000
*.....Q.....*
0007AFC0 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0007AFE0 00000000 00000000 00000000 00000000 00000002 00000000 D3C7E6C1 00518000
*.....LGWA....*
00081560 47F0F034 2FC4C6E2 C6D3D3C7 F060F1F3 F060D3D6 C7C9C3C1 *
.00..DFSFLG0.130.LOGICA*

```

## Recovering from Sx37 abends on GSAM data sets

When an application program is inserting records into a GSAM DASD data set and space on the data set is exceeded, an Sx37 abend occurs. The proper restart procedure depends on the physical characteristics of the GSAM data set and how IMS checkpointed the position in the data set.

### Recovering non-SMS-managed data sets

When an Sx37 abend occurs in a non-SMS-managed data set, you typically solve the problem by copying the data set and allocating more space for the copy.

#### About this task

You can copy the data set with the IEBGENER utility or another utility that reads and writes logical records. Do not do this for blocked GSAM BSAM DASD data sets if you plan to restart by using the copy. You must copy the physical records, not only the logical records. You can use the IEBGENER utility for this task, but you must specify different DCB parameters. A blocked data set has a record format of FB or VB.

To recover from an Sx37 abend on a blocked non-SMS-managed GSAM data set:

#### Procedure

1. Copy the file to a larger data set by using the IEBGENER utility.
2. Specify RECFM=U for the record format for both the input and output data sets. This parameter copies the physical records as they exist. No reblocking is done. The copy must be to a like device type with the same track size. If the data set resides on multiple volumes, only the last volumes of data can be copied. GSAM keeps position by relative volume, by relative track within the volume, and by relative physical block within the track.
3. Change the RECFM parameter for the copied file to its original value: FB or VB. You can change the parameter by using any program that opens the data set, including the IEBGENER utility. Execute the IEBGENER utility with a SYSUT1 DD statement with DISP=SHR and a SYSUT2 DD statement with DISP=MOD. The SYSUT2 DD statement must specify RECFM=xx, where xx is the original GSAM data set record format value. This value causes the IEBGENER utility to open the data set for output. The IEBGENER utility does not copy any records to the data set, but it rewrites the DSCB with the updated RECFM value at close time.

#### Results

You can now use the copy to restart the program from a checkpoint.



### Example JCL for copying a multivolume GSAM data set after an Sx37 abend

The following JCL describes how to copy a multivolume GSAM data set after an Sx37 abend occurs, which includes the following basic steps:

1. Uncatalog the data set that received the Sx37 abend.
2. Allocate a temporary data set for the last volume portion of the data set by using the original LRECL and BLOCKSIZE values, RECFM=U, and the new space allocation.
3. Copy the portion of the original data set from the last volume into the newly allocated temporary data set.
4. Delete the portion of the original data set that resides on the last volume.
5. Rename the temporary data set to the original name.
6. Catalog the data set again, using the new volume label if the data set has moved, and reset the RECFM keyword to its original value. The IEBGENER utility is recommended for this step because the data set has to be opened for the RECFM reset to take effect.
7. Submit the XRST job.

```
//B37C0PY JOB (IMS,xxxxxxx),'GSAM TEST',
//          REGION=0M,
//          CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)
//*****
//* 1. Uncatalog multivolume data set.
//*   The data set needs to be uncataloged before we can copy the
//*   the last volume of a multivolume data set
//*****
//UNCALG EXEC PGM=IEFBR14
//DD1      DD DSN=DBDC.IMS.GSAMTEST,DISP=(OLD,UNCATLG),UNIT=SYSDA
/*
//*****
//* 2. Allocate a new temporary data set with a bigger space allocation
//*   for the last volume portion of a multivolume data set.
//*****
//ALLOC EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=A
//DD2      DD DSN=DBDC.IMS.GSAMTEMP,DISP=(NEW,KEEP),UNIT=SYSDA,
//          VOL=SER=SCR03,
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=80),
//          SPACE=(CYL,(5,1))
/*
//*****
//* 3. Copy current (smaller) GSAM/BSAM data set into a
//*   new (bigger) GSAM/BSAM data set by using DCB=(RECFM=U)
//*****
//COPY1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD DSN=DBDC.IMS.GSAMTEST,DISP=(OLD),
//          UNIT=SYSDA,VOL=SER=SCR03,DCB=(RECFM=U)
//SYSUT2   DD DSN=DBDC.IMS.GSAMTEMP,DISP=OLD,
//          UNIT=SYSDA,VOL=SER=SCR03,DCB=(RECFM=U)
//SYSIN    DD DUMMY
/*
//*****
//* 4. Delete the smaller (original) of the data set that resides on
//*   the last volume
//*****
//BR14D EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=A
//GSAMDS1 DD UNIT=SYSDA,DISP=(OLD,DELETE),VOL=SER=SCR03,
//          DSN=DBDC.IMS.GSAMTEST
/*
//*****
//* 5. Rename the new (copied to) data set to the original (copied
//*   from) data set name
//*****
//RENAME1 EXEC PGM=IEHPROGM
//SYSPRINT DD SYSOUT=A
//GSAMDS1 DD UNIT=SYSDA,DISP=OLD,VOL=SER=SCR03
//SYSIN DD *
//          RENAME DSN=DSN,DBDC.IMS.GSAMTEMP,VOL=VOL,DISK=SCR03,
//          NEWNAME=DSN,DBDC.IMS.GSAMTEST
/*
//*****
//* 6. Re-catalog the multivolume data set using the new volume for
```

```

/*      last volume of the data set, and change the RECFM
/*      from RECFM=U back to its original RECFM=.
/*      IEBGENER utility is recommended because the data set has to
/*      be opened for the RECFM reset to take effect
/*      ****
/*
//CATALG  EXEC  PGM=IEBGENER
//SYSPRINT DD  SYSOUT=A
//SYSUT1  DD  DSN=DBDC.IMS.GSAMTEST,DISP=SHR,UNIT=SYSDA,
//          VOL=SER=(222222,333333,000000,SCR03)
//SYSUT2  DD  DSN=DBDC.IMS.GSAMTEST,DISP=(MOD,CATLG),UNIT=SYSDA,
//          VOL=SER=(222222,333333,000000,SCR03),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=80)
//SYSIN   DD  DUMMY
/*
//

```

## Recovering SMS-managed data sets

When the GSAM data set resides on SMS-managed volumes and is a non-striped data set, you can extend the existing data set if the maximum number of volumes (59) is not exceeded.

### *Extending an SMS-managed, non-striped GSAM data set*

When you convert GSAM data sets to SMS-managed data sets, consider converting to SMS-managed striped data sets, or at minimum, converting to striped data sets, before the maximum number of volumes (59) for a data set is reached.

### Before you begin



**Attention:** If the installation must use non-striped data sets for GSAM databases, and a data set full condition forces you to add a new volume to the data set, be aware that the maximum number of volumes for a data set is 59 volumes. As you approach this limit, copy the data set by using a larger space allocation value before the data set volume limit is reached. If the volume limit is reached and an abend occurs, you will be unable to perform a GSAM XRST. The copy of the data set using the larger space allocation must be completed after a job that has ended normally. After the data set is copied, GSAM XRST cannot be done using checkpoints taken before the data set was copied.

### About this task

To extend an SMS-managed, non-striped GSAM data set:

### Procedure

1. Under SMS, add extra volumes to the storage group, if necessary, and increase the number of volumes allowed for the DATACLAS keyword.
2. Using IDCAMS, enter the command `ALTER dsn ADVOL(*)` to indicate that additional volumes are available to the data set.

### *Converting an SMS-managed, non-striped data set to a striped data set*

Convert SMS-managed, non-striped data sets to SMS-managed, striped data sets before you execute jobs that might abend with Sx37 abends because they exceed the maximum number of volumes (59).

### Before you begin

If an Sx37 abend occurs for an SMS-managed, non-striped data set due to the number of volumes for the data set exceeding 59 volumes, the data set cannot be recovered and an **XRST** cannot be completed.

The recovery cannot be completed because the SMS-managed, non-striped data set must be copied to an SMS-managed, striped data set with a larger primary and secondary allocation space parameter. When copying SMS-managed data sets, the copy process uses all the new space that is defined, starting with the first volume, which results in the checkpointed data record being moved not only to a different

volume, but also having a different relative track and record (TTRZ, or TTTRZ for large format data sets). The copy process thus invalidates the restart position saved in the X'18' log record.

## About this task

Migration from GSAM/BSAM non-striped data sets, which use TTRZ (or TTTRZ for large format data sets) for repositioning, to striped data sets, which use relative block number (RBN), results in an XRST failure or possibly an XRST reposition on the wrong data record. You cannot process a GSAM/BSAM non-striped data set, have an error condition occur, copy the data set to a striped data set, and then attempt to perform an **XRST** command. You must convert GSAM/BSAM non-striped data sets to striped data sets when you learn that your GSAM/BSAM databases are SMS-managed data sets and there is any chance that a job accessing that database might abend with an Sx37 number of volumes exceeded condition and that an **XRST** command is required.

To convert to an SMS-managed, non-striped data set to a striped data set:

## Procedure

1. Copy the non-striped data set to a new striped data set and use the same block size and device type.
2. Specify the RECFM=U record format for both the input and the output data set to avoid any reblocking.
3. When using DFSMS striped data sets, the restriction regarding maintaining the same number of records on the copied from and copied to volumes does not apply because with striped data sets, GSAM uses the RBN for NOTE/POINT, not the TTRZ (or TTTRZ for large format data sets).

## What to do next

In addition, GSAM supports DFSMS striped (extended format) data sets for both VSAM and BSAM.

## Related concepts

[Processing GSAM databases \(Application Programming\)](#)



---

# Chapter 9. DBRC - Database Recovery Control service aids

Diagnostic service aids, including RECON record types, DBRC internal trace, DBRC external trace are described.

---

## Diagnosing from a RECON list

You can use the **LIST** command to list the contents of all or part of the RECON data set.

You can list:

- The copy1 RECON data set
- RECON records for a particular change-accumulation group or for all change accumulation groups
- RECON records for a particular log data set or for all log data sets
- RECON records for a particular database data set or for DBDS groups
- Databases
- Subsystems
- Interim log records

Because some information is not printed when you issue the **LIST.RECON** command, you can issue the **PRINT** command for the access services method that you are using to list all information in hexadecimal format.

---

## RECON record types

The records in the RECON data set store information about logging activity and events that can affect the recovery of the database.

### Content description of the keys in RECON records

To view the layout of the entire RECON record, see the following table. Consider these points as you examine the records:

- The RECON key size is 32 bytes.
- The last three bytes of the key contain either of the following values:
  - Reserved, and contain zeros.
  - First byte=0 and the last 2 bytes=key segment number.
- Time stamps have the following characteristics:
  - Time stamps are 12 bytes.
  - The symbolic UTC format is:  
YYYYDDDFHHMMSSTHMIJUAQQS  
An example of the UTC format is: 2004006F211432800000032D
  - DSPTIMES (DFSTIMES) contains time stamp structure information.

The following table shows the RECON record types.

Table 53. RECON record types

Common name	Part name	List ID	Release	Key fields
RECON Header	DSPRCNRC	RECON	R-1	DBD: hex zeros DDN: hex zeros Type: X'01' Time: hex zeros
RECON Header Extension	DSPRCR1	*****	R-3	DBD: hex zeros DDN: hex zeros Type: X'01' Time: X'000000000008'
Audit Trail Record	DSPMUPHD	*****	2.1	DBD: hex zeros DDN: hex zeros Type: X'02' Time: sequence number
RECON DMB Table Record	DSPRDTRC	*****	9.1	DBD: hex zeros DDN: hex zeros Type: X'03' Time: hex zeros  <b>Note:</b> Not listed in RECON Listing. An IDCAMS print will show the record if it exists.
PRILOG	DSPLOGRC	PRILOG	R-1	DBD: hex zeros DDN: hex zeros Type: X'05' Time: time stamp
Interim PRILOG	DSPLOGRC	IPRI	R-2	DBD: hex zeros DDN: hex zeros Type: X'06' Time: time stamp
LOGALL	DSPLGARC	LOGALL	R-1	DBD: hex zeros DDN: hex zeros Type: X'07' Time: time stamp
SECLOG	DSPLOGRC	SECLOG	R-1	DBD: hex zeros DDN: hex zeros Type: X'09' Time: time stamp
Interim SECLOG	DSPLOGRC	ISEC	R-2	DBD: hex zeros DDN: hex zeros Type: X'0A' Time: time stamp

Table 53. RECON record types (continued)

Common name	Part name	List ID	Release	Key fields
PRISLDS	DSPLOGRC	PRISLD	R-3	DBD: X'FFFFFFFFF00000043' DDN: subsystem name Type: X'43' Time: time stamp
Interim PRISLDS	DSPLOGRC	IPRISL	R-3	DBD: X'FFFFFFFFF00000045' DDN: subsystem name Type: X'45' Time: time stamp
SECSLDS	DSPLOGRC	SECSLD	R-3	DBD: X'FFFFFFFFF00000047' DDN: subsystem name Type: X'47' Time: time stamp
Interim SECSLDS	DSPLOGRC	ISECSL	R-3	DBD: X'FFFFFFFFF00000049' DDN: subsystem name Type: X'49' Time: time stamp
Change Accum Group	DSPCAGRC	CAGRP	R-1	DBD: hex zeros DDN: CA group name Type: X'0F' Time: hex zeros
Change Accum Execution	DSPCHGRC	CA	R-1	DBD: hex zeros DDN: CA group name Type: X'11' Time: time stamp
DBDS Group	DSPDGRC	DBDSGRP	2.1	DBD: X'00000000000000007' DDN: DBDS group name Type: X'16' Time: hex zeros
Database Header	DSPDBHRC	DB	R-2	DBD: DBD name DDN: hex zeros Type: X'18' Time: hex zeros
Partition	DSPPTNRC	DB	7.1	DBD: DBD name DDN: Partition name Type: X'19' Time: hex zeros
Database Data Set	DSPDSHRC	DBDS	R-1	DBD: DBD name DDN: DDN name Type: X'20' Time: hex zeros

Table 53. RECON record types (continued)

Common name	Part name	List ID	Release	Key fields
Area Recovery	DSPDSHRC	DBDS	R-3	DBD: DBD name DDN: area name Type: X'20' Time: hex zeros
Area Auth	DSPDBHRC	DBDS	R-3	DBD: DBD name DDN: area name Type: X'21' Time: hex zeros
ALLOC	DSPALLRC	ALLOC	R-1	DBD: DBD name DDN: DDN or area name Type: X'28' Time: time stamp
Image Copy	DSPIMGRC	IMAGE	R-1	DBD: DBD name DDN: DDN or area name Type: X'2D' Time: time stamp
Reorg	DSPRRGRC	REORG	R-2	DBD: DBD name DDN: DDN or area name Type: X'32' Time: time stamp
Recovery	DSPRCVRC	RECOV	R-1	DBD: DBD name DDN: DDN or area name Type: X'37' Time: time stamp
Backout	DSPBKORC	BACKOUT	4.1	DBD: X'FFFFFFFFF00000035' DDN: subsystem name Type: X'35' Time: hex zeros
Subsystem	DSPSSRC	SSYS	R-2	DBD: X'FFFFFFFFFFFFFFFFF' DDN: subsystem name Type: X'3F' Time: hex zeros
Available CA Execution	DSPCHGRC	CA	R-1	DBD: hex zeros DDN: hex zeros Type: X'51' Time: time stamp
PRIOLDS	DSPOLDRC	PRIOLD	R-3	DBD: X'FFFFFFFFF00000053' DDN: subsystem name Type: X'53' Time: hex zeros



Table 53. RECON record types (continued)

Common name	Part name	List ID	Release	Key fields
Interim PRIOLDS	DSPOLDRC	IPRIOL	R-3	DBD: X'FFFFFFFF00000055' DDN: subsystem name Type: X'55' Time: hex zeros
SECOLDS	DSPOLDRC	SECOLD	R-3	DBD: X'FFFFFFFF00000057' DDN: subsystem name Type: X'57' Time: hex zeros
Interim SECOLDS	DSPOLDRC	ISECOL	R-3	DBD: X'FFFFFFFF00000059' DDN: subsystem name Type: X'59' Time: hex zeros
Available Image Copy	DSPIMGRC	IMAGE	R-1	DBD: DBD name DDN: DDN or area name Type: X'6D' Time: hex zeros

## DBRC internal trace

DBRC internal trace is a useful tool for diagnosing problems that are possibly related to DBRC. The trace is always enabled.

### Types of problems DBRC internal trace can diagnose

The DBRC internal trace can help diagnose many different types of problems, such as:

- RECON data set contention
- RECON data set errors that are indicated by messages
- System abends in which the PSW is pointing to DBRC
- DBRC abends
- Whether DBRC or some other IMS component is causing the problem

Sometimes a problem occurs as a result of the interaction between two different modules performing different tasks. Interpreting trace entries is the best way to determine what each module was doing and when. For example, for RECON data set errors, it is important to know which DBRC modules manipulated the RECON data set and when.

You generally look at the DBRC trace output under the direction of an IBM Software Support representative, who will guide you in collecting data in specific trace fields and in interpreting that data. The DBRC trace entries that follow help you interpret trace data.

### Example of DBRC internal trace

A user receives abend code xxx. The PSW is pointing to DBRC. The user reports the problem to an IBM support representative. Some of the steps that the user diagnostician might take under the guidance of the IBM representative are:

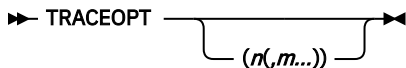
1. Locate the DBRC trace in the trace using the TRACETBL eye catcher.

2. Use the sample trace to verify that you have found the trace and to help you navigate through the trace table entries.
3. Find DBRC and IMS control blocks and data areas by using addresses from selected trace table entries.
4. Determine the events that occurred before the abend.
5. Use the information in the trace and data areas to understand what caused the abend.

Some DBRC functions have the capability of generating additional trace entries that can aid in problem analysis. An IBM representative can assist you in enabling one or more of these expanded trace options through the use of the **CHANGE . RECON** command.

The **CHANGE . RECON** command supports a TRACEOPT parameter that allows you, under the direction of an IBM representative, to select expanded DBRC trace options.

#### **CHANGE.RECON**



**n,m,...**

DBRC TRACEOPT options

TRACEOPT is an optional parameter that you use only under the direction of an IBM Software Support representative for the purpose of gathering documentation for problem analysis. The IBM Software Support will provide the sub-options for the TRACEOPT parameter.

## **DBRC trace input**

When called, DSPTRACE receives a 16-byte parameter list.

### **Contents of the parameter list**

- An 8-character identifier that becomes the first 8 characters of the trace entry.
- A 4-byte control block pointer that points to a DFSBRLSB or the DSPRQB.
- A 4-byte block area pointer. 64 bytes of data from the block area are inserted in the trace entry. If the pointer is 0, the trace entry is 64 bytes long. Otherwise, the trace entry is 128 bytes long.

## **Locating the DBRC trace**

The DBRC trace is in the IMS-formatted portion of an IMS-formatted dump. There are four ways you can locate the DBRC trace.

### **About this task**

#### **Method 1**

Find the trace in the DBRC section of the IMS offline formatted dump.

#### **Method 2**

Find any DSPxxxx module in the Save Area trace of the dump. For most DSPxxxx modules marked ENTERED VIA CALL, register 5 contains the address of the request level control block (RQB). Offset X'38' in the RQB contains the address of router storage. Offset X'1C' in router storage contains the address of the DBRC trace.

In certain situations, register 5 does not point to the RQB. If this is the case, use method 3 or method 4.

#### **Method 3**

The trace is in subpool 0. If the dump has an index, look in the index to locate subpool 0. Scan this portion of the dump for eye catcher "TRACETBL," which identifies the beginning of the trace.

#### Method 4

If you are looking at a dump online, search for eye catcher TRACETBL or GETFEED. If you search for the GETFEED eye catcher, you might first find it within DBRC modules. Search until you find the GETFEED eye catcher in the DBRC trace. Scroll to the beginning of the trace.

#### Related reference

[“Unformatted DBRC internal trace example” on page 255](#)

The module-call entries, module return entries, DSPURI00 trace entries, and other entries (such as GETFEED, DSPCRTR0, and CRTROXIT) are shown in this trace example.

## DBRC trace output

Trace output normally resides in subpool 0 storage, but you can direct output to a Generalized Trace Facility (GTF) data set.

The DBRC internal trace is a wraparound trace. That is, after the trace table is full, tracing starts at the beginning of the table, and each new entry overlays an old entry.

An entry with the identifier "TRACENXT" marks the next entry to be used, which is the logical end of the trace table.

#### Related reference

[“DBRC external trace” on page 263](#)

If you start the Generalized Trace Facility (GTF) and issue the CHANGE.RECON TRACEON command, the DBRC trace (DSPTRACE) creates an external trace record and issues the GTRACE macro to invoke GTF.

## DBRC trace header record

The DBRC trace header record is shown.

```
words 0-1 – Identifier TRACETBL
word 2 – Length of the trace
word 3 – Count of trace calls made
word 4 – Beginning of trace table
word 5 – End of trace table
word 6 – Next entry to update
word 7 – Double word alignment
```

## Module call, module return, and DSPSTACK trace entries

With few exceptions, DBRC modules call module DSPSTGET to obtain initial work space and additional temporary work space (with the DSPGFSTK macro). Upon exit, DSPSTFRE releases the space obtained for the module. This centralized temporary storage management allows DBRC to track the flow of modules, starting with the first call out of DSPCRTR0 (entry point to DBRC).

Three trace entries accomplish this:

- Words 1 and 2 show the following items:
  - An arrow indicating whether the module is being called or is returning.
  - The nesting level of the module being called or returned to. Nesting levels are shown in one or two decimal digits, up to 99. (Nesting level 0 is DSPUIN00)
  - The last five characters of the module name being called or returning.
- DSPSTACK—additional work space trace entry (the result of the currently active module issuing the DSPGFSTK macro that calls DSPSTGET).

The following figure illustrates this processing flow:

1. Module A calls module B, which in turn calls DSPSTGET to obtain initial work space.
2. Module B issues macro DSPGFSTK to obtain additional work space.
3. Module B calls DSPSTFRE to release all temporary storage.

4. Module B returns control to module A.

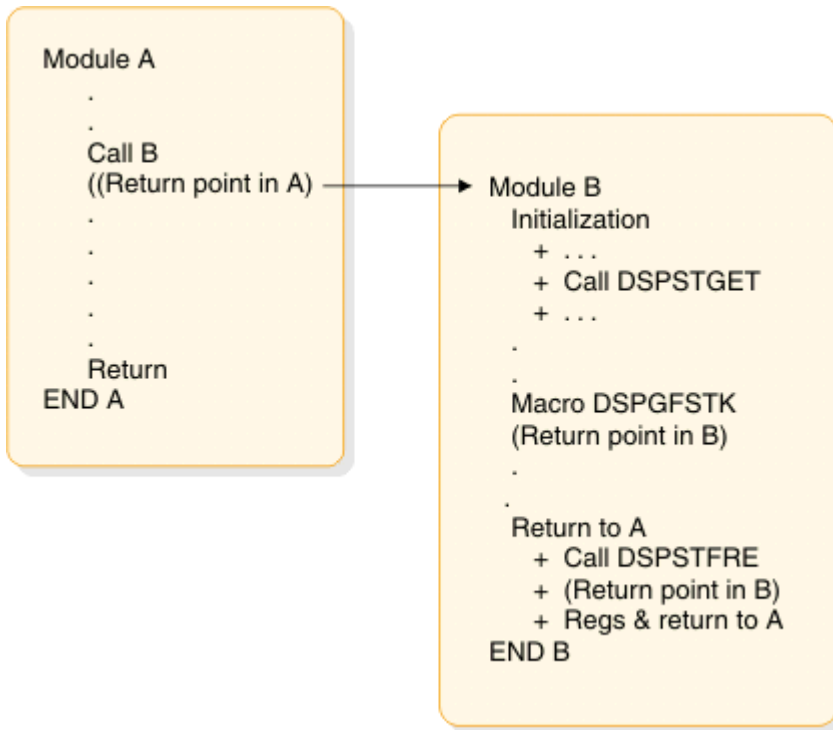


Figure 64. DBRC trace processing flow

"DBRC internal trace" on page 243 illustrates the format of the trace entries associated with this module flow. Each entry occupies one line (8 words) in the DBRC internal trace table. References to specific addresses and locations in modules A and B refer to the diagram above.

#### One-line trace entry produced when module A calls module B

The following figure shows a one-line trace entry that is produced when module A calls module B. A one-line trace entry that is produced when module B calls DSPSTGET to obtain initial work space storage after being called by module A.

```
words 0-1 - Identifier which consists of:
            - An arrow (") indicating that the module is
              being called.
            - The nesting level of module B. Nesting
              levels are shown in one or two decimal digits
              up to 99 (nesting level 0 is DSPUIN00).
            - The last five characters of the module name
              being called.

word 2 - Address in module A of call to module B

word 3 - Entry point address of module B

word 4 - Save area address of the calling module (A)

word 5 - Beginning address of the temporary storage
         obtained for module B (B's save area address)

words 6-7 - Trace time stamp
```

### One-line trace entry produced when module B returns to module A

The following figure shows a one-line trace entry that is produced when module B calls DSPSTFRE to release all of its temporary storage before returning to module A.

```
words 0-1 - Identifier which consists of:
            - The nesting level of module A. Nesting levels
              are shown in one or two decimal digits up to
            - A left arrow (") indicating that the module is
              returning.
            99 (nesting level 0 is DSPUI000).
            - The last five characters of the module name
              returning.

word 2 - Address in module A to which module B returns

word 3 - Offset in module B where it returns to module A

word 4 - Save area address of module A that called module B

word 5 - Beginning address of the temporary storage being
        released for module B by module DSPSTFRE

words 6-7 - Trace time stamp
```

### DSPSTACK trace entry

The following figure shows a one-line trace entry that is produced when module B issues macro DSPGFSTK, which calls DSPSTGET to obtain additional temporary storage.

```
words 0-1 - Identifier DSPSTACK

word 2 - Return point address in the module B to which
        DSPSTGET returns after acquiring additional
        temporary storage for the module.

word 3 - Entry point address of module B

word 4 - Save area address of the module (B)

word 5 - Beginning address of the additional temporary
        storage obtained for module B

words 6-7 - Trace time stamp
```

## BGNCABN0, DSPCABN0, BGNRETRY, DSPCRTR0, and CRTROXIT trace entries

DBRC internal trace is a useful tool for diagnosing problems that are possibly related to DBRC. The trace is always enabled.

In DBRC, the following modules have specific trace calls inserted in their processing flow:

- DSPCABN0
- DSPCRTR0
- DSPUI00

The following figures show the layout of the entries issued from BGNCABN0, DSPCABN0, and DSPCRTR0.

```
words 0-1 - Identifier BGNCABN0

word 2 - A(DSPRQB)

words 3-5 - Zeros

words 6-7 - Time stamp
```

This is normally followed by either DSPCABN0 or a BGNRETRY entry.

The following figure shows that DBRC terminated because of an unrecoverable error.

```
words 0-1 - Identifier DSPCABN0
word 2 - A(DSPRQB)
words 3-5 - Zeros
words 6-7 - Time stamp
```

This is the last logical entry in the trace table.

The following figure shows that DBRC recovered from an abend condition and is beginning to execute a retry sequence of code.

```
words 0-1 - Identifier BGNRETRY
word 2 - A(DSPRQB)
words 3-5 - Zeros
words 6-7 - Time stamp
```

The following figure shows that the router made a trace call before passing control to the next DBRC routine that is scheduled to process the request identified by a DFSBRLSB.

```
Line 1:
words 0-1 - Identifier DSPCRT0
word 2 - A(DFSBRLSB)
words 3-5 - Data from DFSBRLSB: function flags, exit flags, address
            of DSPGDB (These fields are the same as the fields
            that are displayed in the DSPCRT0 entry, but they
            might have been modified by the request.)
words 6-7 - Time stamp
```

```
Line 2:
word 0 - Address of BRLBPRNT field in DFSBRLSB
words 2-7 - Data from DFSBRLSB (next 60 bytes after
            field BRLBPRNT)
```

```
Line 3:
words 0-7 - Data from DFSBRLSB (continued from previous
            line)
```

The following figure shows the function requested in the DSPCRT0 trace entry completed.

```
Line 1:
words 0-1 - Identifier CRT0XIT
word 2 - A(DFSBRLSB)
words 3-5 - Data from DFSBRLSB: function flags,
            exit flags, address of DSPGDB. (These
            are the same fields displayed in the
            DSPCRT0 entry, but they might have been
            modified by the request.)
words 6-7 - Time stamp
```

```
Line 2:
words 0-4 - DFSBRLSB prefix
```

Line 3:  
words 0-7 - DFSBRLSB (continued from previous line)

## DSPURI00 trace entries

A trace entry with the identifier DSPURI00 indicates the beginning of a series of trace calls that show what occurs as DSPURI00 processes an I/O request.

All trace calls from DSPURI00 result in 96-byte trace entries. There are nine separate calls to the trace routine in DSPURI00. The pointer to the DSPRQB follows the trace identifier. The following table shows the 8-character identifier and block-area pointer for each call.

*Table 54. Calls to the trace routine in DSPURI00*

8-character identifier	Block-area pointer	Explanation
DSPURI00	MODIRCAR	DSPURI00 receives control and the function-code value from DSPIRCAR indicates the type of call. (See <a href="#">“DSPURI00 entry trace entry”</a> on page 250.)
OPENER1	FILRESLT(I)	DSPURI00 starts a physical open of a RECON data set.
OPENER2	FILRESLT(I)	DSPURI00 completes a physical open of a RECON data set.
OPENXA8	FILRESLT(I)	DSPURI00 encountered a VSAM OPEN failure with reason code X'A8'.
GETFEED	FILRESLT(I)	After DSPURI00 issues an I/O request, the GETFEED procedure is called to trace specific information related to the I/O operation. Some of this information comes from DSPVFILE, some from the VSAM RPL, some from the record key and some from the I/O parameter block, DSPIOPAR. In addition, the RPL request is translated into a character printable code that describes the I/O operation. See <a href="#">“GETFEED trace entry for one RECON”</a> on page 251.
CLOSER1	FILRESLT(I)	DSPURI00 starts a true close of the RECON data set.
CLOSER2	FILRESLT(I)	DSPURI00 completes a true close of the RECON data set.
VSAMERR	FILRESLT(I)	A VSAM error occurred and the routine to print a VSAM error message was entered.
RDOPTERR	RILRESLT(I)	An invalid read integrity option was specified. An error message was issued.
DSPURI00	ENDIRCAR	DSPURI00 returns to its caller. Relevant exit condition information, if applicable, is traced. (See <a href="#">“DSPURI00 exit trace entry”</a> on page 253.)

The sequence of trace entries identified by DSPURI00, OPENER1, OPENER2, and GETFEED shows DSPURI00 receiving control and doing a physical open of one RECON data set. When DSPURI00 opens the second RECON data set, another sequence of OPENER1, OPENER2, and GETFEED entries follow the entries for the first RECON data set.

The DSPIRCAR data area includes a 1-byte function code and a 3-byte flag field. The function codes are alphabetic characters that identify what operation DSPURI00 does. The flag bytes further identify the type of operation. Pertinent information is extracted from the DSPIRCAR data area and placed in a modified IRCAR area, along with other processing information, to produce both the entry and exit traces within DSPURI00.

The GETFEED trace entry maps 64 bytes of information about the I/O operation. The last two lines of the entry contain this data.

The exit trace entry is similar to the entry trace. It is written upon return from DSPURI00, but only if one or more of the following conditions is true:

- This was a request to locate a specific RECON record.
- The request did not complete successfully (RC greater than 0 was returned).
- The copy 1 or 2 RECON status changed on this entry to DSPURI00.

### DSPURI00 entry trace entry

```
Line 1:
  words 0-1 - DSPURI00
  word 2 - RQB address
  words 3-5 - Binary zeros
  words 6-7 - Time stamp
```

```
Line 2:
  words 0-1 - MODIRCAR
  word 2 - c1c2
  word 3 - Func
  words 4-7 - 16-byte entry message
```

```
Line 3:
  words 0-5 - Key, blank, or repl ddname (key area)
  word 6 - addr
  word 7 - leng
```

### time stamp

Trace time stamp.

### c1c2

The DD statement number (1-3) of the copy 1 and copy 2 RECON, if any, on entry to DSPURI00.

### func

Function and option bits received from caller in DSPIRCAR.

### 16-byte entry message

EBCDIC message readable at the right end of the trace entry, such as LOGICAL OPEN, END MULT, UPDATE, and others. Class and sequential locate requests and configuration requests have a modifier at the end of their message:

#### F

Locate first

#### L

Locate last

#### NX

Locate next

#### P

Locate previous

#### NG

Locate not-greater-than

#### DSNS

Supply dsnames of RECONs in DSPIRCAR

#### STAT

Supply status of all RECONs in DSPIRCAR

#### DUAL

Enter dual mode

#### REPL

Replace RECONx with spare (where x = 1-3, see key area)



**key area**

For all locate, change, insert, and delete requests, contains the 32-byte key of the record involved. For replace requests, contains the ddname of the RECON to be replaced.

**addr**

Address of a record to be changed or inserted.

**leng**

Length of a record to be changed or inserted.

**GETFEED trace entry for one RECON**

```
Line 1:
  words 0-1 - GETFEED
  word 2 - DSPRQBA
  words 3-5 - Binary zeros
  words 6-7 - Time stamp
```

```
Line 2:
  word 0 - RPLFDBWD
  word 1 - FILLRECL
  word 2 - FILNEWCA
  word 3 - FILNEWEX
  word 4 - FILCICNT
  word 5 - FILCACNT
  word 6 - FILEXCNT
  word 7 - FILMAX
```

```
Line 3:
  word 0 - FILCISZ
  word 1 - bytes 1-2 - FILFLAGS
             bytes 3-4 - FILOPERR
  word 2 - FILBUFPT
  word 3 - FILRCDPT
  word 4 - FILRCDLN
  word 5 - bytes 1-2 - SEGMENT NUMBER
             byte 3 - RECON NUMBER
             byte 4 - RPLREQ
  word 6 bytes 1-4 and word 7 bytes 1-2 - PRINTABLE
                                     RPLREQ
  word 7 - bytes 3-4 - NOT USED
```

**dsprqba**

Address of the DSPRQBA.

**time stamp**

Trace time stamp.

**RPLFDBWD**

RPL feedback word.

**FILLRECL**

Logical record length.

**FILNEWCA**

Starting high-used relative byte address (RBA).

**FILNEWEX**

Starting high-allocated RBA.

**FILCICNT**

RECON changed counter value.

**FILCACNT**

Current high-used RBA.

**FILEXCNT**

Current high-allocated RBA.

**FILMAX**

VSAM maximum record size.

**FILCISZ**

Data control interval (CI) size.

**FILFLAGS**

RECON processing status flags (open, reserved, empty).

**FILOPERR**

Open SVC reason code if RC is not 0.

**FILBUFPT**

Pointer to header record buffer.

**FILRCDPT**

Pointer to the record in the VSAM I/O buffer or user area.

**FILRCDLN**

Length of record.

**SEGMENT NUMBER**

Record segment number.

**RECON COPY NUMBER**

Recon number used in this request.

**RPLREQ**

RPL request type.

**RPL REQUEST PRINTABLE CODE**

English word that is later translated into a printable code used to make a request to VSAM.

**RPL REQ PRINTABLE CODE**

This is translation of the RPLREQ field into a printable code that is close to being the English word for the request made to VSAM. The following table shows the translated RPLREQ printable codes.

*Table 55. Translated RPLREQ printable codes*

Printable code	Hexadecimal	RPLEQ	RPL request
GET	00	GET	Retrieve a record
PUT	01	PUT	Write a record
CHECK	02	CHECK	Wait for completion
POINT	03	POINT	Position for access
ENDREQ	04	ENDREQ	Terminate a request
ERASE	05	ERASE	Delete a record
VERIFY	06	VERIFY	Synchronize end of data
*****	07	Not used	Not used
DATPRE	08	DATA PREFORMAT	
IDXPRES	09	INDEX PREFORMAT	
FORCIO	0A	Force I/O	
GETIX	0B	GET INDEX	
PUTIX	0C	PUT INDEX	
SCHBFR	0D	SCHBFR	Search Buffer
MRKBFR	0E	MRKBFR	Mark Buffer
WRTBFR	0F	WRTBFR	Write Buffer
CNVTAD	10	CNVTAD	

Table 55. Translated RPLREQ printable codes (continued)

Printable code	Hexadecimal	RPLEQ	RPL request
MNTACQ	11	MNTACQ	
ACQRNG	12	ACQRANGE	
TRMRPL	13	TERMRPL	
VERREF	14	VERIFY REFRESH	

### DSPURI00 exit trace entry

Line 1:  
 words 0-1 – DSPURI00  
 word 2 – RQB address  
 words 3-5 – Binary zeros  
 words 6-7 – Time stamp

Line 2:  
 words 0-1 – ENDICAR  
 word 2 – c1c2  
 word 3 – Func  
 words 4-7 – 16-byte entry message

Line 3:  
 words 0-5 – Key, blank, or repl ddname (key area)  
 word 6 – addr  
 word 7 – lnrc

### time stamp

Trace time stamp.

### c1c2

The DD statement number (1-3) of the copy 1 and copy 2 RECON, if any, on exit from DSPURI00.

### func

Function and option bits received from caller in DSPIRCAR.

### 16-byte exit message

For locate requests, contains either the message RECORD WAS FOUND or RECORD NOT FOUND, depending on the outcome of the search. Otherwise, contains a repeat of MODIRCAR contents.

### key area

For a successful locate request, contains the 32-byte key of the RECON record returned to caller. Otherwise, contains a repeat of MODIRCAR contents.

### addr

Address of the record found for a successful locate. Otherwise, 0.

### lnrc

Length of the record found for a successful locate, or the return code to be passed back to the module that called DSPURI00.

## Trace entries related to parallel RECON access

When parallel RECON access is enabled, trace entries that are related to parallel access processing are created. Trace calls are made when unexpected conditions occur during processing.

Layout of the DSPURI80 trace entry:

The trace entry is created when DBRC detects that VSAM has shunted data for a RECON data set.

Line 1:  
 words 0-1 – Identifier DSPURI80  
 word 2 – A(DSPRQB)

```
words 3-5 - Zeros
words 6-7 - Time stamp
```

```
Line 2:
word 0 - Index into FILE data for data set with shunted I/O data
words 1-2 - DDNAME of RECON data set with shunted I/O data
words 3-7 - Zeros
```

```
Line 3:
words 0-7 - Zeros
```

Layout of the DSPRSYNC trace entry:

The trace entry is created when a non-zero return code is returned by an z/OS Resource Recovery Services ATRCMIT request or ATRBACK request.

```
Line 1:
words 0-1 - Identifier DSPRSYNC
word 2 - A (DSPRQB)
words 3-5 - Zeros
words 6-7 - Time stamp
```

```
Line 2:
word 0 - Requested RRS request (1=commit, 2=backout)
word 1 - Return code from ATRCMIT request or ATRBACK request
words 2-7 - Zeros
```

```
Line 3:
words 0-7 - Zeros
```

## DBRC group services entries

When DBRC is registered with SCI, module DSPRLX10 makes trace calls in response to group services requests. These calls can be requests to send notifications to other DBRC instances, event notifications, or requests or messages from other DBRC instances.

Requests without an accompanying message are traced in the DSPRLX10 trace entry. Entries with messages are traced in the DSPRLX1M trace entry.

Layout of the DSPRLX10 trace entry:

```
Line 1:
words 0-1 - Identifier DSPRLX10
word 2 - A(DFSBRLSB)
words 3-5 - Data from DFSBRLSB: function flags, exit flags, address of DSPGDB.
           (These fields are the same as the fields that are
            displayed in the DSPCRTR0 entry, but they might have been modified by
            the request.)
words 6-7 - Time stamp
```

Layout of the DSPRLX1M trace entry:

```
Line 1:
words 0-1 -Identifier DSPRLX1M
word 2 - A(DFSBRLSB)
words 3-5 - Data from DFSBRLSB: function flags, exit flags, address of DSPGDB.
           (These fields are the same as the fields that are
            displayed in the DSPCRTR0 entry, but they might have been modified by
            the request.)
words 6-7 - Time stamp
```

```
Line 2:
words 0-1 - 'DSPRLMSG'
word 3 - Message type
word 4 - message subtype
words 5-7 - Zeros
```

```
Line 3:
words 0-3 - Message sender ID
```

word 4 - Message sequence number  
word 5 - Message ID that is being responded to  
words 6-7 - First two words of message specific data

## Unformatted DBRC internal trace example

The module-call entries, module return entries, DSPURI00 trace entries, and other entries (such as GETFEED, DSPCRTR0, and CRTR0XIT) are shown in this trace example.

```
*** DBRC TRACE TABLE ***
0B9A4700 E3D9C1C3 C5E3C2D3 00025900 00000DDA 0B9A4720 0B9C9F20 0B9C9B60 0B9C9FA0
*TRACETBL.....-....*
0B9C9B60 E3D9C1C3 C5D5E7E3 40404040 40404040 40404040 40404040 40404040
*TRACENXT
0B9C9B80 40404040 40404040 40404040 40404040 40404040 40404040 40404040
*
      LINES 0B9C9BA0-0B9C9F1F SAME AS THE ABOVE
0B9A4720 606EF1E3 C9D4C5F0 8B99D3EC 0B939928 0B901E48 0B9A2010 BF9A6FDB F82B9987 *-
>1TIME0.rL..lr.....?.8.rg*
0B9A4740 F04C60E3 C9D4C5F0 0B99D3EC 0000113A 0B901E48 0B9A2010 BF9A6FDB F82CA047 *0<-
TIME0.rL.....?.8...*
0B9A4760 606EF1E4 D9C9F0F0 8B99D5B4 0B94F9C8 0B901E48 0B9A2010 BF9A6FDB F82EF547 *-
>1URI00.rN..m9H.....?.8.5.*
0B9A4780 C4E2D7E4 D9C9F0F0 0B99E480 00000000 00000000 00000000 BF9A6FDB F82F0F47
*DSPURI00.rU.....?.8...*
0B9A47A0 D4D6C4C9 D9C3C1D9 404001D8 D6000600 D7C8E8E2 C9C3C1D3 40D6D7C5 D5404040
*MODIRCAR .Q0...PHYSICAL OPEN *
0B9A47C0 40404040 40404040 40404040 40404040 40404040 40404040 00000000 00000000
*
0B9A47E0 606EF2E4 D9C9F0F1 000007A4 0B954A98 0B9A2010 0B9A2780 BF9A6FDB F82F4047 *-
>2URI01...u.n.q.....?.8..*
0B9A4800 606EF3E4 C3D7F4F0 00000446 0B99A518 0B9A2780 0B9A2F50 BF9A6FDB F8301F07 *-
>3UCP40....r.v.....&...?.8...*
0B9A4820 F24C60E4 C3D7F4F0 0B954EDE 000001BE 0B9A2780 0B9A2F50 BF9A6FDB F8328D07 *2<-
UCP40.n+.....&...?.8...*
0B9A4840 606EF3D9 E3E7C4F0 00000556 0B929418 0B9A2780 0B9A2F50 BF9A6FDB F832A547 *-
>3RTXD0....km.....&...?.8.v.*
0B9A4860 F24C60D9 E3E7C4F0 0B954FEE 000002AE 0B9A2780 0B9A2F50 BF9A6FDB FF6A2270 *2<-
RTXD0.n|.....&...?.8...*
0B9A4880 606EF3E4 D9C9F1F0 00000AC0 0B95C958 0B9A2780 0B9A2F50 BF9A6FDB FF6C1030 *-
>3URI10....nI.....&...?.%.*
0B9A48A0 606EF4E4 D9C9F2F0 00000524 0B95FDA8 0B9A2F50 0B9A3628 BF9A6FDB FF6C4DB0 *-
>4URI20....n.y.....&...?.%(*
0B9A48C0 606EF5E4 C1D3D3F0 000002F8 0B999BD0 0B9A3628 0B9A3A08 BF9A6FDB FF6CF070 *-
>5UALL0...8.r.....&...?.%0.*
0B9A48E0 F44C60E4 C1D3D3F0 0B9600A0 000001DC 0B9A3628 0B9A3A08 BF9A6FDC 010B7535 *4<-
UALL0.o.....&...?.%.*
0B9A4900 606EF5E4 C1D3D3F0 000002F8 0B999BD0 0B9A3628 0B9A3A08 BF9A6FDC 010B98F5 *-
>5UALL0...8.r.....&...?.%q5*
0B9A4920 F44C60E4 C1D3D3F0 0B9600A0 000001DC 0B9A3628 0B9A3A08 BF9A6FDC 0281DA47 *4<-
UALL0.o.....&...?.%a.*
0B9A4940 606EF5E4 C1D3D3F0 000002F8 0B999BD0 0B9A3628 0B9A3A08 BF9A6FDC 02820387 *-
>5UALL0...8.r.....&...?.%b.g*
0B9A4960 F44C60E4 C1D3D3F0 0B9600A0 000001DC 0B9A3628 0B9A3A08 BF9A6FDC 042CF50A *4<-
UALL0.o.....&...?.%5.*
0B9A4980 F34C60E4 D9C9F2F0 0B95CE7C 000002C2 0B9A2F50 0B9A3628 BF9A6FDC 042D294A *3<-
URI20.n.@...B...&.....?.%.*
0B9A49A0 606EF4D9 D3C9F0F0 00000750 0B923570 0B9A2F50 0B9A3628 BF9A6FDC 042D7D4A *-
>4RLI00...&.k.....&...?.%'.*
0B9A49C0 606EF5D9 D3C1E4F0 0000035E 0B922FC0 0B9A3628 0B9A3C20 BF9A6FDC 042E038A *-
>5RLAU0...; .k.....&...?.%.*
0B9A49E0 F44C60D9 D3C1E4F0 0B9238CE 0000047C 0B9A3628 0B9A3C20 BF9A6FDC 05C0B1B2 *4<-
RLAU0.k....@.....&...?.%.*
0B9A4A00 F34C60D9 D3C9F0F0 0B95D0A8 00000C8C 0B9A2F50 0B9A3628 BF9A6FDC 05C0D9F2 *3<-
RLI00.n.y.....&.....&...?.%R2*
0B9A4A20 606EF4E4 D9C9F1F5 00000784 0B95F648 0B9A2F50 0B9A3628 BF9A6FDC 05C0F572 *-
>4URI15...d.n6...&.....&...?.%5.*
0B9A4A40 F34C60E4 D9C9F1F5 0B95D0DC 0000030E 0B9A2F50 0B9A3628 BF9A6FDC 094FABB2 *3<-
URI15.n.....&.....&...?.%|. *
0B9A4A60 606EF4D9 E2E5F0F0 00000796 0B998AE8 0B9A2F50 0B9A3628 BF9A6FDC 094FD6B2 *-
>4RSV00...o.r.Y...&.....&...?.%|0.*
0B9A4A80 F34C60D9 E2E5F0F0 0B95D0EE 00000234 0B9A2F50 0B9A3628 BF9A6FDC 09523432 *3<-
RSV00.n.....&.....&...?.%.*
0B9A4AA0 606EF4E4 D9C9F1D7 000007F4 0B95AC48 0B9A2F50 0B9A3628 BF9A6FDC 09525272 *-
>4URI1P...4.n.....&.....&...?.%.*
0B9A4AC0 D6D7C5D5 C5D9F140 0B99E480 00000000 00000000 00000000 BF9A6FDC 09525BF2
*OPENER1 .rU.....&...?.%$2*
0B9A4AE0 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

```

*.....*
  LINES      0B9A4B00-0B9A4B1F      SAME AS THE ABOVE
0B9A4B20 C4E2D7E2 E3C1C3D2 8B95AEBE 0B95AC48 0B9A3628 0B9A3980 BF9A6FDC 09526932
*DSPSTACK.n...n.....?.....*
0B9A4B40 606EF5E4 D9C9F1D7 000008D0 0B95AC6E 0B9A3628 0B9A39D0 BF9A6FDC 174A3EF5 *-
>5URI1P...n.>.....?.....5*
0B9A4B60 C4E2D7E2 E3C1C3D2 8B95B7BC 0B95AC6E 0B9A39D0 0B9A3D28 BF9A6FDC 174A58B5
*DSPSTACK.n...n.>.....?.....*
0B9A4B80 F44C60E4 D9C9F1D7 0B95B518 00000CE0 0B9A3628 0B9A39D0 BF9A6FDC 1808143C *4<-
URI1P.n.....?.....*
0B9A4BA0 D6D7C5D5 C5D9F240 0B99E480 00000000 00000000 00000000 BF9A6FDC 1808233C
*OPENER2 .rU.....?.....*
0B9A4BC0 00000000 00000000 00000000 00000000 00000000 00024000 00024000 00023000
*.....*
0B9A4BE0 00002000 00000020 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0B9A4C00 F34C60E4 D9C9F1D7 0B95D14C 00000B30 0B9A2F50 0B9A3628 BF9A6FDC 1809373C *3<-
URI1P.nJ<.....&.....?.....*
0B9A4C20 606EF4E4 D9C9F1D7 000007F4 0B95AC48 0B9A2F50 0B9A3628 BF9A6FDC 180957FC *-
>4URI1P...4.n.....&.....?.....*
0B9A4C40 D6D7C5D5 C5D9F140 0B99E480 00000000 00000000 00000000 BF9A6FDC 1809603C
*OPENER1 .rU.....?.....*
0B9A4C60 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
  LINES      0B9A4C80-0B9A4C9F      SAME AS THE ABOVE
0B9A4CA0 C4E2D7E2 E3C1C3D2 8B95AEBE 0B95AC48 0B9A3628 0B9A3980 BF9A6FDC 1809667C
*DSPSTACK.n...n.....?.....@*
0B9A4CC0 606EF5E4 D9C9F1D7 000008D0 0B95AC6E 0B9A3628 0B9A39D0 BF9A6FDC 1E95B4FE *-
>5URI1P...n.>.....?.....n..*
0B9A4CE0 C4E2D7E2 E3C1C3D2 8B95B7BC 0B95AC6E 0B9A39D0 0B9A3D28 BF9A6FDC 1E95CDBE
*DSPSTACK.n...n.>.....?.....n..*
0B9A4D00 F44C60E4 D9C9F1D7 0B95B518 00000CE0 0B9A3628 0B9A39D0 BF9A6FDC 1F48747E *4<-
URI1P.n.....?.....=*
0B9A4D20 D6D7C5D5 C5D9F240 0B99E480 00000000 00000000 00000000 BF9A6FDC 1F487FBE
*OPENER2 .rU.....?.....".....*
0B9A4D40 00000000 00000000 00000000 00000000 00000000 00024000 00024000 00023000
*.....*
0B9A4D60 00002000 00000020 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0B9A4D80 F34C60E4 D9C9F1D7 0B95D14C 00000B30 0B9A2F50 0B9A3628 BF9A6FDC 1F4A063E *3<-
URI1P.nJ<.....&.....?.....*
0B9A4DA0 606EF4E4 D9C9F1D7 000007F4 0B95AC48 0B9A2F50 0B9A3628 BF9A6FDC 1F4A1E7E *-
>4URI1P...4.n.....&.....?.....=*
0B9A4DC0 D6D7C5D5 C5D9F140 0B99E480 00000000 00000000 00000000 BF9A6FDC 1F4A267E
*OPENER1 .rU.....?.....=*

0B9A4DE0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
  LINES      0B9A4E00-0B9A4E1F      SAME AS THE ABOVE
0B9A4E20 C4E2D7E2 E3C1C3D2 8B95AEBE 0B95AC48 0B9A3628 0B9A3980 BF9A6FDC 1F4A2EFE
*DSPSTACK.n...n.....?.....*
0B9A4E40 D6D7C5D5 C5D9F240 0B99E480 00000000 00000000 00000000 BF9A6FDC 2576658C
*OPENER2 .rU.....?.....*
0B9A4E60 00000000 00000000 00000000 00000000 00000000 00000000 00024000 00023000
*.....*
0B9A4E80 00002000 00000020 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0B9A4EA0 F34C60E4 D9C9F1D7 0B95D14C 00000B30 0B9A2F50 0B9A3628 BF9A6FDC 2577744C *3<-
URI1P.nJ<.....&.....?.....<*
0B9A4EC0 606EF4E4 D9C9F2F0 00000B6E 0B95FDCE 0B9A2F50 0B9A3628 BF9A6FDC 26B6A8B8 *-
>4URI20...>.n.....&.....?.....y.*
0B9A4EE0 606EF5E4 C1D3D3F0 000005A6 0B999BD0 0B9A3628 0B9A3A08 BF9A6FDC 2B69DCF8 *-
>5UALL0...w.I.....?.....8*
0B9A4F00 F44C60E4 C1D3D3F0 0B960374 000002AE 0B9A3628 0B9A3A08 BF9A6FDC 2B8387F8 *4<-
UALL0.o.....?.....cg8*
0B9A4F20 F34C60E4 D9C9F2F0 0B95D4C6 000005D4 0B9A2F50 0B9A3628 BF9A6FDC 2B8397B8 *3<-
URI20.nMF...M...&.....?.....cp.*
0B9A4F40 606EF4C4 C5D8F0F0 00000B8E 0B90DE68 0B9A2F50 0B9A3628 BF9A6FDC 2B83B078 *-
>4DEQ00...&.....?.....c..*
0B9A4F60 F34C60C4 C5D8F0F0 0B95D4E6 000007DC 0B9A2F50 0B9A3628 BF9A6FDC 2B842AF8 *3<-
DEQ00.nMW.....&.....?.....d.8*
0B9A4F80 606EF4E4 D9C9F1D7 00001FE6 0B95AC48 0B9A2F50 0B9A3628 BF9A6FDC 30A7E3BA *-
>4URI1P...W.n.....&.....?.....xT.*
0B9A4FA0 D6D7C5D5 C5D9F140 0B99E480 00000000 00000000 00000000 BF9A6FDC 30A7FD3A
*OPENER1 .rU.....?.....x...*
0B9A4FC0 00000000 00000000 00024000 00024000 00000000 00024000 00024000 00023000
*.....*
0B9A4FE0 00002000 00000020 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0B9A5000 C4E2D7E2 E3C1C3D2 8B95AEBE 0B95AC48 0B9A3628 0B9A3980 BF9A6FDC 30A80DBA
*DSPSTACK.n...n.....?.....y..*

```

```

0B9A5020 606EF5E4 D9C9F1D7 000008D0 0B95AC6E 0B9A3628 0B9A39D0 BF9A6FDC 37330F38 *-
>5URI1P.....n.>.....?.....*
0B9A5040 C4E2D7E2 E3C1C3D2 8B95B7BC 0B95AC6E 0B9A39D0 0B9A3D28 BF9A6FDC 373324B8
*DSPSTACK.n...n.>.....?.....*
0B9A5060 F44C60E4 D9C9F1D7 0B95B518 00000CE0 0B9A3628 0B9A39D0 BF9A6FDC 37EE8FB8 *4<-
URI1P.n.....?.....*
0B9A5080 D6D7C5D5 C5D9F240 0B99E480 00000000 00000000 00000000 BF9A6FDC 37EE9BB8
*OPENER2 .rU.....?.....*
0B9A50A0 00000000 00000000 00024000 00024000 0000000C 00024000 00024000 00023000
*.....*
0B9A50C0 00002000 00000020 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0B9A50E0 F34C60E4 D9C9F1D7 0B95E93E 00000B30 0B9A2F50 0B9A3628 BF9A6FDC 37EEA778 *3<-
URI1P.nZ.....&.....?....x.*
0B9A5100 606EF4E4 D9C9F1D7 00001FE6 0B95AC48 0B9A2F50 0B9A3628 BF9A6FDC 3C754CF5 *-
>4URI1P...W.n.....&.....?....<5*
0B9A5120 D6D7C5D5 C5D9F140 0B99E480 00000000 00000000 00000000 BF9A6FDC 3C7560B5
*OPENER1 .rU.....?....-.*
0B9A5140 00000000 00000000 00024000 00024000 0000000C 00024000 00024000 00023000
*.....*
0B9A5160 00002000 00000020 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0B9A5180 C4E2D7E2 E3C1C3D2 8B95AEBE 0B95AC48 0B9A3628 0B9A3980 BF9A6FDC 3C756AF5
*DSPSTACK.n...n.....?....5*
0B9A51A0 606EF5E4 D9C9F1D7 000008D0 0B95AC6E 0B9A3628 0B9A39D0 BF9A6FDC 42F77137 *-
>5URI1P.....n.>.....?....7.*
0B9A51C0 C4E2D7E2 E3C1C3D2 8B95B7BC 0B95AC6E 0B9A39D0 0B9A3D28 BF9A6FDC 42F78477
*DSPSTACK.n...n.>.....?....7d.*
0B9A51E0 F44C60E4 D9C9F1D7 0B95B518 00000CE0 0B9A3628 0B9A39D0 BF9A6FDC 43AB3B78 *4<-
URI1P.n.....?.....*
0B9A5200 D6D7C5D5 C5D9F240 0B99E480 00000000 00000000 00000000 BF9A6FDC 43AB4A78
*OPENER2 .rU.....?.....*
0B9A5220 00000000 00000000 00024000 00024000 0000000C 00024000 00024000 00023000
*.....*
0B9A5240 00002000 00000020 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0B9A5260 F34C60E4 D9C9F1D7 0B95E93E 00000B30 0B9A2F50 0B9A3628 BF9A6FDC 43AB5678 *3<-
URI1P.nZ.....&.....?.....*
0B9A5280 606EF4E4 D9C9F3F0 00000D1E 0B961C46 0B9A2F50 0B9A3628 BF9A6FDC 43AB8B38 *-
>4URI30...o.....&.....?.....*
0B9A52A0 F34C60E4 D9C9F3F0 0B95D676 00000F0E 0B9A2F50 0B9A3628 BF9A6FDC 45367C7E *3<-
URI30.nO.....&.....?....@=*
0B9A52C0 F24C60E4 D9C9F1F0 0B955558 00000458 0B9A2780 0B9A2F50 BF9A6FDC 4536B13E *2<-
URI10.n.....&...?.....*
0B9A52E0 606EF3E4 D9C9F5F0 00002938 0B969170 0B9A2780 0B9A2F50 BF9A6FDC 4537053E *-
>3URI50...o.j.....&...?.....*
0B9A5300 C4E2D7E2 E3C1C3D2 8B96930A 0B969170 0B9A2F50 0B9A35A8 BF9A6FDC 45370F3E
*DSPSTACK.ol..oj.....&...y..?.....*
0B9A5320 C7C5E3C6 C5C5C440 0B99E480 00000000 00000000 00000000 BF9A6FDC 4537967E
*GETFEED .rU.....?....o=*
0B9A5340 00000000 000000280 00024000 00024000 0000000C 00024000 00024000 00023000
*.....*
0B9A5360 00002000 80500000 00000000 0B9012C4 00000280 00000100 C7C5E340 40400000
*.....&.....D.....GET.....*
0B9A5380 F24C60E4 D9C9F5F0 0B9573D0 000002CE 0B9A2780 0B9A2F50 BF9A6FDC 45379EBE *2<-
URI50.n.....&...?.....*
0B9A53A0 C7C5E3C6 C5C5C440 0B99E480 00000000 00000000 00000000 BF9A6FDC 4537B47E
*GETFEED .rU.....?....=*
0B9A53C0 00000000 000000280 00024000 00024000 0000000C 00024000 00024000 00023000
*.....*
0B9A53E0 00002000 80500000 00000000 0BB88000 00000280 00000100 C7C5E340 40400000
*.....&.....GET.....*
0B9A5400 606EF3C4 C5D8F0F0 0000249A 0B90DE68 0B9A2780 0B9A2F50 BF9A6FDC 4539897E *-
>3DEQ00.....&...?....i=*
0B9A5420 F24C60C4 C5D8F0F0 0B956F32 000007DC 0B9A2780 0B9A2F50 BF9A6FDC 453AD3BE *2<-
DEQ00.n?...&...?....L.*
0B9A5440 F14C60E4 D9C9F0F1 0B95016C 0000022E 0B9A2010 0B9A2780 BF9A6FDC 453B1E7E *1<-
URI01.n.%.....?....=*
0B9A5460 C4E2D7E4 D9C9F0F0 0B99E480 00000000 00000000 00000000 BF9A6FDC 453B377E
*DSPURI00.rU.....?....=*
0B9A5480 C5D5C4C9 D9C3C1D9 F1F201D8 D6000600 D7C8E8E2 C9C3C1D3 40D6D7C5 D5404040
*ENDIRCAR12.QO...PHYSICAL OPEN *
0B9A54A0 40404040 40404040 40404040 40404040 40404040 40404040 00000000 00000000
*.....*
0B9A54C0 F04C60E4 D9C9F0F0 0B99D5B4 00000B9E 0B901E48 0B9A2010 BF9A6FDC 453B42BE *0<-
URI00.iN.....?.....*
0B9A54E0 C4E2D7C3 D9E3D9F0 0A7EFCB4 17172002 00000000 0B99E480 BF9A6FF2 C2EBE40E
*DSPCRRTR0.=.....rU.....?2B.U.*
0B9A5500 00000000 00000000 00CB3D28 00000001 0B99E480 00000000 00000000 00000000
*.....rU.....*
0B9A5520 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*

```

0B9A5540	606EF1E2 E2C9C7D5 8000C4D4	0B936468	000067B8	0B9A2010	BF9A6FF2 C2EC458E	*-
>1SSIGN..DM.l.....?2B...*						
0B9A5560	606EF2E4 D9C9F0F0 00000258	0B94F9C8	0B9A2010	0B9A2220	BF9A6FF2 C2EC834E	*-
>2URI00....m9H.....?2B.c+*						
0B9A5580	C4E2D7E4 D9C9F0F0 0B99E480	00000000	00000000	00000000	BF9A6FF2 C2ECAFCF	
*DSPURI00.rU.....?2B...*						
0B9A55A0	D4D6C4C9 D9C3C1D9 F1F201D8	D6000000	40D3D6C7	C9C3C1D3	40D6D7C5 D5404040	*MODIRCAR12.Q0...
LOGICAL OPEN *						
0B9A55C0	40404040 40404040 40404040	40404040	40404040	40404040	00000000 00000000	
*.....*						
0B9A55E0	606EF3E4 D9C9F0F1 000007A4	0B954A98	0B9A2220	0B9A2990	BF9A6FF2 C2ED55CE	*-
>3URI01...u.n.q.....?2B...*						
0B9A5600	606EF4D9 E2E5F0F0 00000EC0	0B998AE8	0B9A2990	0B9A3160	BF9A6FF2 C2ED828E	*-
>4RSV00....r.Y.....-..?2B.b.*						
0B9A5620	F34C60D9 E2E5F0F0 0B955958	00000234	0B9A2990	0B9A3160	BF9A6FF2 C2F2958E	*3<-
RSV00.n.....?2B2n.*						
0B9A5640	606EF4E4 D9C9F3F0 00000F76	0B961C4C	0B9A2990	0B9A3160	BF9A6FF2 C2F2D40E	*-
>4URI30....o.<.....?2B2M.*						
0B9A5660	606EF5E4 D9C9F2F0 00001610	0B95FDDA	0B9A3160	0B9A3AF8	BF9A6FF2 C3FE36FC	*-
>5URI20....n.....8...?2C...*						

0B9A5680	F44C60E4 D9C9F2F0 0B96325C	00000A10	0B9A3160	0B9A3AF8	BF9A6FF2 C3FE683C	*4<-
URI20.o.*.....8...?2C...*						
0B9A56A0	F34C60E4 D9C9F3F0 0B955A0E	00001648	0B9A2990	0B9A3160	BF9A6FF2 C3FE703C	*3<-
URI30.n!.....?2C...*						
0B9A56C0	606EF4E4 D9C9F5F0 00002938	0B969170	0B9A2990	0B9A3160	BF9A6FF2 C3FF047C	*-
>4URI50....oj.....?2C..@*						
0B9A56E0	C4E2D7E2 E3C1C3D2 8B96930A	0B969170	0B9A3160	0B9A37B8	BF9A6FF2 C3FF11BC	
*DSPSTACK.ol..oj.....?2C...*						
0B9A5700	C7C5E3C6 C5C5C440 0B99E480	00000000	00000000	00000000	BF9A6FF2 C3FFBC3C	
*GETFEED.rU.....?2C...*						
0B9A5720	00000000 00000280 00024000	00024000	0000000C	00024000	00024000 00023000	
*.....*						
0B9A5740	00002000 80500000 0B9D4878	0B9012C4	00000280	00000100	C7C5E340 40400000	
*.....&.....D.....GET...*						
0B9A5760	F34C60E4 D9C9F5F0 0B9573D0	000002CE	0B9A2990	0B9A3160	BF9A6FF2 C3FFC6FC	*3<-
URI50.n.....?2C.F.*						
0B9A5780	C7C5E3C6 C5C5C440 0B99E480	00000000	00000000	00000000	BF9A6FF2 C3FFDBFC	
*GETFEED.rU.....?2C...*						
0B9A57A0	00000000 00000280 00024000	00024000	0000000C	00024000	00024000 00023000	
*.....*						
0B9A57C0	00002000 80500000 0B9D4878	0BB84000	00000280	00000100	C7C5E340 40400000	
*.....&.....D.....GET...*						
0B9A57E0	F24C60E4 D9C9F0F1 0B95016C	0000022E	0B9A2220	0B9A2990	BF9A6FF2 C400423C	*2<-
URI01.n.%.....?2D...*						
0B9A5800	F14C60E4 D9C9F0F0 0B9366C0	00000B9E	0B9A2010	0B9A2220	BF9A6FF2 C4005CBC	*1<-
URI00.l.....?2D.*.*						
0B9A5820	606EF2E4 D9C9F0F0 000003F8	0B94F9C8	0B9A2010	0B9A2220	BF9A6FF2 C40084BC	*-
>2URI00...8.m9H.....?2D.d.*						
0B9A5840	C4E2D7E4 D9C9F0F0 0B99E480	00000000	00000000	00000000	BF9A6FF2 C40093BC	
*DSPURI00.rU.....?2D.l.*						
0B9A5860	D4D6C4C9 D9C3C1D9 F1F201D8	D3002000	C4C9D9C5	C3E340D3	D6C3C1E3 C5404040	
*MODIRCAR12.QL...DIRECT LOCATE *						
0B9A5880	FFFFFFFF FFFFFFFF C9D4E2F1	40404040	3F000000	00000000	00000000 00000000	
*.....IMS1.....*						
0B9A58A0	606EF3E4 D9C9F5F0 000038E6	0B969170	0B9A2220	0B9A2990	BF9A6FF2 C400C9BC	*-
>3URI50...w.oj.....?2D.I.*						
0B9A58C0	C4E2D7E2 E3C1C3D2 8B96930A	0B969170	0B9A2990	0B9A2FE8	BF9A6FF2 C400CFFC	
*DSPSTACK.ol..oj.....Y..?2D...*						
0B9A58E0	C7C5E3C6 C5C5C440 0B99E480	00000000	00000000	00000000	BF9A6FF2 C40155BC	
*GETFEED.rU.....?2D...*						
0B9A5900	A9080010 00000280 00024000	00024000	0000000C	00024000	00024000 00023000	
*Z.....*						
0B9A5920	00002000 80500000 0B9D4878	0B9012C4	00000280	00000100	C7C5E340 40400000	
*.....&.....D.....GET...*						
0B9A5940	C7C5E3C6 C5C5C440 0B99E480	00000000	00000000	00000000	BF9A6FF2 C401F87C	
*GETFEED.rU.....?2D.80*						
0B9A5960	04080004 00000280 00024000	00024000	0000000C	00024000	00024000 00023000	
*.....*						
0B9A5980	00002000 80500000 0B9D4878	0B901828	00000280	00000103	D7D6C9D5 E3400000	
*.....&.....D.....POINT...*						
0B9A59A0	F24C60E4 D9C9F5F0 0B9532AE	000002CE	0B9A2220	0B9A2990	BF9A6FF2 C401FD7C	*2<-
URI50.n.....?2D..@*						
0B9A59C0	C7C5E3C6 C5C5C440 0B99E480	00000000	00000000	00000000	BF9A6FF2 C4020DFC	
*GETFEED.rU.....?2D...*						
0B9A59E0	04080004 00000280 00024000	00024000	0000000C	00024000	00024000 00023000	
*.....*						
0B9A5A00	00002000 80500000 0B9D4878	0BB84000	00000280	00000103	D7D6C9D5 E3400000	
*.....&.....D.....POINT...*						
0B9A5A20	C4E2D7E4 D9C9F0F0 0B99E480	00000000	00000000	00000000	BF9A6FF2 C4021A7C	
*DSPURI00.rU.....?2D..@*						



0B9A5A40	C5D5C4C9	D9C3C1D9	F1F20004	D3002000	D9C5C3D6	D9C440D5	D6E340C6	D6E4D5C4	
*ENDIRCAR12..L...RECORD NOT FOUND*									
0B9A5A60	FFFFFFFF	FFFFFFFF	C9D4E2F1	40404040	3F000000	00000000	00000000	00000000	
*.....IMS1									
0B9A5A80	F14C60E4	D9C9F0F0	0B936860	00000B9E	0B9A2010	0B9A2220	BF9A6FF2	C4021E7C	*1<-
URI00.1.-.....?2D..@*									
0B9A5AA0	606EF2E4	D9E3F0F0	00000512	0B971360	0B9A2010	0B9A2220	BF9A6FF2	C40237BC	*-
>2URT00.....p.-.....?2D...*									
0B9A5AC0	606EF3C3	C8D2E6C4	00000278	0B90A010	0B9A2220	0B9A2858	BF9A6FF2	C40248BC	*-
>3CHKWD.....?2D...*									
0B9A5AE0	F24C60C3	C8D2E6C4	0B9715D8	0000017C	0B9A2220	0B9A2858	BF9A6FF2	C4025FBC	*2<-
CHKWD.p.Q...@.....?2D.^.*									
0B9A5B00	606EF3E4	D9C9F0F0	0000037A	0B94F9C8	0B9A2220	0B9A2858	BF9A6FF2	C4026F3C	*-
>3URI00.....m9H.....?2D.?.*									
0B9A5B20	C4E2D7E4	D9C9F0F0	0B99E480	00000000	00000000	00000000	BF9A6FF2	C402753C	
*DSPURI00.rU.....?2D...*									
0B9A5B40	D4D6C4C9	D9C3C1D9	F1F201D8	D6002000	40D3D6C7	C9C3C1D3	40D6D7C5	D5404040	*MODIRCAR12.Q0...
LOGICAL OPEN *									
0B9A5B60	40404040	40404040	40404040	40404040	40404040	40404040	00000000	00000000	
*.....*									
0B9A5B80	606EF4E4	D9C9F0F1	000007A4	0B954A98	0B9A2858	0B9A2FC8	BF9A6FF2	C40291BC	*-
>4URI01...u.n.q.....H...?2D.j.*									
0B9A5BA0	F34C60E4	D9C9F0F1	0B95016C	0000022E	0B9A2858	0B9A2FC8	BF9A6FF2	C402A07C	*3<-
URI01.n.%.....H...?2D..@*									
0B9A5BC0	F24C60E4	D9C9F0F0	0B9716DA	00000B9E	0B9A2220	0B9A2858	BF9A6FF2	C402A63C	*2<-
URI00.p.....?2D.w.*									
0B9A5BE0	606EF3E4	D9E3F7F0	000030C0	0B97F450	0B9A2220	0B9A2858	BF9A6FF2	C402C4BC	*-
>3URT70.....p4&.....?2D.D.*									
0B9A5C00	606EF4E4	D9C9F0F0	0000023E	0B94F9C8	0B9A2858	0B9A2948	BF9A6FF2	C402D7BC	*-
>4URI00.....m9H.....?2D.P.*									
0B9A5C20	C4E2D7E4	D9C9F0F0	0B99E480	00000000	00000000	00000000	BF9A6FF2	C402E07C	
*DSPURI00.rU.....?2D..@*									
0B9A5C40	D4D6C4C9	D9C3C1D9	F1F201D8	C2002000	C2C5C740	D4E4D3E3	40E4D7C4	C1E3C540	
*MODIRCAR12.QB...BEG MULT UPDATE *									
0B9A5C60	40404040	40404040	40404040	40404040	40404040	40404040	00000000	00000000	
*.....*									
0B9A5C80	606EF5E4	D9C9F4F0	00000968	0B966886	0B9A2948	0B9A30B8	BF9A6FF2	C402FD7C	*-
>5URI40.....o.f.....?2D..@*									
0B9A5CA0	606EF6E4	D9C9F3F0	00000236	0B961C46	0B9A30B8	0B9A34C0	BF9A6FF2	C4030EFC	*-
>6URI30.....o.....?2D...*									
0B9A5CC0	F54C60E4	D9C9F3F0	0B966ABC	00000F0E	0B9A30B8	0B9A34C0	BF9A6FF2	C483E738	*5<-
URI30.o.....?2DcX.*									
0B9A5CE0	F44C60E4	D9C9F4F0	0B950330	00000256	0B9A2948	0B9A30B8	BF9A6FF2	C483FA78	*4<-
URI40.n.....?2Dc...*									
0B9A5D00	F34C60E4	D9C9F0F0	0B97F68E	00000B9E	0B9A2858	0B9A2948	BF9A6FF2	C48408F8	*3<-
URI00.p6.....?2Dd.8.*									
0B9A5D20	606EF4E4	D9C9F0F0	000002B8	0B94F9C8	0B9A2858	0B9A2948	BF9A6FF2	C4842138	*-
>4URI00.....m9H.....?2Dd..*									
0B9A5D40	C4E2D7E4	D9C9F0F0	0B99E480	00000000	00000000	00000000	BF9A6FF2	C4842A38	
*DSPURI00.rU.....?2Dd...*									
0B9A5D60	D4D6C4C9	D9C3C1D9	F1F201D8	D3002000	C4C9D9C5	C3E340D3	D6C3C1E3	C5404040	
*MODIRCAR12.QL...DIRECT LOCATE *									
0B9A5D80	FFFFFFFF	FFFFFFFF	C9D4E2F1	40404040	3F000000	00000000	00000000	00000000	
*.....IMS1									
0B9A5DA0	606EF5E4	D9C9F5F0	000038E6	0B969170	0B9A2948	0B9A30B8	BF9A6FF2	C4846A78	*-
>5URI50...w.oj.....?2Dd..*									
0B9A5DC0	C4E2D7E2	E3C1C3D2	8B96930A	0B969170	0B9A30B8	0B9A3710	BF9A6FF2	C48471B8	
*DSPSTACK.ol..oj.....?2Dd...*									
0B9A5DE0	C7C5E3C6	C5C5C440	0B99E480	00000000	00000000	00000000	BF9A6FF2	C484FB78	
*GETFEED.rU.....?2Dd...*									
0B9A5E00	A9080010	00000280	00024000	00024000	0000000C	00024000	00024000	00023000	
*Z.....*									
0B9A5E20	00002000	80500000	0B9D4878	0B9012C4	00000280	00000100	C7C5E340	40400000	
*.....&.....D.....GET									
0B9A5E40	C7C5E3C6	C5C5C440	0B99E480	00000000	00000000	00000000	BF9A6FF2	C4856138	
*GETFEED.rU.....?2De/.*									
0B9A5E60	04080004	00000280	00024000	00024000	0000000C	00024000	00024000	00023000	
*.....*									
0B9A5E80	00002000	80500000	0B9D4878	0B901828	00000280	00000103	D7D6C9D5	E3400000	
*.....&.....POINT									
0B9A5EA0	F44C60E4	D9C9F5F0	0B9532AE	000002CE	0B9A2948	0B9A30B8	BF9A6FF2	C4856638	*4<-
URI50.n.....?2De...*									
0B9A5EC0	C7C5E3C6	C5C5C440	0B99E480	00000000	00000000	00000000	BF9A6FF2	C4856DB8	
*GETFEED.rU.....?2De...*									
0B9A5EE0	04080004	00000280	00024000	00024000	0000000C	00024000	00024000	00023000	
*.....*									
0B9A5F00	00002000	80500000	0B9D4878	00000000	00000280	00000103	D7D6C9D5	E3400000	
*.....&.....POINT									
0B9A5F20	C4E2D7E4	D9C9F0F0	0B99E480	00000000	00000000	00000000	BF9A6FF2	C4857778	
*DSPURI00.rU.....?2De...*									

0B9A5F40	C5D5C4C9	D9C3C1D9	F1F20004	D3002000	D9C5C3D6	D9C440D5	D6E340C6	D6E4D5C4	
*ENDIRCAR12...L...RECORD NOT FOUND*									
0B9A5F60	FFFFFFFF	FFFFFFFF	C9D4E2F1	40404040	3F000000	00000000	00000000	00000000	
*.....IMS1.....*									
0B9A5F80	F34C60E4	D9C9F0F0	0B97F708	00000B9E	0B9A2858	0B9A2948	BF9A6FF2	C4857B78	*3<-
URI00.p7.....?2De#.*									
0B9A5FA0	606EF4E4	D9C9F0F0	000003C0	0B94F9C8	0B9A2858	0B9A2948	BF9A6FF2	C4858B38	*-
>4URI00....m9H.....?2De.*									
0B9A5FC0	C4E2D7E4	D9C9F0F0	0B99E480	00000000	00000000	00000000	BF9A6FF2	C4859238	
*DSPURI00.rU.....?2Dek.*									
0B9A5FE0	D4D6C4C9	D9C3C1D9	F1F20070	E6082000	C9D5E2C5	D9E340D5	C5E640D9	C5C3D9C4	
*MODIRCAR12...W...INSERT NEW RECRD*									
0B9A6000	FFFFFFFF	FFFFFFFF	C9D4E2F1	40404040	3F000000	00000000	00000000	00000000	
*.....IMS1.....*									
0B9A6020	606EF5E4	D9C9F4F0	000020D2	0B966892	0B9A2948	0B9A30B8	BF9A6FF2	C485C538	*-
>5URI40...K.o.k.....?2DeE.*									
0B9A6040	606EF6E4	D9C9F5F0	00000E64	0B969170	0B9A30B8	0B9A34C0	BF9A6FF2	C485E0B8	*-
>6URI50....oj.....?2De.*									
0B9A6060	C4E2D7E2	E3C1C3D2	8B96930A	0B969170	0B9A34C0	0B9A3B18	BF9A6FF2	C485E6B8	
*DSPSTACK.ol..oj.....?2DeW.*									
0B9A6080	C7C5E3C6	C5C5C440	0B99E480	00000000	00000000	00000000	BF9A6FF2	C48631B8	
*GETFEED.rU.....?2Df.*									
0B9A60A0	A9080010	00000000	00024000	00024000	0000000C	00024000	00024000	00023000	
*Z.....*									
0B9A60C0	00002000	80500000	0B9D4878	0B9A3444	00000000	00000100	C7C5E340	40400000	
*.....&.....GET.....*									
0B9A60E0	C7C5E3C6	C5C5C440	0B99E480	00000000	00000000	00000000	BF9A6FF2	C4868EB8	
*GETFEED.rU.....?2Df.*									
0B9A6100	04080004	00000000	00024000	00024000	0000000C	00024000	00024000	00023000	
*.....*									
0B9A6120	00002000	80500000	0B9D4878	0B9A3444	00000000	00000103	D7D6C9D5	E3400000	
*.....&.....POINT.....*									
0B9A6140	F54C60E4	D9C9F5F0	0B9676F6	000002CE	0B9A30B8	0B9A34C0	BF9A6FF2	C4869378	*5<-
URI50.o.6.....?2Df1.*									
0B9A6160	606EF6E4	D9C9F5F0	00000CA0	0B969170	0B9A30B8	0B9A34C0	BF9A6FF2	C48739B8	*-
>6URI50....oj.....?2Dg.*									
0B9A6180	C4E2D7E2	E3C1C3D2	8B96930A	0B969170	0B9A34C0	0B9A3B18	BF9A6FF2	C4873FB8	
*DSPSTACK.ol..oj.....?2Dg.*									
0B9A61A0	C7C5E3C6	C5C5C440	0B99E480	00000000	00000000	00000000	BF9A6FF2	C4D7B4C5	
*GETFEED.rU.....?2DP.E*									
0B9A61C0	00000000	00000084	00024000	00024000	0000000C	00024000	00024000	00023000	
*.....d.....*									
0B9A61E0	00002000	80500000	0B9D4878	0B9E5FA0	00000084	00000101	D7E4E340	40400000	
*.....&.....^.....d.....PUT.....*									
0B9A6200	F54C60E4	D9C9F5F0	0B967532	000002CE	0B9A30B8	0B9A34C0	BF9A6FF2	C4D7E585	*5<-
URI50.o.....?2DPVe*									
0B9A6220	606EF6E4	D9C9F5F0	00000CA0	0B969170	0B9A30B8	0B9A34C0	BF9A6FF2	C4D81D85	*-
>6URI50....oj.....?2DQ.e*									
0B9A6240	C4E2D7E2	E3C1C3D2	8B96930A	0B969170	0B9A34C0	0B9A3B18	BF9A6FF2	C4D82A45	
*DSPSTACK.ol..oj.....?2DQ.*									
0B9A6260	C7C5E3C6	C5C5C440	0B99E480	00000000	00000000	00000000	BF9A6FF2	C51C2985	
*GETFEED.rU.....?2E..e*									
0B9A6280	00000000	00000084	00024000	00024000	0000000C	00024000	00024000	00023000	
*.....d.....*									
0B9A62A0	00002000	80500000	0B9D4AF8	0BA03FA0	00000084	00000201	D7E4E340	40400000	
*.....&.....8.....d.....PUT.....*									
0B9A62C0	F54C60E4	D9C9F5F0	0B967532	000002CE	0B9A30B8	0B9A34C0	BF9A6FF2	C51C3745	*5<-
URI50.o.....?2E...*									
0B9A62E0	F44C60E4	D9C9F4F0	0B951A9A	000004C2	0B9A2948	0B9A30B8	BF9A6FF2	C51CB9C5	*4<-
URI40.n....B.....?2E..E*									
0B9A6300	606EF5E4	D9C9F5F0	00002184	0B969170	0B9A2948	0B9A30B8	BF9A6FF2	C51CD645	*-
>5URI50...d.oj.....?2E.O.*									
0B9A6320	C4E2D7E2	E3C1C3D2	8B96930A	0B969170	0B9A30B8	0B9A3710	BF9A6FF2	C51CDD45	
*DSPSTACK.ol..oj.....?2E...*									
0B9A6340	C7C5E3C6	C5C5C440	0B99E480	00000000	00000000	00000000	BF9A6FF2	C567C4BA	
*GETFEED.rU.....?2E.D.*									
0B9A6360	00000000	00000070	00024000	00024000	0000000C	00024000	00024000	00023000	
*.....*									
0B9A6380	00002000	80500000	0B9D4878	0B9E5FA0	00000070	00000101	D7E4E340	40400000	
*.....&.....^.....PUT.....*									
0B9A63A0	F44C60E4	D9C9F5F0	0B951B4C	000002CE	0B9A2948	0B9A30B8	BF9A6FF2	C567E4BA	*4<-
URI50.n.<.....?2E.U.*									
0B9A63C0	C7C5E3C6	C5C5C440	0B99E480	00000000	00000000	00000000	BF9A6FF2	C568007A	
*GETFEED.rU.....?2E...*									
0B9A63E0	00000000	00000070	00024000	00024000	0000000C	00024000	00024000	00023000	
*.....*									
0B9A6400	00002000	80500000	0B9D4878	00000000	00000070	00000101	D7E4E340	40400000	
*.....&.....PUT.....*									
0B9A6420	606EF5E4	D9C9F5F0	00002184	0B969170	0B9A2948	0B9A30B8	BF9A6FF2	C5681F7A	*-
>5URI50...d.oj.....?2E...*									
0B9A6440	C4E2D7E2	E3C1C3D2	8B96930A	0B969170	0B9A30B8	0B9A3710	BF9A6FF2	C56828FA	
*DSPSTACK.ol..oj.....?2E...*									

```

0B9A6460 C7C5E3C6 C5C5C440 0B99E480 00000000 00000000 00000000 BF9A6FF2 C5AA037C
*GETFEED .rU.....?2E..@*
0B9A6480 00000000 00000070 00024000 00024000 0000000C 00024000 00024000 00023000
*.....*
0B9A64A0 00002000 80500000 0B9D4AF8 0BA03FA0 00000070 00000201 D7E4E340 40400000
*.....8.....PUT...*
0B9A64C0 F44C60E4 D9C9F5F0 0B951B4C 000002CE 0B9A2948 0B9A30B8 BF9A6FF2 C5AA173C *4<-
URI50.n.<.....?2E...*
0B9A64E0 C7C5E3C6 C5C5C440 0B99E480 00000000 00000000 00000000 BF9A6FF2 C5AA2A3C
*GETFEED .rU.....?2E...*
0B9A6500 00000000 00000070 00024000 00024000 0000000C 00024000 00024000 00023000
*.....*
0B9A6520 00002000 80500000 0B9D4AF8 00000000 00000070 00000201 D7E4E340 40400000
*.....&.....8.....PUT...*
0B9A6540 F34C60E4 D9C9F0F0 0B97F810 00000B9E 0B9A2858 0B9A2948 BF9A6FF2 C5AA36FC *3<-
URI00.p8.....?2E...*
0B9A6560 606EF4E4 D9C9F0F0 00000448 0B94F9C8 0B9A2858 0B9A2948 BF9A6FF2 C5AA4DFC *-
>4URI00....m9H.....?2E.(.*
0B9A6580 C4E2D7E4 D9C9F0F0 0B99E480 00000000 00000000 00000000 BF9A6FF2 C5AA567C
*DSPURI00.rU.....?2E..@*
0B9A65A0 D4D6C4C9 D9C3C1D9 F1F201D8 C5082000 C5D5C440 D4E4D3E3 40E4D7C4 C1E3C540
*MODIRCAR12.QE...END MULT UPDATE *
0B9A65C0 40404040 40404040 40404040 40404040 40404040 40404040 00000000 00000000
*.....*
0B9A65E0 606EF5E4 D9C9F4F0 00000994 0B96688C 0B9A2948 0B9A30B8 BF9A6FF2 C5AA713C *-
>5URI40...m.o.....?2E...*
0B9A6600 606EF6E4 D9C9F3F0 000002B0 0B961C46 0B9A30B8 0B9A34C0 BF9A6FF2 C5AA867C *-
>6URI30....o.....?2E.f@*
0B9A6620 F54C60E4 D9C9F3F0 0B966B3C 00000F0E 0B9A30B8 0B9A34C0 BF9A6FF2 C62B48F8 *5<-
URI30.o.....?2F..8*
0B9A6640 606EF6E4 D9C9F5F0 00000E6A 0B969170 0B9A30B8 0B9A34C0 BF9A6FF2 C62C8C38 *-
>6URI50....oj.....?2F...*
0B9A6660 C4E2D7E2 E3C1C3D2 8B96930A 0B969170 0B9A34C0 0B9A3B18 BF9A6FF2 C62C94B8
*DSPSTACK.ol..oj.....?2F.m.*
0B9A6680 C7C5E3C6 C5C5C440 0B99E480 00000000 00000000 00000000 BF9A6FF2 C62D06F8
*GETFEED .rU.....?2F..8*
0B9A66A0 00000000 00000084 00024000 00024000 0000000C 00024000 00024000 00023000
*.....d.....*
0B9A66C0 00002000 80500000 0B9D4878 0BB8C7F8 00000084 00000100 C7C5E340 40400000
*.....&.....G8...d....GET...*
0B9A66E0 F54C60E4 D9C9F5F0 0B9676F6 000002CE 0B9A30B8 0B9A34C0 BF9A6FF2 C62D2D38 *5<-
URI50.o.6.....?2F...*
0B9A6700 606EF6E4 D9C9F5F0 000012E6 0B969170 0B9A30B8 0B9A34C0 BF9A6FF2 C62D3F78 *-
>6URI50...w.oj.....?2F...*
0B9A6720 C4E2D7E2 E3C1C3D2 8B96930A 0B969170 0B9A34C0 0B9A3B18 BF9A6FF2 C62D4578
*DSPSTACK.ol..oj.....?2F...*
0B9A6740 C7C5E3C6 C5C5C440 0B99E480 00000000 00000000 00000000 BF9A6FF2 C62D9B38
*GETFEED .rU.....?2F...*
0B9A6760 00000000 00000084 00024000 00024000 0000000C 00024000 00024000 00023000
*.....d.....*
0B9A6780 00002000 80500000 0B9D4878 0B9A36A0 00000084 00000100 C7C5E340 40400000
*.....&.....d....GET...*

```

```

0B9A67A0 C7C5E3C6 C5C5C440 0B99E480 00000000 00000000 00000000 BF9A6FF2 C66CAE78
*GETFEED .rU.....?2F%...*
0B9A67C0 00000000 00000084 00024000 00024000 0000000C 00024000 00024000 00023000
*.....d.....*
0B9A67E0 00002000 80500000 0B9D4878 0B9A36A0 00000084 00000105 C5D9C1E2 C5400000
*.....&.....d....ERASE...*
0B9A6800 F54C60E4 D9C9F5F0 0B967B72 000002CE 0B9A30B8 0B9A34C0 BF9A6FF2 C66CCD78 *5<-
URI50.o#. ....?2F%...*
0B9A6820 606EF6E4 D9C9F5F0 00000E6A 0B969170 0B9A30B8 0B9A34C0 BF9A6FF2 C66CE0B8 *-
>6URI50....oj.....?2F%...*
0B9A6840 C4E2D7E2 E3C1C3D2 8B96930A 0B969170 0B9A34C0 0B9A3B18 BF9A6FF2 C66CE7B8
*DSPSTACK.ol..oj.....?2F%X.*
0B9A6860 C7C5E3C6 C5C5C440 0B99E480 00000000 00000000 00000000 BF9A6FF2 C66D5138
*GETFEED .rU.....?2F...*
0B9A6880 00000000 00000084 00024000 00024000 0000000C 00024000 00024000 00023000
*.....d.....*
0B9A68A0 00002000 80500000 0B9D4AF8 0BB8C7F8 00000084 00000200 C7C5E340 40400000
*.....&.....8...G8...d....GET...*
0B9A68C0 F54C60E4 D9C9F5F0 0B9676F6 000002CE 0B9A30B8 0B9A34C0 BF9A6FF2 C66D67F8 *5<-
URI50.o.6.....?2F..8*
0B9A68E0 606EF6E4 D9C9F5F0 000012E6 0B969170 0B9A30B8 0B9A34C0 BF9A6FF2 C66D7738 *-
>6URI50...w.oj.....?2F...*
0B9A6900 C4E2D7E2 E3C1C3D2 8B96930A 0B969170 0B9A34C0 0B9A3B18 BF9A6FF2 C66D7D38
*DSPSTACK.ol..oj.....?2F'..*
0B9A6920 C7C5E3C6 C5C5C440 0B99E480 00000000 00000000 00000000 BF9A6FF2 C66DC238
*GETFEED .rU.....?2F_B.*
0B9A6940 00000000 00000084 00024000 00024000 0000000C 00024000 00024000 00023000
*.....d.....*

```

0B9A6960	00002000	80500000	0B9D4AF8	0B9A36A0	00000084	00000200	C7C5E340	40400000	
*.....&.....8.....d....GET	...	...	...	...	...	...	...	...	
0B9A6980	C7C5E3C6	C5C5C440	0B99E480	00000000	00000000	00000000	BF9A6FF2	C6AA75F8	
*GETFEED	.rU.....	..?2F..8*							
0B9A69A0	00000000	00000084	00024000	00024000	0000000C	00024000	00024000	00023000	
*.....d.....	.....	.....	.....	.....	.....	.....	.....	.....	
0B9A69C0	00002000	80500000	0B9D4AF8	0B9A36A0	00000084	00000205	C5D9C1E2	C5400000	
*.....&.....8.....d....ERASE	...	...	...	...	...	...	...	...	
0B9A69E0	F54C60E4	D9C9F5F0	0B967B72	000002CE	0B9A30B8	0B9A34C0	BF9A6FF2	C6AA84F8	*5<-
URI50.o#.....	..?2F.d8*								
0B9A6A00	606EF6E4	D9C9F5F0	00000E6A	0B969170	0B9A30B8	0B9A34C0	BF9A6FF2	C6AA9338	*-
>6URI50.....oj.....	..?2F.l.*								
0B9A6A20	C4E2D7E2	E3C1C3D2	8B96930A	0B969170	0B9A34C0	0B9A3B18	BF9A6FF2	C6AA99B8	
*DSPSTACK.ol..oj.....	..?2F.r.*								
0B9A6A40	C7C5E3C6	C5C5C440	0B99E480	00000000	00000000	00000000	BF9A6FF2	C6AAF3B8	
*GETFEED	.rU.....	..?2F.3.*							
0B9A6A60	00000000	000000B0	00024000	00024000	0000000C	00024000	00024000	00023000	
*.....	.....	.....	.....	.....	.....	.....	.....	.....	
0B9A6A80	00002000	80500000	0B9D4878	0BB8C7F8	000000B0	00000100	C7C5E340	40400000	
*.....&.....G8.....GET	...	...	...	...	...	...	...	...	
0B9A6AA0	F54C60E4	D9C9F5F0	0B9676F6	000002CE	0B9A30B8	0B9A34C0	BF9A6FF2	C6AB0878	*5<-
URI50.o.6.....	..?2F...*								
0B9A6AC0	606EF6E4	D9C9F3F0	000002CE	0B961C46	0B9A30B8	0B9A34C0	BF9A6FF2	C6AC60F8	*-
>6URI30.....o.....	..?2F.-8*								
0B9A6AE0	F54C60E4	D9C9F3F0	0B966B5A	00000F0E	0B9A30B8	0B9A34C0	BF9A6FF2	C72A23F8	*5<-
URI30.o,!.....	..?2G..8*								
0B9A6B00	F44C60E4	D9C9F4F0	0B95035C	00000340	0B9A2948	0B9A30B8	BF9A6FF2	C72A30B8	*4<-
URI40.n.*.....	..?2G...*								
0B9A6B20	F34C60E4	D9C9F0F0	0B97F898	00000B9E	0B9A2858	0B9A2948	BF9A6FF2	C72A3DB8	*3<-
URI00.p8q.....	..?2G...*								
0B9A6B40	F24C60E4	D9E3F7F0	0B974420	000004A0	0B9A2220	0B9A2858	BF9A6FF2	C72A44B8	*2<-
URT70.p.....	..?2G...*								
0B9A6B60	606EF3E4	D9C9F0F0	000003CC	0B94F9C8	0B9A2220	0B9A2858	BF9A6FF2	C72A59B8	*-
>3URI00.....m9H.....	..?2G...*								
0B9A6B80	C4E2D7E4	D9C9F0F0	0B99E480	00000000	00000000	00000000	BF9A6FF2	C72A61F8	
*DSPURI00.rU.....	..?2G../8*								
0B9A6BA0	D4D6C4C9	D9C3C1D9	F1F201D8	C3082000	40D3D6C7	C9C3C1D3	40C3D3D6	E2C54040	*MODIRCAR12.QC...
LOGICAL CLOSE	*								
0B9A6BC0	40404040	40404040	40404040	40404040	40404040	40404040	00000000	00000000	
*.....	.....	.....	.....	.....	.....	.....	.....	.....	
0B9A6BE0	606EF4E4	D9C9F0F1	000007BA	0B954A98	0B9A2858	0B9A2FC8	BF9A6FF2	C72A7A38	*-
>4URI01.....n.q.....H..?	2G...*								
0B9A6C00	F34C60E4	D9C9F0F1	0B950182	0000022E	0B9A2858	0B9A2FC8	BF9A6FF2	C72A8C78	*3<-
URI01.n.b.....	..?2G...*								
0B9A6C20	F24C60E4	D9C9F0F0	0B97172C	00000B9E	0B9A2220	0B9A2858	BF9A6FF2	C72A9278	*2<-
URI00.p.....	..?2G.k.*								
0B9A6C40	F14C60E4	D9E3F0F0	0B93697A	000003F6	0B9A2010	0B9A2220	BF9A6FF2	C72A9A38	*1<-
URT00.l.....6.....	..?2G...*								
0B9A6C60	606EF2E4	D9C9F0F0	00000318	0B94F9C8	0B9A2010	0B9A2220	BF9A6FF2	C72ABD78	*-
>2URI00.....m9H.....	..?2G...*								
0B9A6C80	C4E2D7E4	D9C9F0F0	0B99E480	00000000	00000000	00000000	BF9A6FF2	C72AC378	
*DSPURI00.rU.....	..?2G.C.*								
0B9A6CA0	D4D6C4C9	D9C3C1D9	F1F201D8	C3082000	40D3D6C7	C9C3C1D3	40C3D3D6	E2C54040	*MODIRCAR12.QC...
LOGICAL CLOSE	*								
0B9A6CC0	40404040	40404040	40404040	40404040	40404040	40404040	00000000	00000000	
*.....	.....	.....	.....	.....	.....	.....	.....	.....	
0B9A6CE0	606EF3E4	D9C9F0F1	000007BA	0B954A98	0B9A2220	0B9A2990	BF9A6FF2	C72AD4B8	*-
>3URI01.....n.q.....?	2G.M.*								
0B9A6D00	606EF4C4	C5D8F0F0	0000249A	0B90DE68	0B9A2990	0B9A3160	BF9A6FF2	C72C6D78	*-
>4DEQ00.....	..?2G...*								
0B9A6D20	F34C60C4	C5D8F0F0	0B956F32	000007DC	0B9A2990	0B9A3160	BF9A6FF2	C72F0878	*3<-
DEQ00.n?.....	..?2G...*								
0B9A6D40	F24C60E4	D9C9F0F1	0B950182	0000022E	0B9A2220	0B9A2990	BF9A6FF2	C72F0E38	*2<-
URI01.n.b.....	..?2G...*								
0B9A6D60	F14C60E4	D9C9F0F0	0B936780	00000B9E	0B9A2010	0B9A2220	BF9A6FF2	C72F13B8	*1<-
URI00.l.....	..?2G...*								
0B9A6D80	F04C60E2	E2C9C7D5	0000C4D4	0000033E	000067B8	0B9A2010	BF9A6FF2	C72F1F38	*0<-
SSIGN..DM.....	..?2G...*								
0B9A6DA0	C3D9E3D9	F0E7C9E3	0A7EFCB4	17172002	00000000	0B99E480	BF9A6FF2	C72F36F8	
*CRTR0XIT.....rU.....?	2G..8*								
0B9A6DC0	00000000	0A7D94A0	0A7EFCB4	0A807CB8	00000000	C4C6E2C2	D9D3E2C2	00000100	
*.....'m.....@.....DFSBRLSB.....	...	...	...	...	...	...	...	...	
0B9A6DE0	17172002	00000000	00000000	00000000	00000000	00000000	00000000	00000000	
*.....	.....	.....	.....	.....	.....	.....	.....	.....	

## Related tasks

[“Locating the DBRC trace” on page 244](#)

The DBRC trace is in the IMS-formatted portion of an IMS-formatted dump. There are four ways you can locate the DBRC trace.

## DBRC external trace

If you start the Generalized Trace Facility (GTF) and issue the `CHANGE.RECON TRACEON` command, the DBRC trace (DSPTRACE) creates an external trace record and issues the GTRACE macro to invoke GTF.

### DBRC external trace records

The GTRACE macro passes the address and length of a DBRC external trace record to GTF. A DBRC external trace record is put in the user data area of a GTF trace record.

If more than two DBRC jobs run concurrently, the GTF data set or buffer can contain multiple trace records. Therefore, DBRC external trace records contain either the IMS subsystem ID or a job name. In a DB/DC or DBCTL environment, the SSID is added to the trace record. In other IMS environments, a job name is added to the trace record. The following table shows the format of these records.

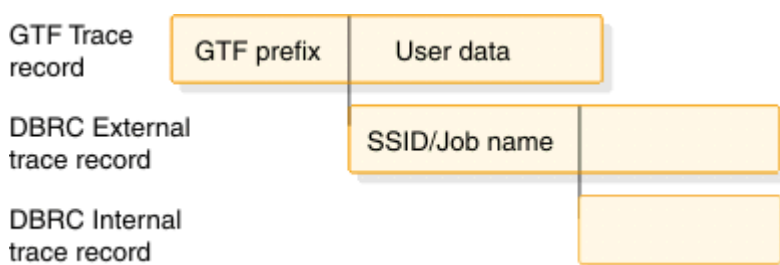


Figure 65. Format of DBRC external trace records

The GTF cataloged procedure is supplied in SYS1.PROCLIB with member name GTF or GRFSNP. If you want the DBRC trace records to be put in the GTF data set, specify `MODE=EXT` on the `EXEC` parameter and `USR` on the `GTF` option in the cataloged procedure. For detailed information about invoking GTF and its cataloged procedure, see *z/OS MVS Diagnosis: Tools and Service Aids*.

You can format and print DBRC trace records in the GTF data set by using the `GTFTRACE` subcommand of `IPCS`. You must specify the exit `AMDUSRF2` on this subcommand. For detailed information about using `IPCS`, see *z/OS MVS Interactive Problem Control System (IPCS) User's Guide*.

### Related concepts

[“DBRC trace output” on page 245](#)

Trace output normally resides in subpool 0 storage, but you can direct output to a Generalized Trace Facility (GTF) data set.

## Examples of DBRC router processing and RECON I/O error processing output

Formatted and unformatted output for DBRC router processing and RECON I/O error processing are shown.

### DBRC external trace output

The following two examples show the unformatted and formatted output for DBRC router processing and RECON I/O error processing.

In the following figure:

- DBRCJOB1 is the job name.
- TIME is the time stamp of the trace entry.
- DSPCRTRO passed control to the next routine to process the request identified by the DFSBRLSB.
- RQB is the address of the request level control block.

- LSB is the address of the DFSBRLSB.
- FUNC indicates the function flags (from the BRLBFFLG field of the DFSBRLSB).
- EXIT indicates the exit flags (from the BRLBEFLG field of the DFSBRLSB).

```
GTF USR Record containing DBRC Unformatted Trace Record Data
HEXFORMAT AID FF FID F2 EID EFAD
+0010 00000000 C4E2D7C3 D9E3D9F0 05F79C94 3 ....DSPCRTR0.7.m 3
+0020 17172002 00000000 00012D78 99085F22 3 .....I... 3
+0030 48397685 00000000 00000000 00000000 00D4C080 3 ...e.....M{. 3
+0040 00000001 00012D78 00000000 00000000 00000000 3 ..... 3
+0050 00000000 00000000 00000000 00000000 00000000 3 ..... 3
+0060 00000000 00000000 00000000 00000000 00000000 3 ..... 3
+0070 00000000 3 .... 3
```

```
Formatted Output
..... TIME=99085F2248397685 DSPCRTR0 RQB=00012D78 LSB=05F79C94 FUNC=17172002 EXIT=00000000
00000000 00000000 00D4C080 00000001 00012D78 00000000 00000000 00000000
*.....M.....*
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
```

In the following figure, a SHOWCB macro instruction is executed after the I/O request is issued.

- IMS1 is the SYSID.
- TIME is the time stamp of the trace entry.
- DSPURI00 has control.
- RQB is the address of the request level control block.
- A locate was done. For a locate, a flag and record key are also shown in the trace record.
- RSCD is the VSAM reason code.

```
GTF USR Record containing DBRC Unformatted Trace Record Data
HEXFORMAT AID FF FID F2 EID EFAD
+0000 00FA2980 C4C2D9D6 C3E3C1D4 C9D4E2F1 | ....DBROCTAMIMS1 |
+0010 40404040 C4E2D7E4 D9C9F0F0 00012D78 | DSPURI00.... |
+0020 00000000 00000000 00000000 99085F22 | .....I.. |
+0030 48398254 C4E2D7C9 D9C3C1D9 00000190 | ..b.DSPIRCAR... |
+0040 D3002000 00000000 00000000 FFFFFFFF | L..... |
+0050 FFFFFFFF C9D4E2F1 40404040 3F000000 | ....IMS1 .... |
+0060 00000000 00000000 00000000 00000000 | ..... |
+0070 00000000
```

```
Formatted Output
DBRCJOB1 TIME=99085F2248398254 DSPURI00 RQB=00012D78 FUNC=LOCATE
FLAG=0020
RECKEY=FFFFFFFFFFFFFFFFC9D4E2F1404040403F00000000000000000000000000000000000000000000
```

## Samples of JCL to create trace output

An example of a job that was used to create unformatted USR(FAD) trace output is shown.

```
//PRTUSRF2 JOB IMSCVT8,MSGLEVEL=1,CLASS=K,MSGCLASS=A,REGION=4096K
//*****
/* JOB NAME: PRINTGTF JCL *
/* JOB DEPENDENCIES: The GTF data set named below must exist. *
/* JOB Source: See the IPCS User's Guide, Appendix B. *
/* JOB DESCRIPTION: This job prints the specified GTF data set using *
/* the Batch IPCS feature. *
//*****
/*ROUTE PRINT THISCPU/IMSM3405
/*OBLIB DD DSN=IMSTESTL.TNUC0,DISP=SHR
/* DD DISP=SHR,DSN=IMSB LD.I710TS25.CRESLIB
/* DD DISP=SHR,DSN=IMSTESTG.IMS710.TSTRES
/* DD DISP=SHR,DSN=IMSTESTG.IMSQA.ACPLIB
/* DD DISP=SHR,DSN=IMSTESTG.IMSQA.PGMLIB
//JOB CAT DD DISP=SHR,DSN=VCATQAV
// DD DISP=SHR,DSN=VCATDCL
//*****
/* Print the SYS1.TRACE data set. *
/* Member BLSCDDIR resides in SYS1.SBLSCLI0, an IPCS system proclib. *
```

```

/* IT ISSUES THE DEFINE CLUSTER FOR 'DBRX06.IPCS.DDIR' ON USER01 AND *
/* catalogs it in SYS1.ECTEST.MASTER.CATALOG. *
/******
//IPCS      EXEC   PGM=IKJEFT01,DYNAMNBR=20,REGION=1500K
//TRACE     DD    DSN=SYS1.TRACE,DISP=SHR,
//           UNIT=SYSDA,VOL=SER=000000
//SYSPROC   DD    DSN=SYS1.SBLSCLI0,DISP=SHR
//SYSTSPRT  DD    SYSOUT=A
//IPCSPRNT  DD    SYSOUT=A
//IPCSTOC   DD    SYSOUT=A
//SYSUDUMP  DD    SYSOUT=A
//SYSTSIN   DD *
//           PROFILE MSGID
//           %BLSCDDIR DSNNAME(DBRX06.IPCS.DDIR) VOLUME(USER01)
//           IPCS NOPARM
//           SETDEF DDNAME(TRACE) NOCONFIRM
//           GTFTRACE USR(FAD)
//           END
/*
/******
/* Delete the IPCS dump directory created by the previous step *
/* so that the re-IPL of the ec machine will not orphan the data *
/* set. *
/******
//AMS01     EXEC   PGM=IDCAMS,COND=EVEN
//SYSPRINT  DD    SYSOUT=A
//DD1       DD    UNIT=SYSDA,VOL=SER=USER01,DISP=SHR
//SYSIN     DD    *
//           DELETE DBRX06.IPCS.DDIR FILE(DD1)
/*

```

The following example is of a job that was used to create the DBRC formatted output:

```

//PRINTHMD JOB IMSCVT8,MSGLEVEL=1,CLASS=K,MSGCLASS=A,REGION=4096K
//*****
/* JOB NAME:          PRINTHMD JCL *
/* JOB DEPENDENCIES: The GTF data set named below must exist. *
/* JOB Source:       See the IPCS User's Guide, Appendix B. *
/* JOB DESCRIPTION: This job prints the specified GTF data set using *
/* the Batch IPCS feature. *
/******
/*ROUTE PRINT THISCPU/IMSM3405
//JOBLIB    DD DSN=IMSTESTL.TNUC0,DISP=SHR
//           DD DISP=SHR,DSN=IMSB LD.I710TS25.CRESLIB
//           DD DISP=SHR,DSN=IMSTESTG.IMS710.TSTRES
//           DD DISP=SHR,DSN=IMSTESTG.IMSQA.ACPLIB
//           DD DISP=SHR,DSN=IMSTESTG.IMSQA.PGMLIB
//JOBCAT    DD DISP=SHR,DSN=VCATQAV
//           DD DISP=SHR,DSN=VCATDCL
//*****
/* Print the SYS1.TRACE data set. *
/* Member BLSCDDIR resides in SYS1.SBLSCLI0, an IPCS system proclib. *
/* IT ISSUES THE DEFINE CLUSTER FOR 'DBRX06.IPCS.DDIR' ON USER01 AND *
/* catalogs it in SYS1.ECTEST.MASTER.CATALOG. *
/******
//IPCS      EXEC   PGM=IKJEFT01,DYNAMNBR=20,REGION=1500K
//TRACE     DD    DSN=SYS1.TRACE,DISP=SHR,
//           UNIT=SYSDA,VOL=SER=000000
//SYSPROC   DD    DSN=SYS1.SBLSCLI0,DISP=SHR
//SYSTSPRT  DD    SYSOUT=A
//IPCSPRNT  DD    SYSOUT=A
//IPCSTOC   DD    SYSOUT=A
//SYSUDUMP  DD    SYSOUT=A
//SYSTSIN   DD *
//           PROFILE MSGID
//           %BLSCDDIR DSNNAME(DBRX06.IPCS.DDIR) VOLUME(USER01)
//           IPCS NOPARM
//           SETDEF DDNAME(TRACE) NOCONFIRM
//           GTFTRACE EXIT(DSPUSRF2)
//           END
/*
/******
/* Delete the IPCS dump directory created by the previous step *
/* so that the re-IPL of the ec machine will not orphan the data *
/* set. *
/******
//AMS01     EXEC   PGM=IDCAMS,COND=EVEN
//SYSPRINT  DD    SYSOUT=A
//DD1       DD    UNIT=SYSDA,VOL=SER=USER01,DISP=SHR
//SYSIN     DD    *

```

```
DELETE DBRX06.IPCS.DDIR FILE(DD1)  
/*
```



# Chapter 10. BPE-based DBRC service aids

BPE-based DBRC service aids help you analyze problems in BPE-based DBRC.

## BPE-based DBRC trace records

BPE-based DBRC trace records are written to one or more trace tables, and provide information that can help you determine the source of errors.

Trace record eye catchers in a formatted dump provide information about which function resulted in an error. You might be able to correct environmental problems immediately. Refer internal problems to IBM Software Support with appropriate documentation, such as system console logs and dumps.

BPE-based DBRC trace records are written to one or more of the trace tables shown in the following table.

Table 56. Trace tables for BPE-based DBRC trace records

Table name	Number of tables	Table description
ERR	1	Used to trace errors that occur within the DBRC address space. Specify as TRCLEV=(ERR,level,DBRC,PAGES=num_pages). The default number of pages for this table is 2. The ERR trace entry is 32-bytes long.
GRPS	1	Used for DBRC group services messages and notification tracing. Specify as TRCLEV=(GRPS,level,DBRC,PAGES=num_pages). The default number of pages for this table is 8.
MODF	N = 1 - 256	Used for DBRC module flow tracing. Specify as TRCLEV=(MODF,level,DBRC,PAGES=num_pages). The default number of pages for this table is 8.
RQST	N = 1 - 256	Used for general DBRC request processing. Specify as TRCLEV=(RQST,level,DBRC,PAGES=num_pages). The default number of pages for this table is 8.

Trace entries have the following general format:

0	1	2	3	4 (WD1)	8 (WD2)	C (WD3)	10	LL-B	LL-1
+	+	+	+	+	+	+	+	+	+
CD SC	THD		DTRCALLER		client ID		user data		STCK
+	+	+	+	+	+	+	+	+	+

### Field name

#### Represents

#### CD

1-byte trace code field that indicates the function that wrote the trace record.

#### SC

1-byte trace subcode field that indicates the category of the trace record.

#### THD

2-byte thread sequence number.

#### WD1 - WD2

2-word trace identifier (DTRCALLR).

#### WD3

The client ID.

#### STCK

2-word system clock (STCK) that indicates the time when the trace entry was created.

The first four words and the last two words are common fields, and the rest of the information is specific to each entry.

DBRC address space trace records are mapped by macro DSPBDTR.

Trace record mapping is based on the trace subcode, which identifies the category of trace record. One particular trace subcode can apply to many trace codes. Each trace record mapping also includes a pictorial representation in a comment block.

Use the trace subcode to locate the trace record mapping in the DSPBDTR macro. The DSPBCODE macro includes 2-byte module identifier codes that are used in trace records when it is necessary to identify a DBRC module. The module identifier represents the module that wrote the trace record. The DBRC module identifiers are defined in macro DSPBCODE.

The following example shows the format of a 64-byte trace entry (MODF or GRPS trace table):

0	1	2	3	4		8		C		10		14		18		1C		1F		
-----+																				
	CD		SC		THD		WD1		WD2		WD3		WD4		WD5		WD6		WD7	
-----+																				
	WD8					WD9		WD10		WD11		WD12		WD13		STCK				
-----+																				
20		24				28		2C		30		34		38		3F				

The following example shows the format of a 128-byte trace entry (RQST trace table):

0	1	2	3	4		8		C		10		14		18		1C		1F		
+																				
	CD		SC		THD		WD1		WD2		WD3		WD4		WD5		WD6		WD7	
+																				
	WD8					WD9		WD10		WD11		WD12		WD13						
+																				
+																				
																STCK				
+																				

## Trace record examples for BPE-based DBRC

These examples show formatted BPE-based DBRC trace record examples.

### DBRC I/O services trace

In the following example, the trace code is in the first byte (X'72'), which the DSPBDTR macro documents as I/O services. The eye catcher is IOSRV. The trace subcode is in the second byte (X'02'), which the DSPBDTR macro documents as RECON data set true open start. The eye catcher is True OPEN start. The 8-byte identifier in the right eye catcher for corresponding subcode is in word 1 - 2.

**Important:** An asterisk at the beginning of a subcode eye catcher indicates an error.

Code	Subcode	Trace Entry
Trace Data		
-----		-----
IOSRV: True OPEN start		72020000 D6D7C5D5 C5D9F140 C9D4E2F1 0BD61000 00000000 00000000 00000000
OPENER1		00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000		00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000		00000000 00000000 00000000 00000000 00000000 00000000 C1C4D082
D9525260		

When the identifier is DSPURI00, the block-area pointer eye catcher follows the eye catcher for subcode in the left eye catcher. In the following example, the trace subcode is in the second byte (X'01'), which the DSPBDTR macro documents as "Enter DSPURI00." The eye catcher is Enter URI00. The block-area

pointer eye catcher that follows is MODIRCAR. The 16-byte entry message for DSPURI00 is in word 12 - 15. The eye catcher is LOGICAL CLOSE.

Code Trace	Subcode Data	Trace Entry
IOSRV: Enter	URI00 MODIRCAR	72010000 C4E2D7E4 D9C9F0F0 C9D4E2F1 0BDC0000 00000000 00000000 00000000
LOGICAL CLOSE		D4D6C4C9 D9C3C1D9 F1F201D8 C3082000 40D3D6C7 C9C3C1D3 40C3D3D6
E2C54040		40404040 40404040 40404040 40404040 40404040 40404040 00000000
00000000		00000000 00000000 00000000 00000000 00000000 00000000 C248B54C
F0FE19A0		

## DBRC module flow trace

The following example shows a DBRC module flow trace.

**Important:** An asterisk at the beginning of a subcode eye catcher indicates an error.

**Module A calls module B:** The following two-line trace entry is produced when module A calls module B. A two-line trace entry is produced when module B calls DSPSTGET to obtain initial workspace storage after being called by module A. The trace code is in the first byte (X'77'), which the DSPBDTR macro documents as "Request processing." The eye catcher is RQST. The trace subcode is in the second byte (X'1A'), which the DSPBDTR macro documents as Process Flow. The eye catcher is Module entry. The 8-byte identifier in the right eye catcher for the corresponding subcode is in word 1 - 2. Word 3 represents the client ID, which is IMS1 in this example.

Code Trace	Subcode Data	Trace Entry
RQST : Module entry		771A0000 606EF1C4 E2E2F0F1 C9D4E2F1 8BD8C38C 0BDF5258 0BE97370 0BE97518
->1DSS01		00000000 00000000 00000000 00000000 00000000 00000000 C3D772C3 AC3CE200

words 1-2	Identifier that consists of:
	- An arrow (->) indicating that the module is being called.
	- The nesting level of module B. Nesting levels are shown in one or two decimal digits, up to 99 (nesting level 0 is DSPUIN00).
	- Last five characters of the module name being called.
word 3	Client ID.
word 4	Offset in module A of call to module B.
word 5	Entry point address of module B.
word 6	Save area address of the calling module (A).
word 7	Beginning address of the temporary storage obtained for module B (save area address of module B).
words 14-15	Trace time stamp.

**Module B calls module A:** The following two-line trace entry is produced when module B calls DSPSTFRE to release all of its temporary storage before returning to module A. The trace code is in the first byte (X'77'), which the DSPBDTR macro documents as Request processing. The eye catcher is RQST. The trace subcode is in the second byte (X'1B'), which the DSPBDTR macro documents as Process Flow. The eye catcher is Module exit. The 8-byte identifier in the right eye catcher for corresponding subcode is in word 1 - 2.

Code Trace	Subcode Data	Trace Entry
RQST : Module exit		771B0000 F04C60C4 E2E2F0F1 C9D4E2F1 0BD8C38C 00000236 0BE97370 0BE97518 <-
DSS01		00000000 00000000 00000000 00000000 00000000 00000000 C3D772D5 50228DC0

```

words 1-2      Identifier that consists of:
                - An arrow (->) indicating that the module is returning.
                - The nesting level of module A. Nesting levels are shown in
                  one or two decimal digits, up to 99 (nesting level 0 is DSPUIN00).
                - Last five characters of the module name returning.
word 3         Client ID.
word 4         Offset in module A to which module B returns.
word 5         Offset in module B where it returns to module A.
word 6         Save area address of module A that called module B.
word 7         Beginning address of the temporary storage being released for module B
                by module DSPSTFRE.
words 14-15    Trace time stamp.

```

**DSPSTACK trace entry:** The following example shows a two-line trace entry that is produced when module B issues macro DSPGFSTK, which calls DSPSTGET to obtain additional temporary storage. The trace code is in the first byte (X'78'), which the DSPBDTR macro documents as System related. The eye catcher is SYS. The trace subcode is in the second byte (X'15'), which the DSPBDTR macro documents as Additional work space. The eye catcher is DBRC stack rqst. The 8-byte identifier in the right eye catcher for corresponding subcode is in word 1 - 2.

Code	Subcode	Trace Entry
SYS : DBRC stack rqst		78150000 C4E2D7E2 E3C1C3D2 C9D4E2F1 8BE0B9AE 0BE0B710 0BF4D7C8 0BE98910
DSPSTACK		00000000 00000000 00000000 00000000 00000000 00000000 C3D772D5 2A1BA7C0

```

words 1-2      Identifier DSPSTACK.
word 3         Client ID.
word 4         Return point address in the module B to which DSPSTGET returns after
                acquiring additional temporary storage for the module.
word 5         Entry point address of module B.
word 6         Save area address of module B.
word 7         Beginning address of the additional temporary storage obtained
                for module B.
words 14-15    Trace time stamp.

```

## DBRC request user exit trace

The following example shows a trace taken before the DBRC request user exit routine is called. The trace subcode is in the second byte of the trace (X'1C'). which the DSPBDTR macro documents as "Before calling user exit," as indicated in the left eye catcher field. The "BRQ0" that follows refers to the module name (DSPBRQ00) that issues the trace call. The 1-byte BRQX\_Flags field is located in the first byte of the second word of the trace. The eye catcher field on the right contains "DBRC IS BYPASSED," corresponding to the flag code (X'80' in this trace). The "END" that follows is the user exit function (BRQX\_Func), which is located in the first byte of the first word (X'02' in this trace).

**Important:** An asterisk at the beginning of a subcode eye catcher indicates an error.

Code	Subcode	Trace Entry
RQST: Before exit call BRQ0		771C0000 02030010 80000000 C9D4E2F1 0A8B0558 07142002 00001000 0C083000
DBRC IS BYPASSED END		00000000 0A8FDE40 0A8B056C 0A91A798 00000008 C4C6E2C2 D9D3E2C2 00000100
		07142002 00001000 00000000 00000000 00000000 00000000 00000000 00000000
		00C57B78 00000001 0C083000 00000000 00000000 00000000 C4BC1081 98B16660

## DBRC group services trace

DBRC group services generates trace entries of two types: group services request or group services send. This type of entry means that global services is requested to do something by either its own DBRC or one of the other DBRC instances in the same IMSplex. The 8-byte identifier in the right eye catcher for corresponding request type is in word 5-6, which represents the DBRC job name.

### Group services request:

The following example shows a group services request entry.

Code Trace	Subcode Data	Trace Entry
GRPSV:REQ - DBRCUP DBR8CSBB		73270000 00000058 00800000 12248000 00000000 C4C2D9F8 C3E2C2C2 F4F70038
C5AFF4C0		1197A6B8 C4D5465B BC96000D 00000000 00000000 00000000 C4D5465B

The following tables describes useful fields and their possible values:

Table 57. Word 1 eye catchers for the group services request

Word 1	Eye catcher	Request type
X'50'	INIT	Initialize global services.
X'51'	NOTQUIESCE	Another DBRC is telling this DBRC that RECON data set access is allowed.
X'52'	QUIESCE	This DBRC wants exclusive RECON data set access.
X'53'	QUACK	Another DBRC acknowledges this DBRC's request for exclusive RECON data set access.
X'54'	ENDQUIESCE	This DBRC releases RECON data sets to other DBRCs again.
X'55'	ENDQUACK	Another DBRC acknowledges renewed RECON data set access.
X'56'	GOTQUIESCE	Another DBRC wants exclusive RECON data set access.
X'57'	GOTENDQUIES	Another DBRC releases its exclusive RECON data set access.
X'58'	DBRCUP	A new DBRC joined the IMSplex.
X'59'	DBRCDOWN	A DBRC left the IMSplex.
X'5A'	SCIUP	Local SCI is active.
X'5B'	SCIDOWN	Local SCI is down.
X'5C'	RCNLOSS	This DBRC wants to notify other DBRCs of a RECON loss.
X'5D'	RESEND	Another DBRC wants this DBRC to resend its last request.
X'5E'	GRP2DONE	Local processing complete for request types 56 or 57.
X'5F'	GOTRCNLOSS	RECON loss notification from another DBRC.
X'60'	ID	Another DBRC informs this DBRC of its SSID.

Table 58. Words 4 -10 request types for the group services request

Word	Request type
Word 4	Message sequence number.

Table 58. Words 4 -10 request types for the group services request (continued)

Word	Request type
	Meaningful only for requests from other DBRCs (request types 51, 53, 55, 56, 5D, 5F, 60). Each DBRC assigns a sequence number to every new request that it sends. If a request is resent, it contains the same sequence number that was originally assigned to it.
Words 5-6	For requests from other DBRCs, the job name of the other DBRC.
Words 7-10	The SCI token associated with the request. Can be used to distinguish between DBRCs if the job name is not unique.

#### Group services send:

The following example shows a group services send entry. This type of entry means that group services is sending a message to another DBRC.

Code Trace	Subcode Data	Trace Entry
GRPSV:SEND	- NOTQUIESCE	73280000 00000001 00000000 00000002 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 C4D5465B C5B11020

The following list describes useful fields and their possible values:

Table 59. Word 1 eye catchers for the group services request

Word 1	Eye catcher	Message type
X'01'	NOTQUIESCE	RECON access is allowed.
X'02'	RCNLOSS	RECON loss notification.
X'03'	QUIESCE	Request for exclusive RECON access.
X'04'	QUIESCECLS	Request for exclusive RECON access and that the recipient close the RECON data sets.
X'05'	QUACK	Acknowledge the request for exclusive RECON access.
X'06'	ENDQUIESCE	Release exclusive RECON access by this DBRC.
X'07'	ENDQUACK	Acknowledge release of exclusive RECON access.
X'08'	RESEND	Request resend of another DBRC's last request.
X'09'	ID	Send SSID to another DBRC.

Table 60. Words 3 and 4 message types for the group services request

Word	Request type
Word 3	Message sequence number.
Word 4	If the message is a response to a message from another DBRC, this is the sequence number of the other DBRC's message.

#### Related concepts

[Planning for recovery \(System Administration\)](#)

## Unformatted BPE-based DBRC internal trace example

The module-call entries, module return entries, DSPURI00 trace entries, and other entries (such as GETFEED, DSPCRTR0, and CRTR0XIT) are shown in this trace example.

In this example, client ID is 5 for all requests.

```
0B9A5440 20010016 C4E2D7C3 D9E3D9F0 00000005 0A315CB4 17172002 00000000 0B99E480
|....DSPCRTR0.....*.....rU.|
0B9A5460 00000000 00000000 00C4CD28 00000001 0B99E480 00000000 00000000 00000000
|.....D.....rU.....|
0B9A5480 00000000 00000000 00000000 00000000 00000000 00000000 00000000
|.....|
0B9A54A0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EDECA7
|.....>....|
0B9A54C0 06010016 606EF1E2 E2C9C7D5 00000005 8000C4E4 0B93A950 000067B8 0B9A2010 |....-
>1SSIGN.....DU.lz&.....|
0B9A54E0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EDF13B7
|.....>....|
0B9A5500 20010018 C4E2D7C3 D9E3D9F0 00000005 0A315CB4 17172002 00000000 0B99E480
|....DSPCRTR0.....*.....rU.|
0B9A5520 00000000 00000000 00C4CD28 00000001 0B99E480 00000000 00000000 00000000
|.....D.....rU.....|
0B9A5540 00000000 00000000 00000000 00000000 00000000 00000000 00000000
|.....|
0B9A5560 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EDECA7
|.....>....|
0B9A5580 20080016 606EF2E4 D9C9F0F0 00000005 00000258 0B953A20 0B9A2010 0B9A2220 |....-
>2URI00.....n.....|
0B9A55A0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EDF4577
|.....>....|
0B9A55C0 06010018 606EF1E2 E2C9C7D5 00000005 8000C4E4 0B93A950 000067B8 0B9A2010 |....-
>1SSIGN.....DU.lz&.....|
0B9A55E0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EDF13B7
|.....>....|
0B9A5600 20080018 606EF2E4 D9C9F0F0 00000005 00000258 0B953A20 0B9A2010 0B9A2220 |....-
>2URI00.....n.....|
0B9A5620 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EDF4577
|.....>....|
0B9A5640 06010018 C4E2D7E4 D9C9F0F0 00000005 0B99E480 00000000 00000000 00000000
|....DSPURI00.....rU.....|
0B9A5660 D4D6C9C4 D9C3C1D9 F1F201D8 D6000000 40D3D6C7 C9C3C1D3 40D6D7C5 D5404040 |MODIRCAR12.QO...
LOGICAL OPEN |
0B9A5680 40404040 40404040 40404040 40404040 40404040 40404040 00000000 00000000
|.....|
0B9A56A0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EDF6E37
|.....>....|
0B9A56C0 20020018 606EF3E4 D9C9F0F1 00000005 000007A4 0B903088 0B9A2220 0B9A2990 |....-
>3URI01.....u...h.....|
0B9A56E0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EDFCE77
|.....>....|
0B9A5700 06010016 C4E2D7E4 D9C9F0F0 00000005 0B99E480 00000000 00000000 00000000
|....DSPURI00.....rU.....|
0B9A5720 D4D6C9C4 D9C3C1D9 F1F201D8 D6000000 40D3D6C7 C9C3C1D3 40D6D7C5 D5404040 |MODIRCAR12.QO...
LOGICAL OPEN |
0B9A5740 40404040 40404040 40404040 40404040 40404040 40404040 00000000 00000000
|.....|
0B9A5760 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EDF6E37
|.....>....|
0B9A5780 20020016 606EF3E4 D9C9F0F1 00000005 000007A4 0B903088 0B9A2220 0B9A2990 |....-
>3URI01.....u...h.....|
0B9A57A0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EDFCE77
|.....>....|
0B9A57C0 20060018 606EF4D9 E2E5F0F0 00000005 00000EE6 0B998AE8 0B9A2990 0B9A3170 |....-
>4RSV00.....W.r.Y.....|
0B9A57E0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EDFF577
|.....>....|
0B9A5800 20060016 606EF4D9 E2E5F0F0 00000005 00000EE6 0B998AE8 0B9A2990 0B9A3170 |....-
>4RSV00.....W.r.Y.....|
0B9A5820 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EDFF577
|.....>....|
0B9A5840 20060018 F34C60D9 E2E5F0F0 00000005 0B903F6E 00000234 0B9A2990 0B9A3170 |....3<-
RSV00.....>.....|
0B9A5860 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE005F7
|.....>....|
0B9A5880 20060016 F34C60D9 E2E5F0F0 00000005 0B903F6E 00000234 0B9A2990 0B9A3170 |....3<-
RSV00.....>.....|
0B9A58A0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE005F7
```

```

|.....>..7|
0B9A58C0 20040018 606EF4E4 D9C9F3F0 00000005 00000F9C 0B961D24 0B9A2990 0B9A3170 |....-
>4URI30.....o.....|
0B9A58E0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE036B7
|.....>...|
0B9A5900 20030018 606EF5E4 D9C9F2F0 00000005 000010E2 0B95FEC2 0B9A3170 0B9A3B08 |....-
>5URI20.....S.n.B.....|
0B9A5920 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE058B7
|.....>...|
0B9A5940 20030018 F44C60E4 D9C9F2F0 00000005 0B962E06 00000A10 0B9A3170 0B9A3B08 |....4<-
URI20.....o.....|
0B9A5960 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE06E77
|.....>..>..|
0B9A5980 20040016 606EF4E4 D9C9F3F0 00000005 00000F9C 0B961D24 0B9A2990 0B9A3170 |....-
>4URI30.....o.....|
0B9A59A0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE036B7
|.....>...|
0B9A59C0 20030016 606EF5E4 D9C9F2F0 00000005 000010E2 0B95FEC2 0B9A3170 0B9A3B08 |....-
>5URI20.....S.n.B.....|
0B9A59E0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE058B7
|.....>...|
0B9A5A00 20030016 F44C60E4 D9C9F2F0 00000005 0B962E06 00000A10 0B9A3170 0B9A3B08 |....4<-
URI20.....o.....|
0B9A5A20 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE06E77
|.....>..>..|
0B9A5A40 20030018 606EF5E4 D9C9F2F0 00000005 000010E2 0B95FEC2 0B9A3170 0B9A3B08 |....-
>5URI20.....S.n.B.....|
0B9A5A60 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE058B7
|.....>...|
0B9A5A80 20040018 F34C60E4 D9C9F3F0 00000005 0B904024 0000106C 0B9A2990 0B9A3170 |....3<-
URI30.....%.....|
0B9A5AA0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE07637
|.....>...|
0B9A5AC0 20050018 606EF4E4 D9C9F5F0 00000005 000029AA 0B969248 0B9A2990 0B9A3170 |....-
>4URI50.....ok.....|
0B9A5AE0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE0F1B7
|.....>..1..|
0B9A5B00 20030016 606EF5E4 D9C9F2F0 00000005 000010E2 0B95FEC2 0B9A3170 0B9A3B08 |....-
>5URI20.....S.n.B.....|
0B9A5B20 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE058B7
|.....>...|
0B9A5B40 20040016 F34C60E4 D9C9F3F0 00000005 0B904024 0000106C 0B9A2990 0B9A3170 |....3<-
URI30.....%.....|
0B9A5B60 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE07637
|.....>...|
0B9A5B80 060A0018 C4E2D7E2 E3C1C3D2 00000005 8B9693E2 0B969248 0B9A3170 0B9A37C8
|....DSPSTACK.....o1S.ok.....H|
0B9A5BA0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE0FAB7
|.....>...|
0B9A5BC0 20070018 606EF5D9 E3E7C4F0 00000005 00000430 0B92D988 0B9A3170 0B9A3870 |....-
>5RTXD0.....krh.....|
0B9A5BE0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE11037
|.....>...|
0B9A5C00 20070018 F44C60D9 E3E7C4F0 00000005 0B969678 000002AE 0B9A3170 0B9A3870 |....4<-
RTXD0.....oo.....|
0B9A5C20 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE15CF7
|.....>..*7|
0B9A5C40 060A0018 C7C5E3C6 C5C5C440 00000005 0B99E480 00000000 00000000 00000000
|....GETFEED.....rU.....|
0B9A5C60 00000000 00000280 00000000 00000000 00000001 00020000 00020000 00007FF8
|....."8|
0B9A5C80 00000080 80100000 0B9CD878 0B9CA57C 00000280 00000100 C7C5E340 40400000
|.....Q...v@.....GET...|
0B9A5CA0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6F11A877
|.....?..y..|
0B9A5CC0 20050016 606EF4E4 D9C9F5F0 00000005 000029AA 0B969248 0B9A2990 0B9A3170 |....-
>4URI50.....ok.....|
0B9A5CE0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE0F1B7
|.....>..1..|
0B9A5D00 060A0016 C4E2D7E2 E3C1C3D2 00000005 8B9693E2 0B969248 0B9A3170 0B9A37C8
|....DSPSTACK.....o1S.ok.....H|
0B9A5D20 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE0FAB7
|.....>...|
0B9A5D40 20070016 606EF5D9 E3E7C4F0 00000005 00000430 0B92D988 0B9A3170 0B9A3870 |....-
>5RTXD0.....krh.....|
0B9A5D60 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE11037
|.....>...|
0B9A5D80 20070016 F44C60D9 E3E7C4F0 00000005 0B969678 000002AE 0B9A3170 0B9A3870 |....4<-
RTXD0.....oo.....|
0B9A5DA0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6EE15CF7
|.....>..*7|
0B9A5DC0 060A0016 C7C5E3C6 C5C5C440 00000005 0B99E480 00000000 00000000 00000000

```



```

|....GETFEED....rU.....|
0B9A5DE0 00000000 00000280 00000000 00000000 00000001 00020000 00020000 00007FF8
|....."8|
0B9A5E00 00000080 80100000 0B9CD878 0B9CA57C 00000280 00000100 C7C5E340 40400000
|.....Q...v@.....GET...|
0B9A5E20 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6F11A877
|.....?..y.|
0B9A5E40 20070018 606EF5D9 E3E7C4F0 00000005 00000876 0B92D988 0B9A3170 0B9A3870 |....-
>5RTXD0.....kRh.....|
0B9A5E60 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6F11BAF7
|.....?..7|
0B9A5E80 20070018 F44C60D9 E3E7C4F0 00000005 0B969ABE 000002AE 0B9A3170 0B9A3870 |....4<-
RTXD0.....o.....|
0B9A5EA0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6F120B77
|.....?..|
0B9A5EC0 20070016 606EF5D9 E3E7C4F0 00000005 00000876 0B92D988 0B9A3170 0B9A3870 |....-
>5RTXD0.....kRh.....|
0B9A5EE0 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6F11BAF7
|.....?..7|
0B9A5F00 20070016 F44C60D9 E3E7C4F0 00000005 0B969ABE 000002AE 0B9A3170 0B9A3870 |....4<-
RTXD0.....o.....|
0B9A5F20 00000000 00000000 00000000 00000000 00000000 00000000 BF3069CF 6F120B77 |....

```



---

## Chapter 11. Data communication service aids

Data communication service aids, such as Terminal communication task trace and DC Trace are described.

### Terminal communication task trace

---

When an output device (such as a terminal, line, or node) hangs, you can use the terminal communication task trace to diagnose the problem.

You can use information you find in the terminal communication task trace to build keywords for your search string, or you can use the information when you are reviewing existing APAR descriptions to determine whether they describe the problem you are experiencing.

All IMS terminal communication tasks are dispatched by the IMS communication analyzer (module DFSICIO0). This module traces its own flow, as well as the flow through device-dependent modules, by using register 0 of the save area of the communication analyzer. (For this reason, this trace is often referred to as the REG0 trace.) The communication analyzer uses the high-order 2 bytes of register 0 to trace the analyzer entry point, and uses the low-order 2 bytes to trace the device-dependent module entry point.

In the DC section of the IMS dump, find the save area sets that hold data about the various IMS processes that were executing prior to the dump. If one of these save areas sets is for DFSICIO0, you can then look at the corresponding register 0 to find the communication task trace entries.

#### Related reference

“Log records” on page 469

To diagnose some problems, you need to examine the content of log records to determine what was going on in the system before the problem occurred. By knowing the layout of the log records, you can set up a DFSERA10 job that will produce the specific log records that you need to examine.

### Analyzer entry points

The high-order 2 bytes of register 0 (for module DFSICIO0) identify the analyzer entry points.

#### Entry points

##### Analyzer entry point (hex)

	Processing description
1	Process an input segment from a terminal.
2	Perform a logical read operation to the terminal.
3	Determine which system function is to be performed next for this line and terminal (or node).
4	Issue a GET NEXT call to message queue.
5	Perform a logical write operation to the terminal.
6	WRITE successful; dequeue message or call the device-dependent module at DD1.
7	Notify master terminal of I/O error; cancel input; return output message to queue.
8	Return output message to queue; cancel input.

9

Generate an error message; cancel input; return output message to queue.

**A**

Idle the line; cancel output; return output message to queue.

**B**

Resend the last message sent from a given logical terminal (LTERM).

**C**

Idle the line.

The low-order 2 bytes of register 0 identify the entry points for the device-dependent modules (DDMs), as listed below:

**DDM entry point (hex)**

## Processing description

**1**

WRITE/SEND setup: Set up output buffer to write current buffer.

**2**

WRITE/SEND interruption: Error check last output operation.

3

READ/RECEIVE setup: Set up to perform a poll or read.

**4**

READ/RECEIVE interruption: Error check, determine terminal responding, and deblock input segment.

5

Cleanup: Restore control blocks after DFSICI00 error.

6

Build: Move output message from a queue buffer (MFS buffer) to a line buffer.

7

Logon: VTAM OPNDST/CLSDST processing.

8

Prepare for output: VTAM

**F**

MFS output format control (DFSCOF0) was entered.

## Trace records

Example trace records for the terminal communication task trace are described.

The entries in the first 2 bytes indicate the processing that the analyzer (DFSICIO0) has completed. The entries in the last 2 bytes indicate the processing that the device-dependent modules (DDMs) have completed. As new entries are added, existing entries shift to the left. When the 2-byte area fills, the oldest entry is overwritten by the next-oldest entry. Therefore, the right-most entry of each 2-byte portion of register 0 identifies the most recent analyzer or device-dependent module activity.

The following figure shows the format of a sample terminal communications task trace record.

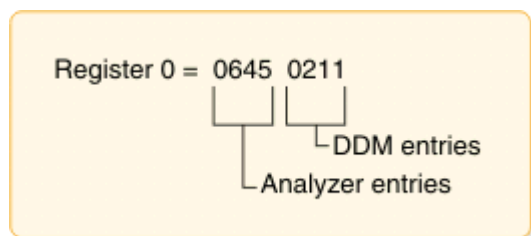


Figure 66. Example of a terminal communication task trace entry

The sample terminal communication task trace entry in the figure indicates that the analyzer entries are 6, 4, and 5; DDM entries are 2, 1, and 1. An analysis of this trace data would yield the flow information shown in the following table.

*Table 61. Example processing flow for a terminal communication task trace entry*

Entry point	Trace ID	Processing description
2	DDM2	A write interrupt occurred.
6	A06	A write completed successfully.
1	DDM1	Another buffer was required.
4	A04	Room in the buffer is allowed for another message segment. (GN was issued to the message queue.)
1	DDM1	This segment was placed in the buffer, filling it or an EOM was detected. Setup for the write operation was completed.
5	A05	An output operation was requested.

## Trace output

You can find the terminal communication task trace in any IMS dump, either in register 0 (corresponding to module DFSIC100) or in the CLB section of the dump for the terminal involved in the problem.

If you look at the CLB section of the dump, the information in field CLBTEMP1 is the same as what is in register 0. Fields CLBTEMP4 and CLBTEMP5 contain the Julian date and time at which the IMS task (ITASK) associated with the line or node returned to the IMS dispatcher (module DFSIDSP0). This information is useful when diagnosing a hung or lost terminal. In an IMS control region dump, you can determine when the last activity occurred on the line or node and what processing path was taken.

## DC trace

The data communication (DC) trace enables you to obtain information about the program flow within the communications analyzer and between the analyzer and the device dependent modules (DDMs).

### About this task

#### Related reference

[“Diagnosing Message Format Service problems” on page 324](#)

The number of physical terminals traced and the number of lines traced can affect completeness of trace records and sequence of trace entries.

## Starting the trace

To start the DC trace for any terminal in the IMS network, enter one of the /TRACE commands from the master terminal or the z/OS console.

### About this task

Specify at least level 3 in the command because buffer contents are usually required for complete diagnosis. If you specify level 4, the trace writes a save area set for certain entries (C00-C12, D05, AER1, and AER2).

- For VTAM terminals:

```
/TRACE SET ON NODE P1 LEVEL 1|2|3|4 MODULE DDM|MFS|ALL
```

- For ISC links:

```
/TRACE SET ON NODE P1 LEVEL=1|2|3|4 MODULE DDM|MFS|ALL
      OR
/TRACE SET ON NODE P1 USER P2
```

- For logical LINKs:

```
/TRACE SET ON LINK P1,...,Pn|ALL LEVEL 1|2|3|4 MODULE DDM|MFS|ALL
```

- For UNITTYPE:

```
/TRACE SET ON UNITTYPE P1,...,Pn LEVEL 1|2|3|4 MODULE DDM|MFS|ALL
UPDATE MSLINK NAME(linkname|linkname*|*) START(TRACE)
```

**Note:** The type-2 command **UPDATE MSLINK NAME(linkname) START(TRACE)** uses the same level and module settings that were used the last time the **/TRACE SET (ON) LINK** command was issued. If a **/TRACE SET (ON) LINK** command has not been issued since the last cold start, this command defaults to MODULE=ALL and LEVEL=4.

- For an XRF environment:

```
/TRACE SET ON NODE xxx TAKEOVER
/TRACE SET ON LINE xxx TAKEOVER
/TRACE SET ON LINK xxx TAKEOVER
UPDATE MSLINK NAME(linkname) START(TRACE) SET(TKOTRC((Y))
```

#### Tip:

- The **/TRACE SET ON NODExxx TAKEOVER** command starts the trace for the specified terminals during takeover only.
- You can enter this command only from the active system in an XRF environment.
- After a terminal has switched successfully, the trace is automatically turned off for that terminal.
- Because this command is recovered across restart and takeover, you need to enter it only once. After a cold start, you must enter the command again.
- Tracing occurs only if the session was active at the time of the takeover.
- If you enter a **/TRACE** command with and without the TAKEOVER keyword, the last command you entered is in effect.
- You can issue this command for VTAM nodes and MSC links during takeover.
- The **/TRACE SET OFF NODE xxx TAKEOVER, /TRACE SET OFF LINE xxx TAKEOVER, /TRACE SET OFF LINK xxx TAKEOVER** or **UPDATE MSLINK NAME(linkname) STOP(TRACE) SET(TKOTRC(N))** command turns off the trace anytime before takeover.

## Stopping the trace

To stop the DC trace, enter one of the **/TRACE SET OFF** commands from the master terminal or the z/OS console.

### About this task

- For VTAM terminals:

```
/TRACE SET OFF NODE P1
```

- For ISC links:

```
/TRACE SET OFF NODE P1
      OR
/TRACE SET OFF NODE P1 USER P2
```

- For logical LINKs:

```
/TRACE SET OFF LINK P1,...,Pn|ALL
```

- For UNITTYPE:

```
/TRACE SET OFF UNITTYPE P1,...Pn  
UPDATE MSLINK NAME(linkname|linkname*|*) START(TRACE)
```

- For an XRF environment:

```
/TRACE SET OFF NODE xxx TAKEOVER  
  
/TRACE SET OFF LINE xxx TAKEOVER  
  
/TRACE SET OFF LINK xxx TAKEOVER  
UPDATE MSLINK NAME(linkname) STOP(TKOTRC)
```

## Printing the trace records

To format and print the trace records, use the File Select and Formatting Print utility (DFSERA10).

### About this task

The DC trace snaps DC control blocks and I/O buffers to the OLDS/WADS as X'6701' log records. These records are archived to the system log data set (SLDS).

To format and print the trace records, use the following method:

- **File Select and Formatting Print utility (DFSERA10)**

To use the File Select and Formatting Print utility (DFSERA10), specify E=DFSERA30 to format the records before printing. The following example shows the JCL you might use to print DC trace records.

```
// JOB jobname  
//S EXEC PGM=DFSERA10  
//SYSPRINT DD SYSOUT=A  
//SYSUT1 DD DSN=DSN of SLDS,.....  
//SYSIN DD *  
CONTROL CNTL  
OPTION PRINT 0=5,V=6701,L=2,T=X,E=DFSERA30  
//
```

where

O = Offset  
L = Length  
V = Value  
T = Type  
E = Exit

Even if the DC trace was started for many terminals, you can print trace entries for a specific terminal by using the following OPTION statement.

```
CONTROL CNTL DDNAME=...  
OPTION PRINT 0=5,T=X,L=1,V=67,C=M  
OPTION PRINT 0=89,T=C,L=8,V=xxxxxxxx,C=E,E=DFSERA30
```

where xxxxxxxx = terminal (node) name

A trace record might span several X'6701' log records. If you use the OPTIONS statements, only the first log record is printed.

### Related reference

[“Content of the trace records” on page 282](#)

You can evaluate DC trace records while debugging errors, building keywords, or evaluating APAR descriptions.

## Content of the trace records

You can evaluate DC trace records while debugging errors, building keywords, or evaluating APAR descriptions.

You can evaluate DC trace records during any of the following tasks:

- Debugging user errors in exit routines or user modifications relating to communications
- Debugging errors in other entities in the communication network (such as programmable terminals or other host processors)
- Building a keyword string to search for known problems
- Evaluating existing APAR descriptions to isolate problems that are most like the one you are experiencing

The first line of each trace record shows the ID:

```
ID= xxx    SEGNO= mm RECNO= nnnnnnnn TIME HH.MM.SS.TT DATE YY.DDD
```

xxx can be any of the following trace record identifiers (IDs):<sup>1</sup>

### ID

#### Description

#### A xx

Communication analyzer activity (DFSICIO0)

#### AERx

Access method error

#### C xx

Communication analyzer activity (DFSCIOC0 in DFSICIO0)

#### CI04

TM shared queues re-read error detected

#### CIO2

Device-dependent module (DDM) SDC read for output

#### CIO3

Device-dependent module (DDM) conditional SDC *wash* output

#### CMEA

Before calling Message Control/Error exit DFSCMUX0

#### CMEB

After calling Message Control/Error exit DFSCMUX0

#### CMEI

Message Control/Error exit interface processing

#### COFC

Entry to the output format control, MFS-supported devices (DFSCOF00)

#### CRTU

Output User Creation user exit routine failure

#### CTTR

ISC TCPIP Trace ABORT.

#### CVCT

VTAM trace. This log record is written even though DC trace is not active on the terminal/link.

---

<sup>1</sup> An asterisk (\*) in this list is a wildcard character, meaning that any character can replace the asterisk.



**CVCV**

XRF class 2 takeover trace. This log record is written for XRF class 2 terminals during takeover, even though DC trace is not active on the terminal.

**D xx**

Device-dependent module (DDM) activity

**DDxx**

Output processing by DFSCOF0

**DSIM**

SIMLOGON attempt of a dynamic terminal

**ESIM**

SIMLOGON error for a dynamic terminal

**FERR**

MFS-block fetch error

**FESx**

Front-end switch user exit routine activity

**FEXT**

Before field edit exit routine

**FMTx**

Message Format Service activity (MFS)

**HCSW**

XRF class 1 takeover trace. This log record is written for XRF class 1 terminals during takeover, even though DC trace is not active on the terminal.

**ICLR**

Message router activity

**INIT**

Device-dependent module (DDM) for ISC TCP/IP that sends session initiation requests.

**IRxx**

Device-dependent module (DDM) for ISC TCP/IP that reads input messages that use the IBM CICS Transaction Server for z/OS IPIC protocol.

**IWxx**

Device-dependent module (DDM) for ISC TCP/IP that builds output messages that use the CICS IPIC protocol.

**MFSP**

MFS activity to detect change in the content of a protected field that is in the input from a 3270 or SLU2 device. Set MFSPFV=Y in the DFSDCxxx member of the IMS PROCLIB data set to configure this option. This log record is written even though DC trace is not active on the terminal.

**MSGs**

Device-dependent module (DDM) for ISC TCP/IP that sends messages that use the CICS IPIC protocol.

**MTRP**

Block verification error

**RESP**

Device-dependent module (DDM) for ISC TCP/IP that sends ACK/NAK messages that use the CICS IPIC protocol.

**SDC1**

Device-dependent module (DDM) SDC output read error

**SDC2**

Device-dependent module (DDM) SDC message reread error

**SEXT**

Before segment edit exit routine

**SGNX**

Signon user exit routine failure

**SPCL**

Close spool data set

**SPOP**

Open spool data set

**SPRE**

Read spool data set

**SPWR**

Write spool data set

**TERM**

Device-dependent module (DDM) for ISC TCP/IP that sends session termination requests.

**TRCE**

Non-SNA 3270 error

**VTPO**

Non-posting of ECB trace (DFSVTPO0)

**Exception:** MSC has its own analyzer module and entry types.

The table below shows the types of data communication (DC) trace records and what each trace record contains. Some of the acronyms used in the table are:

**SEG**

Segment (DECAREA buffer)

**MFS**

MFS input work/MFS output work/MFS protect work

**QBUF**

Queue buffer

**IOPUF**

TP buffer

**S25**

Save area 2-5

**SALL**

Save area all

Table 62. DC trace records

Trace ID	Function	Traced by	When traced or / TRACE option	What is traced
A01	Process input. <a href="#">“1” on page 290</a>	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, DDM	CTB, CLB, CXB, CRB, CIB, CCB, QBUF, IOBUF, INPCNTS, OUTCNTS, EMHB <a href="#">“2” on page 290</a>
A02	Do read. <a href="#">“1” on page 290</a>	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, DDM	CTB, CLB, CXB, CRB, IOBUF, EMHB <a href="#">“2” on page 290</a>
A03	What is next.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, DDM	CTB, CLB, CRB, CTT
A04	Get Next segment.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, DDM	CTB, CLB, CNT
A05	Do write. <a href="#">“1” on page 290</a>	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, DDM	CTB, CLB, CXB, CRB, CCB, IOBUF, EMHB <a href="#">“2” on page 290</a>
A06	After good write.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, DDM	IOB, CTB, CLB, CXB, CRB, CCB

Table 62. DC trace records (continued)

Trace ID	Function	Traced by	When traced or / TRACE option	What is traced
A07	After bad write. <a href="#">“1” on page 290</a>	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, DDM	IOB, CTB, CLB, CRB, CCB, IOBUF, EMHB <a href="#">“2” on page 290</a>
A08	Cancel message, do not DEQ.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, DDM	CTB, CLB, CRB
A09	Generate system message. <a href="#">“1” on page 290</a>	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, DDM	CTB, CLB, CRB, MFS
A10	Quiesce without stopping.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, DDM	CTB, CLB, CRB, CCB
A11	Retrieve last DEQD message.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, DDM	CTB, CLB, CNT, CRB
A12	Wait for ASYNC I/O or output ENQ.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, DDM	CTB, CLB, CRB, CCB, IOBUF, EMHB <a href="#">“2” on page 290</a>
AER1	Access method error.	DFSICIO0 <a href="#">“9” on page 291</a>	Always	CTB, CLB, CNT, QBUF, SALL, CTT, PCB
AER2	Access method error. <a href="#">“3” on page 290</a> , <a href="#">“1” on page 290</a>	DFSICIO0 <a href="#">“9” on page 291</a>	Always	IOB, CTB, CLB, CNT, CXB, CRB, CIB, CCB, QBUF, IOBUF, SALL, CTT, PCB, EMHB <a href="#">“2” on page 290</a>
C00	Get queue buffer.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, MFS	CTB, CNT, CIB, SALL
C01	Reposition queue buffer.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, MFS	CTB, CNT, CIB, SALL
C02	Get Next.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, MFS	CTB, CNT, CIB, SALL
C03	DEQ output.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, MFS	CTB, CNT, CIB, SALL
C04	Place output back in queue.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, MFS	CTB, CNT, CIB, SALL
C05	Find output.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, MFS	CTB, CNT, CIB, SALL
C06	Get new output message or QMGR call.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, MFS	CTB, CNT, CIB, SALL
C07	Free input buffer.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, MFS	CTB, CNT, CIB, SALL
C08	Get output buffer.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, MFS	CTB, CNT, CIB, SALL
C09	User output edit.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, MFS	CTB, CNT, CIB, SALL
C10	Call queue MGR.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, MFS	CTB, CNT, CIB, SALL

Table 62. DC trace records (continued)

Trace ID	Function	Traced by	When traced or / TRACE option	What is traced
C11	Get device-dependent module (DDM) work buffer.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, MFS	CTB, CNT, CIB, SALL
C12	Free device-dependent module (DDM) work buffer.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, MFS	CTB, CNT, CIB, SALL
C13	Free receive-any buffer.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, MFS	CTB, CNT, CIB, SALL
CIO2	Device-dependent module (DDM) SDC read output	DFSCIO20	ALL DDM	copy ctl blk list from CVCT entry
CIO3	Device-dependent module (DDM) SDC wash output	DFSCIO30	ALL DDM	copy ctl blk list from CVCT entry
CMEA	Before call MSG CTRL Error exit.	DFSCMEIO	Before call DFSCMUX0	If ITASK is a CLB or LLB: CTB, CLB, CRB, QBUF, IOBUF, INP/OUTP CNTS, DDM, MSNB
CMEB	After call MSG CTRL Error exit.	DFSCMEIO	After call DFSCMUX0	If ITASK is a CLB or LLB: CTB, CLB, CRB, QBUF, IOBUF, INP/OUTP CNTS, DDM, MSNB If ITASK is a PST: PST, MSGPRFX, SMB, MSNB
CMEI	Error procedure in DFSCMEIO.	DFSCMEIO	On some errors	If ITASK is a CLB or LLB: CTB, CLB, CRB, QBUF, IOBUF, INP/OUTP CNTS, DDM, MSNB If ITASK is a PST: PST, MSGPRFX, SMB, MSNB
COFC	Let MFS edit output.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, DDM	CTB, CLB, CNT, CRB, CIB, IOBUF, EMHB <a href="#">“2” on page 290</a>
CRTU	Output User Creation exit routine failure.	DFSCRTU0	Always	See notes <a href="#">“10” on page 291</a>
CTTR	ISC TCP/IP Trace ABORT	DFSCCTR0	ALL, DDM	CTB,CLB,CRB,SPQB, SPQBEXT,INP_CNTS, CTT, IOBUF, LU6WA, TCPWORK, CTBWORK
CVCT	VTAM TRACE/ABORT. <a href="#">“1” on page 290</a>	DFSCVCT0	ALL, DDM	CTB, CLB, CNT, CRB, IOBUF, CTT, INPCNTS, EMHB <a href="#">“2” on page 290</a>
CVCV	XRF class 2 takeover. <a href="#">“1” on page 290</a>	DFSCVCV0	Always	CLB, CTB, CTT, LLB, LTB, LXB, LU6WA, CNT, CRB, SPQB, CTC, MSNB, EMHB, IOBUF, DDM
D01	Write setup.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, DDM	CTB, CLB, CNT, CRB, CIB, QBUF, S25
D02	Write interrupt. <a href="#">“1” on page 290</a>	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, DDM	IOB, CTB, CLB, CRB, IOBUF, S25, EMHB <a href="#">“2” on page 290</a>
D03	Read setup.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, DDM	CTB, CLB, CNT, CRB

Table 62. DC trace records (continued)

Trace ID	Function	Traced by	When traced or / TRACE option	What is traced
D04	Read interrupt. <a href="#">“1” on page 290</a>	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, DDM	IOB, CTB, CLB, CRB, IOBUF, S25, EMHB <a href="#">“2” on page 290</a>
D05	Cleanup.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, DDM	IOB, CTB, CLB, CNT, CXB, CRB, CIB, CCB, MFS, QBUF, IOBUF, SALL, EMHB <a href="#">“2” on page 290</a>
D07	LOGON. <a href="#">“1” on page 290</a>	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, DDM	CTB, CLB, CNT, CRB
DD6M	Output build (MFS).	DFSCOF00	ALL, DDM	CTB, CLB, CNT, CRB, CIB, SEG, MFS, IOBUF, S25, EMHB <a href="#">“2” on page 290</a>
DD6S	Output build (Non-MFS).	DFSCOF00	ALL, DDM	CTB, CLB, CNT, CRB, CIB, IOBUF, S25, EMHB <a href="#">“2” on page 290</a>
DD8	Prepare for output.	DFSCOF00	ALL, DDM	CTB, CLB, CNT, CRB, CIB, IOBUF, S25, EMHB <a href="#">“2” on page 290</a>
DDM1	Write set up through COFC.	DFSCOF00	ALL, DDM	CTB, CLB, CNT, CRB, CIB, MFS, IOBUF, S25, EMHB <a href="#">“2” on page 290</a>
FERR	MFS block fetch error. <a href="#">“3” on page 290</a>	DFSCFEO0	Always	CIB, CTT, MFSBP00, MFSTRACE <a href="#">“4” on page 290</a>
FES1	Entry to front end switch user exit.	DFSICIO0 <a href="#">“9” on page 291</a>		CTB, CLB, CNT, QBUF, S25
FES2	Exit from front end switch user exit.	DFSICIO0 <a href="#">“9” on page 291</a>		CTB, CLB, CNT, QBUF, S25
FEXT <a href="#">“5” on page 290</a>	Before field edit exit.	DFSCFEI0	MFS	CTB, CIB
FMT1	Return from DFSFEI0 or unformatted input.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, MFS	CTB, CLB, CIB, IOBUF, EMHB <a href="#">“2” on page 290</a>
FMT2	MFS go to DFSFEI0 formatted input.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, MFS	CTB, CLB, CIB, IOBUF, EMHB <a href="#">“2” on page 290</a>
FMT3	MFS complete process MSG segment.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, MFS	CTB, CLB, CIB, MFS, QBUF
FMT4	Get next input.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, MFS	CTB, CLB, CIB
FMT6	Clean up resources.	DFSICIO0 <a href="#">“9” on page 291</a>	ALL, MFS	CTB, CLB, CIB
HCSW	XRF class 1 takeover. <a href="#">“1” on page 290</a>	DFSHCSW0	Always	IOBUF, CNT, CRB, CTT, CTB, CLB
ICLR	Message router.	DFSICLR0	Always	CTB, CLB, CTT, PCB

Table 62. DC trace records (continued)

Trace ID	Function	Traced by	When traced or / TRACE option	What is traced
INIT	Device-dependent module (DDM) for ISC TCP/IP that sends session initiation requests.	DFSCT7E0	All, DDM	CTB,CLB,CRB,SPQB, SPQBEXT,INP_CNDS, CTT, IOBUF, LU6WA, TCPWORK, CTBWORK
IR4B	Device-dependent module (DDM) for ISC TCP/IP that reads input messages that use the CICS IPIC protocol.	DFSCT4B0	All, DDM	CTB,CLB,CRB,SPQB, SPQBEXT,INP_CNDS, CTT, IOBUF, LU6WA, TCPWORK, CTBWORK
IR7A	Device-dependent module (DDM) for ISC TCP/IP that reads input messages that use the CICS IPIC protocol.	DFSCT7A0	All, DDM	CTB,CLB,CRB,SPQB, SPQBEXT,INP_CNDS, CTT, IOBUF, LU6WA, TCPWORK, CTBWORK
IR8B	Device-dependent module (DDM) for ISC TCP/IP that reads input messages that use the CICS IPIC protocol.	DFSCT8B0	All, DDM	CTB,CLB,CRB,SPQB, SPQBEXT,INP_CNDS, CTT, IOBUF, LU6WA, TCPWORK, CTBWORK
IW1B	Device-dependent module (DDM) for ISC TCP/IP that builds output messages that use the CICS IPIC protocol..	DFSCT1B0	All, DDM	CTB,CLB,CRB,SPQB, SPQBEXT,INP_CNDS, CTT, IOBUF, LU6WA, TCPWORK, CTBWORK
IW3B	Device-dependent module (DDM) for ISC TCP/IP that builds output messages that use the CICS IPIC protocol.	DFSCT3B0	All, DDM	CTB,CLB,CRB,SPQB, SPQBEXT,INP_CNDS, CTT, IOBUF, LU6WA, TCPWORK, CTBWORK
IW6A	Device-dependent module (DDM) for ISC TCP/IP that builds output messages that use the CICS IPIC protocol.	DFSCT6A0	All, DDM	CTB,CLB,CRB,SPQB, SPQBEXT,INP_CNDS, CTT, IOBUF, LU6WA, TCPWORK, CTBWORK

Table 62. DC trace records (continued)

Trace ID	Function	Traced by	When traced or / TRACE option	What is traced
IW7A	Device-dependent module (DDM) for ISC TCP/IP that builds output messages that use the CICS IPIC protocol.	DFSCT7A0	All, DDM	CTB,CLB,CRB,SPQB, SPQBEXT,INP_CNDS, CTT, IOBUF, LU6WA, TCPWORK, CTBWORK
IW8A	Device-dependent module (DDM) for ISC TCP/IP that builds output messages that use the CICS IPIC protocol.	DFSCT8A0	All, DDM	CTB,CLB,CRB,SPQB, SPQBEXT,INP_CNDS, CTT, IOBUF, LU6WA, TCPWORK, CTBWORK
MFSP	MFS detected that protected data that is in the input is altered.	DFSCFEI0	Always	CTB, CLB, CIB, MFS, IOBUF
MSG5	Device-dependent module (DDM) for ISC TCP/IP that sends messages that use the CICS IPIC protocol.	DFSCT7E0	All, DDM	CTB,CLB,CRB,SPQB, SPQBEXT,INP_CNDS, CTT, IOBUF, LU6WA, TCPWORK, CTBWORK
MTRP <a href="#">“8” on page 291</a>	Block verification error.	DFSCFE00		CLB, CIB, MFS, CTT
MTRP <a href="#">“7” on page 291</a>	Block verification error.	DFSCFEI0		CLB, CIB, MFS, CTT
RESP	Device-dependent module (DDM) for ISC TCP/IP that sends ACK/NAK messages that use the CICS IPIC protocol.	DFSCT7E0	All, DDM	CTB,CLB,CRB,SPQB, SPQBEXT,INP_CNDS, CTT, IOBUF, LU6WA, TCPWORK, CTBWORK
SDC1	Device-dependent module (DDM) SDC read error	DFSCIO20	ALL DDM	copy ctl blk list from CVCT entry
SDC2	Device-dependent module (DDM) SDC reread error	DFSICIO4	ALL DDM	copy ctl blk list from CVCT entry
SEXT <a href="#">“6” on page 291</a>	Before segment edit exit.	DFSCFEI0	MFS	CTB, CIB

Table 62. DC trace records (continued)

Trace ID	Function	Traced by	When traced or / TRACE option	What is traced
TERM	Device-dependent module (DDM) for ISC TCP/IP that sends session termination requests.	DFSCT7E0	All, DDM	CTB,CLB,CRB,SPQB, SPQBEXT,INP_CNTS, CTT, IOBUF, LU6WA, TCPWORK, CTBWORK
TRCE	Non-SNA 3270 error.	DFSDN130, DFSDN140, DFSDS060	Always	IOB, CTB, CLB, S25, CTT
VTPO	Rejected posting of ECB.	DFSVTPO0	ALL, DDM	See notes <a href="#">“11” on page 291</a>

**Notes:**

1. See [“Diagnosing line and terminal problems” on page 291](#) for more information on this trace code.
2. Fast Path EMHB buff traces (if present) with I/O buffers
3. Module return code saved in CLBTEMP4
4. Return codes from DFSFFRH0 (block fetch), MFSTRACE (when in MFSTEST) or MFSBPCA (when not in MFSTEST); MFSTRACE=MFSTEST trace parameters, MFSBPCA=MFS Buffer Pool Control Area:

**Offset in hex**

**0**

Current pool space in use

**4**

Maximum space used

**5**

Status flag

X'80' I/O active for a task

X'40' Task(s) queued for I/O

X'20' A task dequeued and posted

**9**

Error status

X'BB' BLDL error

X'FF' READ error

**A**

Block name for BLDL error

**10**

BLDL return code on error

**12**

Sense from read error

**14**

CSW status from read error

**16**

Block name for read error

**20**

List for BLDL macro

5. Besides CIB and CTB:



**PARMLIST**

Parameter list to be passed to EXIT

**FIELD**

Field data before exit

6. Besides CIB and CTB:

**PARMLIST**

Parameter list to be passed to EXIT

**SEGMENT**

Segment data before exit

7. SEXT is logged if TRAP 1 is set by /TRACE and a buffer overwrite occurs.

8. MTRP is logged if TRAP 1 is set by /TRACE and a buffer overwrite occurs. In addition to the blocks, the DIF/DOF, MID/MOD, MFBP, and FRE are traced. If in output, R9 is also traced.

9. The MSNB control block content is traced by DFSICIO0 if the /DEQ LTERM, /DEQ NODE, or the /DEQ MSNAME command is entered with the PURGE or PURGE1 keywords.

10. The CRTU trace entry is mapped in [“Format of 6701 log record with CRTU identifier” on page 293.](#)

11. The VTPO trace entry is mapped in [“Format of the 6701 log record with VTPO identifier” on page 294.](#)

**Related tasks**

[“Printing the trace records” on page 281](#)

To format and print the trace records, use the File Select and Formatting Print utility (DFSERA10).

## Diagnosing line and terminal problems

Use trace record identifiers to help in diagnosing line and terminal problems.

The trace records with the following identifier are useful in diagnosing line and terminal problems:

**A01**

TERMINAL INPUT READY FOR IMS PROCESSING

**I TP BUF**

Contains input device segment 6 to 36 bytes from the beginning of the buffer. The data is preceded by a 2-byte length and 2 bytes of zeros.

**A02**

PRIOR TO ISSUING VTAM I/O REQUEST. (LOGICAL READ)

**CLB**

For remote 3270:

**X'0001'**

Special poll (read sense/status)

**X'0401'**

Read initial (general poll)

**X'0082'**

Write initial

**X'0084'**

Write continue

Offset X'0C' contains the address in TP BUF to read into or write from.

**I TP BUF**

The input TP buffer contains data to be written if this is an output operation. For VTAM nodes, the RPL begins at offset X'08'.

**A05**

PRIOR TO ISSUING VTAM I/O REQUEST. (LOGICAL WRITE)

**CLB**

Refer to the information for record A02.

**O TP BUF**

The output TP buffer contains data to be written if this is an output operation. For VTAM nodes, the RPL begins at offset X'08'.

**A07**

GENERATE 'UNABLE TO RECEIVE/OUTPUT' MESSAGE

See the preceding D02 or D04 record for the cause.

**A09**

GENERATE ERROR MESSAGE

See the preceding D02, D04, or D07 record for the cause.

**AER2**

SHOULD NOT OCCUR ERROR HAS OCCURRED

**CLB**

Offset X'3E' contains the error message number in hexadecimal. All available control blocks and buffers are logged. This record is produced even if the trace is not set on.

**CRTU**

OUTPUT USER CREATION EXIT ROUTINE FAILURE

**CVCT**

VTAM DEVICE SUPPORT TRACE

**CLB**

Normally offset X'1C' contains the complemented IMS message key of an IMS master terminal message. All available control blocks and buffers are logged. This record is produced even if the trace is not set on.

**I TP BUF of O BUF**

The VTAM RPL begins at offset X'08'.

**CVCV**

XRF CLASS 2 TAKEOVER TRACE

This log record is written for XRF class 2 terminals during takeover, even though DC trace is not active on the terminal. This record can be used to diagnose subsequent session failures when used in conjunction with CVCT records.

**D02**

VTAM HAS POSTED I/O COMPLETE. (LOGICAL WRITE INTERRUPT)

**CLB**

**Offset X'00' =**

Post code

- X'40' for VTAM = normal completion

Other key fields are DECFLAGS and DECERRST. For VTAM, key fields are CLBVFLAG and CLBLOST.

**O TP BUF**

For VTAM nodes, the VTAM RPL begins at offset X'08'.

**D04**

VTAM HAS POSTED I/O COMPLETE. (LOGICAL READ INTERRUPT)

**CLB**

Refer to the information for record D02.

**IOB**

Refer to the information for record D02.

## I TP BUF

The input TP buffer contains data read from the terminal if the last operation was a read or poll. For VTAM nodes, the RPL begins at offset X'08'.

## D07

DEVICE DEPENDENT INITIALIZATION/TERMINATION

## CLB

Refer to information for record D02.

## O TP BUF

The VTAM RPL begins at offset X'08'.

## HCSW

XRF CLASS 1 TAKEOVER TRACE

This log record is written for XRF class 1 terminals during takeover, even though DC trace is not active on the terminal. This record can be used to diagnose subsequent session failures when used in conjunction with CVCT records.

## VTPO

REJECTED POSTING OF ECB

## Format of 6701 log record with CRTU identifier

A map of the formatted CRTU log record, including offset, hex code, and a description is shown.

Table 63. Map of formatted CRTU log record

Offset	Hex code	Description
+0	H	Length of buffer
+2	XL5	Internal use
+7	X	DFSCRTU0 return code (see below)
+8	XL68	Internal use
+4C	CL8	Input Lterm name
+54	XL52	Internal use

## DFSCRTU0 return codes (decimal)

The DFSCRTU0 return codes and their meanings are listed.

```
4  'ENVIRONMENT' INCORRECT (for example, NO ETO,  
16 NO DFSINSX0 WITH SHARED QUEUES).  
16  DUPLICATE LTERM/SMB NAME.  
20  NO USER DESCRIPTOR COULD BE LOCATED  
20  FOR USE IN CREATING USER STRUCTURE.  
24  INVALID INPUT LTERM NAME.  
28  DFSINSX0 REJECTED USER-CREATION REQUEST.  
32  STORAGE COULD NOT BE OBTAINED TO CREATE  
32  USER STRUCTURE.  
36  STATIC USER ALREADY EXISTS.  
40  INSERT EXIT PRAMETER ERROR: INVALID LTERM  
40  NAME, BAD FORMAT.  
48  AVAILABLE.  
52  LATCHING ERROR OCCURRED.  
56  STORAGE MANAGER ERROR - DFSPPOOL.  
60  ERROR IN ADDING DYNAMIC SMB TO HASH TABLE.  
64  INSERT EXIT (DFSINSX0) PARAMETER ERROR:  
64  INVALID DYNAMIC TRANSACTION DATA.  
68  LOCAL CNT FOUND, BUT DESTINATION REGISTERED  
72  TO RESOURCE MANAGER AS A TRANSACTION.  
72  LOCAL SMB FOUND, BUT DESTINATION REGISTERED  
76  TO RESOURCE MANAGER AS AN LTERM.  
76  DESTINATION REGISTERED TO RESOURCE MANAGER AS  
80  A CPIC TRANSACTION, APPC DESCRIPTOR, OR MSNAME.  
80  DESTINATION COULD NOT BE VALIDATED IN RESOURCE  
MANAGER DUE TO AN RM INTERFACE ERROR.
```

84 SMB CREATION REQUESTED, BUT DESTINATION WAS ALREADY  
REGISTERED TO RESOURCE MANAGER AS AN LTERM, CPIC  
TRANSACTION, APPC DESCRIPTOR, OR MSNAME.  
88 SMB CREATION REQUESTED, BUT SHARED QUEUES IS NOT  
ACTIVE.

## Format of the 6701 log record with VTPO identifier

If an APPC or OTMA message is discarded because of a send type error, IMS does not log a type 6701–CMEA/CMEB record for the error.

The lack of type 6701–CMEA/CMEB records makes debugging for the Message Control/Error exit routine (DFSCMUX0) difficult. It does log type 6701–CMEA/CMEB records for errors related to other devices, however. The following table shows the VTCB posting in DFSVTPO0.

Table 64. VTCB posting in DFSVTPO0

Offset	Hex code	Description
+0	X	Function code
	X'00'	VTCB is to be posted
	X'04'	VTCB is to be released
	X'08'	Check if ACB can be closed
	X'0C'	Delete a VTCB
	X'10'	Stacked logon for static CLB
	X'14'	NSEXIT for static CLB
	X'18'	NSEXIT for dynamic CLB
	X'1C'	LOSTERM for static CLB
	X'20'	LOSTERM for dynamic CLB
+1	X	Type of checking RQD for post
	X'04'	Post if node is active
	X'08'	Post if node not active
	X'0C'	Post if idle and not active
	X'10'	Hard post the node
	X'14'	Post an MSC LLB
+2	X	Conditional data for posting
	X'80'	Type is ISC parallel session
	X'40'	Type is MSC LLB
	X'20'	Z-NET cancel in progress
		On detection of an error, this byte contains one of the following reject codes:
	X'01'	VTCB not specified
	X'02'	Inspection failed—check subcode
	X'03'	Node not idle
	X'04'	RQR failed—check subcode
	X'05'	Node active—check subcode
	X'06'	Node not alive—check subcode
	X'07'	Invalid request
	X'08'	MSC link already posted
	X'09'	MSC send outstanding
	X'0A'	Node already dispatched
	X'20'	No VTCB to delete
	X'30'	CINIT rejected by PLU (NSX)
	X'31'	VTAM error (NSX)
	X'40'	Stacked logon procedure failure

Table 64. VTCB posting in DFSVTPO0 (continued)

Offset	Hex code	Description
+3	X	Posting-rejection subcode <sup>1</sup>
	X'01'	Node already dispatched (RQR)
	X'02'	Node already posted (RQR)
	X'03'	Unpostable I/O (RQR)
	X'04'	Clear issued (RQR)
	X'05'	Inact performed (RQR)
	X'01'	SPQB not found (INSPECT)
	X'02'	No match on CLB ADDR (INSPECT)
	X'03'	VOPEN not on (INSPECT)
	X'04'	VTCB not found by scan (INSPECT)
	X'05'	No match on VTCBs (INSPECT)
	X'06'	CIDs do not match (INSPECT)
	X'07'	VOPEN not set (INSPECT)
	X'08'	Temporary VTCB (INSPECT)
	X'01'	No /idle node CMD (POSTRTN)
	X'02'	Node inoperable (POSTRTN)
	X'03'	Node dispatched (POSTRTN)
	X'04'	Line already posted (POSTRTN)
	X'05'	V2SND is set (POSTRTN)
	X'06'	Not XRF sync mode (POSTRTN)
	X'07'	Not SCIP exit with clear (POSTRTN)
	X'08'	SCIP exit bindrace done (POSTRTN)
+4	0F	Post code
+4	X	NSEXIT flag
	X'80'	Cleanup RU
	X'40'	Notify RU
+5	X	NSEXIT type for CLBLOST
+6	X	Reason code for CTBRTERM
+7	X	Notify reason code
+8	F	VTCB address
+C	CL8	VTAM node name
+14	F	CID
+18	CL8>	SPQB name if parallel session
+20	0F	CLBNCID for a stacked logon
+20	F	Sense data (NSEXIT)

**Note:**

1. This byte contains an additional qualifier subcode.

An example of a Data Communication (DC) trace record is shown. This example shows part of a normal VTAM terminal logon flow.

## 296 IMS: Diagnosis

```

02235040 000040 10308050 00000000 80800000 44000000 00000000 00000000 00000000 00000000
*...&;.....*
02235060 000060 00000000 00000000 80008010 00000000 00000000 00000000 00000000 00000000
*...&;.....*
02235080 000080 00000000 00440000 D0000040 00000000 02248078 C2F0D7F0 F6404040 D9C5C3D6
*...B0P06 RECO*
022350A0 0000A0 D9C44040 00000000 00000000 41080002 00000001 00000000 00000000 00000000
*RD .....*
022350C0 0000C0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*...&;.....*
022350E0 0000E0 TO 02235160 000160 SAME AS ABOVE
02235180 000180 00000000 00000000 00000000 00000000 FF00403F C181AA55 01900000 00000000
*...AA.....*
022351A0 0001A0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*...&;.....*
022351C0 0001C0 00000000 00000000
*...&;.....*
INTERNAL TRACE RECORD ID = D 07 SEGNO=00 RECNO = 0000013E TIME 08.41.00.43 DATE 88.047
CLB
02248078 000000 40000000 00000000 00000000 02235008 00000000 00000000 00000000 00000000
*...&;.....*
02248098 000020 00000000 00000000 C2F0D7F0 F6404040 10020100 022480FC 00000000 00050007
*...B0P06.....*
022480B8 000040 0840598F 0088047F 00010000 00000000 022480FC 80000000 00000000 00000000
*...H.....*
022480D8 000060 00000000 02235000 00000000 00000000 00000000 00000000 40000000 00000000
*...&;.....*
022480F8 000080 00000000
*...&;.....*
*.....*

CTB
022480FC 000000 00038CC8 02248078 00000000 000B2000 00000000 082A0000 0000FFFF 0003614C
*...H.....*/<*
0224811C 000020 00000000 00000000 022481C4 00004040 40404040 40400000 00000000 00000000
*...AD.....*
0224813C 000040 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*...&;.....*
0224815C 000060 SAME AS ABOVE
INP CNTS
0003614C 000000 00000000 00000000 00000000 00000000 00000000 00820084 00000000 C2F0D7F0
*...B.D....B0P0*
0003616C 000020 F6404040 00000001 022480FC 000371F0 FFFF0909 00000000 00000000
*6.....*
NEXT CNT
000371F0 000000 00000000 00000000 00000000 00000000 00000000 00820084 00000000 D4E3D6D4
*...B.D....MTOM*
00037210 000020 C1E2E340 00000001 022480FC 00000000 FFFF0909 00000000 00000000
*AST .....*

```

## Diagnosing problems in the Queue Control Facility Message Requeuer

The Queue Control Facility Message Requeuer (MRQ) processor module (DFSQMRQ0), which is part of the IMS Transaction Manager, provides diagnostic information for diagnosing errors while running the IBM IMS Queue Control Facility for z/OS (QCF).

Although problems can be diagnosed separately in the QCF product by using SCRAPLOG records and in the Queue Control Facility processor module by using 6701-MRQE diagnostic records, QCF and the Queue Control Facility processor work together to allow inserting and loading, querying, recovering, deleting and unloading, recovering, and viewing messages on the IMS message queue data sets and shared message queue structures.

In this topic, the information about SCRAPLOG records also applies to SCRAPSEL and SCRAPCAN records. The SCRAPSEL, SCRAPCAN, and SCRAPLOG data sets are generated by the IQCSELECT, IQCCANCL, and IQCINSRT modules of QCF, respectively, and are identical in both format and function.

- QCF functions help you accomplish the following tasks:
  - Message queue recovery when you want to return messages to the IMS queue for reprocessing.
  - Application recovery when you want to return messages to the IMS queue for reprocessing.
  - IMS queue maintenance (you can query, browse, unload, and load IMS nonshared queue environments).
  - Message queue migration and fallback.
  - Stress, regression, and application testing when transaction data is needed to simulate production loads or application input.

- A queue overflow protection function monitors queue usage and takes action to prevent queue utilization from reaching critical thresholds (non-shared queues environment).
- An ISPF front-end enables you to select QCF functions and selection criteria to complete the following tasks:
  - Query messages (or IMS status) on the queue
  - Unload (delete) messages from the queue
  - Load messages onto the IMS message queues
  - Release or terminate waiting tasks (nonshared queue environment)
  - Maintain the tables associated with queue overflow protection (nonshared queue environment)

## **IBM IMS Queue Control Facility for z/OS interface**

You can invoke the IBM IMS Queue Control Facility for z/OS functions through either a user control card input or a TSO/ISPF interface. The functions are Browse, Query, Load/Insert, Queue Overflow Protection, Recover, and Unload.

The following figure describes the IMS Queue Control Facility interface to IMS.



## QCF V2.1

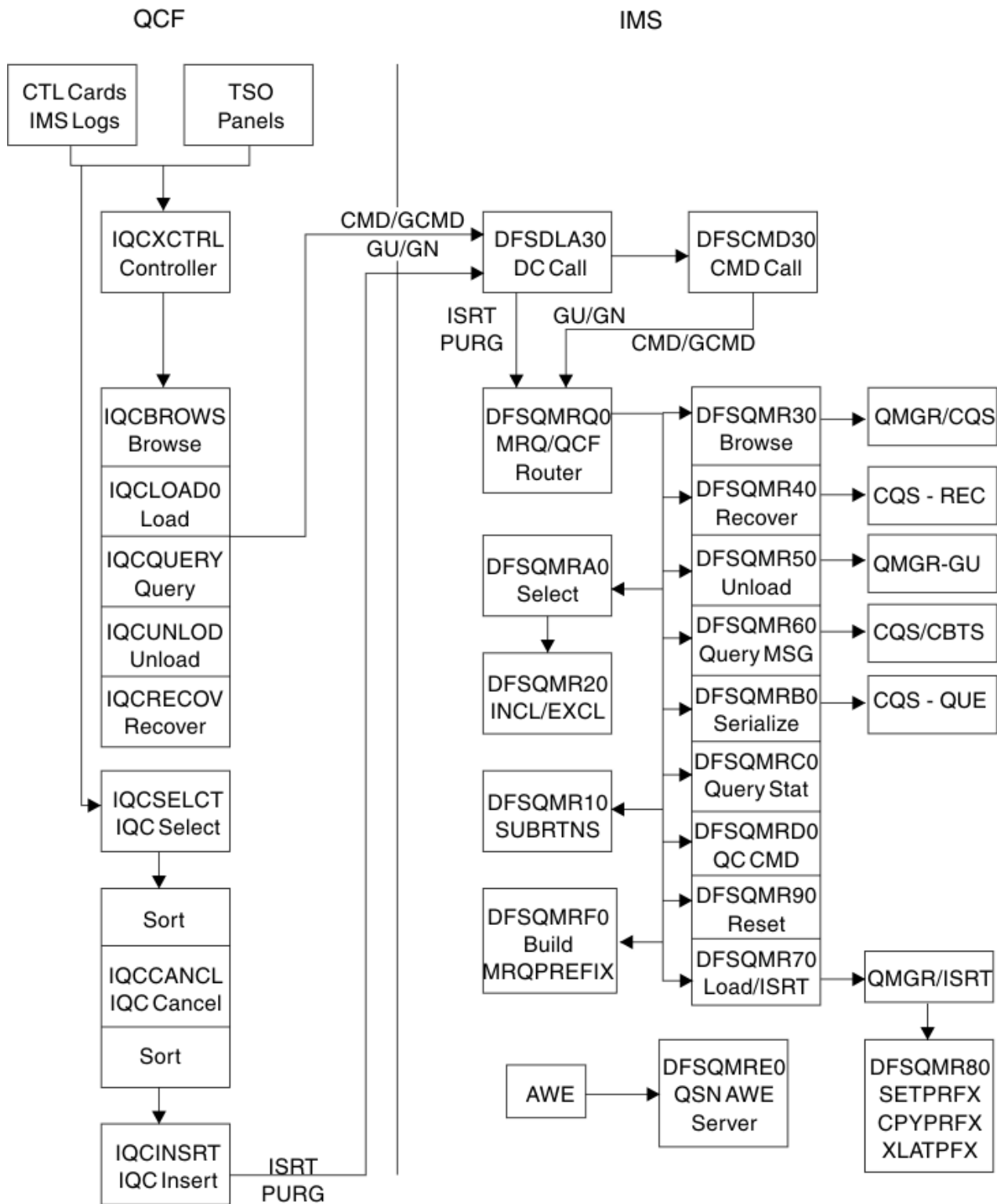


Figure 67. IMS Queue Control Facility interface to IMS

The functions are processed by function routines within IMS Queue Control Facility, and passed to corresponding function routines in IMS through a BMP application program interface (API). GCMD calls are used to invoke the function, and the messages, query, and status data is exchanged through GCMD, ISRT/PURGE, and GU/GN calls.

The IMS Queue Control Facility function routines interface with the IMS Queue Manager and Common Queue Server (CQS) routines.

By using the standard AIB interface, errors detected are recorded with an AIB return code = 000000F0, a unique AIBREASN code for each error, a TPCBSTAT code of MR, and a 6701-MRQE log record is written to the IMS online log data set (OLDS). The AIBREASN codes are printed in the reports (Browse, Query, Load, Recover, and Unload) and are documented in the DFSMRAEQ macro.

After the error is reported and logged, IMS Queue Control Facility and IMS skip to the next message, function, or terminate the BMP, depending on the error. The IMS Queue Control Facility routines in IMS do not abend. To diagnose the error, the 6701-MRQE log records should be printed and analyzed. The API calls may also be traced by IMS Queue Control Facility (Trace control card), or within IMS by issuing the /TRACE SET ON PROGRAM MRQPSB. The IMS Queue Control Facility trace sends output to the QCFPRINT DD data set. The IMS trace logs type 6701 records to the OLDS.

### Related reference

[“6701-MRQE diagnostic records” on page 302](#)

An IMS error detected while QCF is requeuing messages results in the logging of a 6701-MRQE diagnostic record.

## SCRAPLOG diagnostic records

By analyzing SCRAPLOG records, you can sometimes determine that a logical terminal (LTERM) to which messages are to be requeued does not exist. In this case, you can fix the problem and run the job again so that the messages are requeued.

As part of diagnosing problems with the Queue Control Facility/Message Requeuer, you use SCRAPLOG records. The SCRAPLOG record consists of a X'100' or X'140' MRQ prefix that is mapped by DFSMRQPF, followed by the actual message that is being inserted. The message is either a 4002 record (a message from a DUMPQ or SNAPQ checkpoint) or a 01 (input) or 03 (output) message record. IMS messages are mapped by the QLOGMSGP macro.

### Sample QCF record from scraplog data set

An example showing a message that is scrapped by QCF and written to the SCRAPLOG data set is shown.

The first X'140' byte is the QCF prefix, mapped by the DFSMRQPF macro. Offset X'88' into DFSMRQPF is the AIBREASN code = 00001084 = message is nonrecoverable (in other words, INQUIRY=NORECOV on the IMS TRANSACT macro TRAN31B0).

The rest of the data is the message (offset 04 = X'03' = type 03 output message), mapped by macro QLOGMSGP.

### QCF prefix mapped by DFSMRQPF

```
5B RECORD
QCF prefix mapped by DFSMRQPF
00000000 000000 04610000 5BD8C3C6 D4E2C700 08100102 01400000 00000000 00000000 00000000
*./..$QCFMSG.....*
00000020 000020 00000000 02000100 2001304F 22581647 4184032D E2E8E2F3 40404040 B6AB6C0E
*.....|.D..SYS3 ..%.*
00000040 000040 26E03901 E2E8E2F3 40404040 B6AB6C0E 26E03901 00000000 00000000 00000000
*...SYS3 ..%.....*
00000060 000060 0001004C 00000000 08000002 40404040 40404040 E3D9C1D5 F3F1C2F0 D3F6F2D4 *...<.....
TRAN31B0L62M*
00000080 000080 E5E2F140 00000000 00000000 81000000 0000D4D9 000000F0 00001084 000A000A
*VS1 .....A.....MR...0...D....*
000000A0 0000A0 E3D9C1D5 F3F1C2F0 40404040 40404040 00000000 00000000 D8C3C6E5 F1D9F240
*TRAN31B0 .....QCFV1R2 *
000000C0 0000C0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
000000E0 0000E0 TO 00000120 000120 SAME AS ABOVE
IMS message mapped by QLOGMSGP
00000140 000140 03110000 01D18194 08000002 08000002 02E40000 E2E8E2F3 40404040 B6AB6C0E
*.....JAM.....U..SYS3 ..%.*
00000160 000160 26E03901 E2E8E2F3 40404040 B6AB6C0E 26E03901 00000000 00000000 00000000
*...SYS3 ..%.....*
00000180 000180 00408100 C8000000 00000000 00000000 00010000 00000000 00000000 00000001 *.
A.H.....*
000001A0 0001A0 FFFFFFFF 0C027700 E3D9C1D5 F3F1C2F0 00000000 00000000 40404040 40404040
*.....TRAN31B0.....*
```

```

000001C0 0001C0 00108600 0264FC00 00000000 00000000 011E8700 00C2D588 8000D600 C9D4E2D5
*..F.....G..BNH..O.IMSN*
000001E0 0001E0 C5E34040 D3F6F2D4 E5E2F140 D3F6F2D4 C4C5F0F1 40404040 40404040 00000000 *ET L62MVS1
L62MDE01
00000200 000200 00000000 0C027700 40404040 40404040 40404040 40404040 0C505A70 00000002
*.....&.....*
00000220 000220 E3D9C1D5 F3F1C2F0 D3F6F2D4 E5E2F140 00000000 B6AB6C0E 24746405 00000000
*TRAN31B0L62MVS1.....%.....*
00000240 000240 00000000 00000000 00000000 00000000 00000000 00000000 00000008 E3D9C1D5
*.....TRAN*
00000260 000260 F3F1C2F0 50018046 15519555 55555555 55555555 55555555 55555555 55555555
*31B0&.....N.....*
00000280 000280 55555555 55555555 55555555 55555555 55555555 86A3A781 B0B7A415 55555555
*.....FTXA..U.....*
000002A0 0002A0 55555555 09151515 15151515 15151515 15151515 00000000 00000000 00000000
*.....*
000002C0 0002C0 00000000 00E2E8E2 F3404040 40000000 00000000 00000000 00000000 00000000
*.....SYS3.....*
000002E0 0002E0 00000000 00000000 00000000 00000016 88004040 40404040 40404040 40404040
*.....H.....*
000002E0 0002E0 00000000 00000000 00000000 00000016 88004040 40404040 40404040 40404040
*.....H.....*
00000300 000300 4040D600 00108900 00018000 B6AB6C0E 26E49E81 00188A00 2001304F 22581647 *
O...I.....%.U.A.....|.....*
00000320 000320 4184032D 00000000 00000000 00688B00 00000000 00000000 00000000 00000080
*.D.....*
00000340 000340 00000000 00000000 00000000 00000000 00000000 00000000 0000000A 00000000
*.....*
00000360 000360 00000000 00000000 0000000A 000A000A E2E8E2F3 40404040 B6AB6C0E 26E03901
*.....SYS3.....%.....*
00000380 000380 00000000 00000000 00000000 00000000 00000000 00908C00 0000000A 00000000
*.....*
000003A0 0003A0 00000003 E3D9C1D5 F3F1C2F0 FDFFFFFF 0C027700 0A0A014C 40080000 00000000
*...TRAN31B0.....<.....*
000003C0 0003C0 00000000 00000000 00000000 00000000 00000000 00000000 08100000 00000000
*.....*
000003E0 0003E0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
00000400 000400 00000000 00000810 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
00000420 000420 00000000 002D0300 E3D9C1D5 F3F1C2F0 40D6E4E3 C2D6E4D5 C440D4C5 E2E2C1C7 *.TRAN31B0
OUTBOUND MESSAG*
00000440 000440 C540E3D6 40E3D9C1 D5F3F1C2 F0404040 40B6AB6C 0E26E5DF 01000000 00000001 *E TO
TRAN31B0 ..%.V.....*
00000460 000460 E5
*V *

```

## Key fields of SCRAPLOG records and their offsets

Key fields of SCRAPLOG records, including offset, label, length, value and a description will help you in diagnosing problems.

The following table shows key fields of the QCF records and their offsets.

Table 65. Key fields in QCF records and their offsets				
Offset	Label	Length	Value	Description
04	MSGMRQID	08	\$QCFMSG	Prefix ID (First character is 5B, which causes DFSERA30 to print as 5B rec.)
74	MRPREDST	08	TRAN31B0	Destination name
94	MRPRETRN	04	000000F0	AIBRETRN code, always this value for QCF errors
98	MRPREASN	04	00001084	AIBREASN code = message non recoverable

The following table shows the key fields in messages.

Table 66. Key fields in messages (offset 0140=offset 00 into message)				
Offset	Label	Length	Value	Description
0140	MSGLRLL	02	0361	Length of message
0144	MSGLCODE	01	01	Log code, 01=input message, 03=output message

Table 66. Key fields in messages (offset 0140=offset 00 into message) (continued)

Offset	Label	Length	Value	Description
0150	MSGPRFLL	02		Length of total message prefix (user segments start at this offset)
01A8	MSGODSTN	08	TRAN31B0	Message destination name

## Sample JCL for printing SCRAPLOG records

Use SCRAPLOG records in combination with 6701-MRQE records to effectively diagnose QCF problems.

The following figure shows sample JCL for printing SCRAPLOG records.

### Sample JCL for printing SCRAPLOG records

```
//SCRAPPRT JOB
//* PRINT IQCSELECT SCRAPSEL
//JOB LIB DD DISP=SHR,DSN=IMS610.RESLIB
//SELECT EXEC PGM=DFSERA10,REGION=512K
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=QCF.SCRAPSEL,DISP=SHR
//SYSIN DD *
CONTROL CNTL
OPTION PRINT E=DFSERA30
END
/*
//CANCEL EXEC PGM=DFSERA10,COND=EVEN,REGION=256K
//* PRINT IQCCANCL SCRAPCAN
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=QCF.SCRAPCAN,DISP=SHR
//SYSIN DD *
CONTROL CNTL
OPTION PRINT E=DFSERA30
END
//INSERT EXEC PGM=DFSERA10,COND=EVEN,REGION=256K
//* PRINT IQCINSRT SCRAPLOG
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=MRQ.SCRAPLOG,DISP=SHR
//SYSIN DD *
CONTROL CNTL
OPTION PRINT E=DFSERA30
END
/*
```

## 6701-MRQE diagnostic records

An IMS error detected while QCF is queueing messages results in the logging of a 6701-MRQE diagnostic record.

The message that is being queued is then discarded (and written to the SCRAPLOG), and the QCF BMP (IQCINSRT) proceeds to the next message. Each type of error is accompanied by a unique reason code that is set in the application interface block reason code field (AIBREASN).

When the IQCINSRT step completes, a report of messages scrapped and grouped by reason code is produced. A report of messages scrapped and grouped by destination name is also produced.

### Related reference

[“IBM IMS Queue Control Facility for z/OS interface” on page 298](#)

You can invoke the IBM IMS Queue Control Facility for z/OS functions through either a user control card input or a TSO/ISPF interface. The functions are Browse, Query, Load/Insert, Queue Overflow Protection, Recover, and Unload.

## Sample JCL for printing the 6701-MRQE diagnostic records

The following figure shows sample JCL for printing 6701-MRQE records.

### Sample JCL for printing 6701-MRQE records

```
//LOGPRNT JOB
//JOB LIB DD DISP=SHR,DSN=IMS610.RESLIB
//IMSLOG0 EXEC PGM=DFSERA10,REGION=512K
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=IMS610.OLDSP0,DISP=SHR
//SYSIN DD *
CONTROL CNTL
OPTION PRINT 0=5,V=6701,L=2,C=M,E=DFSERA30
OPTION PRINT 0=9,V=MRQE,L=4,T=C,C=E,E=DFSERA30
END
/*
```

## Control blocks logged at time of error (and their mapping macros)

The following table shows the 6701-MRQE diagnostic record control blocks and data areas that are logged if they are available at the time of the error.

Table 67. Control blocks and data areas logged at time of error for 6701-MRQE records

Block	Description	Mapping macro
AIB	Application interface block	DFSAIB
	AIBRETRN, AIBREASN codes	DFSMRAEQ
CMDMSGP	Command call buffer for browse, load, QSN, query, recover, or unload command	DFSMRQCW (OCO)
DFSSQQR	Query buffer	CQSQRQT
I/O AREA	Input/Output area	QLOGMSGP
MRQCMDWK	Command call buffer for browse, load, QSN, query, recover, or unload command	DFSMRQCW (OCO)
MRQPREFX	QCF prefix buffer	DFSMRQPF
MRQWORK	MRQ/QCF work area	Mapping macro
MRSELROW	Include/Exclude work area	DFSMRQCT (OCO)
MRSELWK	Select work buffer	DFSMRQSW (OCO)
PCB	Program communication block	IDLI TPCBBASE=0,CALLER=IMS
PST/EOB	Partition specification table / end of block	IDLI PSTBASE=0
PSTDCA	DL/I call parameter area	No DSECT
QMBA	Queue manager buffer area	DFSQMGR FUNC=QDSECT
QSAPWKAD	Queue manager work area	QSAPWKAD
QTPDST	Queue manager destination block	ICLI CNTBASE=0, or IAPS SMBASE=0 (CNT/LNB or SMB) DSECT for QAB/TIB not provided

Table 67. Control blocks and data areas logged at time of error for 6701-MRQE records (continued)

Block	Description	Mapping macro
REG14-12	Registers 14 through 12	No DSECT
WORKMSG	Work message buffer	QLOGMSGP

## Normal errors and their AIBREASN codes

Some errors might be normal. It is important to determine the AIBREASN code, destination name, and other characteristics of the message to determine whether or not the error is normal.

For example, the following AIBREASN codes are considered normal:

### AIBREASN

#### Explanation

#### 00001080

Message destination is an LU 6.2 synchronous logical unit (LU) name and as such is considered nonrecoverable.

#### 00001084

Message destination is nonrecoverable either because the destination transaction code name was defined as NORECOV or the message was received from a LU 6.2 LU in synchronous conversation mode, which implies nonrecoverable.

#### 00001088

Message was already canceled by IMS. Most likely the cause of this is an output message that was canceled when the application program abended or issued a ROLL or ROLB call.

#### 000010A4

The message being passed by IQCINSRT is an internal IMS message that is not recoverable.

#### 00002014

The message is being purged (enqueued to a temporary destination) and the temporary destination name of the message is an inquiry type LTERM.

## Abnormal errors that can be expected

Some errors might be expected. It is important to determine the AIBREASN code, destination name, and other characteristics of the message to determine whether or not the error is expected.

For example, when a source or destination name is not found, an error might occur if the system was redefined and the resource name was deleted.

## Obtaining diagnostics in addition to SCRAPLOG and 6701-MRQE

When the 6701-MRQE diagnostic records and the SCRAPLOG records do not provide enough diagnostic detail to adequately diagnose a problem, you can obtain additional diagnostic details by issuing the **/TRACE SET ON PROGRAM** command. **/TRACE SET ON** *pgmname* causes the logging of additional 6701-MRQB records when the QCF BMP is processing.

### About this task

When the 6701-MRQE diagnostic records and the SCRAPLOG records do not provide enough diagnostic detail to adequately diagnose a problem, you can obtain additional diagnostic details by issuing the following command:

```
/TRACE SET ON PROGRAM pgmname
```

where *pgmname* is the name of the appropriate MRQPSB.

6701-MRQB diagnostic records are almost identical to 6701-MRQE records, with the exception of MRQB appearing where MRQE normally does. The *pgmname* value is the default QCF PSBNAME. This value

might have been overridden on the MSGQUEUE MRQPSBN parameter at system definition. To determine if your installation has overridden the name, either consult with your IMS system administrator or issue the IMS command **/DISPLAY PROGRAM MRQPSB**.

If PROGRAM MRQPSB displays as an invalid name, your installation has overridden the default MRQPSB. Consult with your system administrator for the correct name for your installation.

The records that are contained in this program are in addition to the existing program trace records logged by DFSDLA30. Records logged by DFSDLA30 are types 6701-LA3A and 6701-LA3B, which contain the TPCB, I/O AREA (64 bytes), and PST control blocks.

With the program trace set on, for each ISRT call to insert a message (or segment of a message), there is an LA3A, MRQB, and LA3B record. For each PURG call (which completes and enqueues a message) there is one LA3A and LA3B log record. If an error is detected while processing either call, an additional MRQE record is logged. The MRQE records are logged regardless of whether the program trace is on when an error is detected.

### Related tasks

[“IMS transaction trace” on page 317](#)

The IMS transaction trace writes entries to the IMS log at entry to and exit from the DC call analyzer (DFSDLA30).

## Determining when messages are successfully queued

Messages that are successfully queued by the Queue Control Facility/Message Requirer are logged to the OLDS with an identical 01 (input) or 03 (output) log record as the original, except when MSGCFLG3=MSGC3MRQ (that is, message+19=45) is set to indicate that this message was queued by the Queue Control Facility/Message Requirer.

This flag is propagated to other messages that originate from this message. (That is, if the message is an input transaction message the flag is propagated to the output response messages when the transaction message is processed. Or, if the message is an MSC message, it is propagated to messages in other IMS/MSB systems when the message is sent across the MSC link.)

The following figure shows an input transaction to TRANCODE=TRAN31V0 from LTERM=IMSUS02 that was queued by QCF.

### Sample log record showing successfully queued message

```
01
RECORD
00000000 000000 01EE0000 01C18110 08000055 08000055 01CE1000 E2E8E2F3 40404040 B6AB6CBC
*.....AA.....SYS3 ..%.*
00000020 000020 C4E84B83 C9D4E2F1 40404040 B7BD992F E30E2241 80000100 00000000 00000000
*DY.CIMS1 ..R.T.....*
00000040 000040 00408100 C8400000 C4E3E2D3 E4F2F0F2 00020000 00000000 00000000 00000001 *.
A.H ..DTSLU202.....*
00000060 000060 C9D4E2E4 E2F0F240 E3D9C1D5 F3F1E5F0 00000000 00000000 C4C6E2D4 D6F24040 *IMSUS02
TRAN31V0.....DFSM02 *
00000080 000080 00108600 014E7C00 00000000 00000000 00168800 C9D4E2E4 E2F0F240 40404040
*.F..+@.....H.IMSUS02 *
000000A0 0000A0 40404040 E4000018 89000000 0000B6AB 6CBCC4EA CD030000 00000000 00000018 *
U...I.....%D.....*
000000C0 0000C0 8A002001 304F2301 19573676 032D0000 80000000 00000068 8B000000 00000000
*.....|.....*
000000E0 0000E0 00000000 00000000 00800000 00000000 00000000 00000000 00000000 00000000
*.....*
00000100 000100 00000000 000A0000 00000000 00000000 00000000 000A0024 000AE2E8 E2F34040
*.....SYS3 *
00000120 000120 4040B6AB 6CBCC4E8 4B830000 00000000 00000000 00000000 00000000 00000090
*...%.DY.C.....*
00000140 000140 8C000000 000A0000 00000000 0003E3D9 C1D5F3F1 E5F0C9D4 E2E4E2F0 F240240A
*.....TRAN31V0IMSUS02 ..*
00000160 000160 014C0008 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.<.....*
00000180 000180 00000810 00000000 00000000 00000000 00000000 01E3D9C1 D5F3F1E5 F0404040
*.....TRAN31V0 *
000001A0 0001A0 40404040 00000000 00000000 00000000 08100000 00000000 00000000 00000000
*.....*
000001C0 0001C0 00000000 00000000 00000000 00000010 0301E3D9 C1D5F3F1 E5F040C8 C94BB7BD
*.....TRAN31V0 HI...*
```

## Diagnosing message routing problems

User exits are consolidated into user exit DFSMSCE0. Several traces, messages, and information fields in the message prefix area can be used to diagnose message routing problems in the user exits and in IMS.

### DFS070 UNABLE TO ROUTE MESSAGE RSN=xyxy

Message DFS070 is issued when any one of three conditions occur.

- IMS attempts to enqueue a message.
- These TM/MSC exits attempt to reroute a message:
  - DFSMSCE0–Message Routing.
  - DFSMSTRO–Terminal Routing.
- A **/FORMAT** command is entered and an error is encountered while routing a message.

### DFS070 diagnostic message

Diagnostic messages from DFS070 are described.

Here is an example of the DFS070 diagnostic message:

```
DFS070 UNABLE TO ROUTE MESSAGE RSN=0104
```

The RSN code identifies the module that issued the message (01 = DFSICIO0) and the reason for the error (04 = Prefix buffer length is too large).

In this case DFSICIO0 called the message generator (DFSCLMR0) with R1 = 00680046.

```
Where x'00680046' = module identifier, reason code,message key
                  x'0068' = 0104 (decimal)
                    01 = Module that issued message = DFSICIO0
                    04 = Prefix buffer length is too large

                  x'0046' = 70 (decimal) = DFS070 MESSAGE KEY
```

The following table shows:

- The label used for the module identifier
- The module identifier
- The module function or name

The labels shown in the following table can be used to scan the module source code to locate where the message was issued from.

Table 68. DFS070 module identifier table

Label	Default module identifier	Function (module name)
MSUK	00	Unknown module or DFSMSCEC requestor
MSTR	01	DC Communication Manager (DFSICIO0)
MSTRAP	02	LU 6.2 Receive LU Manager (DFSRLM10)
MSTROT	03	OTMA Receive LU Manager (DFSYTIB0)
MSPR	04	DC Call Handler (DFSDLA30)
MSLR	05	MSC Analyzer (DFSCMS00)



Table 68. DFS070 module identifier table (continued)

Label	Default module identifier	Function (module name)
MSFM	06	<b>/FORMAT</b> Command Processor (DFSICLK0)
MSTE	08	IMS Termination (DFSTRM00)
MSINIT	10	IMS Initialization (DFSIINB0)

The following table shows:

- The label used for the reason code
- The reason code value
- The description of the error

The labels shown in the following table can be used to scan the module source code to locate where the message was issued from.

Table 69. DFS070 reason (RSN) codes table

Label	Reason code decimal/hexadecimal	Description
PFXUPRER	02/02	User requested 2 user prefix segments (code 8E).  Programmer response: The routine that was setting up to call the DFSMSCE0 user exit determined that a user prefix segment had already been obtained. The programmer may need to turn on the DFSMSCE0 trace to determine which routine is setting the field, MSCEUPR (DFSMSCEP) or the flag MSCEB2RET (DFSMSCEB).
PFXIPRER	03/03	User requested two Workload router prefix segments (code 8F).  Programmer response: The routine that was setting up to call the DFSMSCE0 user exit determined that a user prefix segment had already been obtained. The programmer may need to turn on the DFSMSCE0 trace to determine which routine is setting the field MSCEUPR (DFSMSCEP) or the flag MSCEB2RET (DFSMSCEB).
PFTOOBIG	04/04	Prefix buffer length is too large.  Programmer response: The user prefix segment size field MSCEUPRL (DFSMSCEP) or the workload router prefix segment size field MSCEIPRL (DFSMSCEP) is greater than 512. The programmer may need to turn on the DFSMSCE0 trace to determine which routine is setting the field MSCEUPR or MSCEIPR (DFSMSCEP) to a value larger than 512.
GBPFER	05/05	DFSPPOOL error on get prefix buffer.  Programmer response: Failure to get storage for the user prefix segment or the workload router prefix segment through the DFSPPOOL macro from the HIOP pool.
URCERR1	06/06	User exit return code negative.  Programmer response: User exit DFSMSCE0 returned a negative return code.

Table 69. DFS070 reason (RSN) codes table (continued)

Label	Reason code decimal/hexadecimal	Description
URCERR2	07/07	DFSBCB error getting BCB block.  Programmer response: User exit DFSMSCEO returned a negative return code returned an invalid return code.
GMSBERR	08/08	DFSBCB error getting BCB block.  Programmer response: Failure to get storage for the MSEB block through the DFSBCB macro.
LRBADSID	09/09	Bad SYSID detected.  Programmer response: In getting the address for the LNB that is associated with either the origin SID or the SID that is specified by the caller, a bad SYSID was detected.
IPFX	10/0A	Queue Manager insert prefix error.  Programmer response: In an effort to update the MESSAGE PREFIX (01/03) log record, a prefix update call was made (DFSQMGR0) to add the user prefix segment or the workload router segment, or both. The prefix update routine was unable to add the segment.
ICLR1ERR	11/0B	Non zero return code from DFSICLR1 (DFSICLR0).
AVMLKERR	12/0C	Destination is an invalid type for AVM/ISC link.
MSCEFL1E	15/0F	DFSMSCEC user exit routing flag is in error.  Programmer response: An invalid option was requested for the user routing exit flag 1 (MSTRFL1/MSLRFL1/MSPRFL1). Refer to the DFSMSCEP macro for valid options. Check the user exit parameter in the 6701-MSCE record to determine which option was requested. These options are usually set by IMS code.
USRXIFER	16/10	DFSUSRX interface error.  Programmer response: The macro DFSMSCEC invoking DFSUSRX0 through the DFSUSRX macro received a non-zero return code. The value is in field MSCEBRC in the DFSMSCEB block. Possible values returned are:  1. 04 the user exit routine specified has not been defined (the address in UXDT is zero)  2. Unable to get an interface block using the DFSBCB macro. DFSBCB return code is in field, MSCEBSSRC in the DFSMSCEB block.
IONAMCHG	18/12	User exit changed the destination name of the I/O PCB message.  Programmer response: The user exit (DFSMSCEO) set flag MSPR2CHG in field MSPRFL2 to request that the destination name MSPRDEST be changed. The PCB is the I/O PCB that cannot be changed. Check the user exit parameter in the 6701-MSCE record to determine which option was requested.

Table 69. DFS070 reason (RSN) codes table (continued)

Label	Reason code decimal/hexadecimal	Description
IOROUTE	19/13	<p>User exit requested reroute I/O PCB message.</p> <p>Programmer response: The user exit DFSMSCEO requested a routing option of MSPR2RMT, /MSPR2LSQ, /MSPR2SRC, /MSPR2NDR in field MSPRFL2. This is invalid if the PCB is the I/O PCB.</p> <p>Refer to the user exit parameter in the 6701-MSCE record to determine which command was requested.</p>
CMDINV	20/14	<p>User exit changed the destination name to a command (such as: / <b>CMDVERB</b>).</p> <p>Programmer response: The user exit DFSMSCEO changed the destination name to a command.</p> <p>Refer to the user exit parameter in the 6701-MSCE record to determine which command was requested.</p>
SQGINV	21/15	<p>User Link receive exit override MSNAME in segment because destination is not an MSNAME.</p> <p>Programmer response: User exit DFSMSCEO in a shared queues group link receive exit failed due to the destination not being an MSNAME.</p>
REGFAIL	22/16	<p>Local shared queue registration (DFSSQIF FUNC=INFRM) failed for the transaction when the user exit requested MSLR2LSQ=1 or MSTR2LSQ=1.</p>
NOTRANCD	23/17	<p>Terminal routing exit routed the message to a remote IMS (MSTR2RMT=1) but the destination type at MSTRDEST is an unsupported TRANCODE (such as remote routing is not allowed for LTERM or FAST PATH exclusive TRANCODE).</p>
DSIDINV	24/18	<p>The Terminal, Link Receive or the Program Routing exit returned an invalid destination SYSID (for example, either field MSTRDSID, MSLRDSID, or MSPRDSID is invalid).</p>
DMSNINV	25/19	<p>The Terminal, Link Receive, or Program routing exit returned an invalid destination MSNAME (for example: either field, MSTRDMSN, MSLRDMSN, or MSPRDMSN is invalid).</p>
SSIDINV	26/1A	<p>The Link Receive exit rerouted an intermediate message (MSLR1INT=1) to this local IMS by setting MSLR2LOC=1, but the message had an invalid return (source) SYSID so this IMS could not accept it locally.</p>
RMT2INV	27/1B	<p>The Terminal, Link Receive, or Program routing exit indicated routing the message to a remote MSC link by setting MSTR2RMT, MSLR2RMT, or MSPR2RMT; however, the exit did not set either of the corresponding destination SYSID or MSNAME fields (for example, either MSTRDSID, MSLRDSID, or MSPRDSID was left set to zero, or MSTRDMSN, MSLRDMSN, or MSPRDMSN was left set to blanks).</p>

Table 69. DFS070 reason (RSN) codes table (continued)

Label	Reason code decimal/hexadecimal	Description
SRC2INV	28/1C	<p>The Program routing exit requested the message be routed to the source MSC system by setting MSPR2SRC=1 however the message cannot be routed because either:</p> <ul style="list-style-type: none"> <li>• MSC is not available.</li> <li>• Or the source SYSID is not valid because the application program has not issued a get unique (GU).</li> <li>• The application program is a non-message driven BMP.</li> </ul>
NDR2INV	29/1D	<p>The Program Routing exit requested a direct routing message be overridden by setting MSPR2NDR=1; however, either:</p> <ul style="list-style-type: none"> <li>• MSC is not available.</li> <li>• This is not a direct routed message with a MSNAME destination.</li> <li>• The overriding name in the front of the I/O area is not valid.</li> </ul>
RMT2FSR	30/1E	<p>The Terminal routing exit indicated to route the message to a remote MSC link by setting MSTR2RMT=1, but the input ISC node was set to process the message as a Front End Switch message by the user Front End Switch exit (DFSFEBJ0). Front End Switch messages cannot be routed to MSC links.</p>
RSPROUTE	31/1F	<p>The Link receive exit requested that a response message (MSLR1RSP=1) be rerouted by either setting one of the MSLRFL2 reroute flags. Response messages may not be rerouted.</p>
INBCHGID	33/21	<p>CHANGEID not supported.</p> <p>Programmer response: The user exit (DFSMSCE0) did not use the DFSMSCSV macro or generate module entry code. IMS initialization expects a branch instruction around the character information of entry code.</p> <p>Refer to the sample version of the provided user exit DFSMSCE0's use of DFSMSCSV for more information.</p>
INBIDLNG	35/23	<p>Character string 'VECTOR' not present.</p> <p>Programmer response: The user exit (DFSMSCE0) did not use the DFSMSCSV macro or generate module entry code. IMS initialization expects the entry code to contain a length of the module entry code at a given offset.</p> <p>Refer to the sample version of the provided user exit DFSMSCE0's use of DFSMSCSV for more information.</p>
INBNVECT	35/23	<p>Character string 'VECTOR' not present.</p> <p>Programmer response: The user exit DFSMSCE0 did not use the DFSMSCSV macro or module entry code to provide the character string "VECTOR" in its entry code.</p> <p>Refer to the sample version of the user exit DFSMSCE0's use of DFSMSCSV for more information.</p>

Table 69. DFS070 reason (RSN) codes table (continued)

Label	Reason code decimal/hexadecimal	Description
PFXUINVA	36/24	<p>Upon return from the user exit IMS detected that the user prefix at MSCEUPR is invalid.</p> <p>Possible causes are:</p> <ul style="list-style-type: none"> <li>• Length not in range of 5 to 512 bytes.</li> <li>• Address of prefix is invalid. Must be address obtained by IMS or within HIOP pool.</li> <li>• Length has been changed (MSCEBUPRL).</li> <li>• Address of user exit prefix has changed (MSCEBUPR).</li> <li>• Prefix code not 8E.</li> </ul> <p>The programmer may need to turn on the DFSMSCE0 trace to trace the fields MSCEBUPR and MSCEBUPRL within the DFSMSCEB block.</p>
PFXIINVA	37/25	<p>Upon return from the user exit, IMS detected the Workload Router prefix at MSCEIPR is invalid.</p> <p>Programmer response:</p> <ul style="list-style-type: none"> <li>• Length not in range of 5 to 512 bytes.</li> <li>• Address of prefix is invalid. Must be address obtained by IMS or within HIOP pool.</li> <li>• Length has been changed (MSCEBIPRL).</li> <li>• Address of workload router prefix has changed (MSCEBIPR).</li> <li>• Prefix code is not 8F.</li> </ul> <p>The programmer may need to turn on the DFSMSCE0 trace to trace the fields MSCEBIPR and MSCEBIPRL within the DFSMSCEB block.</p>
EXIOVLAY	38/26	<p>User exit overlaid the 512 byte user work area buffer.</p> <p>Programmer response: The user exit DFSMSCE0 appears to have overlaid the 512 byte workarea.</p> <p>The overlay character string SCDSMCON is inserted at the end of the 512 byte workarea MSEBIBOV before calling the user exit DFSMSCE0 and is checked on return.</p> <p>Refer to the user exit DFSMSCEB in the 6701-MSCE record to help determine the overlay.</p>
EXBOVLAY	39/27	<p>User exit overlaid the MSEB BCB block name (Overlay Check).</p> <p>Programmer response: The user exit (DFSMSCE0) appears to have overlaid the DFSMSCEB block. The DFSBCB system service inserts a character string (MSEB) at the end of the DFSMSCEB block. IMS will abend when the DFSMSCEB block is returned by way of a DFSBCB release request. The DFS070 message will assist in determining when the overlay occurred.</p> <p>Refer to the user exit parameter in the 6701-MSCE record to help determine the overlay.</p>

Table 69. DFS070 reason (RSN) codes table (continued)

Label	Reason code decimal/hexadecimal	Description
EXPOVLAY	40/28	<p>User exit overlaid the parameter list (Overlay Check).</p> <p>Programmer response: The user exit DFSMSCEO appears to have overlaid the user exit parameter list (DFSMSCEP). The overlay character string SCDSMCON is inserted at the end of the parameter list DFSMSCEP before calling the user exit DFSMSCEO and is checked on return.</p> <p>Refer to the user exit parameter in the 6701-MSCE record to help determine the overlay.</p>

Codes 41 thru 52, shown in the following table, apply to the **/FORMAT** command.

Table 70. DFS070 reason (RSN) codes table for the /FORMAT command

Label	Reason code decimal/hexadecimal	Description
FMFND	41/29	The CNT for the terminal to be formatted was not found.
FMRCNT	42/2A	The specified terminal is a remote LTERM.
FMDLNB	43/2B	The specified terminal is a dynamic MSNAME (LNB).
FMMFST	44/2C	The destination terminal (different from the input terminal) is not MFS-formatted.
FMLRESMD	45/2D	The destination terminal is in line response mode.
FMTRESMD	46/2E	The destination terminal is in terminal response mode.
FMCONV	47/2F	Conversation is active on the destination terminal (when LTERM was specified in the command).
FMINP	48/30	The terminal is in input mode only.
FMEXCL	49/31	The terminal was in exclusive mode (when LTERM was specified in the command).
FMQBUF	50/32	The call to Queue Manager failed for a <b>PUT LOCATE</b> call.
FMIPREF	51/33	The <b>INSERT PREFIX</b> call to Queue Manager failed.
FMMSGNR	52/34	The call to enqueue the message failed.

## DFSMSCEO TM/MS Message Routing exit trace

The DFSMSCEO TM/MS Message Routing exit trace can be activated individually for each exit entry point that processes a message routing request. This trace is useful for diagnosing problems in both the user exit and in IMS.

The DFSMSCEO TM/MS Message Routing Exit trace writes a 6701-MSEA log record when the exit is entered, and a 6701-MSEB log record when the exit returns to IMS to process the reroute request. The following information is traced:

- Exit parameter area, DFSMSCEP
- 512 byte work area
- Message

- Message prefix
- Message segment being inserted
- Other work area storage

## Displaying DFSMSCE0 trace status

Use the **/DISPLAY TRACE EXIT** command to display the DFSMSCE0 trace status.

### About this task

To display the DFSMSCE0 trace status, issue the following command:

```
/DISPLAY TRACE EXIT
```

The display will show ON, OFF, or N/A for each DFSMSCE0 trace entry point.

## Starting and stopping the DFSMSCE0 trace

To start or stop the DFSMSCE0 trace, issue one of the **/TRACE** commands.

### About this task

```
/TRACE SET (ON|OFF) EXIT (DFSMSCE0) (ALL|TRBT|TRVT|TR62|
TROT|LRTR|LRLT|LRIN|
LRDI|PRCH|PRIS)
```

Any combination of TRBT, TRVT, TR62, TROT, LRTR, LRLT, LRIN, LRDI, PRCH, and PRIS is valid.

## DFS081 trace exit command unsuccessful RSN=xyyy message

Message DFS081 is issued for a variety of reasons. The module identifier and function name are listed as well as the reason codes and descriptions.

This message is issued when one or more of the following scenarios occurs:

- IMS attempts to enqueue a message.
- The following user exits attempt to reroute a message:
  - The TM/MSD message routing exit, DFSMSCE0.
  - The Terminal Routing exit, DFSMSTRO.
- A **/FORMAT** command was entered.
- An error was encountered while routing the message.

*Table 71. DFS081 module identifier table*

Label	Module identifier (decimal)	Function (module name)
ICLN	01	Trace Command Processor (DFSICLN5)

The following table shows:

- The label used for the reason code
- The reason code value
- The description of the error

The labels shown in the following table can be used to scan the module source code to locate where the message was issued from.

Table 72. DFS081 reason (RSN) codes table

Label	Reason code decimal/hexadecimal	Description
EXTIKW	01/01	Invalid keyword for trace exit
EXTIPT	02/02	Invalid parameter type for trace exit command.
EXTNPT	03/03	No parameter type was specified for trace exit command.
EXTMPT	04/04	Multiple parameter types for trace exit command.
EXTMCB	05/05	Missing DFSMSCB control block for the trace exit DFSMSCE0 command.
EXTIPS	06/06	Invalid parameter subtype for the trace exit command.
EXTENS	07/07	Trace exit is not supported for this environment.
EXTENL	09/09	Required exit is not loaded for start trace command.
EXTSCF	10/0A	System command failure.
EXTIPL	11/0B	Invalid parameter length.

### DFS070 diagnostic message

This is an example of the DFS070 diagnostic message.

```
DFS070 UNABLE TO ROUTE MESSAGE RSN=0104
```

The RSN code identifies the module that issued the message (01 = DFSICIO0) and the reason for the error (04 = Prefix buffer length is too large).

In this case DFSICIO0 called the message generator (DFSCLMR0) with R1 = 00680046.

```
Where x'00680046' = module identifier, reason code,message key
      x'0068' = 0104 (decimal)
              01 = Module that issued message = DFSICIO0
              04 = Prefix buffer length is too large
      x'0046' = 70 (decimal) = DFS070 MESSAGE KEY
```

The following table shows:

- The label used for the module identifier
- The identifier
- The module function or name

The labels shown in the following table can be used to scan the module source code to locate where the message was issued from.

### Contents of the DFSMSCE0 trace records

DFSMSCE0 records are type X'6701' with a trace ID of MSEA (entry) or MSEB (exit).

Refer to the DFSMSCEB macro for contents of the MSCEB block.

#### PROGRAM ROUTING

- MSCEB (Message routing exit interface block)
  - (CHNG/ISRT call)
- PCB (CHNG/ISRT call)
- MESSAGE PREFIX (CHNG/ISRT call)



- MESSAGE SEGMENT (ISRT call) maximum of 256 bytes

#### LINK RECEIVE

- MSCEB (Message routing exit interface block)
- MESSAGE PREFIX

#### TERMINAL ROUTING

- MSCEB (Message routing exit interface block)
- MESSAGE SEGMENT maximum of 256 bytes

**Important:** To assist in diagnosing DFSMSCE0 exit problems, the MSCEB block will maintain the following information:

- 8-byte eye catcher 'DFSMSCEB'
- 4-byte Routing exit type:

```
TRTB|TRVT|TR62|TROT|LRTR|LRLT|LRIN|LRDI|PRCH|PRIS
```

- 4 byte Address of ECB
- 4 byte Address of interface block
- 4 byte Address of DFSMSCE0 exit parameter list

## Diagnosing routing errors by using the transaction trace or program trace

The transaction trace or program trace can be used to diagnose routing error problems that are related to the user program routing exit DFSMSCE0.

By setting this trace on for a transaction or program, IMS logs a 6701-LA3A record at entry to DFSDLA30, and a 6701-LA3B when DFSDLA30 returns to the application program. In addition IMS logs a 6701-MSEA record when the exit is entered, and a 6701-MSEB when the exit returns to IMS. IMS also logs a 6701-MSCE error record, for each DFSMSCE0 related routing error.

Module DFSDLA30 receives control for every user application program call to a TPPCB (such as an I/O TPPCB or an alternate TPPCB). The DFSMSCE0 routing exit can be tailored to receive control for the first ISRT call of each new message to an I/O TPPCB or alternate TPPCB, or for each CHNG call to a alternate modifiable TPPCB.

For example, if the transaction trace is active for TRANA, and a TRANA message is processed and the user application program issues a ISRT to an alternate TPPCB, and the DFSMSCE0 exit is being used to route ISRT calls, IMS traces the following records with this command:

```
/TRACE SET ON TRANSACTION transaction_name
```

```
6701-LA3A - DFSDLA30 called to process ISRT call
6701-MSEA - DFSMSCE0 called to process ISRT route
6701-MSEB - DFSMSCE0 returns
6701-MSCE - Logged if routing error detected, even if tran/prog trace
            is not active
6701-LA3B - DFSDLA30 returns (ISRT/route processed)
```

To trace the DL/I portion of data communication for a specific program, issue the following command:

```
/TRACE SET ON PROGRAM program_name
```

For program routing exit (DFSMSCE0) call errors, TPPCB status, AIBRETRN, and AIBREASN codes are set.

### Related tasks

[“IMS transaction trace” on page 317](#)

The IMS transaction trace writes entries to the IMS log at entry to and exit from the DC call analyzer (DFSDLA30).

## TPCB STATUS, AIBRETRN, and AIBREASN codes for DFSDLA30 routing errors

The TPCB STATUS, AIBRETRN, and AIBREASN codes for DFSDLA30 routing errors are listed and described.

TPCB STATUS, AIBRETRN, and AIBREASN codes for DFSDLA30 routing errors are given in the following format:

TPCBSTAT	AIBRETRN	AIBREASN	COMMENTS
A1	00000104	MSERQINV(0560)	EXIT ROUTE REQUEST INVALID (DFSMSCE0)

The complete list of return and reason codes is available in the DFSAIBEQ macro.

## Diagnosing routing problems by using the DC LINE/NODE/LINK TRACE

The DC trace traces line, node, and MSC link activity. It can be used in conjunction with the DFSMSCE0 exit trace to diagnose message routing problems in the terminal routing, input message routing, and link receive exits.

These traces log 6701 log records with a variety of trace IDs (such as: 6701-A01). If any of these traces is active, IMS logs a 6701-MSEA record when the message routing exit is called and a 6701-MSEB log record when the exit returns. For example, if the node trace is active, the following trace records are logged:

```

6701-A01 - DC analyzer (DFSIC100) is called to process the message
          LINK the DFSMSCE0 trace will log X'6701' records with a
          trace ID of MSEA (entry) or MSEB (exit) for terminal
          routing or link receive. Refer to DFSMSCEB macro for
          the contents of the MSCEB block.

6701-MSEA - DFSMSCE0 called to process the message

6701-MSEB - DFSMSCE0 returns

6701-MSCE - Logged if routing error detected, even if the line, node,
          or link trace is not active

6701-A03 - DC Analyzer determines what to do next

```

## Using 01/03 log record trace

The 01/03 log record trace reflects the user exit routines called and the user options requested by the varies user exits.

A double word trace to reflect the user routing request is included in the Transaction Management Router Segment of the 01/03 log records. The trace reflects:

```

BYTE 1 - user parameter list (DFSMSCEP) flag 1
          indicates the user routing exits called.

BYTE 2-3 - User Terminal Routing flags 2 and 3
          (DFSMSCEP MSTRFL2 and MSTRFL3) indicates
          the user Terminal Routing options.

BYTE 4-5 - User Link Receive Routing flags 2 and 3
          (DFSMSCEP MSLRFL2 and MSLRFL3) indicates
          the user LINK Routing options.

BYTE 6-7 - User Program Routing flags 2 and 3
          (DFSMSCEP MSPRFL2 and MSPRFL3) indicates
          the user Program Routing options.

BYTE 8 - Currently unused

```

## IMS transaction trace

---

The IMS transaction trace writes entries to the IMS log at entry to and exit from the DC call analyzer (DFSCLA30).

### About this task

#### Starting the trace

To start the trace, issue one of the two following /TRACE commands.

To trace the DL/I portion of data communication for a specific transaction:

```
/TRACE SET ON TRAN transaction name
```

To trace the DL/I portion of data communication for a specific program:

```
/TRACE SET ON PROGRAM program name
```

#### Content of the trace records

DFSCLA30 records are type X'6701' with a trace ID of LA3A (entry) or LA3B (exit). They contain:

- PCB
- Maximum of 64 bytes of the I/O area
- MODNAME
- PST
- SMB of the transaction (if the program in the IMS control region is an MPP or a message driven BMP)

The PCB and PST areas are always logged. The I/O area, MODNAME, and SMB are additional areas that are logged when available and applicable to the call type:

- The I/O area can be logged only on entry or exit. For example, a GN call logs the I/O area on exit, while an ISRT call logs the I/O area on entry. Depending on the call type, the I/O area can be logged on both entry and exit.
- The MODNAME is logged only on an entry trace.
- The SMB is logged on both the entry and exit traces.

Field PSTSYNFC in the PST contains the following calls:

**04**

ABTERM IN PROGRESS

**08**

SYNC POINT PHASE 1

**0C**

SYNC POINT PHASE 2

**10**

PURGE TP PCBS

**14**

PHASE 1 SYNC POINT ENQ OUTPUT TO TEMP DEST

**18**

ROLB CALL

**1C**

INVALID ABENDU0820

**20**

ABORT

Field PSTFUNCT in the PST contains the following calls:

**01**  
 GU  
**03**  
 GN  
**41**  
 ISRT  
**50**  
 SETO  
**67**  
 INQY  
**83**  
 CHNG  
**85**  
 CHKP  
**87**  
 CMD  
**88**  
 GCMD  
**89**  
 ROLB  
**8A**  
 ROLS  
**8C**  
 SETS  
**8F**  
 AUTH  
**90**  
 PURG

The following figure is an example of a IMS transaction trace.

### Example of IMS transaction trace records

INTERNAL TRACE RECORD		ID = LA3A	SEGN0=00	RECNO = 0000009A	TIME 07.45.06.42	DATE 93.014
PCB						
0271B084	000000	00300038	00010018	40404040	40404040	006DD054 00000000 00009F58 C9D6D7C3
*.....IOPC*						
0271B0A4	000020	C2404040	00000000	00000000	00000000	0271B084 E6E3D6D9 40404040
*B.....DWTOR*						
0271B0C4	000040	12004040	0093014F	0745063F	00000006	40404040 40404040 40404040
*...L...*						
0271B0E4	000060	40404040	40404040			
*.....*						
I/O AREA						
02825000	000000	00340000	C3E4E2E3	D6D4C5D9	40D9C5D8	E4C5E2E3 E240C9D5 C6D6D9D4 C1E3C9D6 *...CUSTOMER REQUESTS
INFORMATIO*						
02825020	000020	D540D6D5	40D7C1F2	F860F1F6	F140D4D6	C4C5D3E2 00000000 00000000 *N ON PA28-161
MODELS.....*						
MODNAME						
82825850	000000	D4D6C4F4	F0F0F4F2			
*MOD40042*						
SMB						
027CA754	000000	00000000	00000000	00000000	00000000	00810075 00020002 D7C1D9E3
*.....A.....PART*						
027CA774	000020	40404040	41416000	0700A704	FFFFFFF	00000002 FFFFFFFF 00001D1D 027D5410
*.....X.....*						
027CA794	000040	00000000	0100FFFF	0000FFFF	00000000	027CA7C8 00000000 C4C6E2E2 C1D4F0F2
*.....@XH....DFSSAM02*						
027CA7B4	000060	40404040	40404040			
*.....*						
PST						
0271B060	000000	00000000	82801A39	02978C04	02CB51DC	00000000 00000000 00000000 00000000
*...B...P.....*						
0271B080	000020	02819040	00300038	00010018	40404040	40404040 006DD054 00000000 00009F58
*...A.....*						
0271B0A0	000040	C9D6D7C3	C2404040	00000000	00000000	00000000 00000000 0271B084 E6E3D6D9

```

*IOPCB .....DWTOR*
0271B0C0 000060 40404040 12004040 0093014F 0745063F 00000006 40404040 40404040 40404040
*.....L.I.....*
0271B0E0 000080 40404040 40404040 40404040 00000000 00000000 00000000 02C5A758 00000000
*.....EX.....*
0271B100 0000A0 00000000 00000000 04000002 02CB5148 00000000 027CA754 0094000E 01420080
*.....@X..M.....*
0271B120 0000C0 02CB5138 04000002 00000000 00000001 00000001 00000000 02825840 0280E610
*.....B...W.*
0271B140 0000E0 006D3D08 00000000 00000000 02825000 0275DC40 00000000 00000080 00000000
*.....B&;.....*
0271B160 000100 00000000 028BB020 01020304 00000000 00000000 00000000 D4D7D740 40404040
*.....MPP.....*
0271B180 000120 D4D7D740 40404040 00000000 00000000 00000000 00000000 00000000 00000000
*MPP.....*
0271B1A0 000140 00000000 00000000 00000000 00000000 00000001 00000000 00000001 00000000
*.....*
0271B1C0 000160 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0271B1E0 000180 00000000 00000000 00000000 00000000 00000000 00000000 00000000 028258A8
*.....B..Y*
0271B200 0001A0 00000000 00000000 C5C0FFFF 8A000000 C9E2D9E3 4140BA07 0271B084 00000000
*.....E.....ISRT.....D.....*
0271B220 0001C0 00000002 00C53D20 00000000 00000000 00000000 00000000 00000000 00000000
*.....E.....*
0271B240 0001E0 00000000 00000000 000000A0 02000000 00000000 00000000 00000000 00000000
*.....*
0271B260 000200 00000000 00000000 8299C762 00000000 00000000 00000000 00000000 00000000
*.....BRG.....*
0271B280 000220 00000000 00000000 00000000 0290B210 00000000 00000000 00000000 00000000
*.....*
0271B2A0 000240 00000000 00000000 00000000 00000000 00000000 00000002 000E0300 02825000
*.....B&;*
0271B2C0 000260 02707540 00000000 027573A4 00000000 00000000 00000000 00000000 00000000
*.....U.....*
0271B2E0 000280 C9E2D9E3
*ISRT
INTERNAL TRACE RECORD ID = LA3A SEGN0=01 RECNO = 0000009B TIME 07.45.06.42 DATE 93.014
CONTINUE
0271B2E4 000284 00000000 00000000 00000000 0271B084 00000000 00000000 00000000 00000000
*.....D.....*
0271B304 0002A4 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0271B324 0002C4 00000000 00000000 00000000 00000004 00F76180 00000000 00000840 00000000
*.....7/.....*
0271B344 0002E4 00011C00 0271B3D8 10000000 00000000 00000000 00008000 C0808000 24008000
*.....Q.....*
0271B364 000304 00000000 00000000 00000000 0275DC54 00000000 00000000 00000000 00000000
*.....*
0271B384 000324 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0271B3A4 000344 SAME AS ABOVE
0271B3C4 000364 00000000 00000000 00000000 00000BC8 08000000 0271B060 00000000 00000000
*.....H.....*
0271B3E4 000384 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0271B404 0003A4 00000000 00000000 00000000 00000000 00000000 028225A8 00000000 00000000
*.....B..Y.....*
0271B424 0003C4 00000000 00000000 026DE040 00004B00 000E15E6 00196FF2 00000000 00000000
*.....W..?2.....*
0271B444 0003E4 00000000 00000000 00000000 00000000 00000000 028BB000 00000000
*.....*
0271B464 000404 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0271B484 000424 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0275D83B
*.....Q.....*
0271B4A4 000444 0275D73C 00000000 07004040 40404040 00000000 00000000 00000000 0275DA3C
*.....P.....*
0271B4C4 000464 00000000 0275D83C 00000000 00000000 00000000 00000000 00000000 00000000
*.....Q.....*
0271B4E4 000484 00000000 028FCA40 02759040 00000000 00C16190 00000000 0271BC28 00000000
*.....A/.....*
0271B504 0004A4 00000000 00000000 00000000 0280E450 0280E714 A6E4A497 78F98705 00F741B0
*.....U&;X.WUUP.9G..7..*
0271B524 0004C4 026EB048 006DD000 00340000 00000000 00800000 00000000 00000000 02CD2469
*.....>.....*
0271B544 0004E4 AD2CD246 0275DD0C 00000000 00000001 00000000 00000000 0275DD38 0271BD18
*.....K.....*
0271B564 000504 00000000 C4C6E2E2 C1D4F0F2 0280E610 00000000 00000000 000C4040 00000000
*.....DFSSAM02..W.....*
0271B584 000524 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0271B5A4 000544 00000000 00000000 02757040 00000000 00000000 00000000 569ABC9A 1D1D014C
*.....<*
0271B5C4 000564 00000000 00001D1D 0271B0BC 00000000 014C0000 E6E3D6D9 40404040 00000000
*.....<..WTOR.....*
0271B5E4 000584 000056E0 00000000 00196D3D 0280E524 00000000 00000001 0000FFFF 827BEC70
*.....V.....B#..*
0271B604 0005A4 80000000 0271B648 829B8A1E 82978630 00000000 0271B060 827BEC70 028BB068
*.....B...BPF.....B#.....*
0271B624 0005C4 0271B060 0280E610 02825840 829B891C 02825000 026EB048 00000064 00C53D20 *...-..W..B.
B..I..B&;>..E..*
0271B644 0005E4 029B81E8 00000000 0271B600 0271B690 82978774 0297C2FE 00000000 02707540

```

INTERNAL TRACE RECORD				ID = LA3A	SEGN0=02	RECNO = 0000009C	TIME 07.45.06.42	DATE 93.014
CONTINUE								
0271B6C4	000664	0297C488	0271B060	0297C4A0	00C53D20	0297C2FE	00000000	0271B690 0271B720
*.PDH...-.PD..E...PB.....*								
0271B6E4	000684	82C45D79	82999D60	00C53D20	0271B060	00000410	00000584	0272E61C 02707598
*BD).BR..-E.....D..W....Q*								
0271B704	0006A4	02707554	0271B6C4	0272E5A4	0271B060	82C45E38	00C53D20	02C45B80 00000000 *.....D..VU...-
BD;..E...D\$......*								
0271B724	0006C4	0271B6D8	0271B768	8299B341	0299BAEC	000E3E8D	00003E8D	0299AD60 000E3E8D
*...Q....BR...R.....R.....*								
0271B744	0006E4	00000000	000000410	0271B068	8299A28A	00C19E00	0271B060	00C19000 00C53D20
*.....BRS..A.....-A...E...*								
0271B764	000704	82999D60	00000000	0271B720	0271B7B0	8299B341	0299BAEC	00C53D20 027BED10
*BR.....BR...R...E...#...*								
0271B784	000724	0299AD60	00C53D20	00000000	02707540	0272E594	829B891C	0272E078 0271B060
*.R..-E.....VMB.I.....*								
0271B7A4	000744	00C19000	00C53D20	82999D60	00000000	0271B768	0271B7F8	8299BD25 829CE580
*.A...E..BR.....8BR...B.V...*								
0271B7C4	000764	00C53D20	027BED10	027BED10	00000024	00000004	02707540	0272E594 829B891C
*.E...#...#.....VMB.I...*								
0271B7E4	000784	0272E078	0271B060	00C19000	00C53D20	0299BC2C	00000000	0271B7B0 0271B840
*.....-A...E...R.....*								
0271B804	0007A4	8299BF17	829CA578	00C53D20	027BED10	027BED10	00000028	0272E604 02707588
*BR..B.V..E...#...#.....W...H*								
0271B824	0007C4	02707550	0271B8C48	0272E5A4	0271B060	00C19000	00C53D20	0299BE12 00000000
*...&.....VU...-A...E...R.....*								
0271B844	0007E4	0271B7F8	0271B888	8299BD25	829CE580	00C53D20	027BED10	027BED10 00000024
*...8...HBR..B.V..E...#...#.....*								
0271B864	000804	00000004	02707540	0272E594	00000832	0272E078	0271B060	00C19000 00C53D20
*.....VM.....-A...E...*								
0271B884	000824	0299BC2C	0271B600	0271B840	0271B8D0	82957237	829A19C8	00000000 0275F260
*.R.....BN..B..H.....2...*								
0271B8A4	000844	C3D5E340	000001FF	02C5A758	02C5A758	00000000	00C34200	00C2A1C8 0271B060
*CNT.....EX..EX.....C...B.H...*								
0271B8C4	000864	0271B060	00C53D20	0295703E	00000000	0271B888	0271B918	829A1ACB 029A248E
*...E...N.....H.....B.....*								
0271B8E4	000884	00000000	00F76180	C3D5E340	000001FF	02C5A758	02C5A758	0271B04C 00000000
*.....7/.CNT.....EX..EX...<...*								
0271B904	0008A4	027BEC70	0271B060	0275F260	00C53D20	829A19C8	00000000	0271B8D0 0271B960
*.#. ....2..E..B..H.....2...*								
0271B924	0008C4	829A2579	829554F8	00C53D20	00C2A238	C3D5E340	000001FF	02C5A758 02C5A758
*B...BN.8.E...BS.CNT.....EX..EX...*								
0271B944	0008E4	00C2A238	00000000	027BEC70	0271B060	0275F260	00C53D20	029A248E 00000000
*.BS.....#.....2..-E.....*								
0271B964	000904	0271B918	0271B9A8	8011566B	829CA578	000053E8	02766910	00000000 02B54000
*.....Y...B.V...Y.....*								
0271B984	000924	000053E8	0276					

```

0271BBE4 000B84 0743506F 00000000 00000000 00000000 00000000 00000000
*..&?;.....*
0271BC04 000BA4 00000000 00000000 00000000 02707540 00000000 00000000
*.....*
0271BC24 000BC4 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0271BC44 000BE4 00000000
*.....*

```

### Related concepts

“Diagnosing routing errors by using the transaction trace or program trace” on page 315

The transaction trace or program trace can be used to diagnose routing error problems that are related to the user program routing exit DFSMSCE0.

### Related tasks

“Obtaining diagnostics in addition to SCRAPLOG and 6701-MRQE” on page 304

When the 6701-MRQE diagnostic records and the SCRAPLOG records do not provide enough diagnostic detail to adequately diagnose a problem, you can obtain additional diagnostic details by issuing the /

**TRACE SET ON PROGRAM** command. **/TRACE SET ON** *pgmname* causes the logging of additional 6701-MRQB records when the QCF BMP is processing.

## Receive-any buffer analysis

While talking with level 1 or level 2 IBM Software Support representatives, you might need to determine if you are out of receive-any (RECANY) buffers. Use either the IMS IPCS panel interface or the manual procedure to help you determine if this is the case. As you proceed through the steps, write down the information you gather.

## IMS IPCS Dump Formatter panel

The IMS IPCS Dump Formatter provides a panel-driven interface to perform analysis.

### Procedure

Using this IMS IPCS Dump Formatter panel, choose the RECANY (receive any) selection in the EDA/TM (Enhanced Dump Analysis/Transaction Manager) option, shown in the following figure, to create an output that contains the RECANY information.

```

----- RECEIVE ANY BUFFERS FORMATTING OPTIONS -- Row 1 to 3 of 3
COMMAND ==> Scroll ==> PAGE

The Receive Any Buffer contains input data that IMS receives from
VTAM terminals.

S = SELECT Select choice plus required ARGument
M = SELECT (minimum data) and hit enter to process. Use UP/DOWN
X = SELECT (maximum data) to scroll.

Cmd Option Type ARG Argument description
v-----vvvvvvvv-----
_ RECANY TYPE All Receive Any Buffers (leave ARG blank)
_ RECANY FILTER Receive Any Buffer filtering option
_ RECANY FILTER ALLOCATE Filter for allocated buffers
***** Bottom of data *****

```

Figure 68. IPCS Dump Formatter EDA/TM option

### Related tasks

[“Manual process to determine receive-any buffers space issues” on page 322](#)

While talking with level 1 or level 2 IBM Software Support representatives, you might need to determine if you are out of receive-any (RECANY) buffers. As you proceed through the steps, write down the information you gather.

## Manual process to determine receive-any buffers space issues

While talking with level 1 or level 2 IBM Software Support representatives, you might need to determine if you are out of receive-any (RECANY) buffers. As you proceed through the steps, write down the information you gather.

### Procedure

1. Find the address of the first RECANY buffer.
  - a) SCD+X'91C' = pointer to the first RECANY buffer (SCDRECPT)
  - b) SCD+X'920' = size of each RECANY buffer (SCDRCSIZ)
  - c) SCD+X'922' = number of RECANY buffers (SCDRCANY)
2. Offset X'04' in the RECANY buffer points to the next RECANY buffer. You can follow the chain of RECANY buffers using the pointer at offset X'04'.
3. Examine offset X'90' in each RECANY buffer (4 bytes). This field contains either an address of a CLB or zeros. If it contains a CLB address, the buffer is in use. If it contains zeros, in most cases the buffer is available.
4. If the buffer is tied to a CLB, the data you find in the following fields in the CLB is helpful in problem diagnosis.
  - a) CLB+X'00'-> Event Control Block (ECB) (4 bytes)
  - b) CLB+X'20'-> VTAM CID of the session (CLBCID) (4 bytes)
  - c) CLB+X'24'-> QE for queued receive-any buffers (CLBQE) (4 bytes)



- d) CLB+X'30' = Flag bytes (CLBFLAG1) (4 bytes)
- e) CLB+X'68' -> Input buffer (CLBINBUF) (4 bytes)
- f) CLB+X'6C' -> Output buffer (CLBOUTBF) (4 bytes)
- g) CLB+X'70' = QE for responses (CLBQERES) (4 bytes)
- h) CLB+X'74' = Flag bytes (CLBVFLAG) (4 bytes)

#### Related tasks

[“IMS IPCS Dump Formatter panel” on page 321](#)

The IMS IPCS Dump Formatter provides a panel-driven interface to perform analysis.

## Finding the active save set

To analyze data communication (DC) problems, you need to find the active save set at the time of abend.

### About this task

Use the following steps to locate the active save set.

### Procedure

1. Locate the registers at entry to abend (error registers). Register 13 points to the address of the active save set.
2. The active save sets begin under eye catcher \*\*\* SAVE AREA SET\*\*\*.
3. Find the save area (SA) address that matches the address in error register 13.

### Example of a Save Area Set

If error register 13 contains 320548, you would analyze the save set flow as shown the following figure. The registers in this save set are the registers that are saved on entry to each module.

```
***SAVE AREA SET***

EP DFSICI00
SA 22FE930

EP DFSCFEI0
SA 22E930

EP DFSCFEP0
SA 22E990

EP DFSCI0C0
SA 229490

EP DFSQMGR0
SA 22D990

EP DFSAOS80
SA 320548
```

## IMS VTAM interface

The basic functions of an IMS DC operation are establishing communications, sending and receiving messages, and terminating communications. The execution of these functions is shared among the elements that make up the network: the terminal, the controller, the VTAM system, the IMS system, and the application.

The communications analyzer (DFSICI00) uses the request parameter list (RPL) block to communicate with VTAM, and VTAM returns its status to IMS in the RPL. Therefore, it is important to analyze the RPL.

## IBM 3270 error recovery analysis

---

When the IBM 3270 detects an error, it sends the processor one of four sense-status messages.

- Intervention required, such as printer out of paper
- DEVICE END, which indicates the end of an operation
- DEVICE BUSY, normally caused by an operational error
- Hardware I/O error within the 3270 complex, such as a data check, control check, or equipment check

If IMS receives a sense-status message other than a DEVICE END, it issues message DFS973I.

## Diagnosing Message Format Service problems

---

The number of physical terminals traced and the number of lines traced can affect completeness of trace records and sequence of trace entries.

- Completeness of the trace record, (that is, whether or not all module activity related to a particular I/O action is traced), is affected if only one physical terminal (PTERM) is traced. The device-dependent module occasionally can change the current PTERM pointer before returning to the analyzer. Because the trace switch is kept in the CTB and is checked upon entry of a particular code, some module trace entries might be missing if the current CTB is not always maintained.
- Sequence of entries can be broken if more than one line is traced at a time. In this case, entries for a particular line have to be related by CLB.

Trace records with the following identifiers are useful in diagnosing MFS problems.

### **DD6M**

EDIT SEGMENT INTO TP BUFFER

#### **CIB**

MOD/DOF name

#### **MFS SEG**

SEGMENT created by MFS from output message and MOD/DOF

### **D01/DDM1**

PREPARE TO WRITE TO TERMINAL

#### **CIB**

Offset X'00' contains 8-byte MOD name.

Offset X'0C' contains 8-byte DOF name.

### **A05**

PRIOR TO ISSUING VTAM I/O REQUEST (NORMALLY A WRITE)

#### **O TP BUF**

Contains the data to be written to the terminal and the RPL for VTAM devices. Refer to the previous A05 record.

### **A01**

TERMINAL INPUT READY FOR IMS PROCESSING

#### **I TP BUF**

Contains input "device segment" 6 to 36 bytes from the beginning of the buffer. The data is preceded by a 2-byte length and 2 bytes of zeros.

### **FMT2**

ENTRY TO MFS INPUT PROCESSING

#### **CIB**

Offset X'00' contains MID name.

Offset X'22' indicates if PFK or PA key is used.

**X'80'**

PA key

**X'40'**

PFK key

**X'21'**

PA or PFK number

**FMT1**

MESSAGE TO BE EDITED BY BASIC EDIT, NOT MFS

**FMT3**

MFS HAS COMPLETED A MESSAGE SEGMENT

**MFS SEG**

Shows input segment created by MFS.

**MFS I WK**

Shows complete input message (all segments) and internal segment control information used by DFSCFEI0.

**MFS P WK**

This trace record is displayed only when the content of a protected field that is returned from a 3270 or SLU2 device is altered.

Offset 4 contains the address of the altered field in the input buffer.

Offset 8 contains MOD name.

Offset x'10' contains DOF name.

Offset x'2C' points to the protected field entry (PFVENTRY).

**PFVENTRY**

Offset 0 points to the next entry.

Offset 4 is the length of the entry, which consists of an 8-byte header length and data length.

Offset 6 is the address of the device field buffer of the protected field.

Offset 8 is the original content of protected field.

**ICLR**

A message satisfied MSGDEL=NONIOBCB for its destination PTERM and was deleted. The relevant control blocks are traced:

- Destination CTT
- Telecommunication program communication block (TP PCB)
- Destination CLB
- Destination CTB

This trace record is produced when any trace level is active for the destination PTERM.

**Note:** To examine the segments placed in the message queue, see X'01' and X'03' log records. X'01' log records contain input message segments. X'03' log records contain output message segments.

**Related tasks**

[DC trace \(Diagnosis\)](#)

## Message Format Service module traces

The Communications Interface Block (CIB) contains two module traces: CIBSTRAC and CIBTRACE.

### CIBSTRAC trace

CIBSTRAC is located in the CIB + X'50'. This 4-byte trace entry contains information indicating which MFS modules received control and in what order. The following figure shows the format.

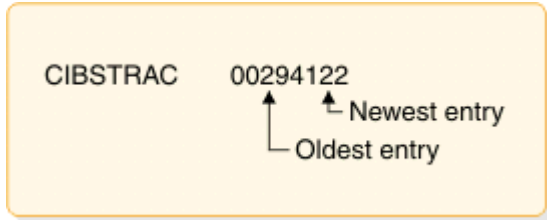


Figure 69. Example of CIBSTRAC trace

The leftmost nonzero digit shows the oldest entry and the high-order 4 bits of the rightmost byte show the newest. You can ignore the rightmost digit because it is always the same as the digit to its left. The trace entries are described in the following list.

#### Value (hex)

##### Meaning

- 1** Entry to DFSCFEQ0 (MFS resource cleanup).
- 2** Entry to DFSCFEI0 (MFS input editing occurred).
- 3** See value 8. Value 3 usually follows value 8 and is obtained by ORing 1 and 2.
- 4** INIT or DDFIN entry to DFSCFEO0 (either initial entry or after DDM6 finished current segment).
- 5** CONT entry to DFSCFEO0 (4 ORed with 1; after successful WRITE, next output segment was requested).
- 6** PAGEPOS entry to DFSCFEO0 (4 ORed with 2; entry after paging request).
- 7** DDNEXT entry to DFSCFEO0 (4 ORed with 3; DDM6 wanted next segment).
- 8** Entry to DFSCFEP0 (3 in the next slot; DFSCFEP0 flushed input message by calling DFSCFEQ0. After returning to DFSCFEP0, page position was established and exit to analyzer D was made. (Entry 8 was shifted left by DFSCFEQ0 entry and entry 1 was written. After returning to DFSCFEP0 1 was ORed with 2.)  
  
5 in the next slot; DFSCFEP0 flushed input message by calling DFSCFEQ0. After returning to DFSCFEP0, message dequeue routine was entered. Entry 8 was shifted and entry 1 was written by calling DFSCFEQ0. After returning to DFSCFEP0, DEQ routines ORed 1 with 4 resulting in 5.
- 9** Entry to DFSCFEP0 and exit to analyzer 3 entry. (8 ORed with 1).
- A** Entry to DFSCFEP0 (page position established) (8 ORed with 2).
- C** Entry to DFSCFEP0 and message dequeue requested. (8 ORed with 4).

## F

Noninitial entry to DFSCFEI0

### CIBTRACE trace

CIBTRACE is located in the extended CIB at CIB+X'70'. If the CIBSEXT flag is on (X'80'), an extended CIB exists. The following figure shows the format.

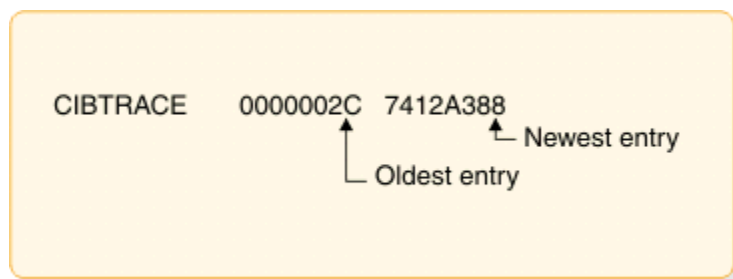


Figure 70. Example of CIBTRACE trace

The leftmost nonzero digit shows the oldest entry and high-order 4 bits of the rightmost byte show the newest. You can ignore the rightmost digit since it is always the same as the digit to its left. The trace entries are described in the following list.

#### Value (hex)

##### Meaning

#### 0

ENDMSG entry to DFSCFEI0 (Tests for EOT and spanned operation). If spanned, ENQWORK; if not, set EOM and setup for spanned operation.

#### 1

CPP100 entry to DFSCFEI0. Data was moved to message field.

#### 2

CPP10 entry to DFSCFEI0. Field was padded with fill character or literal has been moved into field.

#### 3

GETLBUF entry to DFSCFEI0. Acquire next line buffer. Return at entry GETLBUF2 with address of line buffer segment in register 1.

#### 4

NOFIT entry to DFSCFEI0. Sets up for spanned operation.

#### 7

GETWORK entry to DFSCFEI0. Acquire work buffer and initialize work buffer header. Moved data from QBUF to work buffer.

#### 8

REFRESH2 entry to DFSCFEI0. DIF table was cleared and setup.

#### 9

ENQWORK entry to DFSCFEI0. Segment in work buffer was moved to QBUF for processing.

#### A

FINQBUF entry to DFSCFEI0. Compress nulls out of segmenting work buffer.

#### B

NULLFDE entry to DFSCFEI0. Process all NULLFDEs.

#### C

PROCQBUF entry to DFSCFEI0. Return to analyzer to process QBUF.

#### D

GETQBUF entry to DFSCFEI0. Branches to analyzer entry C0 to acquire a QBUFFER.

#### F

ISRTNULL entry to DFSCFEI0. Inserts all null segments and processes them for move data.

## Tracing errors in module DFSCNXA0

DFSCNXA0 is the interface module between IMS and VTAM for all logon processing and abnormal session termination processing. It is often the first module to be notified when a failure occurs on a session and is always the first to get control when a node connects to IMS.

The session attributes are verified and the IMS session control blocks are built before the connection request is passed on to signon processing in IMS. The module consists exclusively of calls to VTAM exit routines.

### Location codes for DFSCNXA0 error messages

Message DFS3672I contains the location codes of the DFSCNXA0 error messages and the message also identifies the exit routine in which the error occurred.

Message DFS3672I contains the location codes listed in the following table.

Session failures might occur that do not cause any DFS messages to be issued by DFSCNXA0. In these cases, only message DFS3672 appears.

The format of the DFS3672I message is as follows:

```
DFS3672I SESSION ERROR. TYPE=aaa CODE=bb QUAL.=cc MSG=dddd
```

where

**aaa**

is the VTAM exit which was driven when the error occurred.

**bb**

is the location code of the error.

**cc**

is the location qualifier of the error.

Table 73. Location codes for DFSCNXA0 error messages

Location code (decimal)	Location code (hexadecimal)	Msg# (DFS)	Exit	Explanation
19	13	3862	LOG	Non-master terminal initiating a session on the alternate system.
20	14	3100	LOG	Node in FP input mode.
21	15	3645	LOG	Generic Resource name used but VGR for ISC was disabled.
147	93	3645	LOG	Parsing for userdata failed for AUTOSIGN terminal
151	97	3644	LOG	Could not get SOPB storage for AUTOSIGN terminal
22	16	3645	SCIP	Generic Resource name used but VGR for ISC was disabled.
1	1	N/A	LOST	No CID in VTAM parameter list.
2	2	N/A	LOST	CLB not found.
3	3	N/A	LOST	Stacked logon chaining error.
4	4	N/A	LOST	CLBs do not match (stacked logon situation).

Table 73. Location codes for DFSCNXA0 error messages (continued)

Location code (decimal)	Location code (hexadecimal)	Msg# (DFS)	Exit	Explanation
5	5	N/A	LOST	CLBs do not match (nonstacked situation).
1	1	N/A	NSXT	No CLB in USERFLD of NIB (Cleanup RU).
2	2	N/A	NSXT	No CID.
3	3	N/A	NSXT	CLB not found (Cleanup RU).
4	4	N/A	NSXT	CLB addresses do not match.
5	5	N/A	NSXT	IMS APPLID not found in resolve in-doubt vector list.
7	7	N/A	NSXT	Polarity mismatch on MSC link.
8	8	N/A	NSXT	Polarity mismatch on MSC link.
10	A	N/A	NSXT	Not Cleanup, NSPE, or Notify—RU is invalid.
11	B	N/A	NSXT	Invalid session key for NSPE.
12	C	N/A	NSXT	Invalid vector key for NOTIFY.
13	D	N/A	NSXT	Invalid session key for NOTIFY.
21	15	2061	NSXT	NSPE/NOTIFY processed.
22	16	2061	NSXT	NSPE/NOTIFY processed, AHDR not cleaned up.
23	17	2061	NSXT	CLB not found (NOTIFY RU).
1	1	N/A	RELQ	VTCTB not found.
2	2	N/A	RELQ	Terminal defined with NORELQ option.
3	3	N/A	RELQ	No CID in nonparallel-session VTCTB.
4	4	N/A	RELQ	No CID in any parallel-session VTCTBs.
1	1	1915	SCIP	No pointer to RPL.
2	2	1917	SCIP	Node not found.
3	3	3862	SCIP	VTCTB not found (XRF Alt.).
4	4	3862	SCIP	Invalid temporary VTCTB (XRF Alt.).
5	5	3862	SCIP	BIND not on surveillance link (XRF Alt.).
6	6	3101	SCIP	BIND not from same APPLID.
7	7	3101	SCIP	BIND rejected after setting VLGFF.
8	8	2104	SCIP	Non-LU 6.1 node.
9	9	3111	SCIP	Node stopped.
10	A	3101	SCIP	Logoff requested.
11	B	3101	SCIP	SPQB already allocated. Another 3672 (code=2D) is sent, after the -resp is sent.

Table 73. Location codes for DFSCNXA0 error messages (continued)

Location code (decimal)	Location code (hexadecimal)	Msg# (DFS)	Exit	Explanation
12	C	3101	SCIP	BIND not from same APPLID.
13	D	3101	SCIP	BIND rejected after setting CLBVLGFF flag.
14	E	2104	SCIP	CLEAR for non-ISC node.
15	F	970	SCIP	UNBIND entry message sent (after posting).
16	10	1931	SCIP	ASR processing begins.
17	11	2104	SCIP	SDT for non-ISC node.
18	12	1915	SCIP	Invalid command in RPL.
22	16	79	SCIP	Queues not available.

### Codes related to ISC processing

The codes in the following table relate to ISC processing—either as a result of LOGON or SCIP exits being driven. This is reflected in the DFS3672 message with 'I' appended to the exit type.

Table 74. Codes related to ISC processing

Location code (decimal)	Location code (hexadecimal)	Msg# (DFS)	Exit	Explanation
1	1	79	ISC	IMS shutting down.
2	2	1914	ISC	Bad INQUIRE return code.
3	3	1914	ISC	Bad INQUIRE feedback.
4	4	2066	ISC	USERFLD is zeros.
5	5	2066	ISC	First structured field not 0.
6	6	2066	ISC	User field length = 0.
7	7	2066	ISC	Primary Session Qualifier length = 0.
8	8	2066	ISC	Primary Session Qualifier length > 8.
9	9	2066	ISC	Secondary Session Qualifier length = 0.
10	A	2066	ISC	Secondary Session Qualifier length > 8.
11	B	3107	ISC	SPQB found but allocated.
12	C	3107	ISC	SPQB CRB pointer <> 0.
13	D	2049	ISC	VTCB not found and no dynamic terminals.
14	E	3101	ISC	No available VTCBs.
15	F	3107	ISC	Session initialization already begun.
16	10	3101	ISC	Second SCIP entry for same session.
17	11	3105	ISC	No CNTs on SPQB.



Table 74. Codes related to ISC processing (continued)

Location code (decimal)	Location code (hexadecimal)	Msg# (DFS)	Exit	Explanation
18	12	3107	ISC	Nonzero CID for existing session.
19	13	3111	ISC	Session blocked (3STOP).
20	14	3111	ISC	Session stopped.
21	15	3107	ISC	Ran out of CLBs.
22	16	3101	ISC	SPQB CRB pointer = 0.
23	17	1916	ISC	LOGON, but previous session was secondary.
24	18	1916	ISC	SCIP, but previous session was primary.
25	19	2066	ISC	User data length from INQUIRE = 0.
26	1A	3663	ISC	LU type in BIND = '0602' (LU 6.2)
27	1B	3107	ISC	SPQB found but allocated.
28	1C	3107	ISC	SPQB CRB pointer <> 0.
29	1D	3101	ISC	Second logon entry for same session.

The codes in the following table might occur during ISC BINDRACE processing.

Table 75. Codes related to ISC BINDRACE processing

Location code (decimal)	Location code (hexadecimal)	Msg# (DFS)	Exit	Explanation
41	29	N/A	ISC	SESSIONC not issuable—VTAM terminating.
42	2A	N/A	ISC	SESSIONC issued.
43	2B	N/A	ISC	SESSIONC not issuable—VTAM terminating.
44	2C	N/A	ISC	BIND not received.
45	2D	N/A	ISC	SESSIONC issued.

## Codes related to MSC errors

Codes related to MSC and MSC SCIP errors are listed, including their location codes, DFS message number, exit, and an explanation.

The codes in the following table relate to MSC errors.

Table 76. Codes related to MSC errors

Location code (decimal)	Location code (hexadecimal)	Msg# (DFS)	Exit	Explanation
51	33	3101	MSC	CID already present.
52	34	3213	MSC	3213 message issued. Code=4.

Table 76. Codes related to MSC errors (continued)

Location code (decimal)	Location code (hexadecimal)	Msg# (DFS)	Exit	Explanation
53	35	3213	MSC	3213 message issued. Code=8.
54	36	3213	MSC	3213 message issued. Code=24.
55	37	3213	MSC	3213 message issued. Code=32.

The codes in the following table relate to MSC SCIP errors.

Table 77. Codes related to MSC SCIP errors

Location code (decimal)	Location code (hexadecimal)	Msg# (DFS)	Exit	Explanation
71	47	N/A	MSC	CID already present.
72	48	N/A	MSC	No USERFLD provided.
73	49	N/A	MSC	RPL not initialized.

## Codes related to dynamic logon

Dynamic logon codes, their locations, DFS message number, exit and explanation are listed.

Table 78. Codes related to dynamic logon errors

Location code (decimal)	Location code (hexadecimal)	Msg# (DFS)	Exit	Explanation
81	51	2264	LOG	Do not accept logons.
82	52	3862	LOG	Nonexistent VTCB trying to logon to alternate system.
83	53	2037	LOG	/STA DC not done.
84	54	2104	LOG	Invalid temporary VTCB exists.
85	55	3862	LOG	Invalid temporary VTCB exists.
86	56	3862	LOG	Logon not for XRF link.
87	57	3111	LOG	Node stopped.
88	58	2264	LOG	Logons not accepted and SIMLOG not in effect.
89	59	3862	LOG	In backup but not preopen.
90	5A	3862	LOG	In backup preopen but backup session not allowed.
91	5B	2037	LOG	/STA DC not done.
92	5C	79	LOG	Queues not available.
93	5D	3111	LOG	Node not started.
94	5E	79	LOG	Shutting down and MTO logging not on.

Table 78. Codes related to dynamic logon errors (continued)

Location code (decimal)	Location code (hexadecimal)	Msg# (DFS)	Exit	Explanation
95	5F	3111	LOG	Node stopped.
96	60	3101	LOG	Node logging off.
97	61	3101	LOG	Session terminating.
98	62	3101	LOG	CID already exists.
99	63	3111	ISC	Node stopped on temporary VTCB.

## Codes related to existing ISC session errors

ISC session error codes are listed, including their location, DFS message number, exit, and explanation.

Table 79. Codes related to existing ISC session errors

Location code (decimal)	Location code (hexadecimal)	Msg# (DFS)	Exit	Explanation
111	6F	3645	ISC	QSAVE could not be gotten.
112	70	3645	ISC	Parsing failed.
113	71	3645	ISC	Dynamic terminals not allowed.

## Codes related to user-logon-exit routine processing

User-logon-exit routine processing codes are listed, including their location, DFS message number, exit, and explanation.

Table 80. Codes related to user-logon exit routine processing

Location code (decimal)	Location code (hexadecimal)	Msg# (DFS)	Exit	Explanation
121	79	3645	LOG	Could not get QSAVE for signon parameters.
122	7A	3645	LOG	Parsing failed.
123	7B	3645	LOG	User logon exit rejected logon.
124	7C	3645	LOG	User logon exit rejected logon.
125	7D	3645	LOG	Invalid ALOT or ASOT value from user logon exit routine
126	7E	3645	N/A	User logon exit routine erased all descriptors.
127	7F	3645	LOG	A dynamically created logging-on STSN VCTB must have user data.
128	80	3645	LOG	Existing dynamic logging-on STSN VTCB must have user data.

## Codes related to logon errors

Logon error codes are listed, with their locations, DFS message number, exit, and explanation.

*Table 81. Codes related to logon errors*

<b>Location code (decimal)</b>	<b>Location code (hexadecimal)</b>	<b>Msg# (DFS)</b>	<b>Exit</b>	<b>Explanation</b>
141	8D	3645	N/A	Dynamic terminals not allowed.
142	8E	3646	N/A	Inconsistent attributes.
143	8F	3646	N/A	Inconsistent attributes.
144	90	3645	N/A	Could not get SOPB storage.
145	91	3645	N/A	Parsing of userdata failed.
146	92	3645	N/A	Terminal is the primary or secondary master terminal for the alternate system in an XRF environment.
148	94	3644	N/A	Could not get SOPB storage.
149	95	3644	N/A	Could not get SOPB storage.
150	96	2066	LOG	The LUTYPE in BIND/CINIT conflicts with static ISC block LUTYPE.
161	A1	3671	N/A	Invalid descriptor specified in userdata.
162	A2	3651	N/A	No default descriptor found.
163	A3	3671	N/A	User logon exit routine returned invalid descriptor.
164	A4	3644	N/A	Could not get SOPB storage.
165	A5	3651	N/A	No default descriptor found.

### Related reference

[“Qualifier codes” on page 335](#)

Qualifier codes for ETO parsing errors, VTCB creation errors, screen-attribute errors, are listed with their location codes, DFS message number, exit, and explanation.

## Codes related to logon descriptor processing

Logon descriptor processing codes are listed along with their location, DFS message number, exit, and explanation.

*Table 82. Codes related to logon descriptor processing*

<b>Location code (decimal)</b>	<b>Location code (hexadecimal)</b>	<b>Msg# (DFS)</b>	<b>Exit</b>	<b>Explanation</b>
181	B5	3663	LOG	LU type must be < 7.
182	B6	3663	LOG	LU type must be >= 0.
183	B7	3663	LOG	Invalid LU type specified.

Table 82. Codes related to logon descriptor processing (continued)

Location code (decimal)	Location code (hexadecimal)	Msg# (DFS)	Exit	Explanation
184	B8	3663	LOG	Invalidly-specified non-SNA 3270 VTAM device. Make sure mode-table is properly defined and referenced.
185	B9	3663	LOG	Invalid LU 1 or NTO device type.

## Codes related to logging-on device characteristics

Logging-on device characteristic codes are listed along with their location, DFS message number, exit and explanation.

Table 83. Codes related to logging-on device characteristics

Location code (decimal)	Location code (hexadecimal)	Msg# (DFS)	Exit	Explanation
191	BF	3646	LOG	Invalid SLU 1 device logging on.
192	C0	3646	LOG	Device LU type does not match descriptor.
193	C1	3646	LOG	Non-SNA 3270 VTAM logon descriptor invalid for the logging-on device.
194	C2	3646	LOG	Invalid SLU P or 3600 type device mismatch with the logon descriptor.
195	C3	3646	LOG	TS type or LU type mismatch.

## Qualifier codes

Qualifier codes for ETO parsing errors, VTCB creation errors, screen-attribute errors, are listed with their location codes, DFS message number, exit, and explanation.

## Codes related to ETO parsing errors

### Codes related to ETO parsing errors

The Qualifier codes in the following table relate to ETO parsing errors that are associated with message DFS3645I.

Table 84. Qualifier codes related to ETO parsing errors

Location code (decimal)	Location code (hexadecimal)	Msg# (DFS)	Exit	Explanation
1	1	N/A	N/A	Invalid logon descriptor name—no name specified.
2	2	N/A	N/A	Invalid logon descriptor name—name is greater than 8 characters.
3	3	N/A	N/A	Invalid logon descriptor name—no name specified.

## Codes related to VTCB creation errors

### Codes related to VTCB creation errors

The Qualifier codes in the following table relate to VTCB creation errors that are associated with a DFS3644 message.

Table 85. Qualifier codes related to VTCB creation errors

Location code (decimal)	Location code (hexadecimal)	Msg# (DFS)	Exit	Explanation
1	1	N/A	N/A	QSAVE not gotten.
2	2	N/A	N/A	VTCB could not be created.
3	3	N/A	N/A	Could not put VTCB into hash table.

## Codes related to screen-attribute errors

### Codes related to screen-attribute errors

The Qualifier codes in the following table relate to screen-attribute errors (associated with a DFS3646I message).

Table 86. Qualifier codes related to screen-attribute errors

Location code (decimal)	Location code (hexadecimal)	Msg# (DFS)	Exit	Explanation
1	1	N/A	N/A	No Device Characteristics Table. Run the MFS DCT (DFSUTB00) utility.
2	2	N/A	N/A	No match for screen size and feature. Update MFS DCT (DFSUTB00) for the missing entry.
3	3	N/A	N/A	Screen size control byte incorrectly specified. The byte itself might be invalid. If X'7F' is specified, then a valid screen size must also be specified.

### Related reference

“Codes related to logon errors” on page 334

Logon error codes are listed, with their locations, DFS message number, exit, and explanation.

## IDC0 trace table entries

IDC0 trace table entries, including error messages issued by DFSCNXA0 are listed along with the codes and DFS message number.

### Error messages issued by DFSCNXA0

#### Error messages issued by DFSCNXA0

The following table shows codes that identify error messages that are issued by DFSCNXA0. The code is placed in the MsgID field of an IDC0 trace entry.

Table 87. Codes that identify error messages issued by DFSCNXA0

Code (decimal)	Code (hexadecimal)	Msg# (DFS)
0	00	2104

Table 87. Codes that identify error messages issued by DFSCNXA0 (continued)

Code (decimal)	Code (hexadecimal)	Msg# (DFS)
4	04	3111
8	08	2037
12	0C	79
16	10	1915
20	14	1917
24	18	1931
28	1C	3862
32	20	970
36	24	1916
40	28	1914
44	2C	2066
48	30	3107
52	34	3105
56	38	3101
60	3C	N/A
64	40	2049
68	44	3213
72	48	2264
76	4C	3644
80	50	3645
84	54	3646
88	58	3651
92	5C	3663
96	60	N/A
100	64	3671
104	68	2061

The following internal trace formats map IDC0 trace table entries:

**Format 1 (IDC0)**

**XL1**

Function Code = X'B8' (set by 'DFSTRACE')

**XL1**

Subcode

**XL2**

Unusable

**XL1**

RPLRTNCD - RPL return code

**XL1**

RPLFDB2 - RPL feedback

**XL1**

Reserved

**XL1**

Error type

X'80' = 2061 error

X'40' = 2062 error

X'20' = 970 error

**CL8**

Nodename

**CL8**

Mode-table entry name

**CL8**

Applid (if applicable)

or

**CL8**

Time stamp

**Format 2 (CNXA)**

One event can span two entries.

**First Entry****XL1**

Function Code = X'B9' (set by 'DFSTRACE')

**XL1**

Subcode

**XL2**

Unusable

**XL1**

VTAM-exit indicator

00 --> You are looking at the '2nd' entry

04 --> LOGON EXIT ENTERED

08 --> SCIP EXIT ENTERED

0C --> NSEXIT EXIT ENTERED

10 --> LOSTERM EXIT ENTERED

14 --> RELREQ EXIT ENTERED

**XL1**

Error location code

**XL1**

Location code qualifier

**XL1**

Processing flag at error time

80 VTCB LATCH HELD

40 LOGON DESCRIPTOR NAME IN CINIT/BIND

20 VTCB DOES NOT YET EXIST

10 VTCB ATTEMPTING CONNECTION FOUND

08 SPQB FOUND

04 IMS CORRELATION ID IN USERDATA

02 ISC PROCESSING ENTERED

01 EXISTING VTCB IN LOGOFF PROCESS



**CL8**

Nodename

**XL4**

LOSTERM reason code

**XL4**

CLB address

**XL4**

CID

**XL1**

LU type

**XL1**

TS profile

**XL1**

MSG ID of error message

**XL1**

Reserved

**2nd entry (in the case of LOGON or SCIP exits being driven)****XL1**

Function Code = X'B9' (set by 'DFSTRACE')

**XL1**

Subcode

**XL2**

Unusable

**XL4**

Reserved

**CL8**

Nodename

**CL8**

Descriptor name or subpool name

**XL8**

Time stamp

## APPC/IMS diagnostic aids

---

APPC/IMS diagnostic aids include LU manager trace, LU 6.2 module-to-code cross references, APPC/MVS verb-to-code cross references, DDFS1959e message information, diagnostic information for APP and OTMA with shared queues, and information on SNAPs and dumps.

### LU manager trace

The LU manager trace records the flow of control through the IMS LU 6.2 components. Analyzing the trace entries together with the MVS/ESA APPC trace entries is useful in determining the problem.

#### About this task

## Starting the LU manager trace

The **/TRACE SET ON TABLE LUMI** command activates the trace and sends the entries to an internal table.

### About this task

You can format the table using the Offline Dump Formatter under IPCS, using either the VERBX command or the IMS Dump Formatter panels.

If a SNAP dump is taken, the table is formatted as part of the IMS dump.

If you add the OPTION LOG parameter to the **/TRACE** command, IMS sends the output to an external data set. You can use the File Select and Formatting utility (DFSERA10) with exit DFSERA60 to format the trace entries.

### Related concepts

[“Formatting IMS dumps offline” on page 512](#)

Two methods are available for formatting IMS dumps offline: interactive formatting, performed through a series of panels which provide formatting choices, and formatting by using JCL.

## Formatting the LU manager trace

The general format of an LU manager trace record is shown. Each record is 8 words long. Word 0 holds standard information for each record.

### LU manager trace record format

Table 88. LU manager trace record format

WORD 0		WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7
ID	SEQ NUM							

where  
represents

#### ID

Two-byte trace ID.

#### SEQ NUM

Two-byte trace sequence number assigned by the IMS trace component.

Words 1 thru 7 contain data specific to each trace entry, as described below:

**TRACE ID = X'7B01'** LUM module entry

#### Word 1

**byte 0:**Module number **bytes 1-3:** Reserved

#### Word 2

A(ECB)

#### Word 3

Register 1

#### Words 4-5

Optional user data

#### Words 6-7

Time stamp (STCK)

**TRACE ID = X'7B02'** LUM module exit

#### Word 1

**byte 0:**Module number **bytes 1-3:** Reserved

**Word 2**

A(ECB)

**Word 3**

Return code

**Words 4-5**

Optional user data

**Words 6-7**

Time stamp (STCK)

**TRACE ID = X'7B03'** IMS internal LUM error**Word 1****byte 0:** Module number **bytes 1-3:** 0**Word 2**

Error code

**Word 3**

TIB address

**Word 4**

TIB prefix address

**Word 5**

0

**Words 6-7**

Time stamp (STCK)

**TRACE ID = X'7B04'** IMS APPC Status Change**Word 1**

- **byte 0:** Module number
- **byte 1:** AWE function requested code
  - **X'01':** Initialization request
  - **X'02':** Dependent region connected
  - **X'03':** Start APPC
  - **X'04':** Stop APPC
  - **X'05':** Purge APPC
  - **X'06':** Cancel APPC
  - **X'07':** Terminate APPC
  - **X'08':** Attach request
  - **X'09':** APPC initialized
  - **X'0A':** APPC stopped
  - **X'0B':** LU activated
  - **X'0C':** LU deactivated
  - **X'0D':** XRF takeover
  - **X'0E':** Clear TIBs
  - **X'0F':** Build LU6.2 descriptors
- **byte 2:** Current APPC status
  - **X'C1':** Starting
  - **X'C3':** Cancelled
  - **X'C4':** Disabled
  - **X'C5':** Enabled

- **X'C6'**:Failed
- **X'D6'**:Outbound
- **X'D7'**:Purging
- **X'E2'**:Stopped
- **byte 3:** Desired/requested APPC status
  - **X'C1'**:Starting
  - **X'C3'**:Cancelled
  - **X'C4'**:Disabled
  - **X'C5'**:Enabled
  - **X'C6'**:Failed
  - **X'D6'**:Outbound
  - **X'D7'**:Purging
  - **X'E2'**:Stopped

#### **Word 2**

A(ECB)

#### **Word 3**

- **byte 0:** Last APPC status
  - **X'C1'**:Starting
  - **X'C3'**:Cancelled
  - **X'C4'**:Disabled
  - **X'C5'**:Enabled
  - **X'C6'**:Failed
  - **X'D6'**:Outbound
  - **X'D7'**:Purging
  - **X'E2'**:Stopped
- **byte 1:** Last Desired/requested APPC status
  - **X'C1'**:Starting
  - **X'C3'**:Cancelled
  - **X'C4'**:Disabled
  - **X'C5'**:Enabled
  - **X'C6'**:Failed
  - **X'D6'**:Outbound
  - **X'D7'**:Purging
  - **X'E2'**:Stopped
- **bytes 2-3:** 0

#### **Word 4**

0

#### **Word 5**

0

#### **Words 6-7**

Time stamp (STCK)

**TRACE ID = X'7B05'** LUM module IWAIT

#### **Word 1**

**byte 0:**Module number **bytes 1-3:** Reserved

**Word 2**

A(ECB)

**Word 3**

TIB\_SYNC\_PTR

**Words 4**

A(TIB)

**Words 5**

0

**Words 6–7**

Time stamp (STCK)

**TRACE ID = X'7B06'** LUM module IPOST**Word 1****byte 0:**Module number **bytes 1-3:** 0**Word 2**

A(ECB)

**Word 3**

TIB\_SYNC\_PTR

**Words 4**

A(TIB)

**Words 5**

0

**Words 6–7**

Time stamp (STCK)

**TRACE ID = X'7B07'** z/OS cross-system coupling facility sendmsg**Word 1**

- **Byte 0:** Module number - see [Table 89 on page 348](#)
- **Bytes 2-3:** Function Return Code

**Word 2**

A(TIB)

**Words 3-6**

TIB MSG PREFIX URTOKEN

**Word 7**

Time stamp (STCK)

**TRACE ID = X'7B08'** AWE server TIB POST**Word 1**

- **Byte 0:** Module number - see [Table 89 on page 348](#)
- **Bytes 2-3:** Return code

**Word 2**

A(ECB)

**Word 3**

AOS POST CODE

**Word 4**

A(TIB)

**TRACE ID = X'7B09'** Request sent to z/OS Resource Recovery Services AWE server**Word 1**

- **Byte 0:** Module number - see [Table 89 on page 348](#)

- **Bytes 2-3:** AWRRFUNC

#### **Word 2**

A(ECB)

#### **Word 3**

AOS POST CODE

#### **Words 4-7**

MSG PREFIX URTOKEN

**TRACE ID = X'7C01'** Normal return from APPC/MVS

#### **Word 1**

- **byte 0:** Module number - See [Table 89 on page 348](#).
- **byte 1:** ATB call number - See [Table 90 on page 350](#).
- **byte 2:** ATB flags
  - **bit 0:** Verb issued for asynchronous processing
  - **bit 1:** Return code is from asynchronous processing
  - **bit 2:** CID given and all zeros
  - **bit 3:** TPID field has user data
  - **bit 4:** CID field has user data
- **byte 3:** Optional user data

#### **Words 2-3**

TPID or user data

#### **Words 4-5**

CID or user data

#### **Word 6**

Return code

#### **Word 7**

A(ECB)

**TRACE ID = X'7C02'** Unexpected return code from APPC/MVS

#### **Word 1**

- **byte 0:** Module number
- **byte 1:** ATB call number
- **byte 2:** ATB flags
  - **bit 0:** Verb issued for asynchronous processing
  - **bit 1:** Return code is from asynchronous processing
  - **bit 2:** CID given and all zeros
  - **bit 3:** TPID field has user data
  - **bit 4:** CID field has user data
- **byte 3:** Optional user data

#### **Words 2-3**

TPID or user data

#### **Words 4-5**

CID or user data

#### **Word 6**

Return code

#### **Word 7**

A(ECB)

**TRACE ID = X'7C03'** APPC/MVS asynchronous verb entry

**Word 1**

- **byte 0:** Module number
- **byte 1:** ATB call number
- **byte 2:** ATB flags
  - **bit 0:** Verb issued for asynchronous processing
  - **bit 1:** Return code is from asynchronous processing
  - **bit 2:** CID given and all zeros
  - **bit 3:** TPID field has user data
  - **bit 4:** CID field has user data
- **byte 3:** Optional user data

**Words 2-3**

TPID or user data

**Words 4-5**

CID or user data

**Word 6**

Reserved (FFFFFFFF)

**Word 7**

A(ECB)

**TRACE ID = X'7F01'** APPC Attach from APPC/MVS

**Word 1**

Reserved

**Word 2**

XCF message type

**Words 3-4**

TPID for XCF message

**Words 5-6**

Local LU to which ATTACH request was directed

**Word 7**

Time stamp (STCK)

**TRACE ID = X'7F02'** IMS LU activating or deactivating

**Word 1**

Reserved

**Word 2**

XCF message type

**Word 3**

XCF message LU flags **bit 0:** LU is base LU

**Words 4-5**

LU name

**Word 6**

0

**Word 7**

Time stamp (STCK)

**TRACE ID = X'7F03'** APPC/MVS starting or stopping

**Word 1**

Reserved

**Word 2**

XCF message type

**Words 3-6**

0

**Word 7**

Time stamp (STCK)

**TRACE ID = X'7F04'** CPOOL storage shortage**Word 1**

Reserved

**Word 2**

XCF message type

**Word 3**

XCF message length

**Words 4-5**

TPID from XCF message

**Word 6**

0

**Word 7**

Time stamp (STCK)

**TRACE ID = X'7F05'** CPOOL block too small for XCF message**Word 1**

Reserved

**Word 2**

XCF message type

**Word 3**

XCF message length

**Word 4**

Cell size

**Words 4-5**

TPID from XCF message

**Word 6**

0

**Word 7**

Time stamp (STCK)

**TRACE ID = X'7F06'** Invalid request from XCF**Word 1**

Reserved

**Word 2**

XCF message type

**Word 3**

0

**Words 4-5**

MEPLSRCE map

**Word 6**

0

**Word 7**

Time stamp (STCK)



**TRACE ID = X'7F07'** APPC/MVS not enabled for Attach

**Word 1**

Reserved

**Word 2**

XCF message type

**Word 3**

- **byte 0:** LSCD status (disabled, failed, stopped)
- **byte 1:** LSCD IN flags (LSCD - APPC/IMS global control block)
- **byte 2:** LSCD OUT flags
- **byte 3:** LSCD flags

**Word 4**

0

**Words 5-6**

TPID from XCF message

**Word 7**

Time stamp (STCK)

**TRACE ID = X'7F09'** TP deallocate failed

**Word 1**

Reserved

**Word 2**

XCF message type

**Word 3**

Return code

**Words 4-6**

0

**Word 7**

Time stamp (STCK)

## LU manager trace example

The following LU manager trace example shows calls to DFS62FD0 caused by **/DISPLAY** commands, a clean address space caused by a non-LU 6.2 transaction ending, and a synchronous LU 6.2 transaction being executed.

### LU manager trace example

The trace has been formatted by the File Select and Formatting utility (DFSERA10) with exit DFSERA60, which places the module number after word 7.

```
OPTION PRINT 0=5,V=67FA,EXITR=DFSERA60
END
FUNCTION      WORD 0      WORD 1      WORD 2      WORD 3      WORD 4      WORD 5      WORD 6      WORD 7
* LU1 TRACE TABLE - DATE 91323 TIME 11323667 SKIP 0000 TOTAL SKIP 00000000 RECORD NUMBER 00000167
Module Exit    7B023DD8    20000000    03080330    00000004    10800000    00000000    A4D224D2    27C7AB05    32
Module Exit    7B023E22    20000000    03080330    00000004    10800000    00000000    A4D224D2    34020504    32
Module Exit    7B023E2B    20000000    03080330    00000004    10400000    00000000    A4D224D2    340ACC04    32
Module Entry   7B01554C    0B000000    028E0060    02942244    00020080    00000000    A4D22B41    9C54DB04    11
APPC/MVS Exit  7C01554F    0B120000    FFFFFFFF    FFFFFFFF    FFFFFFFF    FFFFFFFF    00000004    028E0060    11-
ATBCMAS
Module Exit    7B025552    0B000000    028E0060    00000000    00000000    00000000    A4D22B41    9C5EEA04    11
APPC ATTACH    7F01AC63    00000000    00000001    037AE648    00000002    D3F6F2C9    D4E2F140    48CE0D51
Module Exit    7B02AC8D    20000000    02D02020    00000000    40100000    0310E2B0    A4D24448    CEE7BE05    32
Module Entry   7B01AC97    06000000    0310E2B0    0294D538    00000000    00000000    A4D24448    CEF77405    06
Module Entry   7B01AC9C    10000000    0310E2B0    03036334    01000000    0310E5B2    A4D24448    CF163105    16
Module Exit    7B02AC9D    10000000    0310E2B0    00000000    404008C1    00000000    A4D24448    CF169505    16
Module Entry   7B01ACA2    10000000    0310E2B0    03036334    04020000    0310E5B2    A4D24448    CF1AA305    16
Module Exit    7B02ACA3    10000000    0310E2B0    00000000    404008C1    00000000    A4D24448    CF1B0905    16
APPC/MVS Entry 7C03ACA8    060D8040    037AE648    00000002    037B6018    00000002    FFFFFFFF    0310E2B0    06-
```

```

ATBRCVW
APPC/MVS Exit 7C01ACB0 060DC000 037AE648 00000002 037B6018 00000002 00000000 0310E2B0 06-
ATBRCVW
APPC/MVS Entry 7C03ACB7 060D8040 037AE648 00000002 037B6018 00000002 FFFFFFFF 0310E2B0 06-
ATBRCVW
APPC/MVS Exit 7C01ACBF 060DC001 037AE648 00000002 037B6018 00000002 00000000 0310E2B0 06-
ATBRCVW
Module Entry 7B01ACC4 22000000 0310E2B0 03035E98 C1D7D6D3 F1F14040 A4D24448 E8BD6C05 34
Module Exit 7B02ACC5 22000000 0310E2B0 00000000 00000000 00140014 A4D24448 E8C11D05 34
Module Exit 7B02ACEF 06000000 0310E2B0 00000000 00000000 00000000 A4D24448 E9427C05 06
Module Entry 7B01AD41 0A000000 028E0060 02942C78 80000080 028E00F8 A4D24448 F43CDC04 10
Module Exit 7B02AD48 20000000 028E0060 00000000 00100000 0310E2B0 A4D24448 F44ABE04 32
Module Exit 7B02AD4B 0A000000 028E0060 00000000 028E00F8 028E00AC A4D24448 F44D9404 10
APPC/MVS Exit 7C01AD59 3E110000 037AE648 00000002 00000000 00000000 00000000 028E0060 62-
ATBASOC
Module Entry 7B01AD5B 10000000 028E0060 02938040 01000000 02CF9AFE A4D24448 F9BF9F04 16
Module Exit 7B02AD5C 10000000 028E0060 00000000 00000000 00000000 A4D24448 F9C01704 16
Module Entry 7B01AD78 0A000000 028E0060 02942240 00800080 028E00F8 A4D24449 5C418704 10
Module Entry 7B01AD7B 01000000 028E0060 02B921A8 80000000 028E00EC A4D24449 5C4E4D04 01
Module Entry 7B01AD9A 22000000 028E0060 02B929C0 C1D7D6D3 F1F14040 A4D24449 5D101404 34
Module Exit 7B02AD9B 22000000 028E0060 00000000 04000000 00270027 A4D24449 5D10D104 34
APPC/MVS Entry 7C03ADA0 010F8000 037AE648 00000002 037B6018 00000002 FFFFFFFF 028E0060 01-
ATBSEND
APPC/MVS Exit 7C01ADA8 010FC000 037AE648 00000002 037B6018 00000002 00000000 028E0060 01-
ATBSEND
Module Entry 7B01ADAD 22000000 028E0060 02B929C0 C1D7D6D3 F1F14040 A4D24449 5E1F7B04 34
Module Exit 7B02ADAE 22000000 028E0060 00000000 04000000 00260026 A4D24449 5E202704 34
APPC/MVS Entry 7C03ADB3 010F8000 037AE648 00000002 037B6018 00000002 FFFFFFFF 028E0060 01-
ATBSEND
APPC/MVS Exit 7C01ADBB 010FC000 037AE648 00000002 037B6018 00000002 00000000 028E0060 01-
ATBSEND
APPC/MVS Entry 7C03ADC0 01068000 037AE648 00000002 037B6018 00000002 FFFFFFFF 028E0060 01-
ATBFLUS
APPC/MVS Exit 7C01ADC8 0106C000 037AE648 00000002 037B6018 00000002 00000000 028E0060 01-
ATBFLUS
Module Exit 7B02ADDB 01000000 028E0060 00000000 00010000 00000000 A4D24449 5E828004 01
Module Exit 7B02ADDE 0A000000 028E0060 00000000 028E00F8 00000000 A4D24449 5E855A04 10
Module Entry 7B01ADEE 0B000000 028E0060 02942240 00400080 028E00F8 A4D24449 5E9D0E04 11
Module Exit 7B02ADED 0B000000 028E0060 00000000 028E00F8 00000000 A4D24449 5E9E4C04 11
Module Entry 7B01ADF8 0A000000 028E0060 02942240 00040080 00000000 A4D24449 5EAAA104 10
Module Exit 7B02ADF9 0A000000 028E0060 00000000 00000000 028E00AC A4D24449 5EABB204 10
Module Entry 7B01AE09 0A000000 028E0060 02942240 00200080 028E00F8 A4D24449 5EB48D04 10
APPC/MVS Entry 7C03AE0C 0A048000 037AE648 00000002 037B6018 00000002 FFFFFFFF 028E0060 10-
ATBDEAL
APPC/MVS Exit 7C01AE14 0A04E000 037AE648 00000002 037B6018 00000002 00000000 028E0060 10-
ATBDEAL
Module Exit 7B02AE19 20000000 028E0060 00000000 80100000 00000000 A4D24449 5EF81604 32
Module Exit 7B02AE1C 0A000000 028E0060 00000000 028E00F8 00000000 A4D24449 5F104504 10
Module Entry 7B01AE3F 0B000000 028E0060 02942244 00020080 00000000 A4D24449 5F2BD704 11
APPC/MVS Exit 7C01AE42 0B150000 037AE648 00000002 FFFFFFFF FFFFFFFF 00000004 028E0060 11-
ATBCMTTP
Module Exit 7B02AE45 0B000000 028E0060 00000000 00000000 00000000 A4D24449 D2E40205 11
Module Entry 7B01AE5A 0B000000 028E0060 02942244 00020080 00000000 A4D24449 D5D0AD05 11
APPC/MVS Exit 7C01AE5D 0B120000 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000004 028E0060 11-
ATBCMAS
Module Exit 7B02AE60 0B000000 028E0060 00000000 00000000 00000000 A4D24449 D5DB1205 11
DFS707I END OF FILE ON INPUT
DFS708I OPTION COMPLETE
DFS703I END OF JOB

```

## LU 6.2 module-to-code cross-reference table

You can use the module number, module name, and description to associate code xx in message DFS1959E and the module number in trace records X'7Bxx' and X'7Cxx' with a module.

Table 89. LU 6.2 module-to-code cross-reference table

Module number (decimal)	Module number (hexadecimal)	Module	Description
01	01	DFSSLUM0	Synchronous output LU manager
02	02	DFSAPPC0	DFSAPPC message switch processor
03	03	DFSCMD00	LU 6.2 command interface

Table 89. LU 6.2 module-to-code cross-reference table (continued)

Module number (decimal)	Module number (hexadecimal)	Module	Description
04	04	DFSALM00	Asynchronous output LU manager
05	05	DFSRLM00	Receive LU manager server
06	06	DFSRLM10	Receive LU manager receiver
08	08	DFSAPP10	DFSAPPC keyword parser
09	09	DFSATB00	APPC/MVS verb execution/trace
10	0A	DFS6LUS0	LU 6.2 services interface 1
11	0B	DFS6LUS1	LU 6.2 services interface 2
12	0C	DFS6LUS2	LU 6.2 services interface 3
16	10	DFSRAC60	RACF interface module
21	15	DFS6RST0	LU 6.2 restart processor
22	16	DFS6CKP0	LU 6.2 checkpoint processor
24	18	DFSGIDC0	Read and build LU 6.2 descriptors
31	1F	DFS6ECT0	LU 6.2 z/OS cross-system coupling facility message processor
32	20	DFS62FD0	LU 6.2 Find destination routine (QABs/TIBs)
33	21	DFSLUDIO	LU 6.2 User Destination exit
34	22	DFSLIEE0	LU 6.2 User Data Edit exit
35	23	DFSHCI00	XRF takeover processing
36	24	DFS6QFX0	LU 6.2 Nonrecoverable message cleanup
37	25	DFSHAV70	XRF termination/takeover
38	26	DFS62FD1	LU 6.2 Find destination routine (LUBs/DESCs)
40	28	DFSCMLC0	MSC SQ APPC/OTMA Message Router
41	29	DFSCMS00	MS Analyzer
50	32	DFSXLUM0	LUM TCB Initialization routine
51	33	DFSYIOE0	OTMA Input and Output user exit
52	34	DFSXXCF0	XCF TCB initialization
53	35	DFSXRM00	RLUM TCB initialization
54	36	DFSXALM0	ALUM TCB initialization
55	37	DFSXALC0	ALUM allocate TCB initialization
56	38	DFSFLUM0	LUM TCB ESTAE routine
60	3C	DFSICM20	LU 6.2 command processor
61	3D	DFSTMR00	TM ABEND retry eligibility module
62	3E	DFSTMAS0	TM ASSOCIATE TPI and create accessor environment element (ACEE)

Table 89. LU 6.2 module-to-code cross-reference table (continued)

Module number (decimal)	Module number (hexadecimal)	Module	Description
63	3F	DFSTMCD0	CONNECT/DISCONNECT support
71	47	DFSAOSW0	APPC/OTMA SMQ AWE server
72	48	DFSRGFS0	z/OS Resource Recovery Services Server, AWE PROCESSOR
90	5A	DFSXAOS0	DFSXAOS0 APPC/OTMA SMQ Enablement Initialization

#### Related reference

“DFS1959E message information” on page 351

APPC/IMS issues message DFS1959E when a severe internal error occurs.

## APPC/MVS verb-to-code cross-reference table

You can use the verb number, verb name, and verb description to associate the ATB call number in trace records X'7Cxx' with an APPC/MVS verb.

Table 90. APPC/MVS verb-to-code cross-reference table

Verb number (hexadecimal)	Verb name	Verb description
01	ATBALLC	Allocate a conversation
02	ATBCFM	Send a confirmation request
03	ATBCFMD	Send a confirmation reply
04	ATBDEAL	Deallocate a conversation.
05	ATBDFTP	Define TPID
06	ATBFLUS	Empty the local LU's send buffer
07	ATBGTA2	Get conversation attributes
08	ATBGETC	Accept conversation
09	ATBGETP	Get TP properties
0A	ATBGETT	Get conversation type
0B	ATBPTR	Enter receive state
0C	ATBRCVI	Receive data, if available
0D	ATBRCVW	Wait to receive data
0E	ATBRTS	Enter send state
0F	ATBSEND	Send data
10	ATBSERR	Send error
11	ATBASOC	Associate TPID
12	ATBCMAS	Clean address space
13	ATBMIGRP	Join z/OS cross-system coupling facility message group
14	ATBSASA	Set address space attributes

Table 90. APPC/MVS verb-to-code cross-reference table (continued)

Verb number (hexadecimal)	Verb name	Verb description
15	ATBCMTP	Clean TPID
16	ATBCNTL	APPC/MVS control call
17	ATBCONN	Connect address space to scheduler
18	ATBDCON	Disconnect address space from scheduler
19	ATBEXAI	Extract conversation information
1A	ATBIDEN	Identify scheduler to APPC/MVS
1B	ATBUNID	Unidentify scheduler from APPC/MVS
1C	ATBIDN4	Identify scheduler to APPC/MVS
1D	ATBUID4	Unidentify scheduler from APPC/MVS
1E	ATBVERS	Version service
1F	ATBALC5	Allocate a conversation
20	ATBSTO5	Set timeout value
21	ATBLEAVE	Leave XCF message group

## DFS1959E message information

APPC/IMS issues message DFS1959E when a severe internal error occurs.

The message format is:

```
DFS1959E SEVERE IMS INTERNAL FAILURE, REASON CODE=xxyy
```

Variable xx is a decimal number that identifies the module. . Variable yy is an internal reason code.

If you receive this message, contact IBM Software Support with the module number and reason code supplied in the message, and, if requested, output from the LU manager trace.

The following lists provide an explanation of the reason codes listed in the DFS1959E message. Contact IBM Software Support for actions to take in response to these IMS internal failures.

The following two reason codes are module INDEPENDENT. xx denotes the specific IMS module performing the macro call:

### RC

#### Description

#### xx98

Failure in DFSPPOOL to acquire storage for PL/AS variables using the DFSLUMGT macro.

#### xx99

Failure in DFSPPOOL to release storage for PL/AS variables using the DFSLUMRL macro.

### Related reference

[“LU 6.2 module-to-code cross-reference table” on page 348](#)

You can use the module number, module name, and description to associate code xx in message DFS1959E and the module number in trace records X'7Bxx' and X'7Cxx' with a module.

## **DFSALM00**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFSALM00 are given. The DFSALM00 reason code is module dependent.

### **RC**

#### **Description**

##### **0401**

Failure to clear asynchronous control block work pending bit.

##### **0402**

Failure to get LUMP pool buffer using DFSPPOOL macro.

##### **0403**

Failure to free LUMP pool buffer using DFSPPOOL macro.

##### **0408**

Missing LUNAME from LU 6.2 message prefix.

##### **0409**

Missing TPNAME from LU 6.2 message prefix.

##### **0410**

Unsupported sync level specified in asynchronous control block or LU 6.2 message prefix.

##### **0411**

Invalid conversation type specified in asynchronous control block or LU 6.2 message prefix.

##### **0412**

Invalid control data in message segment from GU call.

##### **0413**

Invalid control data in message segment from GN call.

##### **0414**

No data, redundant DFSQMGR Get Next call. RC=4.

##### **0415**

Unknown return code on DFSQMGR Get Next call.

##### **0416**

Missing LU 6.2 prefix on DFSQMGR Get Unique call.

##### **0417**

Queue already in read status on DFSQMGR Get Unique call. RC >= X'C'.

##### **0418**

Failure to dequeue output message. "No message on queue status" is indicated. DFSQMGR Dequeue call, RC=8.

##### **0419**

Unknown return code from dequeue call. DFSQMGR Dequeue call, RC is other than 0 or 8.

##### **0421**

Unknown return code from DFSLIEEO LU 6.2 user edit exit. RC is other than 0, 4, or 8.

## **DFSAOSW0**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFSAOSW0 are given.

### **RC**

#### **Description**

**7101**

Unknown request code.

**7109**

Zero TIB address for send output.

**7110**

Failure in QUERY of DFSXCF macro.

**7116**

Zero header address for send output.

**7121**

Failure to get AWE storage using DFSBCB macro.

**7133**

Transaction not found for notify.

**7134**

Other than transaction found for notify.

**7136**

Wrong message number in SEND DFS MESSAGE function.

**7144**

z/OS cross-system coupling facility parameter length too large.

**7144**

Unknown subfunction for Common XCF Communications.

**7150**

Failure to get LUMP storage using DFSPPOOL macro.

**7190**

Failure in QUERY in DFSXCF macro.

**DFSAPPC0**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFSAPPC0 are given.

**RC****Description****0201**

DFSQMGR Get Unique call failure, RC not 0.

**0202**

DFSQMGR Get Next call failure, RC not 0 and QTP1EOM=0.

**0203**

DFSQMGR Enqueue call failure, RC not 0.

**0204**

DFSQMGR Dequeue call failure, RC not 0.

**0205**

DFSQMGR Insert Move call failure, RC not 0.

**0206**

DFSQMGR Insert Move call failure, RC not 0.

**0207**

DFSQMGR Cancel Input call failure, RC not 0.

**0208**

Failure to read DFSAPPC message from shared queues.

**0209**

DFSQMGR Insert Move without LU62 MSG PREFIX call failure, RC not 0.

**0210**

DFSQMGR Get Next call failure, RC not 0 and QTP1EOM=0.

**0211**

DFSQMGR Get Next call failure, RC not 0.

**0212**

DFSQMGR Get Unique call failure, RC not 0.

**0250**

Failure to find or create asynchronous control block.

**0260**

Router call failure. DFSICLR0 call, RC not 0.

**0270**

DFSUSE FUNC=NOUSE call failure, RC not 0.

**DFSATB00**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFSATB00 are given.

**RC****Description****0901**

Calling module requesting unsupported APPC/MVS verb name.

**DFSCMD00**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFSCMD00 are given.

**RC****Description****0301**

DFSQMGR Get Unique call failure, RC not 0.

**0302**

DFSQMGR Get Next call failure, RC not 0.

**0304**

DFSQMGR Dequeue call failure, RC not 0.

**0306**

DFSQMGR Insert Locate call failure, RC not 0.

**0321**

Failure to get LUMP pool buffer using DFSPPOOL macro.

**0322**

Failure to free LUMP pool buffer using DFSPPOOL macro.

**DFSCMLC0**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFSCMLC0 are given.

**RC****Description****4001**

Failure in LUMIF GU call through DFSCMAP0. Type 6701-MSS1/MSS2 records were logged.



**4002**

Failure in processing a remote keyed message. Type 6701-MSS1/MSS2 records were logged.

**4003**

Failure in an INSERT call. Type 6701-MSS1/MSS2 records were logged.

**4004**

Failure in DFSICLR0 message router. Type 6701-MSS1/MSS2 records were logged.

**4005**

DFSCOND0 was called to process an error scratchpad area segment for a APPC or OTMA client in conversation mode and an error (RC=08) was returned. Type 6701-MSS1/MSS records were logged.

**4006**

Conversation scratch pad (SPA) message did not have the correct SPA message flags in the message prefix MSGMSFL1 and MSGMSFL2 flags. Type 6701-MSS1/MSS2 records were logged.

**4007**

DFSCONM0 was called to process a normal scratch pad segment for a APPC or OTMA client in conversation mode and an error (RC=0C) was returned. Type 6701-MSS1/MSS2 records were logged.

**DFSCMS00**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFSCMS00 are given.

**RC****Description****4101**

Failure in LUMIF GU call using DFSCMAP0.

**4102**

Failure in LUMIF GU call using DFSCMAP0.

**4103**

Failure in LUMIF GU call using DFSCMAP0.

**DFSHCI00**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFSHCI00 are given.

**RC****Description****3501**

Failure to get AWE storage using DFSBCB.

**DFSRLM00**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFSRLM00 are given.

**RC****Description****0501**

AWE extension not a FMH5 Attach request.

**0502**

Synchronous control block creation failure using DFS62DST FUNC=FINDD. This could be storage related.

**0503**

Error freeing XAWE. Unknown storage pool.

**0504**

Error freeing XAWE using STORAGE macro.

**0505**

AWE not an FMH5 Attach request.

**0506**

Error posting DFSRLM10 using DFSSERVER macro.

**0507**

Failure in Identify Protected Conversation Context.

**0508**

Blank or zero LUNAME received

**DFSRLM10**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFSRLM10 are given.

**RC****Description****0601**

Failure in DFS62FD0 releasing a synchronous control block (DFS62DST FUNC=RELEASE).

**0602**

Failure in DFSICLF0 FindDest routine looking up trancode. RC >= X'10'.

**0603**

Failure in DFSRAC60. DFSRAC6 FUNC=RACINIT RC not 0.

**0604**

Failure in DFSRAC60. DFSRAC6 FUNC=FRACHECK RC>= X'44'.

**0605**

Failure in DFSTM0 building a CPI-C dynamic SMB RC not 0.

**0606**

Failure in DFSICLR0 message router. Enqueue to SMB RC not 0.

**0607**

Failure to get LUMP pool buffer using DFSPPOOL macro.

**0608**

Failure to free LUMP pool buffer using DFSPPOOL macro.

**0609**

Failure in DFSQMGR updating message to nonrecoverable RC not 0.

**0610**

Failure in DFSTM0 to ENQ prefix to CPIC dynamic SMB RC not 0.

**0611**

Failure in DFSQMGR to insert Data for SMB or DFSAPPC DFSQMGR Insert Move call failure, RC not 0.

**0612**

Failure in DFSCMD00 processing IMS command. RC not 0.

**0613**

Failure in DFSAPPC0 processing Message Switch RC not 0.

**0614**

Failure in DFSQMGR to cancel a message in progress. RC not 0.

**0615**

Failure in DFSQMGR to enqueue message for Cmd or DFSAPPC. RC not 0.

**0616**

Failure in DFSQMGR to update APPC Message Prefix. RC not 0.

**0617**

Failure in DFSHEILO unrecognized return code from Fast Path RC other than 0, 4, 8, or 12.

**0618**

Failure in DFSBCB to free AWE.

**0619**

Failure in DFS6LUS0 RLUM reposted and not running conversational transaction.

**0620**

Failure in DFSQMGR to update modname RC not 0.

**0621**

Failure in DFSQMGR to update a message to response mode.

**0622**

Failure in DFSQMGR to cancel a message, RC not 0.

**0623**

Failure in DFSQMGR to delete a message, RC not 0.

**0624**

Failure in DFS62FD0 getting an asynchronous control block (DFS62DST FUNC=FINDD).

**DFSSLUM0**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFSSLUM0 are given.

**RC****Description****0101**

Failure in DFSQMGR Get Unique or GN call. RC not 0 and QTP1EOM=0.

**0103**

Failure in DFSQMGR Dequeue or Cancel call. RC not 0.

**0107**

Failure to get AWE using DFSBCB macro.

**0108**

QMGR GU call failed with RC08.

**0111**

DFSSLUM0 has been called to deliver a message with zero length to the front-end IMS system. A DFS2224 message will be sent instead.

**0121**

Failure to get LUMP pool buffer using DFSPPOOL macro.

**0122**

Failure to free LUMP pool buffer using DFSPPOOL macro.

**DFS6CKP0**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFS6CKP0 are given.

**RC****Description****2201**

Invalid checkpoint type specified in parameter list. Should be ALL or STATUS.

**2202**

Data block too large for log record.

**DFS6ECT0**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFS6ECT0 are given.

**RC****Description****3101**

Error freeing XAWE using DFSBCB macro.

**3102**

Error freeing XAWE using STORAGE macro.

**3104**

Invalid AWE request.

**3105**

Failure in DFSTM0 to connect all dependent regions FUNC=CONALL.

**3107**

Failure in DFSBCB to get AWE storage

**3109**

Error detected in DFS6IDC0 building user descriptors.

**3110**

Error getting CIOP storage using DFSPPOOL macro.

**3111**

Error freeing CIOP storage using DFSPPOOL macro.

**3112**

VTAM MODIFY USERVAR failed during activation of XRF alternate.

**3113**

VTAM VARY NET TERM failed for termination of primary system.

**3114**

Error Posting asynchronous control block using DFSSERVER macro.

**3115**

Error Checking synchronous control block using DFSSERVER macro.

**3116**

VTAM MODIFY USERVAR failed for activation of primary system.

**DFS6IDC0**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFS6IDC0 are given.

**RC****Description****2401**

Unable to obtain storage for BPAM buffer using STORAGE macro.

**2402**

Unable to release storage for BPAM buffer using STORAGE macro.

**2403**

Unknown DFS warning message number.

**2404**

Failure to get LUMP pool buffer using DFSPPOOL macro.

When this reason code is issued, the error message is followed by a U0732 abend.

**2405**

Failure to free LUMP pool buffer using DFSPPOOL macro.

**DFS6LUS0**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFS6LUS0 are given.

**RC****Description****1004**

No synchronous control block given in SEND service call.

**1007**

TIB was released while the task was waiting to synchronize.

**1008**

TIB\_SYNC\_PTR was changed, but not to zero.

**1010**

Unknown service call in main program.

**1012**

Unable to get storage for LU 6.2 message prefix using DFSBCB macro.

**1013**

Unable to create an asynchronous control block using DFS62DST FUNC=FUNC.

**1015**

No LUM block given in BLDPRE service call.

**1016**

Unable to find asynchronous control block or create a new one in CHNG service call. DFS62DST FUNC(FUNC).

**1018**

Conversation-id zero at send time.

**1020**

Return Code X'1C' from Queue Manager Get Unique call.

**1022**

Unable to free storage for LU 6.2 message prefix using DFSBCB macro.

**1027**

Expect input LU 6.2 msg prefix in COPYPF62 service call.

**1029**

Expect input synchronous/asynchronous control block in COPYPF62 service call.

**1031**

Invalid TPN=DFSSIDE in CHNG service call.

**1032**

Unable to find LU 6.2 descriptor entry in BLDPRE service call using DFS62DST macro.

**1060**

Failure in DFSBCB to get AWE.

**1061**

Failure in DFSBCB to free AWE.

**1062**

Failure to get LUMP pool buffer using DFSPPOOL macro.

**1063**

Failure in SENDMSG using DFSXCF macro.

**1064**

Failure to free LUMP pool buffer using DFSPPOOL macro.

**DFS6LUS1**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFS6LUS1 are given.

**RC****Description****1110**

Unknown service call in main program.

**1117**

No message prefix or synchronous/asynchronous control block given in INQY service call.

**1123**

Unable to clean up TP.

**1124**

Unable to clean up in the address space.

**1125**

No synchronous control block is given in TIBINFO service call

**1126**

Unable to find the asynchronous or restart synchronous control block in GETQABTIB service call.

**1127**

DFS6LUS1 cannot find TIB/QAB.

**1130**

Unable to post RLM back in CONVCONT service call.

**1133**

Unable to find LU 6.2 descriptor entry in INQY service call.

**1134**

No message prefix supplied in GETQABTIB service call.

**1140**

DFSQMGR Get Unique or Insert Move call failed in MSGROUTE service call.

**1142**

Unable to find or to create a synchronous control block in FPGU service call.

**1143**

Unable to free a synchronous control block (DFS62DST FUNC=RELEASE).

**1150**

Return code from ATBRCVW in PH1 service call (abort synchpoint).

**1151**

Return code from ATBGTA2 in PH1 service call (abort synchpoint).

**DFS6LUS2**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFS6LUS2 are given.

**RC****Description**

**1201**

No PCB given in READSQ service.

**1202**

No control block given in READSQ service.

**1203**

Invalid control block type in READSQ service.

**1204**

DFSQMGR Get Unique failure in READSQ service.

**1205**

DFSQMGR Enqueue failure in READSQ service.

**1206**

DFSQMGR Dequeue failure in READSQ service.

**1207**

Failure to get LUMP pool buffer using DFSPPOOL macro.

**1208**

Failure to free LUMP pool buffer using DFSPPOOL macro.

**1209**

Failure to get MSEB storage using DFSBCB macro.

**1210**

Failure to free MSEB storage using DFSBCB macro.

**1211**

Failure to get HIOP storage using DFSPPOOL macro.

**1212**

Failure to free HIOP storage using DFSPPOOL macro.

**1224**

CQS not available in READSQ service.

**DFS6QFX0**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFS6QFX0 are given.

**RC****Description****3601**

Failure in creating a restart control block.

**3602**

Failure in DFSCIR to create restart ITASK.

**3603**

Failure in IXCTL to run under restart ITASK.

**3604**

Failure in DFSCIR to delete restart ITASK.

**3682**

Issue /STO APPC if APPC/IMS was started; then issue /STA APPC.

## DFS6RST0

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFS6RST0 are given.

### RC

#### Description

#### 2101

Log record type not X'22', X'23', or X'24'.

#### 2102

Log record code not X'40'.

## DFS62FD0

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFS62FD0 are given.

### RC

#### Description

#### 3201

Failure in DFSBCB to release LU block.

#### 3202

Failure in DFSBCB to release asynchronous control block.

#### 3203

Failure in DFSBCB to get asynchronous control block.

#### 3204

Failure in DFSBCB to release asynchronous control block. (Second location within module.)

#### 3205

Failure in DFSTCBTB FUNC=LOCATE.

#### 3206

Failure in DFSCIR to create ITASK.

#### 3207

Failure in DFSBCB to get synchronous control block.

#### 3208

Failure in DFSCIR to delete ITASK for asynchronous message.

#### 3209

Failure in DFSCIR FUNC=DTASK to release duplicate ITASK for asynchronous message.

#### 3210

Synchronous control block to be released not found in chain.

#### 3211

Input parameter list is invalid, unknown type.

#### 3213

DFSCS failed adding synchronous control block to chain.

#### 3216

IMODULE DELETE failed while releasing asynchronous control block.

#### 3217

Blank LUNAME or nonblank SIDENAME with TPNAME='DFSSIDE'.

#### 3220

Invalid parameters on module entry.

#### 3221

Invalid parameters on module entry.



**3222**

Failure to free HIOP storage using DFSPOOL macro.

**3223**

Failure to free HIOP storage using DFSPOOL macro.

**3224**

Failure to free MSEB storage using DFSBCB macro.

**DFS62FD1**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFS62FD1 are given.

**RC****Description****3801**

Input parameter list is invalid, unknown type.

**3802**

Failure in DFSBCB FUNC=GET to get LU block.

**3803**

Failure in DFSBCB FUNC=REL to release LU block.

**3804**

Failure in DFSBCB FUNC=GET to get descriptor.

**3805**

Failure in DFSCS for inserting descriptor into table.

**3806**

IMODULE DELETE failed for delete of restart synchronous control block hash table.

**3807**

Failure in DFSBCB FUNC=GET to get synchronous control block.

**3808**

Failure in DFSBCB FUNC=REL to release restart asynchronous control block.

**DFSLUM00**

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFSLUM00 are given.

**RC****Description****5101**

Failure in DFSQMGR Get Unique for notify message.

**5102**

Failure in DFS62FD0 finding an asynchronous control block for notify message.

**5109**

Unknown return code from z/OS clean address space call.

**5110**

Unknown return code from z/OS unidentify call.

**5111**

IXCLEAVE unsuccessful.

## DFSHAV70

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFSHAV70 are given.

### RC

#### Description

#### 3709

Unknown return code from z/OS clean address space call.

#### 3710

Unknown return code from z/OS unidentify call.

#### 3711

IXCLEAVE unsuccessful.

## DFSXLUM0

APPC/IMS issues message DFS1959E when a severe internal error occurs. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message. Reason codes for message DFSXLUM0 are given.

### RC

#### Description

#### 5009

Unknown return code from z/OS clean address space call.

#### 5010

Unknown return code from z/OS unidentify call.

#### 5011

IXCLEAVE unsuccessful.

## DFS1965E APPC/MVS call failures

The APPC/IMS diagnostic aids include DFS1965E APPC/MVS call failures. If you receive this message, contact IBM® Software Support with the module number and reason code supplied in the message.

A call to APPC/MVS had an unexpected return code. The call for FUNCTION=aaaaaaa was issued, and a return code xx from APPC/MVS was the result. Return code xx denotes the specific IMS module performing the APPC call. Error return codes that represent anticipated conditions are handled by IMS, and do not result in this message. This message is produced when an unexpected result is encountered, which might represent an abnormal condition in some system component.

### RC

#### Description

#### xx90

Synchronous call failure

#### xx91

Asynchronous call failure

## Diagnostics for use with synchronous APPC and OTMA with shared queues

Synchronous APPC and OTMA message processing in the shared queues environment introduces additional diagnostic considerations for the message flow.

In addition to the APPC and OTMA traces already used, other facilities include:

- IMS Resource Recovery Trace
- z/OS Resource Recovery Trace
- z/OS APPC Trace

- Console memory dumps of the z/OS Resource Recovery Services and APPC address and data spaces.

If IMS is using z/OS cross-system coupling facility for communication, a memory dump of the RRS address space is unavailable.

#### Related tasks

[“Resource Recovery Services trace” on page 611](#)

The Resource Recovery Services trace (RRST) provides information about relevant z/OS Resource Recovery Services (RRS) events in the IMS dependent region. Use the trace under direction of IBM Software Support when problems are suspected in the RRS area.

## SNAPs and dumps

For errors that do not result in an abend, IMS writes a X'67D0' log record or produces an SDUMP, depending on the error. The minimum data dumped for LU 6.2 problems are the control blocks that are associated with the task in error and the appropriate trace tables.

## OTMA diagnostic aids

OTMA diagnostic aids include OTMA trace, module-to-code cross references, verb-to-code cross references, DFS1296E message information, log records, and SNAPs and dumps.

#### About this task

### OTMA trace

The OTMA trace records the flow of control through IMS OTMA. Turn on the OTMA trace only if the IBM support representative requests it.

#### About this task

### Starting the OTMA trace

The **/TRACE SET ON TABLE OTMT** command activates the trace and sends the entries to an internal table.

You can format the table using the Offline Dump Formatter under IPCS, using either the VERBX command or the IMS Dump Formatter panels.

If a SNAP dump is taken, the table is formatted as part of the IMS dump. If you add the **OPTION LOG** parameter to the **/TRACE** command, IMS sends the output to an external data set. You can use the File Select and Format utility (DFSERA10) with exit routine DFSERA60 to format trace entries.

#### Related concepts

[“Formatting IMS dumps offline” on page 512](#)

Two methods are available for formatting IMS dumps offline: interactive formatting, performed through a series of panels which provide formatting choices, and formatting by using JCL.

### Format of OTMA trace records

Each record of the OTMA trace record format is eight words long, and word 0 holds standard information.

*Table 91. OTMA trace record format*

WORD 0		WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7
ID	SEQ NUM							

**where**  
**represents**

**ID**  
2-byte trace ID

**SEQ NUM**  
2-byte trace sequence number assigned by the IMS trace component

Words 1 through 7 contain data specific to each trace entry, as described below:

Trace ID = X'5A01' OTMA module entry

**Word 1**  
Byte 0: Module number  
Bytes 1-3: Reserved

**Word 2**  
A(ECB)

**Word 3**  
Register 1

**Words 4-5**  
Optional user data

**Words 6-7**  
Time stamp (STCK)

TRACE ID = X'5A02' OTMA module exit

**Word 1**  
Byte 0: Module number  
Bytes 1-3: Reserved

**Word 2**  
A(ECB)

**Word 3**  
Return code

**Words 4-5**  
Optional user data

**Words 6-7**  
Time stamp (STCK)

TRACE ID = X'5A03' IMS internal OTMA error

**Word 1**  
Byte 0: Module number  
Bytes 1-3: 0

**Word 2**  
A(ECB)

**Word 3**  
Error code

**Word 4**  
Optional user data

**Word 5**  
0

**Words 6-7**  
Time stamp (STCK)

TRACE ID = X'5B01' XCF/z/OS entry

**Word 1**

Byte 0: Module number

Byte 1: XCF call number

**Words 2-7**

Control message

TRACE ID = X'5B02' XCF/z/OS exit

**Word 1**

Byte 0: Module number

Byte 1: XCF call number

**Word 2**

A(ECB)

**Word 3-4**

XCF token

**Word 5**

Return code

**Word 6**

Reason code

**Word 7**

Time stamp (short)

TRACE ID = X'5CX' OTMA AWE function

**Word 1**

Byte 0: Module number

**Words 2-6**

Reserved

**Word 7**

Time stamp (short)

TRACE ID = X'5C71' OTMA DFSYPSIO input trace entry

**Word 1**

Byte 0: module number X'25'

Byte 1:

**X'01'**

an input transaction with reroute name specified

**X'02'**

a NAK with reroute request

**X'03'**

a NAK with purge request

Byte 2-3: 0

**Word 2**

Addr(ECB)

**Word 3**

Addr(YQAB) if byte 1 of word 1 is X'01'. Otherwise, it will be Addr(YTQAB).

**Word 4**

0

**Word 5**

Bytes 0-3 of reroute tpipe name

**Word 6**

Bytes 4-7 of reroute tpipe name

**Word 7**

Time stamp (short)

TRACE ID = X'5C72' OTMA DFSYQAB0 output trace entry

**Word 1**

Byte 0: module number X'29'

Byte 1: X'03' reroute on SendOnly output

Byte 2-3: 0

**Word 2**

Addr(ECB)

**Word 3**

Addr(YQAB)

**Word 4**

0

**Word 5**

Bytes 0-3 of reroute tpipe name

**Word 6**

Bytes 4-7 of reroute tpipe name

**Word 7**

Time stamp (short)

TRACE ID = X'5D01' OTMA Find Tpipe or scan Tpipe was invoked

**Word 1**

Return code

**Word 2**

Location code

**Word 3 - 4**

8-byte characters that could be TrcTPIPE, NO TPIPE, or tpipe name.

**Word 5**

0 or member block address

**Word 6-7**

Time stamp (STCK)

TRACE ID = X'5D02' OTMA Find YTIB or scan YTIB was invoked

**Word 1**

Return code or YTIB address

**Word 2**

Location code or TPIPE address

**Word 3 - 4**

8-byte characters that could be NO YTIB or tpipe name

**Word 5**

0 or member block address

**Word 6-7**

Time stamp (STCK)

TRACE ID = X'5D03' Find YQAB or scan YQAB was invoked

**Word 1**

Return code or YQAB address

**Word 2**

Location code or TPIPE address

**Word 3 - 4**

8-byte characters that could be NO YQAB or tpipe name

**Word 5**

0 or member block address

**Word 6-7**

Time stamp (STCK)

## OTMA trace entries for parallel RESUME TPIPE request processing (X'5A20')

OTMA trace entries for parallel RESUME TPIPE request processing have the trace ID X'5A20' in byte 0 and 1 of word 0.

### ID X'23' - Trace entries for the DFSYMEM0 module

*Table 92. Trace entry for the activation or deletion of a RESUME TPIPE YQAB (MQAB) for a RESUME TPIPE request*

Word	Byte	Contents
Word 1	Byte 0	Module ID X'23'
	Byte 1	Location ID: <b>X'20'</b> DFSYMEM0 activated a RESUME TPIPE YQAB (MQAB) for a RESUME TPIPE request. <b>X'2D'</b> DFSYMEM0 deleted a resume tpipe YQAB (MQAB) for a resume tpipe request.
	Byte 2	0
	Byte 3	0
Words 2 - 3		YQAB token (MQAB token) for a RESUME TPIPE request
Words 4 - 5		RESUME TPIPE token
Words 6 - 7		STCK

*Table 93. Trace entry for queuing a RESUME TPIPE request to a tpipe*

Word	Byte	Contents
Word 1	Byte 0	Module ID X'23'
	Byte 1	Location ID: <b>X'21'</b> The DFSYMEM0 module queued a RESUME TPIPE request to the tpipe.
	Byte 2	0
	Byte 3	0
Words 2 - 3		YQAB token for a hold queue
Words 4 - 5		RESUME TPIPE token
Words 6 - 7		STCK

*Table 94. Trace entry for processing the exit of a client from the XCF group*

<b>Word</b>	<b>Byte</b>	<b>Contents</b>
Word 1	Byte 0	Module ID X'23'
	Byte 1	Location ID: <b>X'22'</b> DFSYMEM0 processed the XCF leave for the client that is identified in words 2 - 5.
	Byte 2	0
	Byte 3	0
Words 2 - 5		The name of the OTMA client member that left the XCF group
Words 6 - 7		STCK

*Table 95. Trace entry for a super member that processes the exit of a client from the XCF group*

<b>Word</b>	<b>Byte</b>	<b>Contents</b>
Word 1	Byte 0	Module ID X'23'
	Byte 1	Location ID: <b>X'23'</b> DFSYMEM0, which is a super member, processed the exit from the XCF group of the client that is named in words 4 - 5.
	Byte 2	0
	Byte 3	0
Words 2 - 3		The name of the super member.
Words 4 - 5		The first eight bytes of the client member name.
Words 6 - 7		STCK



*Table 96. Trace entry for a change in the number of inactive RESUME TPIPE requests that are queued to the tpipe*

Word	Byte	Contents
Word 1	Byte 0	Module ID X'23'
	Byte 1	Location ID: <b>X'24'</b> The number of inactive RESUME TPIPE requests that are queued to the tpipe before cleanup logic is shown in word 3. <b>X'25'</b> The number of inactive RESUME TPIPE requests that are queued to the tpipe after cleanup logic is shown in word 3. <b>X'28'</b> An MQAB was reused for a queued RESUME TPIPE request. The total count of MQABs that are associated with the tpipe hold queue after cleanup logic is shown in word 3.
	Byte 2	0
	Byte 3	0
Word 2		0
Word 3		<b>YQAB_RT_QUEUE_CT</b> The number of number of inactive RESUME TPIPE requests that are currently queued to the tpipe.
Words 4 - 5		The hold queue YQAB token.
Words 6 - 7		STCK

*Table 97. Trace entry for the number of active RESUME TPIPE requests (MQABs) on the tpipe*

Word	Byte	Contents
Word 1	Byte 0	Module ID X'23'
	Byte 1	Location ID: <b>X'26'</b> The number of active RESUME TPIPE YQABs (MQABs) for the tpipe before cleanup processing is shown in word 3. <b>X'27'</b> The number of active RESUME TPIPE YQABs (MQABs) for the tpipe after cleanup processing is shown in word 3.
	Byte 2	0
	Byte 3	0

Table 97. Trace entry for the number of active RESUME TPIPE requests (MQABs) on the tpipe (continued)

Word	Byte	Contents
Word 2		<b>TPIPE_MQAB_CT</b> The total number of MQABs, including disabled MQABs. :
Word 3		<b>TPIPE_MQAB_DCT</b> The total number of disabled MQABs.
Words 4 - 5		The YQAB token of the tpipe hold queue.
Words 6 - 7		STCK

Table 98. Trace entry for the number of MQABs that a tpipe hold queue

Word	Byte	Contents
Word 1	Byte 0	Module ID X'23'
	Byte 1	Location ID: <b>X'2A'</b> A cancel resume TPIPE request was received from a client. The associated MQAB is deleted. <b>X'2B'</b> A cancel resume TPIPE request was initiated by DFSYQAB0 upon the completion of a resume TPIPE request. The associated MQAB is deleted.
	Byte 2	0
	Byte 3	0
Words 2 - 3		The name of the tpipe.
Words 4 - 5		The RESUME TPIPE token
Words 6 - 7		STCK

Table 99. Trace entry for an attempt to delete the YQAB of an active RESUME TPIPE request

Word	Byte	Contents
Word 1	Byte 0	Module ID X'23'
	Byte 1	Location ID: <b>X'2C'</b> An attempt was made to delete this RESUME TPIPE YQAB, but it could not be deleted because the RESUME TPIPE is active.
	Byte 2	0
	Byte 3	0
Word 2		0
Word 3		<b>TPIPE_MQAB_DCT</b> The total number of disabled MQABs.

Table 99. Trace entry for an attempt to delete the YQAB of an active RESUME TPIPE request (continued)

Word	Byte	Contents
Words 4 - 5		The RESUME TPIPE token.
Words 6 - 7		STCK

Table 100. Trace entry for when the number of active RESUME TPIPE requests reaches the LIMITRTP value

Word	Byte	Contents
Word 1	Byte 0	Module ID X'23'
	Byte 1	Location ID: <b>X'30'</b> The number of active RESUME TPIPE requests reached the LIMITRTP value that is currently in effect for this tpipe.
	Byte 2	0
	Byte 3	0
Word 2		0
Word 3		<b>MCB_MQAB_LIMIT</b> The LIMITRTP value in effect for this tpipe.
Words 4 - 5		The tpipe name.
Words 6 - 7		STCK

## ID X'29' - Trace entries for the DFSYQAB0 module

Table 101. Trace entry for the activation of a RESUME TPIPE YQAB (MQAB)

Word	Byte	Contents
Word 1	Byte 0	Module ID X'29'
	Byte 1	Location ID: <b>X'01'</b> A RESUME TPIPE YQAB (MQAB) was activated
	Byte 2	0
	Byte 3	0
Words 2 - 3		Resume TPIPE YQAB token
Words 4 - 5		The name of the tpipe.
Words 6 - 7		STCK

*Table 102. Trace entry for the processing of a synchronous callout request by a RESUME TPIPE YQAB (MQAB)*

Word	Byte	Contents
Word 1	Byte 0	Module ID X'29'
	Byte 1	Location ID: <b>X'02'</b> A RESUME TPIPEYQAB (MQAB) processed a synchronous callout request
	Byte 2	0
	Byte 3	0
Words 2 - 3		Resume TPIPE YQAB token
Words 4 - 5		AWE token for the synchronous callout request
Words 6 - 7		STCK

*Table 103. Trace entry for RESUME TPIPE YQAB(MQAB) processing*

Word	Byte	Contents
Word 1	Byte 0	Module ID X'29'
	Byte 1	Location ID: <b>X'03'</b> A synchronous callout request was processed by the RESUME TPIPErequest that is identified by the token in words 4 and 5. <b>X'04'</b> An asynchronous callout request was processed by the RESUME TPIPErequest that is identified by the token in words 4 and 5. <b>X'05'</b> A NAK was received for a RESUME TPIPE YQAB (MQAB). <b>X'06'</b> ACK was received for a RESUME TPIPE YQAB (MQAB). <b>X'07'</b> A request to delete a RESUME TPIPE YQAB (MQAB).
	Byte 2	0
	Byte 3	0
Words 2 - 3		Resume TPIPE YQAB token
Words 4 - 5		RESUME TPIPE token
Words 6 - 7		STCK

## OTMA trace entry for user exits

Trace IDs and their words are listed for the user exits.

### OTMA trace entry for TRACE ID = X'5A05' (OTMAIOED user exit entry)

Table 104. Contents of words 1 - 7 for X'5A05' trace entries in OTMAIOED user exit entry

Word		Contents
Word 1	Byte 0	Module number X'33'
	Byte 1-3	0
Word 2		A(ECB)
Word 3		0
Word 4		0
Word 5		0
Word 6-7		Time stamp (STCK)

### OTMA trace entry for TRACE ID = X'5A06' (OTMAIOED user exit entry)

Table 105. Contents of words 1 - 7 for X'5A06' trace entries in OTMAIOED user exit entry

Word		Contents
Word 1	Byte 0	Module number X'33'
	Byte 1-3	0
Word 2		A(ECB)
Word 3		Exit RC set by the module
Word 4		0
Word 5		0
Word 6-7		Time stamp (STCK)

### OTMA trace entry for TRACE ID = X'5A07' (OTMAYPRX user exit entry)

Table 106. Contents of words 1 - 7 for X'5A07' trace entries in OTMAYPRX user exit entry

Word		Contents
Word 1	Byte 0	Module number X'31'
	Byte 1-3	0
Word 2		A(ECB)
Word 3		0
Word 4		0
Word 5		0
Word 6-7		Time stamp (STCK)

**OTMA trace entry for TRACE ID = X'5A08' (OTMAYPRX user exit entry)***Table 107. Contents of words 1 - 7 for X'5A08' trace entries in OTMAYPRX user exit entry*

Word		Contents
Word 1	Byte 0	Module number X'31'
	Byte 1-3	0
Word 2		A(ECB)
Word 3		Exit RC set by the module
Word 4		0
Word 5		IMS internal processing code
Word 6-7		Time stamp (STCK)

**OTMA trace entry for TRACE ID = X'5A09' (user exit DFSYDRU0 module entry)***Table 108. Contents of words 1 - 7 for X'5A09' trace entries in user exit DFSYDRU0 module entry*

Word		Contents
Word 1	Byte 0	Module number X'32'
	Byte 1-3	0
Word 2		A(ECB)
Word 3		0
Word 4		0
Word 5		0
Word 6-7		Time stamp (STCK)

**OTMA trace entry for TRACE ID = X'5A0A' (user exit DFSYDRU0 module exit)***Table 109. Contents of words 1 - 7 for X'5A0A' trace entries in user exit DFSYDRU0 module exit*

Word		Contents
Word 1	Byte 0	Module number X'32'
	Byte 1-3	0
Word 2		A(ECB)
Word 3		Exit RC set by the module
Word 4		0
Word 5		IMS internal processing code
Word 6-7		Time stamp (STCK)

**OTMA trace entry for TRACE ID = X'5A0D' (OTMA ALTPCB in DFSYFND0)***Table 110. Contents of words 1 - 7 for X'5A0D' trace entries in OTMA ALTPCB in DFSYFND0*

<b>Word</b>		<b>Contents</b>
Word 1	Byte 0	Module number X'1F'
	Byte 1	<b>X'01'</b> Input transaction from legacy <b>X'02'</b> Input transaction from OTMA <b>X'03'</b> Match descriptor entry found <b>X'04'</b> Use descriptor, ignore any exits <b>X'05'</b> No matching descriptor found <b>X'06'</b> Descriptor with EXIT=YES found <b>X'07'</b> No DFSYPRX0 exist <b>X'08'</b> Take OTMA finddest processing <b>X'09'</b> Take legacy finddest processing <b>X'10'</b> Error finddest processing <b>X'11'</b> No DRU0 exit found <b>X'12'</b> DRU0 exit found <b>X'13'</b> DRU0 exit sets legacy destination <b>X'14'</b> DRU0 exit sets OTMA destination <b>X'15'</b> DRU0 exit sets RC=101 <b>X'16'</b> DRU0 sets RC=8 <b>X'17'</b> DRU0 sets RC=100 <b>X'18'</b> DRU0 error processing <b>X'19'</b> Destination TPIPE set by descriptor <b>X'20'</b> Destination TPIPE set by DRU exit <b>X'21'</b> User data exists in OTMA input

Table 110. Contents of words 1 - 7 for X'5A0D' trace entries in OTMA ALTPCB in DFSYFND0 (continued)

Word		Contents
Word 1 (cont'd)	Byte 1	<b>X'22'</b> No user data input to DRU0 exit
		<b>X'23'</b> DRU0 sets user data for RC=0 or RC=101
		<b>X'24'</b> DRU0 does not set user data for RC=0 or RC=101
		<b>X'25'</b> Prepare ICON user data based on descriptor
		<b>X'26'</b> Prepare MQ user data based on descriptor
	Byte 2-3	0
Word 2-3		0
Word 4-5		Original CHNG call value
Word 6-7		Time stamp (STCK)

## OTMA trace entry for synchronous callout

OTMA trace entries for synchronous callout have the trace ID X'5A04'.

### Trace entries in module DFSYSCP0

Byte 0 of word 1 is the module identifier X'10'. Byte 1 of word 1 is a location ID that identifies the trace event.



Table 111. Contents of words 1 - 7 for trace entries in module DFSYSCPO

Word	Byte	Contents
Word 1	Byte 0	Module ID: X'10'
	Byte 1	Location ID:
		<b>X'11'</b> Ready to chain the ICAL AWE to DFSYQAB0
		<b>X'12'</b> Failed to activate DFSYQAB0
		<b>X'13'</b> Post back by DFSYSCS0 for ICAL
		<b>X'14'</b> Post back by DFSYSCS0 for ICAL
		<b>X'15'</b> ICAL was rejected by an IMS command
		<b>X'16'</b> Processing a response
		<b>X'17'</b> Processing a timeout
		<b>X'18'</b> Processing an invalid post from DFSYSCS0
		<b>X'19'</b> OTMA ACK timeout occurred
	Byte 2	PSTFLAG2
	Byte 3	0
Word 2		PST address of the ICAL
Word 3		YQAB address
Word 4		AWORNAME (1:4) <b>Note:</b> This identifier is the unique ICAL ID
Word 5		AWORNAME (5:8) <b>Note:</b> This identifier is the unique ICAL ID
Word 6		First half of STCK
Word 7		Second half of STCK

### Trace entries in module DFSYQAB0

Byte 0 of word 1 is the module identifier X'29'. Byte 1 of word 1 is a location ID that identifies the trace event.

Table 112. Contents of words 1 - 7 for trace entries in module DFSYQAB0

Word		Contents
Word 1	Byte 0	Module ID: X'29'
	Byte 1	Location ID: <b>X'11'</b> Pre-call to the Edit_N_Send for ICAL <b>X'12'</b> Waking up to process an ACK or NAK <b>X'13'</b> Late ACK received after response <b>X'14'</b> Getting an ACK for ICAL <b>X'15'</b> Getting a NAK for ICAL
	Byte 2	PSTFLAG2
	Byte 3	0
Word 2		State of this ICAL: AWOSTATE
Word 3		YQAB address
Word 4		AWORNAME (1:4) <b>Note:</b> This identifier is the unique ICAL ID
Word 5		AWORNAME (5:8) <b>Note:</b> This identifier is the unique ICAL ID
Word 6		First half of STCK
Word 7		Second half of STCK

### Trace entries in module DFSYMEM0

Trace entries in this module have the module ID X'23' and are organized in one of two formats. If the location identifier is X'10', words 2 - 5 are 0 as shown in the following table.

Table 113. Contents of words 1 - 7 for trace entries in module DFSYMEM0 for location ID X'10'

Word		Contents
Word 1	Byte 0	Module ID: X'23'
	Byte 1	Location ID: X'10'
		<b>X'10'</b>
	Byte 2	PSTFLAG2
	Byte 3	0
Word 2		0
Word 3		0
Word 4		0
Word 5		0

Table 113. Contents of words 1 - 7 for trace entries in module DFSYMEM0 for location ID X'10' (continued)

Word	Contents
Word 6	First half of STCK
Word 7	Second half of STCK

The following table shows the trace entry format for all other location identifiers in module DFSYMEM0.

Table 114. Contents of words 1 - 7 for trace entries in module DFSYMEM0 for location IDs other than X'10'

Word	Contents
Word 1	Byte 0 Module ID: X'23'
	Byte 1 Location ID: <b>X'11'</b> Getting a response message from the client <b>X'12'</b> An invalid AWE state was detected for ICAL <b>X'13'</b> An internal post error was reported by DFSYSCS0 <b>X'14'</b> The AWE of the ICAL has an unknown state
	Byte 2 PSTFLAG2
	Byte 3 0
Word 2	State of this ICAL: AWOSTATE
Word 3	YQAB address
Word 4	AWORNAME (1:4) <b>Note:</b> This identifier is the unique ICAL ID
Word 5	AWORNAME (5:8) <b>Note:</b> This identifier is the unique ICAL ID
Word 6	First half of STCK
Word 7	Second half of STCK

### Trace entries in module DFSYMOM0

Byte 0 of word 1 is the module identifier X'22'. Byte 1 of word 1 is a location ID that identifies the trace event.

Table 115. Contents of words 1 - 7 for trace entries in module DFSYMOM0

Word	Contents	
Word 1	Byte 0	Module ID: X'22'
	Byte 1	Location ID: <b>X'11'</b> The ICAL has timed out
	Byte 2	PSTFLAG2
	Byte 3	0
Word 2	State of this ICAL: AWOSTATE	
Word 3	YQAB address	
Word 4	AWORNAME (1:4)	
	<b>Note:</b> This identifier is the unique ICAL ID	
Word 5	AWORNAME (5:8)	
	<b>Note:</b> This identifier is the unique ICAL ID	
Word 6	First half of STCK	
Word 7	Second half of STCK	

#### Related concepts

[Synchronous callout requests \(Communications and Connections\)](#)

# OTMA trace entry for synchronous program switch

OTMA trace entries for synchronous program switch have the trace ID X'5A0B'.

Table 116. Contents of trace entry X'5A0B' for module DFSYSCP0

Word	Byte	Contents
Word 1	Byte 0	Module ID X'10'
	Byte 1	Location ID: <b>X'21'</b> Post DFSYTIB0 to process the synchronous program switch. Word 3 contains the YTIB CLB address. <b>X'30'</b> Received a NACK from DFSYTIB0. Word 2 contains the return code to DFSDLA40. Word 3 contains the extended return code. <b>X'31'</b> Response returned. Word 2 contains the return code to DFSDLA40. Word 3 contains the extended return code. <b>X'32'</b> Target application abend. Word 2 contains the return code to DFSDLA40. Word 3 contains the extended return code. <b>X'33'</b> A CM0 application abend was detected. Word 3 contains the extended return code. <b>X'34'</b> A CM0 response was detected. Word 3 contains the extended return code. This location is used when REPLYCHK=NO and there are multiple responses to the ICAL request. <b>X'35'</b> The ICAL was canceled by an IMS command. Word 2 has the return code to DFSDLA40. Word 3 has the extended return code.

Table 116. Contents of trace entry X'5A0B' for module DFSYSCP0 (continued)

Word	Byte	Contents
Word 1 (continued)	Byte 1 (continued)	<b>X'36'</b> Timeout occurred. Word 2 has the return code to DFSDLA40. Word 3 has the extended return code.
		<b>X'41'</b> Bad transaction code, CPIC transaction, or conversational transaction. Word 2 has the return code to DFSDLA40. Word 3 has the extended return code.
		<b>X'42'</b> Security failed in APPC RACINIT. Word 2 has the return code to DFSDLA40. Word 3 has the extended return code.
		<b>X'43'</b> Security failed in OTMA RACINIT. Word 2 has the return code to DFSDLA40. Word 3 has the extended return code.
		<b>X'44'</b> Security failed in DFSDAUT0. Word 2 has the return code to DFSDLA40. Word 3 has the extended return code.
	Byte 2	PSTFLAG2
	Byte 3	0
Word 2		PST address of the ICAL that initiated the synchronous program switch, or the return code to DFSDLA40.
Word 3		Extended return code, or the YTIB CLB address of the ICAL.
Word 4		YTIB token (1:4) <b>Note:</b> This identifier is the unique ICAL ID
Word 5		YTIB token (5:8) <b>Note:</b> This identifier is the unique ICAL ID
Word 6		First half of STCK
Word 7		Second half of STCK

Table 117. Contents of trace entry X'5A0B' for module DFSYTIB0

Word	Byte	Contents
Word 1	Byte 0	Module ID X'28'
	Byte 1	Location ID: <b>X'21'</b> SPS shared queues AOS=X enqueue.
	Byte 2	YTIB_SPS_STATUS
	Byte 3	0
Word 2		YTIB_SPS_Flags
Word 3		YTIB CLB address of the ICAL
Word 4		YTIB token (1:4) <b>Note:</b> This identifier is the unique ICAL ID.
Word 5		YTIB token (5:8) <b>Note:</b> This identifier is the unique ICAL ID.
Word 6		First half of STCK
Word 7		Second half of STCK

Table 118. Contents of trace entry X'5A0B' for module DFSYSLM0

Word	Byte	Contents
Word 1	Byte 0	Module ID X'2D'
	Byte 1	Location ID: <b>X'21'</b> CM1 LUMIF SEND was invoked to process a potential CM1 response. This response could be a late response, a DFS2082 message, a regular response, or a timed-out message. Word 3 contains the front-end YTIB CLB address.
		<b>X'22'</b> CM1 response reroute is needed. Word 2 contains the YTIB_SPS_Flags.
		<b>X'23'</b> SPS shared queues back-end message processing. Word 2 contains the YTIB_SPS_Flags.
		<b>X'24'</b> SPS shared queues front-end message processing. Word 2 contains the YTIB_SPS_Flags.
	Byte 2	YTIB_SPS_STATUS
	Byte 3	0
Word 2		LUP_FE_TIB_PTR or YTIB_SPS_Flags

Table 118. Contents of trace entry X'5A0B' for module DFSYSLM0 (continued)

Word	Byte	Contents
Word 3		YTIB CLB address of the ICAL
Word 4		YTIB token (1:4) <b>Note:</b> This identifier is the unique ICAL ID.
Word 5		YTIB token (5:8) <b>Note:</b> This identifier is the unique ICAL ID.
Word 6		First half of STCK
Word 7		Second half of STCK

Table 119. Contents of trace entry X'5A0B' for module DFSYLUS0

Word	Byte	Contents
Word 1	Byte 0	Module ID X'2A'
	Byte 1	Location ID: <b>X'21'</b> DFSYLUS0 is ready to post back to DFSYMEM0 for an ICAL response. <b>X'22'</b> No ICAL response will be sent because of a timed-out or late message, or a <b>PSTOP</b> command. <b>X'23'</b> No ICAL response will be sent because REPLYCHK=NO and DFS2082 message support is enabled. <b>X'24'</b> Freed the YTIB control block in DFSYLUS0. <b>X'25'</b> SPS shared queues back-end GU.
	Byte 2	YTIB_SPS_STATUS
	Byte 3	0
Word 2		YTIB_SPS_Flags
Word 3		YTIB CLB address of the ICAL
Word 4		YTIB token (1:4) <b>Note:</b> This identifier is the unique ICAL ID.
Word 5		YTIB token (5:8) <b>Note:</b> This identifier is the unique ICAL ID.
Word 6		First half of STCK
Word 7		Second half of STCK



Table 120. Contents of trace entry X'5A0B' for module DFSYQAB0 with location ID X'21' or X'22'

Word	Byte	Contents
Word 1	Byte 0	Module ID X'29'
	Byte 1	Location ID: <b>X'21'</b> A late message was detected and dequeued. <b>X'22'</b> A late message was detected and rerouted.
	Byte 2	LUP_MSG_FLAG4
	Byte 3	0
Word 2		LUP_SPS_EXPTM
Word 3		LUP_FE_TIB_PTR (the original YTIB CLB for the ICAL)
Word 4		LUP_MSG_ARRIVAL_TIME (1:4) <b>Note:</b> This identifier is the saved YTIB_TOKEN value
Word 5		LUP_MSG_ARRIVAL_TIME (5:8) <b>Note:</b> This identifier is the saved YTIB_TOKEN value
Word 6		First half of STCK
Word 7		Second half of STCK

Table 121. Contents of trace entry X'5A0B' for module DFSYQAB0 with location ID X'23'

Word	Byte	Contents
Word 1	Byte 0	Module ID X'29'
	Byte 1	Location ID: <b>X'23'</b> A late message was detected and dequeued.
	Byte 2	YTIB_SPS_STATUS
	Byte 3	0
Word 2		YTIB_EVENT
Word 3		The original YTIB CLB for the ICAL
Word 4		YTIB token (1:4) <b>Note:</b> This identifier is the unique ICAL ID.
Word 5		YTIB token (5:8) <b>Note:</b> This identifier is the unique ICAL ID.
Word 6		First half of STCK
Word 7		Second half of STCK

Table 122. Contents of trace entry X'5A0B' for module DFSYQAB0 with location ID X'21' or X'22'

Word	Byte	Contents
Word 1	Byte 0	Module ID X'23'
	Byte 1	Location ID: <b>X'21'</b> The original YTIB was found. A response to the ICAL call will be sent if the state is OK. <b>X'22'</b> The ICAL region ended abnormally and IMS is processing a cleanup request for the ICAL call.
	Byte 2	YTIB_SPS_STATUS
	Byte 3	0
Word 2		YTIB_EVENT
Word 3		The original YTIB CLB for the ICAL
Word 4		YTIB token (1:4) <b>Note:</b> This identifier is the unique ICAL ID.
Word 5		YTIB token (5:8) <b>Note:</b> This identifier is the unique ICAL ID.
Word 6		First half of STCK
Word 7		Second half of STCK

## OTMA module-to-code cross-reference table

The OTMA module-to-code cross-reference table consists of module numbers, module name, and a description. You can use this information to associate code xx in message DFS1269E and the module number in trace records X'5A'xx, X'5B'xx and X'5C'xx with a module.

Table 123. OTMA module-to-code cross-reference table

Module number (decimal)	Module number (hexadecimal)	Module/User exit type	Description
19	13	DFSYLUS0	OTMA fast services
20	14	DFSYST00	OTMA storage manager
21	15	DFSYRR00	OTMA destination reroute setup routine
22	16	DFSYIO00	OTMA input/output setup routine
23	17	DFSYCM20	OTMA command processor
24	18	DFS6DC0	Read and build LU 6.2 descriptors
25	19	DFSYCLH0	OTMA /TRA services
26	1A	DFSYRAC0	OTMA security
27	1B	DFSYMGX0	OTMA z/OS cross-system coupling facility message exit
28	1C	DFSYGRX0	OTMA XCF group exit

Table 123. OTMA module-to-code cross-reference table (continued)

Module number (decimal)	Module number (hexadecimal)	Module/User exit type	Description
29	1D	DFSXYMO0	OTMA attach member OIM TCB
30	1E	DFSYC480	OTMA STA/ST0 (join/leave) interface
31	1F	DFSXFND0	OTMA FINDDEST processor
32	20	DFSXFDO0	OTMA control block processor
33	21	DFSXFDO10	OTMA control block processor
34	22	DFSXMOM0	OTMA AWE server DFSXMOM0
35	23	DFSXMEM0	OTMA member AWE server DFSXMEM0
36	24	DFSXIMI0	OTMA getting storage for new member
37	25	DFSXPSI0	TPIPE input AWE server DFSXPSI0
38	26	DFSXPSO0	TPIPE output AWE server DFSXPSO0
39	27	DFSXSND0	OTMA XCF interface
40	28	DFSXTIB0	OTMA synchronous processor DFSXTIB0
41	29	DFSXQAB0	OTMA asynchronous processor DFSXQAB0
42	2A	DFSXLUS0	OTMA service module number 0
43	2B	DFSXCMD0	OTMA command service
44	2C	DFSXCKP0	OTMA check point
45	2D	DFSXSLM0	OTMA synchronous send module
46	2E	DFSXRST0	OTMA restart
47	2F	DFSXIDC0	OTMA descriptor builder
48	30	DFSXQFX0	OTMA queue fixer
49	31	OTMAYPRX	OTMA Destination Resolution user exit
50	32	DFSXDRU0	OTMA default DRU exit routine DFSXDRU0
51	33	OTMAIOED	OTMA Input/Output Edit user exit

## OTMA verb-to-code cross-reference table

The OTMA verb-to-code cross-reference table consists of verb numbers, verb name, and a description. You can use this information to associate the z/OS cross-system coupling facility call number in trace record X'5B' xx with a z/OS XCF verb.

Table 124. z/OS XCF verb-to-code cross-reference table

Verb number (hexadecimal)	Verb name	Verb description
01	IXCCREAT	Defines a member to XCF
02	IXCJOIN	Enables a member to join a group
03	IXCQUERY	Return information about groups and members

Table 124. z/OS XCF verb-to-code cross-reference table (continued)

Verb number (hexadecimal)	Verb name	Verb description
04	IXCMGO	Sends a message to another active member
05	IXCMGI	Receives a message on an active member
06	IXCLEAVE	Disassociates a member from XCF

## DFS1269E message information

OTMA issues message DFS1269E when a severe internal error occurs. If you receive this message, contact the IBM Support Center with the module number and reason code supplied in the message, and, if requested, output from the OTMA trace.

The message format is:

```
DFS1269E SEVERE IMS INTERNAL FAILURE, REASON CODE=xxyy
```

Variable xx is a decimal number that identifies the module. To determine the module associated with the code, see [Table 123 on page 388](#). Variable yy is an internal reason code.

The following two reason codes are module independent. Variable xx represents the specific IMS module issuing the macro call.

### Reason code Description

#### xx98

Failure in DFSPPOOL to acquire storage for a variable with the DFSYMGAT macro.

#### xx99

Failure in DFSPPOOL to release storage for a variable with the DFSYMARL macro.

Other reason codes are module dependent.

## OTMA log records

To activate OTMA logging, use one of the **/TRA SET ON** commands.

### About this task

To activate OTMA logging, enter one of the following trace commands from the master terminal or the z/OS console.

```
/TRA SET ON tmember client1.  
/TRA SET ON tmember client1 tpipe tpipe1.
```

## SNAPs and dumps

For errors that do not result in an abend, IMS writes log record X'67D0', or produces an SDUMP, depending on the error. The minimum data dumped for OTMA problems are the control blocks that are associated with the task in error and the appropriate trace tables.

## Diagnosing Fast Path problems related to print data set options: IMS Spool API support

---

IMS provides an expansion of the DL/I application program interface that allows applications to interface directly to JES and create print data sets on the JES spool. These print data sets can then be made available to print managers and spool servers to serve the needs of the application.

### Understanding parsing errors

The IMS Spool API support provides feedback to the application program when IMS detects errors in the print data set options included on either the CHNG calls or SETO calls. This section describes the high-level processing of the parameters associated with the CHNG and SETO calls, including some examples of errors and the types of feedback information that can be expected.

Error codes provides a summary of the error codes that can be expected to be returned if the application provides a feedback area. It might be useful for the application to develop ways to display these errors by sending a message to an IMS printer or some other technique that allows examination of the parameter lists and feedback area without having to look at a dump. This section discusses each error code and provides some examples of when the error code might be expected. This discussion applies to these calls when used with the IMS Spool API support.

When diagnosing multiple parsing error return codes, the first code returned should be the most meaningful. Errors detected with incorrect length fields or previously invalid keywords can result in valid keywords being reported as errors.

### Keywords

The parameter lists used with CHNG and SETO calls contain two types of keywords. The two types are those keywords valid for the calls (that is, IAFP, PRTO, TXTU, and OUTN), and the keywords provided as operands of the PRTO keyword (for example, CLASS, FORMS).

This separation of keywords is used to determine what type of keyword validation IMS should perform. When looking for valid keywords on the calls, one set of keywords is valid, and when looking at keywords following the PRTO keyword, another set of keywords are valid. For this reason, incorrectly specified length fields may cause one scan to terminate prematurely and keywords to be invalid because they are incorrectly positioned in the call list.

### Status codes

We can deduct what might be the source of the error code by looking at the status code returned for the call. As a general rule, a status code of AR is given when the keyword is associated with the call and a status code of AS is given when the keyword is invalid as a PRTO option. There might be exceptions to this rule, but in general this will hold true.

### Error code examples

These examples describe errors and the resulting error codes.

Some length fields are omitted from the examples when they are unnecessary to the example. Feedback and options lists that are shown on multiple lines are contiguous, as they are in the working storage of an application.

Subsections:

- [“Error code \(0002\)” on page 392](#)

- [“Error code \(0004\)” on page 392](#)
- [“Error code \(0006\)” on page 392](#)
- [“Error code \(0008\)” on page 393](#)
- [“Error code \(000A\)” on page 393](#)
- [“Error code \(000C\)” on page 393](#)
- [“Error code \(000E\)” on page 394](#)

## Error code (0002)

This code indicates that an invalid keyword was discovered within the call options. The error code of (0002) indicates that the keyword scan being performed is associated with keywords that are valid for the call. For example:

```
CALL = SETO
      01
OPTIONS LIST = PRT0=04DEST(018),CLASS(A),TXTU=SET1
FEEDBACK = TXTU(0002)
STATUS CODE = AR
```

In this example, the options list contains both the keywords PRT0 and TXTU. The TXTU keyword is not valid for the SETO call.

Another example of an error code of (0002) in the feedback is created when the length field that represents the PRT0 options is specified as shorter than the actual length of the options. For example:

```
CALL = CHNG
      01
OPTIONS LIST = IAFP=N0M,PRT0=0FDEST(018),LINECT(200),CLASS(A),
              COPIES(80),FORMS(ANS)
FEEDBACK = COPIES(0002),FORMS(0002)
STATUS CODE = AR
```

In this example, the length field of the PRT0 options (001F) is too short to contain all the options. As a result, IMS finds the keywords COPIES and FORMS outside the PRT0 options list area and indicates that these keywords are not allowed as keywords on the CHNG call.

## Error code (0004)

This error code indicates that an option variable that follows a keyword in the options list for the CALL is not within the length limits for the option. An example of this type of error is the OUTN keyword. The name of the OUTPUT JCL statement must be from 1- to 8-characters. For example:

```
CALL = CHNG
OPTIONS LIST = IAFP=N0M,OUTN=OUTPUTDD1
FEEDBACK = OUTN(0004)
STATUS CODE = AR
```

The operand for the OUTN keyword is 9 bytes and exceeds the maximum value.

## Error code (0006)

This error occurs when IMS is scanning for valid keywords that are associated with the call and encounters the PRT0 keyword. On interrogation of the length field associated with the PRT0 keyword,

IMS discovers that the total length of the options list for the call is too short to contain all of the operands within the PRT0 keyword. For example:

```
CALL = CHNG
      0400          05
OPTIONS LIST = 0800IAFP=N0M,PRT0=0ADEST(018),LINECT(200),CLASS(A),
               COPIES(3),FORMS(ANS)

FEEDBACK = PRT0(0006),LINECT(0002),CLASS(0002),COPIES(0002),
           FORMS(0002)

STATUS CODE = AR
```

This example provides an options list that is hexadecimal, 48 (decimal 72) bytes long and the correct length for the options list. The length field of the PRT0 keyword incorrectly indicates a length of hexadecimal 5A. The length of the PRT0 options exceeds the length of the entire options list so the PRT0 keyword is ignored and the rest of the options list scanned for valid keywords. The feedback area contains the PRT0(0006) as we would expect to indicate a length error for this keyword, but we also find that the PRT0 keywords are reported to be in error (0002). This is because the keywords beyond the first PRT0 keyword, up to the length specified in the options list length field have been scanned in search of valid keywords for the call. The status code of AR tells us that the keywords are considered invalid for the call and not the PRT0 keyword.

### Error code (0008)

This error is returned when IMS finds that one of the options for the IAFP keyword has not been specified correctly. For example:

```
CALL = CHNG
      00
OPTIONS LIST = IAFP=N0Z,PRT0=0BDEST(018)

FEEDBACK = IAFP(0008) INVALID VARIABLE

STATUS CODE = AR
```

The message option of the IAFP keyword has been incorrectly specified as 'Z'. This results in the error code (0008).

### Error code (000A)

This error indicates that not all of the necessary keywords have been specified for this call. For example:

```
CALL = CHNG

OPTIONS LIST = TXTU=SET1

FEEDBACK = TXTU(000A)

STATUS CODE = AR
```

For this call, a valid keyword of TXTU was specified but the call also requires that the IAFP keyword be specified if the TXTU keyword is used. Since the IAFP keyword is missing, the error code of (000A) is returned when the TXTU keyword is found.

### Error code (000C)

The error code is reporting a condition in which a set of mutually exclusive keywords are used in the same call options list. A clue to the problem being with the call options and not the PRT0 options is given by issuing of the status code of AR and not the status code of AS. For example:

```
CALL = CHNG
      00
OPTIONS LIST = IAFP=A00,PRT0=0BCOPIES(3),TXTU=SET1
FEEDBACK = TXTU(000C)
STATUS CODE = AR
```

In this case, the call options list contains both the keywords of PRT0 and TXTU. These options are mutually exclusive and cannot be used in the same options call list. The result is error code of (000C) returned along with status code of AR.

## Error code (000E)

This error code indicates that while parsing the actual print data set descriptors, an error was detected with one or more of the operands. For the most part, IMS does not do any checking for these print descriptors. Instead IMS uses z/OS services (SJF) to do the validation of the print descriptors. When SJF is called, the validation requested is the same as for the TSO OUTDES command. For this reason, IMS is insensitive to changes in output descriptors and the valid descriptors for your system are a function of the z/OS release level.

You can obtain a list of the valid descriptors and the proper syntax by using the TSO HELP OUTDES command.

IMS must first establish that the format of the PRT0 options is in a format such that SJF services can be requested. If not, IMS returns status code AS and error code of (000E) and a descriptive error message. If the error has been detected during the SJF process, the error message from SJF includes information of the form, (R.C.=xxxx,REAS.=yyyyyyyy) and an error message indicating the error. The return codes and reason are further identified in the *z/OS MVS Programming: Authorized Assembler Services Reference Vol 1*.

The range of some variables are controlled by the JES initialization parameters. Values for the maximum number of copies, allowable remote destination, classes, and form names are examples of variables influenced by the JES initialization parameters.

The following examples show parsing errors and the resulting error messages:

```
CALL = CHNG
      01
OPTIONS LIST = IAFP=A00,PRT0=0BCOPIES((3),(8,RG,18,80))
FEEDBACK = PRT0(000E) (R.C.=0004,REAS.=00000204) COPIES/RG VALUE
                MUST BE NUMERIC CHARACTERS
STATUS CODE = AS
```

For this example, the COPIES parameter has the incorrect value 'RG' specified as one of its operands. The error message indicates that the values for these operands must be numeric.

```
CALL = CHNG
      00
OPTIONS LIST = IAFP=A00,PRT0=0AXYZ(018)
FEEDBACK = PRT0(000E) (R.C.=0004,REAS.=000000D0) XYZ
STATUS CODE = AS
```

This example includes an invalid PRT0 operand. The resulting reason code of X'000000D0' indicates that the operand shown (XYZ) is invalid.



## Debugging and diagnostic aids provided by IMS Spool API

In addition to providing feedback related to parsing errors, the IMS Spool API also provides other aids you can use in your diagnosis.

While debugging suspected problems with either the IMS Spool API or the application using the support, keep in mind that multiple services are involved in providing the total environment. Certain JES specifications might affect which options and specifications can be used by the IMS Spool API on behalf of an application program.

### Internal trace table

Each dependent region that uses the IMS Spool API creates a trace table that is used to trace module flow and significant events during IMS Spool API processing. This trace table is of the internal wrap around type, is always active for IMS Spool API functions, and cannot be written to an external device. It appears in any dumps that are produced by the dependent region. The first four words of the trace table are the header and contain the following information.

#### Word one

This is the trace table eye catcher. The eye catcher is IWB.

#### Word two

This is the offset from the beginning of the trace table (that is, trace table header) to the last entry traced. Since the entry is an offset, relocation of the trace table does not affect the use of this word to obtain the address of the last trace entry. The offset value is added to the relocated trace table address to obtain the last trace entry. If the value is zero, no entries have been traced.

#### Word three

This is the offset from the beginning of the trace table (the header) to the last trace entry in the table.

#### Word four

Reserved.

### Log records produced by the IMS Spool API

The IMS Spool API produces log records to record the significant events during IMS Spool API processing. A log record of the type X'68' is written for each data set that is opened. This log record contains the information necessary for identification of the data set. If any significant event occurs during spool processing, a diagnostic log record, 67D0 is produced to record diagnostic information about the error or event. The writing of the 67D0 records is normally associated with the DFS0013E message sent to the IMS MTO for these errors.

### Specialabend processing

The IMS Spool API places control blocks in both extended common storage area (ECSA) and dependent region private storage. When a dependent region dump is produced, and IMS abnormal termination routines are allowed to execute, the following control block relocation is performed to provide diagnostic information in the dependent region dump.

The master control block for the dependent region and any active data set control blocks in ECSA are copied to the dependent region. These control blocks are copied without modification and the ECSA address of each print data set control block, IAFPCB, is appended to the front of each relocated block.

A dummy module, DFSIAFD0, is loaded into the dependent region to serve as a place holder for the addresses of the relocated IMS Spool API control blocks. Obtain the address of module DFSIAFD0 by checking the dependent regions Job Pack Queue for the Contents Directory Entry (CDE) that represents module DFSIAFD0. The first three words of this dummy module contain the address of the relocated control blocks as follows.

#### Word one

This is the address of the relocated master control block (IAFPMCB) for the dependent region. The ECSA address of the master control block is appended in front of the relocated control block area. The eye catcher for the block is IAFPMCB.

**Word two**

This is the address of the first relocated IMS Spool API data set control block for a print data set (IAFPDCB). When this block is copied to the dependent region, the ECSA address of the original block is appended to the front of the relocated block. This is so that the chaining of the blocks can be verified. Any additional IAFPDCB control blocks are relocated following the first relocated block with the ECSA address of each block appended to the front of each relocated block. The eye catcher for the block is IAFPDCB.

**Word three**

This is the address of the trace table for the IMS Spool API. The eye catcher for the trace table is IWB.

**Service error log record 67D0**

The IMS Spool API creates Service Error log records, log record type 67D0, whenever a service error or unexpected condition is encountered. The 67D0 log record contains the service in error and detailed information about the system status at the time the error is detected. When problem determination is being attempted for suspected IMS Spool API errors, obtain the 67D0 log records from the IMS systems log. If the IMS Spool API issues message DFS0013E, a service error log record is also written.

In addition to the errors reported through message DFS0013E, service error log records are written if the IMS Spool API code encounters inconsistent control block structures or is unable to properly process print data sets during abend processing. These service error log records are printed using the File Select and Formatting Print utility (DFSERA10).

Some examples of events that cause service error log records 67D0 to be produced are:

- Error during storage obtain/free
- Open or Close errors
- Allocation or deallocation errors
- Errors during Output Descriptor processing
- BSAM write errors
- Invalid IAFP Control Block encountered
- Unable to process print data sets due to abending dependent region
- OTMA experiences a severe internal error or rejects a synchronous callout ICAL call from an IMS application

The writing of these service error log records occurs automatically.

# Chapter 12. DRA - Database Resource Adapter service aids

In a Database Control (DBCTL) environment, if you think the coordinator controller (CCTL) did not cause the problem, then start your analysis here.

## About this task

This section provides service aids and tips that can help you analyze problems in a Database Control (DBCTL) environment.

The DRA is the interface between DBCTL and the CCTL. The functions of the DRA are to:

- Request connection to and disconnection from DBCTL.
- Tell the CCTL when DBCTL has failed or when the operator has requested a shutdown.
- Manage threads.

## DRA dumps

The DRA creates a dump when a DRA request fails or when DRA processing fails. A DRA request failure produces either a system abend or an IMS pseudoabend. A DRA processing failure produces a system abend.

For either type of failure, the DRA first tries to create a z/OS SDUMP. If that fails, the DRA creates a SNAP dump. In some situations, the DRA creates a SNAP dump without attempting to create an SDUMP. For certain types of pseudoabends, the DRA creates neither an SDUMP nor a SNAP dump.

To determine what type of dump the DRA created, check field PAPLRETC in the DFSPAPL (the parameter list that passes information between the CCTL and the DBCTL). The field PAPLRETC has the format *hhsssuuu*, where *hh* indicates the type of dump.

The following table shows the values for *hh* and indicates which dumps the DRA creates for different types of failures.

Table 125. Determining the type of dump created by the DRA

hh	Type of dump	Failures
X'80'	SDUMP or SNAP	An SDUMP is created for all IMS abend codes not listed in this table, and for all z/OS abend codes that can be retried. If the SDUMP fails, a SNAP dump is created.
X'84'	SNAP	A SNAP dump is created for IMS abend codes U0260, U0261, and U0263.
X'88'	No	No SDUMP or SNAP dump is created for the following codes: <ul style="list-style-type: none"><li>• IMS abend codes U0775, U0777, U2478, U2479, U3303</li><li>• z/OS abend codes that can be retried (for example, S222 and S13E)</li><li>• DRA return codes</li></ul>

## DRA SDUMP output

The DRA creates a dump when a DRA request fails or when DRA processing fails. The DRA first tries to create a z/OS SDUMP.

DRA SDUMP output contains:

- IMS control region
- DLISAS address space
- Key 0 and key 7 CSA
- Selected parts of DRA private storage, including the address space control block (ASCB), task control block (TCB), and request blocks (RBs)

A DRA SDUMP has its own SDUMP option list. To add to the SDUMP option list of a DRA, you can use the CHNGDUMP parameter. However, you cannot use the CHNGDUMP parameter to delete areas from the list.

You can format the IMS control blocks by using the Offline Dump Formatter (ODF). The ODF does not format DRA storage. You can use IPCS to format the z/OS blocks in the private storage of the CCTL.

## SNAP dump output

The SNAP dump data sets are dynamically allocated whenever a SNAP dump is needed. A parameter in the DRA startup table defines the SYSOUT class.

SNAP dump output contains:

- Selected parts of DRA private storage, including the address space control block (ASCB), task control block (TCB), and request blocks (RBs)
- Thread blocks of the DBCTL

## Recovery tokens

In a DBCTL environment, you need to correlate the information that is produced by the CCTL with information that is produced by the DBCTL. The link between the CCTL and DBCTL is the recovery token, which uniquely identifies each unit of recovery (UOR).

### Recovery token details

The recovery token appears in the DRA dump (both SDUMPs and SNAP dumps) and in the dump title. The recovery token contains a mixture of EBCDIC and hexadecimal data as shown:

#### **CCTL subsystem ID 8 bytes (EBCDIC)**

Unique UOR ID (created by the CCTL) 8 bytes (hexadecimal)

## Analyzing DRA problems

---

To analyze DRA problems, first investigate any external conditions that might have caused the problem. If you can eliminate external causes, an unexpected DBCTL return code or another IMS function might have caused the problem.

### About this task

Follow these steps to analyze the problem.

### Procedure

1. Did external conditions cause the problem?
  - For CCTL external problems, check the status of applications or transactions. DBCTL and the DRA do not control these resources.
  - For DBCTL external problems, check the status of databases, PSBs, and dependent regions (BMPs and CCTLs) by using the **/DISPLAY** commands.
  - For DRA external problems:

- Make sure you are using the correct DRA startup table for this DBCTL/CCTL session. Values such as Fast Path buffer allocations and minimum/maximum thread specifications can cause scheduling and resource problems.
- Become familiar with the CCTL control exit.

The DRA calls the control exit to notify the CCTL of certain events, such as a DRA failure, an identify failure, a DBCTL failure, and so on. The DRA passes this information in a parameter list (DFSPAPL). The CCTL responds by passing back a return code in field PAPLRETC to tell the DRA what action to perform. Understanding which actions the CCTL is allowed to request can help you distinguish between valid actions and failures.

- The DRA does not issue any messages that report the actions it performed.
  - If an external condition caused the problem, stop here and fix the problem. Otherwise, continue with the next step.
2. You reach this point by eliminating external reasons as the cause of the problem.
- Determine if DBCTL returned a nonzero return code, indicating that the request from the CCTL was not successfully completed.
- If yes, take a z/OS online dump of the CCTL and contact IBM Software Support.
  - If no, then other functions might be involved in the problem. Use the appropriate section in this information to analyze the problem. Keyword procedures are useful in narrowing the problem to a specific cause.

## Results

To determine the source of problems in a DBCTL environment, create a dump of the CCTL address space. Dumps that are produced by SDUMP and by specifying the DUMP option on the CCTL **/SHUTDOWN** command are acceptable for problem diagnosis. If IBM Software Support needs to analyze the CCTL dump, send the unformatted dump so that they can obtain DBCTL DRA storage.

## Related concepts

[“Selecting the keywords” on page 34](#)

You select the proper keywords to search the IBM Software Support database for a problem similar to the one you are experiencing. The keywords you select depend on the component that is experiencing the problem and the type of failure that occurred.



# Chapter 13. FP - Fast Path service aids

Service aids for Fast Path include diagnosis, control interval, external trace, and locating control blocks.

## About this task

## Diagnosing Fast Path problems

Before diagnosing problems in Fast Path, you must understand the structure of its dumps, especially the dependent region dumps.

When a dependent region abends, the structure of the dump varies, depending on a number of conditions. For example, if you requested and were able to perform offline dump formatting, the structure of the dump is different than if you had not requested offline dump formatting. Furthermore, if the dependent region that abends is an MPP executing in mixed mode, the structure of the dump might be different from that of an IFP region. The recommended approach is to request and use the offline dump formatting option.

## ABENDU1026 analysis

To analyze ABENDU1026 failures, you determine the documentation to obtain, and how to find and interpret diagnostic data from the documentation. Gather the necessary data before searching an IBM Software Support database or calling the IBM Software Support.

## About this task

Several modules issue ABENDU1026 to indicate conditions that should not occur. The dependent region abends, but the IMS control region continues processing. Message DFS2712I accompanies ABENDU1026.

This analysis is based on using a dump that you can format with the Offline Dump Formatter (ODF). The following tables describes where to find ODF information.

*Table 126. Locating information about the Offline Dump Formatter (ODF)*

For information about	Refer to
Obtaining dumps that are suitable for input to the ODF	<a href="#">“Input for the IMS Offline Dump Formatter” on page 513</a>
Running the ODF	<i>IMS Version 15.5 Database Utilities</i>
Using the ODF to solve problems	<a href="#">“Formatting IMS dumps offline” on page 512</a>

Before beginning the analysis, you need the following information:

- A copy of the DFS2712I message
- A dump formatted by the ODF
- A copy of *IMS Version 15.5 Messages and Codes, Volume 3: IMSAbend Codes*

If an authorized program analysis report (APAR) is necessary, you might also need the following information:

- The last successful image copy of the database encountering the problem
- The IMS logs from the time of the last successful image copy to the point of failure
- A copy of the Fast Path trace, if Transaction Retry was invoked

The following example procedure takes you through the analysis of an actual ABENDU1026 until you have collected enough data to search an IBM software support database or call the IBM Support Center.

This example procedure uses the sample message DFS2712I in the following figure. Message DFS2712I is sent to the console. Save a printed copy of the message.

```
DFS2712I  MODULE NAME:   DBFMRCU0
DFS2712I  ABEND SUBCODE: 0053
DFS2712I  AREA  NAME:   DB21AR0

DFS2712I  MLTE:
DFS2712I  02A923BC  02919E60  00000000  00000000  00001008
DFS2712I  02A923CC  02903310  00005A08  00001008  00040400
DFS2712I  02A923DC  03018000  001C0008  029328B4  00060000
DFS2712I  02A923EC  00000000  00000000  00000000  02A92178
DFS2712I  02A923FC  02A92470  0072F70A  00000000  40800000
DFS2712I  02A9240C  00000000  00000000  00000000  00000000
DFS2712I  02A9241C  00000000  00000000  00060000  00000000
DFS2712I  02A9242C  00000000

DFS2712I  BUFFER CONTENTS:
DFS2712I  02919E58  016C0802  40000000  99000000  5C08015E
DFS2712I  02919E68  C1C140E3  C8C9E240  C9E240E3  C8C540C6
DFS2712I  02919E78  C9D9E2E3  40F3D9C4  40D3C5E5  C5D340E2
DFS2712I  02919E88  C5C7D4C5  D5E34040  40404040  40404040
DFS2712I  02919E98  40404040  40404040  40404040  40404040
.      .      .      .      .      .      .      .      .      .

DFS2712I  R0-R3  00000008  00000053  02919E60  02A92010
DFS2712I  R4-R7  02A923BC  008138D4  00000008  00005A00
DFS2712I  R8-R11 00000004  02903310  0070B040  0086DF20
DFS2712I  R12-R15 00818BA0  0070767C  80818C62  00000018
```

## Procedure

Use the following steps to analyze ABENDU1026:

1. Locate the module name and subcode associated with the abend. This information appears in the first few lines of message DFS2712I.

In the previous example, the module name is DBFMRCU0 and the subcode is 0053.

2. To find the meaning of the subcode, look up ABENDU1026 in *IMS Version 15.5 Messages and Codes, Volume 3: IMSAbend Codes*. Find module DBFMRCU0 and subcode 0053.

The description of subcode 0053 is: MLTE segment code (Reg4 + X'1E') is not equal to the DSEGCODE of the segment pointed to by register 2.

This means that the segment code in field MLTESGCD in MLTE (a Fast Path control block) does not match the segment code of the segment in the buffer (DSEGCODE). Therefore, your next step is to determine what the mismatched values are.

3. Check *IMS Version 15.5 Messages and Codes, Volume 3: IMSAbend Codes* again to determine which registers you must examine.

The important registers are:

Register 8 = MLTESGCD

Register 2 = Address of the segment; DSEGCODE is the first byte

In the previous example, the register contents appear at the bottom of message DFS2712I.

4. Use the registers and the buffer contents in the message to compare the segment code in the segment in the buffer (DSEGCODE) with the segment code in field MLTESGCD in the MLTE. These codes must match.
  - Register 8 contains the segment code from field MLTESGCD in the MLTE. In the example, register 8 has a value of 00000004.
  - Register 2 contains the address of the segment in the buffer. The first byte of the segment is the segment code (DSEGCODE). In the example, DSEGCODE has a value of 99.



- Because the segment code from the MLTE (04) does not match the segment code of the segment (99), ABENDU1026 occurred.

There are several ways to find this data. To find the segment code in field MLTESGCD in MLTE, you can also use register 4 + X'1E'. To find the DSEGCODE, you can also use register 6 (00000008), which is the offset in the buffer to the DSEGCODE.

5. Look at the module save area set to determine the module flow leading to the abend. You can use the Offline Dump Formatter (ODF) to format the save area set in a dump by specifying FMTIMS DB,MIN.
  - Register 13 in message DFS2712I contains the address of the save area for the PST that suffered the abend.
  - In the example message in the previous example, register 13 contains the address 0070767C.
  - In the \*\*DPST section of the formatted dump in the following example, search for a save area (SA) with address 0070767C. If you are searching online, the second occurrence you find is the save area.

\*\*\*SAVE AREA SET\*\*\*

```

EP DBFMCLX005/06/8804.27PL24768 ABCD
SA 0070755C WD1 8071B310 HSA 80000000 LSA 007075A4 RET 8088070E EPA 00812FE0
R0 00000519
R1 8071B310 R2 C7D5D740 R3 02A92010 R4 0001A000 R5 00707050
R6 00000000 R7 8072F624 R8 00707050 R9 0072F6CC R10 0070B040 R11 0086DF20
R12 00880042
EP DBFMGNX003/03/8820.09PL22770 AB
SA 007075A4 WD1 00000000 HSA 0070755C LSA 007075EC RET 808131A0 EPA 00814528
R0 00000519
R1 8071B3AB R2 C7D5D740 R3 02A92010 R4 02A92090 R5 008138D4
R6 FFFFFFFD80 R7 FEE06FD4 R8 00707050 R9 0072F6CC R10 0070B040 R11 0086DF20
R12 00812FE0
EP DBFMPUG005/11/8800.59PL26682 ABCDE
SA 007075EC WD1 00000000 HSA 007075A4 LSA 00707634 RET 8081466A EPA 00816900
R0 00000519
R1 8071B3AB R2 00000000 R3 02A92010 R4 02A92178 R5 008138D4
R6 FFFFFFFD80 R7 FEE06FD4 R8 00707050 R9 0072F6CC R10 0070B040 R11 0086DF20
R12 00814528
EP DBFMRCU003/21/8618.02PT01119 0
SA 00707634 WD1 00000000 HSA 007075EC LSA 0070767C RET 80816ABE EPA 00818BA0
R0 00000519
R1 8071B3AB R2 02A92178 R3 02A92010 R4 02A923BC R5 008138D4
R6 FFFFFFFD80 R7 00005A08 R8 0291AE66 R9 0072F6CC R10 0070B040 R11 0086DF20
R12 00816900
EP DBFMPG0002/04/8617.58PP35272 1B
SA 0070767C WD1 00000000 HSA 00707634 LSA 007076C4 RET 80818C62 EPA 00818FD8
R0 00000008
R1 8071B3AB R2 02919E60 R3 02A92010 R4 02A923BC R5 008138D4
R6 00000008 R7 00005A00 R8 00000004 R9 02903310 R10 0070B040 R11 0086DF20
R12 00818BA0
EP DBFMSRB002/13/8716.56PP58251 AB
SA 007076C4 WD1 00000000 HSA 0070767C LSA 0070770C RET 80822377 EPA 008285F0
R0 FFFF4040
R1 02903310 R2 02932A08 R3 02903278 R4 808222E0 R5 00822638
R6 00005A00 R7 00BBCF78 R8 02932A08 R9 02903310 R10 0070B040 R11 0086DF20
R12 008221B8
EP DBFXSL3007/08/8819.02PL28384 AB
SA 0070770C WD1 00000000 HSA 007076C4 LSA 00707754 RET 808286D7 EPA 00823D38
R0 00000000
R1 0070B040 R2 02932A70 R3 02903278 R4 02903310 R5 0071A250
R6 00005A00 R7 00BBCF78 R8 02932A08 R9 02903310 R10 0070B040 R11 0086DF20
R12 008285F0

```

6. In the example above, the module flow, reading from the top down, is DBFMCLX0, DBFMGNX0, DBFMPUG0, and DBFMRCU0, which is where the abend occurred. Notice that other modules follow DBFMRCU0 in the flow. You can ignore these modules now. However, they might be important later in the problem analysis.
7. Information from other sources might help you while searching the IBM software support database or talking with the IBM Support Center representative.

If an MPP or an IFP received the ABENDU1026, the Transaction Retry function should have retried the transaction.

Look in your MTO log for messages DFS0663I, DFS0784I, DFS0785I, DFS0787I, and other messages associated with a retry to find out what happened.

## Results

After you complete these steps, you have most of the following information:

- The abend code (ABENDU1026).
- The subcode (SUBCODE053).
- The module name (DBFMRCU0).
- The save area flow leading to the abend.
- The field in error (MLTESEGCD or DSEGCODE). You might not be sure which field is incorrect.
- Any messages produced by a transaction retry (for example, MSGDFS0663I).

With this information you are ready to search the database or contact IBM Software Support.

## Related concepts

[“Input for the IMS Offline Dump Formatter” on page 513](#)

The dump data set you use for input to the IMS Offline Dump Formatter must include Key 0 and Key 7 CSA, the CVT, and SQA. CSA is not required for batch or CICS-local DL/I. The dump must be machine readable.

[“Formatting IMS dumps offline” on page 512](#)

Two methods are available for formatting IMS dumps offline: interactive formatting, performed through a series of panels which provide formatting choices, and formatting by using JCL.

## Fast Path Transaction Retry

Fast Path Transaction Retry (FPTR) is designed for IMS Fast Path users who cannot run the Fast Path trace permanently on their system because of its impact on performance, but want to have the trace turned on when Fast Path failures occur.

Fast Path problems can be resolved much faster when trace information is available to show the logic flow of a call or transaction.

FPTR is activated only when certain Fast Path failures occur. FPTR automatically allocates a trace data set, turns on the trace, and retries the transaction. If no abend occurs on the retry, FPTR issues a message, turns off the trace, and the system continues processing. If an abend does occur on the retry of the transaction, Fast Path trace writes the trace data, FPTR turns off the trace, and the system continues with Fast Path trace inactive. FPTR is not invoked for abends in BMP regions.

When you report certain IMS Fast Path problems to the IBM Support Center, you will be asked if the Transaction Retry function failed. The following topics will help you determine what information to report.

## Processing flow

A summary of the processing flow of FPTR follows:

- The ESTAE exit of the dependent region controller receives control for abends U1026 and U1027, and all system abends except 122 and 222.
- The ESTAE exit provides debugging information including:
  - Name of abending module
  - Last applied APAR of the abending module
  - Date and time of assembly of module

If the failing module cannot be identified, a message informs the operator.

- The ESTAE exit decides if the transaction can be retried. If so, the ESTAE requeues the failing input message for retry and produces a dump of the first abend.
- Message DFS554A is sent to the master terminal.
- The retry process starts in an eligible dependent region.
  - FPTR dynamically allocates a trace data set and starts Fast Path trace.
  - FPTR writes message DFS0785A to the master terminal and the JES2 job log.
- When the retry of the transaction is complete, FPTR deallocates the trace data set and spools the contents of the trace data set to the SYSOUT class specified in the MSGCLASS parameter on the JOB statement of the dependent region.

## System programmer tasks

The system programmer should:

- Print the job log.
- Print the spooled trace data set information.
- Save and analyze the above information.
- Contact IBM Software Support for assistance, if needed.

## DEDB control interval (CI) problem assistance aids

---

When you print portions of the DEDB, the control intervals (CIs) have these identifying characteristics.

After you have performed the analysis described in [“ABENDU1026 analysis”](#) on page 401, review the contents of the various control blocks. Included in message DFS2712I is a dump of the control block that is related to the logical inconsistency. This control block is in the format of one of the control intervals (CIs). You can (possibly with help from IBM Software Support) obtain the RBA of the affected CI from the buffer. You can then use this RBA:

- When you extract the CI from the image copy of the DEDB
- When you choose the criteria for selecting and printing the IMS log records (with DFSERA10)

The acronyms used in this topic are:

### DOVF

Dependent overflow

### IOVF

Independent overflow

### RAP BLOCK

Root-anchor point block

### SDEP

Sequential dependent

## CI type identification

Each CI has an identifier at X'02' in the CI, with the exception of the first and second CIs. The first is the IMS control CI and the second contains the DMAC control block for this area.

### CI Type

Identifier

### REORG CI

00

### RAP

01

### DOVF

02

## IOVF (SPACE MAP)

04

## IOVF

08

## SDEP

10

## DEDB CI formats

These are the CI types and the data that is common to all CIs (except the SDEP CI).

### CI 0

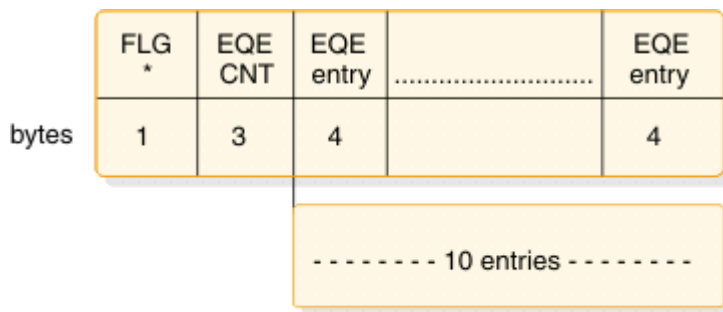
The IMS control CI.

0	8	10	18	1C	20	28	32
Creation	Restart	ERStart	RBA of	Characters	Cisize	Org	
Date/Time	Date/Time	Date/Time	Last CI	DBF1.000	- 7	"D"	

### CI 1

The DMAC control block for this area is located here.

The Error Queue Element (EQE) list is also located in this CI. This list is 44 bytes long and immediately precedes the trailer information, (for example, CUSN, RBA, RDF and CIDF). The following figure shows the EQE list format: FLG (1 byte), EQE CNT (3 bytes), 10 available EQE entries (40 bytes).



\* A flag setting of X'80' indicates that there are more than 10 EQEs or an error in the second CI.

Figure 71. EQE list in CI 1

### RAP CI

The following figure shows the RAP CI.

0	2	4	8
FSEAP	0203	RBA of current	Segments, FSEs and Scraps
		overflow CI	
	(02)	- Indicates CUSN	
		is in this CI.	

### First DOVF CI

The first DOVF CI has the format shown in the following figure.

0	2	4	8
FSEAP	0203	RBA of current	Segments, FSEs and Scraps
		overflow CI	
	(02)	- Same as RAP CI -- these two bits combined	
	(01)	- Look here for space -- make the 03 in byte 3.	

**Exception:** From here on, the key bits are shown, but byte 3 is not shown.

## Other DOVF CIs

All DOVF CIs except the first one have the format shown in the following figure.

0	2	4	8
FSEAP	02	RBA of next DOVF CI with space, last contains zeros	Segments, FSEs and Scraps

## First IOVF CI

The CI shown in the following figure is a space map and is the first in each group of 120 CIs. The 119 CIs that follow are data CIs.

0	2	4	6	8
0000	04	8000xxxx	offset to 1st free	8 (119 words mapping next 119 CIs) 8000xxxx free and offset to next free
		4000uow#	2000uow#	4000uow# allocated
		40000000	no free	2000uow# used by reorg
				space in this space map CI

## Other IOVF CIs

The following figure is a data CI - 119 data CIs follow each space map CI.

0	2	4	8
FSEAP	0802	4000uow#	Segments, FSEs and Scraps (allocated, to UOW number; 0 is the first UOW).
0008	0802	80000000	FSE (CI not allocated).
		(02)	indicates CUSN is in this CI

## SDEP CI

**Exception:** SDEP CIs do not contain FSEs and have no FSEAP or CUSN. User segments have a time stamp added at the end. The following figure shows the SDEP CI.

0	2 3	4	8
0000	1000	Partner name	Segments inserted sequentially and cannot be updated
	(01)	- Time stamp exists	
	(04)	- SDEP CI is full.	

## FSEAP

FSEAP is the offset of the first FSE in the CI. Fast Path FSEs are chained from the highest RBA, in order, to the lowest RBA in the CI.

FSE---X'8offssss' off=offset of next FSE in CI  
ssss=size (length) of the free space including the FSE.

X'8000ssss' indicates this is the last FSE on the chain in this CI.

If the CI is empty, the FSE is X'15' bytes less than the CI size, or X'13' less than the CI size if no CUSN exists. The RDF and CIDF are X'7' bytes less than the CI size. Here are some examples:

CI	512	X'200'	1024	X'400'	2048	X'800'	4096	X'1000'
FSE	800001EB		800003EB		800007EB		80000FEB	
RDF	0001F9		0003F9		0007F9		000FF9	
CIDF	01F90000		03F90000		07F90000		0FF90000	

## Scraps

Scraps are less than 4 bytes. They begin with X'7n' if less than 8 segment types, or X'Fn' if more than 8. For example,

1 byte-X'71' or X'F1'  
2 bytes-X'72' or X'F2'  
3 bytes-X'73' or X'F3'

## Data Common to All CIs

The last X'0D' bytes of a CI all have the same use. The last line of a CI looks like this in a dump.

```
data  data  data  data  data
x-x   x-x   x-x   x-x   xxxxxxxx -D -C-B-A-9 -8-7-6-5 -4-3-2-1
                                xxxxxxxx xxxbbbbbb bbbbbbbb
```

The bytes with bbbbbs do not print and will show as blanks in the dump. The fields from -D to -1 are:

CUSN	-D,C	These 2 bytes represent updates to the CI. The 02 bit in byte 3 of a CI indicates a CUSN exists in the CI.
RBA	-B,A,9,8	These 4 bytes are the beginning RBA of the CI.
RDF	-7,6,5	
CIDF	-4,3,2,1	

**Recommendation:** Use the RBA of the CI when you select log records to format and print with the DFSERA10 utility.

SDEP CIs do not contain FSEs and do not have a CUSN. SDEP CIs end at -B (the RBA). Data can occupy the space up to that location.

## Analyzing control interval (CI) contention

When CI contention occurs in a DEDB, Fast Path passes both lock requests to program isolation (PI) modules. The PI trace, if active, traces the locks. To format the PI trace records (log record type X'67FA'), use the File Select and Formatting Print utility (DFSER10) with exit DFSERA40. For information about running this utility, see *IMS Version 15.5 System Utilities*.

Using the trace records, find the RBA field of the CI. The digits in the CI RBA field are shifted right 8 bits. For example, an RBA of 00468000 is displayed as 00004680.

You must translate the value in the DMB field to a relative DMAC number. (DMAC numbers are relative to the DATABASE definitions.)

For example, if the first DMAC is X'FFFE', the second DMAC is X'FFFD', the third DMAC is X'FFFC', and so on. Because databases are chained alphabetically in the DDIR, if the DMB field is X'FFF6', you calculate the relative DMAC number as follows:

```
X'FFFF' - X'FFF6' = X'19' = 25 (decimal)
```

This calculation means that X'FFE6' is the 25th area relative to the first area of the first DEDB in the DDIR.

## Related reference

[“Log records” on page 469](#)

To diagnose some problems, you need to examine the content of log records to determine what was going on in the system before the problem occurred. By knowing the layout of the log records, you can set up a DFSERA10 job that will produce the specific log records that you need to examine.

## Locating Fast Path control blocks and tables

Many of the Fast Path control blocks are extensions of IMS full-function control blocks. The names of these Fast Path control blocks are the same as in full-function.

The acronyms for these Fast Path control blocks start with "E."

### Example:

#### SCD

System Contents Directory (full-function IMS)

#### ESCD

Extended System Contents Directory (Fast Path)

To view the layout of the Fast Path control blocks for your system, assemble DFSADSCT from IMS.ADFSSMPL and use XREF(FULL).

The following table shows the Fast Path control blocks and work areas that appear as a load list in an IMS dump. This information is relevant when you are working on an abend U1011 in module DBFINI20. Message DFS2703A generally accompanies the abend. Abend U1011 results from either a GEN problem or a storage fragmentation problem.

*Table 127. Fast Path control blocks and work areas that appear in IMS dumps*

<b>Load list name</b>	<b>Fast Path block/work area</b>	<b>Appearance in dump</b>
DFSEPhnnn	Fast Path EPSTs (nnn=000-999)	IMS STM task
DBFCONT1	ECNTs/MSDBs	IMS STM task
DBFCONT3	DMHRs/buffers	IMS STM task
DBFCONT4	DEDB blocks	IMS STM task
DBFCONT5	OTHRs	IMS STM task
DBFCONT6	BALGs	IMS STM task
DBFCONT7	Miscellaneous buffers	IMS STM task
AREALIST	AREA list	IMS STM task

At Fast Path initialization, modules DBFINI21, DBFINI23, DBFINI24, DBFINI25, DBFINI26, and DBFINI27 calculate the amount of contiguous ECSA storage that is needed to load the buffers, buffer headers, MSDBs, and other related control blocks into separate work areas named DBFCONTx.

If module DBFINI2x cannot obtain a large enough contiguous block of storage, abend U1011 is issued, along with an error message. When this abend occurs, you can restart IMS, or you can stop other jobs that might prevent module DBFINI2x from obtaining the necessary storage.

The following table describes the control block structures that are defined during IMS startup. This table can help you determine which control blocks are needed in your Fast Path environment.

*Table 128. Control blocks that are allocated during IMS startup*

<b>Control block/table</b>	<b>With MSDB/ DEDB</b>	<b>Without DEDB</b>	<b>Without MSDB</b>	<b>Without MSDB/DEDB</b>
ECNT DBFCONT1	X	X	X	X
BHDR DBFCONT1	X	X		
MSDB DBFCONT1	X	X		
DMHR DBFCONT3	X	X	X	
BUFF DBFCONT3	X	X	X	
DEDB blocks (DMCBs, DMACs, MRMBs, segment names, and field names) DBFCONT4	X		X	
OTHR DBFCONT5	X		X	
BALG DBFCONT6	X	X	X	X
LBUF DBFCONT7	X	X	X	X

Table 128. Control blocks that are allocated during IMS startup (continued)

Control block/table	With MSDB/ DEDB	Without DEDB	Without MSDB	Without MSDB/DEDB
FPAL AREALIST	X		X	

## Locating IMS blocks and work areas by using load list elements

IMS loads IMS blocks and work areas using the IMS IMODULE facility. IMS generates a load list element from which you can obtain the unique name and location of each work area.

### Load list areas

Load list areas are areas that appear formatted as the load list in an IMS control region dump. Global areas are in the common storage area (CSA).

Table 129. Load list areas

Load list name	IMS block/work area	Pool type
DFSABSxx	Abend Diagnostic Area, xx=PST number	Global
DFSBFSP	DL/I Buffer Handler Pool	Global
DFSBLK0x	SCD, x=same as nucleus suffix	Global
DFSBWLOG	BG Write Log Work Area	Local
DFSCBTHD	Control block table header that points to the storage pools defined in DFSCBT00	Global <sup>"1" on page 412</sup>
DFSCBT10	Storage pool headers for the pools defined in DFSCBT00	Global <sup>"1" on page 412</sup>
DFSDLWxx	Retrieve Work Area, xx=PST number	Global
DFSDMBRS	Resident DMBs	Global
DFSDSET	OLDS Data Set Entry Table	Local
DFSEOVOS	OSAM DCB Work Area	Global
DFS01FXL	Fixlist for OSAM I/O Driver	Local
DFSINTRS	Resident Intent Lists	Global
DFSIPB	Initialization Parameter Block	Local
DFSISIT	Ident Table and ISI Storage	Global
DFSLCD	Logger LCD	Global <sup>"2" on page 412</sup>
DFSLCDST	IMS Monitor Logger LCD	Global
DFSLLOG	X'06' and X'42' Log Records	Local
DFSLOCP	Storage Management Local Pool	Local
DFSLOGxx	Log Work Area, xx=PST number	Global
DFSLXBC	Link Extension Blocks for MSC CTC	Global
DFSLXBM	Link Extension Blocks and I/O Buffers for MSC MTM links	Global
DFSMFDDH	MFS Pool Dynamic Directory Hash Table	Local <sup>"4" on page 412</sup>



Table 129. Load list areas (continued)

Load list name	IMS block/work area	Pool type
DFSMFDDP	MFS Pool Dynamic Directory Prime Area	Local <a href="#">“4” on page 412</a>
DFSMFDD0	MFS Pool Dynamic Directory Entry Area	Local <a href="#">“4” on page 412</a>
DFSMFPDS	MFS Pool PDS Directory Indexes	Local <a href="#">“4” on page 412</a>
DFSMFSTG	MFS Pool Staging Buffers	Local <a href="#">“4” on page 412</a>
DFSMTCLB	CLB (ECB) for DFSCMTIO	Global
DFSMTIOT	Monitor TIOT Table	Global
DFSMTMH	MSC Main Storage-to-Main Storage Queue Header	Local <a href="#">“3” on page 412</a>
DFSMTMW	MSC Main Storage-to-Main Storage Window	Local <a href="#">“3” on page 412</a>
DFSOFPL	OSAM Buffer Pool	Global <a href="#">“2” on page 412</a>
DFSOFWA	OSAM Buffer Pool Work Area	Local
DFSOLRnn	OLDS Read DCB where nn must be numeric	Local
DFSOSDEB	OS/VS2 "Fake" OSAM DEB	Global
DFSPCWAP	Communications Work Pool	Local
DFSPDBWP	Database Work Pool	Global
DFSPDMB	DMB Pool	Global
DFSPFBP	MFS Pool	Local
DFSPFWA	Prefetch Work Area, ECB and Save Sets	Local
DFSPPSBW	PSB and PSB Work Pool	Global
DFSPQBUF	Queue Manager Buffers	Local
DFSPSBRs	Resident PSBs	Global
DFSPSTQE	Scheduler Sequence Queue	Global
DFSPSTxx	SAP Work Area, xx=PST number	Global
DFSPTPDB	Communications Pool	Local
DFSPWKAP	Working Storage General Pool	Global <a href="#">“2” on page 412</a>
DFSRSTEB	Restart ECB and Save Sets	Local
DFSRSTWA	Restart Work Area	Local
DFSSBBUF	Sequential buffering: SBUF	Local
DFSSBCA1	Sequential buffering: SCAR	Global
DFSSBDCB	Sequential buffering: SDCB	Local
DFSSBDSE	Sequential buffering: EDSG	Local
DFSSBD SG	Sequential buffering: SDSG	Local
DFSSBITA	Sequential buffering: ITASK storage for overlapped I/O	Global
DFSSBPSS	Sequential buffering: SBPSS	Global
DFSSBPST	Sequential buffering: SBPST	Local

Table 129. Load list areas (continued)

Load list name	IMS block/work area	Pool type
DFSSBRAN	Sequential buffering: SRAN	Local
DFSSBSBU	Sequential buffering buffers	Local
DFSSBSCD	Sequential buffering: SBSCD	Global
DFSSBWO	Sequential buffering: DFSSBWO	Local
DFSSLX	SCD Latch Extension	Global
DFSSSCT	Subsystem Control Table	Local <sup>"3" on page 412</sup>
DFSSTAEB	STAE Work Area	Local
DFSSTPEB	Stop Region ECB, Save Sets and Work Area	Local
DFSSTPWA	Stop Region Message Work Area	Local
DFSTRMWK	Modify/Terminate Task Save Sets, ECB and Work Area	Local
DFSTSAV	Temporary Save Sets	Local
DFSVRFXL	Fixlist for EXCPVR	Local
DFSXCWxx	Exclusive Control Enqueue/Dequeue Work Area, xx=01-99	Global <sup>"2" on page 412</sup>
DFSZIBxx	ZIB/FAQE Pool, xx=01-99	Global

**Notes:**

1. A large number of storage pools are defined in module DFSCBT00. The contents directory element (CDE) name for storage in a given control block table (CBT) pool is #xxxxyyy, where xxxx is the pool name, and yyy is a number from 001 to 999. See [Table 130 on page 412](#) for a description of the CBT pools.
2. When you use the local storage option (LSO), all these areas are obtained from local storage. When you use Fast Path and LSO, DFSLCD, DFSDBUFF, and DFSXCWxx remain in global storage. When you select LSO = S, DFSLCD and DFSPWKAP remain in global storage.
3. IMS constructs these areas at abend time. They consist of copies of the subject areas preceded by one word containing the original address of the area.
4. IMS builds these areas in extended private storage.

## Control block table pools

Modules DFSBCB00, DFSBCB30, and DFSBCB90 support get/release requests for blocks in specific storage pools, referred to as 'CBT' pools.

Table 130. CBT pool names and descriptions

CBT pool	Description
AHDR	Autologon LU headers
ADSC	Fast Path DEDB area data set control block
AESL	Fast Path DBRC parameter area
AWE	Work-to-do element for task communication
BCPT	Checkpoint ID table

Table 130. CBT pool names and descriptions (continued)

CBT pool	Description
BQEL	Used when a buffer is altered and released at sync point
BXQE	Storage manager queue elements
CBLK	LU 6.2 CPI communications driven control block
CCB	Conversational control block
CLLE	Common latch list element
CMWU	Save sets/ECB for ITASKs which do not require a PST
CSAG	Callable services anchor block (ECSA storage)
CSAL	Callable services anchor block (E-private storage)
DBPB	Database purge block
DBRC	DBRC work area
DDIR	Database directories
DDRE	DMB directory extension
DESC	LU 6.2 descriptor block
DG2W	Dispatcher work area section 2 (global storage)
DL2W	Dispatcher work area section 2 (local storage)
DPST	Dependent region PST: The following blocks are associated with the dependent region structure:  D1WA, DG2W, EPST, FSRB, GQMW, IDT, IOSB, IRLM, KLSD, LCRE, SAP, SLOG, STTR, XPST.
D1WA	Dispatcher work area section 1
EPST	Fast Path PST extension
EQEL	Recoverable in doubt structure queue elements
EZS	External subsystem storage
FEIB	Front-end message switch interface block
FNCB	Used by Fast Path for global command notifies
FPB6	Fast Path 64-bit buffer manager pool
FPCP	Used by Fast Path for local commands
FSRB	Fast Path wake up/sleep SRBs
GESE	Represents a defined external subsystem
GIOB	IOB for batch
GOWA	OSAM channel programs for batch
GQMW	Global queue manager work area
GS24	Global 24-bit save area
GSAV	Global save area
IAFP	IMS advanced future print block

Table 130. CBT pool names and descriptions (continued)

CBT pool	Description
IDT	Block used to keep track of identified regions
IEQE	In-flight/in-doubt data buffers
IOSB	I/O supervisor block for OSAM
IRLM	Dependent region block, if IRLM is used
KLSD	LSO=X,Y block for each dependent region
LCLL	Local common latch list element (E-private storage)
LCRE	Local Recovery element (persists across restart)
LG24	Below the 16MB line dynamic SAP save sets
LGND	Block used to hold logon descriptor representations
LGWA	Log work area
LGWX	Log work area extension (private)
LPST	PSTs for IMS internal use in local storage
LQB	Local queue block (SPQBs and CNTs)
LQMW	Local queue manager work area
LS24	Local 24-bit save area
LSAV	Dynamic SAP save sets
LUB	LU 6.2 LU block
L56X	Fast Path database control log record
MSGP	Message buffers in global storage
OLRK	OLR database block
OSWA	OSAM channel program areas
PCIB	MFS Partition CIB
PDIR	Program directories
PF62	LU 6.2 message prefix block
PST	PSTs for IMS internal use in global storage
QAB	LU 6.2 queue anchor block
QMBA	Queue manager global buffer area
QSAV	Save sets with AWEs
RACW	RACF workarea
RCNT	Remote communication name table
RCTE	Fast Path routing codes
RECA	VTAM receive any buffers
RPST	Restart PST
RRE	Represents an active thread to an external subsystem

Table 130. CBT pool names and descriptions (continued)

CBT pool	Description
SAP	Save area prefix – Includes fixed and dynamic SAPs
SIDX	One for each identified external subsystem
SLOG	IMS Monitor parameter area block
SMB	Scheduler message blocks
SOPB	Sign-on parameter list block
SRBC	Common SRBs used for data sharing asynchronous NOTIFYs
STAT	Database Control (DBCTL) and Database Resource Adapter (DRA) statistics area
STTR	Retrieve trace area
SVPG	System service parameter list block (global-ECSA)
SVPL	System service parameter list block (local-private)
TCBT	TCB table
TIB	LU 6.2 transaction instance block
TTAB	Trace table (31-bit storage)
TT24	Trace table (24-bit storage)
USMU	Security block
USRD	Blocks used to represent user control block structure
UXIC	User exit instance blocks in common storage
UXIP	User exit instance blocks in private storage
UXSC	User exit static work area in common storage
UXSP	User exit static work area in private storage
VRPL	VSAM RPL with two save areas
VTGB	VTAM terminal control blocks
VWA	Volatile work area
WLMB	Work area of parameter list for the Workload Manager classification service.
XMCI	Cross memory ITASK block
XPST	Dependent region PST extension
X124	DL/I pool below the 16MB line for MVS/ESA

## Fast Path external trace

The Fast Path external trace is a tool for diagnosing problems with Fast Path DL/I calls.

Unexpected DL/I status codes or abends such as U1026 are examples of problems with Fast Path DL/I calls. Fast Path external trace is best suited for diagnosing problems that can be easily recreated, and is not intended to be run routinely because the overhead and output volume of the trace can be large. Fast Path external trace is intended for use primarily by IBM Software Support, but users might also find it useful. IBM Software Support specialists might ask you to capture Fast Path external trace data for analysis by IBM specialists.

Fast Path external trace traces only dependent region activity; you cannot use it to collect data on control region processes. Because most Fast Path DL/I call flow is normally processed in the dependent region, this limitation is not serious. However, if you specify the PARDLI=1 option, DL/I processing is performed under the CTL TCB, which limits the usefulness of Fast Path external trace.

**Recommendation:** Do not trace PARDLI=1 execution. Create the problem again, if possible, without specifying the PARDLI=1 option.

The Fast Path trace entries are documented in [“Fast Path trace entries” on page 641](#).

## Trace activation

You can activate trace using the Fast Path Transaction Retry, the CCTL DRA thread, or using the `/TRA SET ON TABLE FAST` command.

- The Fast Path Transaction Retry function normally attempts to activate Fast Path external trace when a transaction is retried in an MPP or IFP region after an abend in Fast Path code. In this case, the trace is activated internally for the dependent region that is executing the retry and not for other dependent regions. The trace is deactivated after one retry attempt. The Fast Path Transaction Retry function dynamically allocates an FPTRACE DD statement as a JES SPOOL file and closes and deallocates an FPTRACE DD statement when the trace is deactivated at the end of the retry operation. The intent of the Fast Path Transaction Retry function is to provide first-failure data capture.
- A CCTL DRA thread can also request that Fast Path external trace be activated for a particular thread during the create thread process.
- The trace can also be activated with a `/TRA SET ON TABLE FAST` command. The Fast Path external trace writes diagnostic data to a FPTRACE DD statement in the dependent region JCL. After the trace is activated, the presence or absence of the FPTRACE DD statement determines whether data is traced for each active dependent region, including CCTL DRA threads. A spool file (`SYSOUT=x`) can be used for a FPTRACE DD statement, or a DASD file can be used. DCB attributes are forced to `LRECL=133, BLKSIZE=133, RECFM=FA` by IMS when the DCB is opened.

### Recommendations:

- Use a spool file (`SYSOUT=x`) rather than a disk file. A certain amount of data related to the trace activation itself is traced (written to FPTRACE) before the determination is made that the trace is active or inactive.
- Do not include an FPTRACE DD statement in your standard dependent region JCL. Add it only as required and then remove it after the trace data is collected.

## Trace deactivation

The trace is deactivated with a `/TRA SET OFF TABLE FAST` command.

## Diagnostic data

Data is formatted as it is written. No offline formatting of the trace data is required.

### About this task

#### *Trace point identifiers*

The FP trace captures module flows, and at certain points, logic flows within modules. In most cases, there is a trace point at entry to a module and another at exit from the module. There might be additional trace points within the module. Each trace point has a unique 4-character identifier. To indicate nesting within call flows, using this unique 4-character identifier, the identifier is shifted right at each level. Each trace entry is prefixed by the identifier located in columns 1-13. The relative position of the identifier within columns 1-13 indicates nesting level, for example:

```
IRC1.....  
.MCL0.....
```

Because the identifier has 4-characters and 13 positions are available, 9 levels of nesting are possible. Output lines with no identifier in columns 1-13 are continuations of the previous entry. The module entry and exit trace entries differ by only one character. Usually, the module exit identifier is the same as the module entry identifier, except for one character. Normally, the first character of the identifier is shifted up one alphabetically, for example:

```
.MCL0.....
.NCL0.....
```

### **Trace point time stamps**

Trace point time stamps are labeled with TOD=xxxxxxx. The hexadecimal digits are the middle 4-bytes of an 8-byte STCK time stamp. The high order digit is approximately 1 second.

**Note:** Fields that are labeled TIME and DATE within trace entries refer to the compile date and time of the module involved, and are not related to trace time.

### **Trace initialization entries**

Entries COT1 to TRAN at the beginning of the trace file refer to FP Trace initialization and can be ignored. These entries are produced if a FPTRACE DD statement is specified, even if the trace is not enabled.

### **Key trace point data items**

The data that is traced for each trace point varies. However, each field has a label to make it easier to determine the contents of the entry. Some of the common and useful labels include:

#### **Ra#b,Rab,**

Registers a-b follow.

#### **CALL**

DL/I call function.

#### **TOKN**

UOR recovery token.

#### **MODU**

Module entry point address.

#### **EPST**

EPST address of the dependent region.

#### **SSA**

The first 30 bytes of the call SSA. Might contain residual data for short SSAs.

## **Fast Path external trace examples**

The following examples show the Fast Path external trace.

### **Trace entries from COT1 to SIEX are tracing the initialization of FPTRACE**

```
COT1.....TOD=B6B46252 WKAR=0A6FCC94 R0#F=00000031 007BF6B0 8AD8D6D8 00004700 8A6FC634 00000001 00CC4B20 7ABC7570
8AD8D6D8 007F6A
.....D8
BLTE.....TOD=B6B46259 R15=8AE1A060 DATE=01/0310.3 0#10=00004700 007AE900 8AD8D6C4 00004700 8A6FC634 0A6FCC94
00CC4B20 7ABC7570 8
.....AD8D6D8 007F6AD8 0A6FC040
BLTX.....R0#F=000000DD 007AE900 000000DC 007AF360 8A6FC634 0000000C 000000DD 007AE91C 00000001 007F6AD8 0A6FC040
007AE91C 8AE1A0
.....60 0A5BE6F0 000000DE 00000000 EPST=0A6FC040 TFTD=007AE900C4C2C6E3C6E3D6D400000000
TSTD=00000000000000000000000000000000
.....XTOM=00000000
COT2.....R0#F=00004700 007AE900 007AE900 00004700 8A6FC634 0A6FCC94 00CC4B20 7ABC7570 8AD8D6D8 007F6AD8 0A6FC040
00CC5B78 8ADC00
.....D0 0A5BE6F0 8ADC033E 00000000 EPST=0A6FC040 TFTD=007AE900C4C2C6E3C6E3D6D400000000
TSTD=00000000000000000000000000000000
.....XTOM=00000000
COT3.....R0#F=00004700 007AE900 007AE900 00004700 8A6FC634 0A6FCC94 00CC4B20 7ABC7570 8AD8D6D8 007F6AD8 0A6FC040
00CC5B78 8ADC00
.....D0 0A5BE6F0 8ADC033E 00000000 EPST=0A6FC040 TFTD=007AE900C4C2C6E3C6E3D6D400000000
TSTD=00000000000000000000000000000000
.....XTOM=00000000
COTX.....TOD=B6B465B8 WKAR=0A6FCC94 R0#F=00004700 007AE900 007AE900 00004700 8A6FC634 0A6FCC94 00CC4B20 7ABC7570
8AD8D6D8 007F6A
```

```

.....D8 0A6FC040 00CC5B78 8ADC00D0 0A5BE6F0 8ADC033E 00000000 MODU=8ADC00D0 DATE=08/01/031 TIME=10.32U
LCHA=UP9HCT011
COTE.....TOD=B6B465B9 WKAR=0A6FCC94 R0#F=00000032 007BF6B0 000063AC 8ADC00D0 8A6FC624 00000001 00CC4B20 7ABC7570
00CC5B78 007F6A
.....D8 0A6FC040 00CC5B78 8ADC00D0 0A5BE6F0 8ADC010C 00000000 MODU=8ADC00D0 DATE=08/01/031 TIME=10.32U
LCHA=UP9HCT011
COTY.....TOD=B6B465BA WKAR=0A6FCC94 R0#F=00000030 007BF6B0 8AD91DD8 8ADC00D0 8A6FC624 00000001 00CC4B20 7ABC7570
00CC5B78 007F6A
.....D8 0A6FC040 00CC5B78 8ADC00D0 0A5BE6F0 8ADC019E 00000000 MODU=8ADC00D0 DATE=08/01/031 TIME=10.32U
LCHA=UP9HCT011
STS9.....TOD=B6B46614 R15=8ADDC410 DATE=01/0310.3 0#10=00000950 007BF6B0 000063AC 8ADDC410 8A6FC624 00000001
00CC4B20 7ABC7570 0
.....0CC5B78 007F6AD8 0A6FC040
STXS.....TOD=B6B4664D R15=00000000 DATE=01/0310.3 0#10=00000002 0A5BE060 007AE900 8ADDC410 8A6FC624 00000001
00CC4B20 7ABC7570 0
.....0CC5B78 007F6AD8 0A6FC040
STEX.....TOD=B6B4664E R15=00000000 DATE=01/0310.3 0#10=00000950 007BF6B0 000063AC 8ADDC410 8A6FC624 00000001
00CC4B20 7ABC7570 0
.....0CC5B78 007F6AD8 0A6FC040

```

## End of trace initialization

```

FPR3.....TOD=B6B60ECA WKAR=0A6FCC94 R0#F=000121F8 0A69602C C7C8E440 00000000 00000001 0A5BE060 0BC14F5B C4C5D7C1
D9E3D4E3 4D0060
.....18 0A6FC040 00010000 8ADBA272 0001EE48 0001EC80 0E48B350 MODU=8ADBA272 DATE=08/05/03P TIME=PQ6040 LCHA=01
1IAB

```

## DL/I call start in DBFIRC10

```

IRC1.....TOD=B6B60ECC R015=00000000 00000C17 C7C8E440 0A69602C 0BC04F54 0A5BE060 00000000 8A71C580 D9E3D4E3
0A9F1048 0A6FC040 00
.....CC5B78 8ADBA272 0A5BE618 0A6FC9F2 00000001 LCRE=0A9F1048 TOKEN=E2E8E2F3404040400000000600000000
EPCB=0A69602C PCBA=0A52F
.....35C PCBD=000121F8 ESCD=00CC5B78 MADR=8AD477B0 EPST=0A6FC040 WKAR=0A6FCC94 SVIO=0BC04F54 SVIL=0000 SVSN=0001
.MCLO.....TOD=B6B60ECE CALL=GHU EPST=0A6FC040 WKAR=0A6FCC94 EPCB=0A69602C CCID=00 LCID=00 PRGP=00000000
LKFP=00000000 PCB=0A52F3
.....5C SSA=DEPARTMT(DEPTKEY = R1210000001 MODU=0AD4AD80 DATE=08/01/031 TIME=10.25E LCHA=E}
i"}

```

## Call is GHU - first 30 bytes of SSA are traced

```

..SAGE.....TOD=B6B60ED0 R0#9=00000000 00000C17 C7C8E440 0A69602C 0BC04F54 0AD4B67C FFFFE20 FEE02D87 00CC4B20
000000FF R14=8AD4AF9
.....0 R15=0AD728C8 EPCB=0A69602C FLGM=00 DMAC=00000000 ARBA=00000000 UOWO=00000000 CCID=8C CCNT=00000001
UBLK=00000000 KUBL
.....=00000000 DEDB=10 LKFP=00000000 PRGP=00000000 CLEV=0A6960EC MLTE=0BC04F54 POPT=00 SGLS=00 CLOC=00000000
DMHR=00000000 P
.....RBA=00000000 CRBA=00000000 NRBA=00000000 GRBA=00000000 XRBA=00000000 SGCD=00 PROF=0000 LEVL=00 KEYL=00
FLGA=00 FLGB=00
.....ACCK=0000 KEYO=0000 SDBS=00000000 MLTE=0BC04F54 SFRX=00 SFSX=00 SFWX=00 SFZX=00 SFMX=00 PREF=0000
SCNT=00000000 SWC1=00
.....SWC2=00 SWC3=00 LOPR=00 SNAP=00000000 EPST=0A6FC040 SCVL=00 REOP=00 FDLN=00 FDOF=0000 DEDB=00
COMP=00000000 STAT=
.....WCH=00 MODU=0AD728C8 DATE=08/01/031 TIME=10.28E LCHA=E}
i"}

```

## SSA handler for GET type calls

```

...SAGI.....TOD=B6B60EDB R0#9=00000000 00000C17 C7C8E440 0A69602C 00000000 0A6FC588 FFFFE20 FEE02D87 00CC4B20
000000FF R14=8AD7293
.....6 R15=0AD72030 EPCB=0A69602C FLGM=00 DMAC=00000000 ARBA=00000000 UOWO=00000000 CCID=8C CCNT=00000001
UBLK=00000000 KUBL
.....=00000000 DEDB=10 LKFP=00000000 PRGP=00000000 CLEV=0A6960EC MLTE=00000000 POPT=80 SGLS=CE CLOC=040C0000
DMHR=00FC7B38 P
.....RBA=070C2000 CRBA=00000000 NRBA=814171C0 GRBA=00000000 XRBA=00000000 SGCD=9E PROF=0008 LEVL=07 KEYL=0C
FLGA=10 FLGB=00
.....ACCK=80CE KEYO=B594 SDBS=078D2000 MLTE=00000000 SFRX=00 SFSX=00 SFWX=04 SFZX=0C SFMX=00 PREF=8002
SCNT=00000000 SWC1=00
.....SWC2=00 SWC3=04 LOPR=00 SNAP=00000000 EPST=0A6FC040 SCVL=00 REOP=00 FDLN=00 FDOF=0000 DEDB=00
COMP=00000000 STAT=
.....WCH=00 MODU=0AD72030 DATE=08/01/031 TIME=10.28E LCHA=E}
i"}
....VSNA.....TOD=B6B60EDF R0#9=00000000 00000C17 C7C8E440 0A69602C 00000000 0A6FC588 8BC14F5B FEE02D87 00CC4B20
000000FF R14=8AD7208
.....6 R15=0AD74478 EPCB=0A69602C FLGM=00 DMAC=00000000 ARBA=00000000 UOWO=00000000 CCID=8C CCNT=00000001
UBLK=00000000 KUBL
.....=00000000 DEDB=10 LKFP=00000000 PRGP=00000000 CLEV=0A6960EC MLTE=00000000 POPT=80 SGLS=CE CLOC=040C0000
DMHR=00FC7B38 P
.....RBA=070C2000 CRBA=00000000 NRBA=814171C0 GRBA=00000000 XRBA=00000000 SGCD=9E PROF=0008 LEVL=07 KEYL=0C
FLGA=10 FLGB=00
.....ACCK=80CE KEYO=B594 SDBS=078D2000 MLTE=00000000 SFRX=00 SFSX=00 SFWX=04 SFZX=0C SFMX=00 PREF=8002
SCNT=00000000 SWC1=00

```





```
=00000000 DEDB=10 LKFP=00000000 PRGP=00000000 CLEV=0A6960EC MLTE=0A6960EC POPT=00 SGLS=01 CLOC=00000000
DMHR=00000000 P
.....RBA=00000000 CRBA=00000000 NRBA=00000000 GRBA=00000000 XRBA=00000000 SGCD=01 PROF=0000 LEVL=01 KEYL=0B
FLGA=88 FLGB=00
.....ACCK=000C KEYO=0008 SDBS=0AD09210 MLTE=0A6960EC SFRX=00 SFSX=00 SFWX=00 SFZX=00 SFMX=00 PREF=0006
SCNT=00000001 SWC1=00
.....SWC2=02 SWC3=00 LOPR=82 SNAP=0A696168 EPST=0A6FC040 SCVL=01 REOP=80 FDLN=0B FDOF=0008 DEDB=80
COMP=0A4998D0 STAT=
.....WCH=00 MODU=0AD6B640 DATE=08/01/031 TIME=10.27P LCHA=PQ69789 A
```

## Retrieve by qualified call

```
...MCTL.....R0#F=00000000 00000C17 C7C8E440 0A69602C 0A6960EC 00000001 FFFFE20 FEE02D87 00CC4B20 0A5BE060 0A6FC040
00CC5B78 0AD4C4
.....98 0A5BE6F0 8AD6BDE0 0AD4C498 EPCB=0A69602C FLGM=00 DMAC=00000000 ARBA=00000000 UOWO=00000000 CCID=8C
CCNT=00000001 UBL
.....K=00000000 DEDB=10 LKFP=00000000 PRGP=00000000 CLEV=0A6960EC DBPC=0A52F35C STC= A LEV=00 SFD=DEDB
.....FD=D9F1F2F1F0F0F5F1F0F0F3C1404040404040404040404040404040400000E2D7C3C20000000001000301 MLTE=0A6960EC POPT=00
SGLS=01 CLOC=
.....00000000 DMHR=00000000 PRBA=00000000 CRBA=00000000 NRBA=00000000 GRBA=00000000 XRBA=00000000 SGCD=01
PROF=0000 LEVL=01
.....KEYL=0B FLGA=88 FLGB=00 ACCK=000C KEYO=0008 SDBS=0AD09210 MLTE=0A6960EC SFRX=00 SFSX=00 SFWX=00 SFZX=00
SFMX=00 PREF=00
.....06 SCNT=00000001 SWC1=00 SWC2=02 SWC3=00 LOPR=82 SNMT=0A696168 NAME=DEPARTMTÿ EPST=0A6FC040 SCVL=01
REOP=80 FDLN=0B FDO
....SSA9....TOD=B6B613F3 R0#9=00000004 00000000 C7C8E440 0A69602C 0A6960EC 00000001 00000001 FEE02D87 00CC4B20
00000000 R14=8AD4C7A
.....4 R15=0AD712D0 EPCB=0A69602C FLGM=00 DMAC=00000000 ARBA=00000000 UOWO=00000000 CCID=8C CCNT=00000001
UBLK=00000000 KUBL
.....=00000000 DEDB=10 LKFP=00000000 PRGP=00000000 CLEV=0A6960EC MLTE=0A6960EC POPT=00 SGLS=01 CLOC=00000000
DMHR=00000000 P
.....RBA=00000000 CRBA=00000000 NRBA=00000000 GRBA=00000000 XRBA=00000000 SGCD=01 PROF=0000 LEVL=01 KEYL=0B
FLGA=88 FLGB=00
.....ACCK=000C KEYO=0008 SDBS=0AD09210 MLTE=0A6960EC SFRX=00 SFSX=00 SFWX=00 SFZX=00 SFMX=00 PREF=0006
SCNT=00000001 SWC1=00
.....SWC2=02 SWC3=00 LOPR=82 SNAP=0A696168 EPST=0A6FC040 SCVL=01 REOP=80 FDLN=0B FDOF=0008 DEDB=80
COMP=0A4998D0 STAT=
.....WCH=00 MODU=0AD712D0 DATE=08/01/031 TIME=10.28P LCHA=PQ73448 A
```

## Search SSA for data

```
....BACK....TOD=B6B613F6 R15=00000000 DATE=01/0310.2 0#10=00000004 00000000 C7C8E440 0A69602C 0A6960EC 00000001
00000001 FEE02D87 0
.....0CC4B20 00000000 0A6FC040
....MDRA....TOD=B6B613F7 R0#9=00000004 00000000 00000000 0A69602C 0A6960EC 00000001 00000001 FEE02D87 00000001
00000000 R14=8AD7186
.....6 R15=0AD4E428 EPCB=0A69602C FLGM=04 DMAC=00000000 ARBA=00000000 UOWO=00000000 CCID=8C CCNT=00000001
UBLK=00000000 KUBL
.....=00000000 DEDB=10 LKFP=00000000 PRGP=00000000 CLEV=0A6960EC MLTE=0A6960EC POPT=00 SGLS=01 CLOC=00000000
DMHR=00000000 P
.....RBA=00000000 CRBA=00000000 NRBA=00000000 GRBA=00000000 XRBA=00000000 SGCD=01 PROF=0000 LEVL=01 KEYL=0B
FLGA=88 FLGB=00
.....ACCK=000C KEYO=0008 SDBS=0AD09210 MLTE=0A6960EC SFRX=00 SFSX=00 SFWX=00 SFZX=00 SFMX=00 PREF=0006
SCNT=00000001 SWC1=00
.....SWC2=02 SWC3=00 LOPR=82 SNAP=0A696168 EPST=0A6FC040 SCVL=01 REOP=80 FDLN=0B FDOF=0008 DEDB=80
COMP=0A4998D0 STAT=
.....WCH=00 MODU=0AD4E428 DATE=08/01/031 TIME=10.26 LCHA=
i~}}
i~}}
```

## Determine the possibility of randomizing

```
....MD03....TOD=B6B613FB WKAR=0A6FCC94 R0#F=00000004 0A4998C0 0A4998C6 0A69602C 0A6960EC 00000010 00000001 FEE02D87
00000001 000000
.....01 0A6FC040 00CC5B78 0AD4E428 0A5BE780 8AD71866 00000000 MODU=0AD4E428 DATE=08/01/031 TIME=10.26 LCHA=
i~}}
```

## No position

```
....MD49....TOD=B6B613FC WKAR=0A6FCC94 R0#F=00000004 0A4998C0 0A4998C6 0A69602C 0A6960EC 00000010 00000001 FEE02D87
00000001 000000
.....01 0A6FC040 00CC5B78 0AD4E428 0A5BE780 8AD71866 0A4998D0 MODU=0AD4E428 DATE=08/01/031 TIME=10.26 LCHA=
i~}}
```

## Use randomizer

```
....MDRA....TOD=B6B613FD R0#9=00000004 0A4998C0 0A4998C6 0A69602C 0A6960EC 00000010 00000001 FEE02D87 00000001
00000001 R14=8AD7186
.....6 R15=0A4998D0 EPCB=0A69602C FLGM=04 DMAC=00000000 ARBA=00000000 UOWO=00000000 CCID=8C CCNT=00000001
UBLK=00000000 KUBL
```

```

.....=00000000 DEDB=10 LKFP=00000000 PRGP=00000000 CLEV=0A6960EC MLTE=0A6960EC POPT=00 SGLS=01 CLOC=00000000
DMHR=00000000 P
.....RBA=00000000 CRBA=00000000 NRBA=00000000 GRBA=00000000 XRBA=00000000 SGCD=01 PROF=0000 LEVL=01 KEYL=0B
FLGA=88 FLGB=00
.....ACCK=000C KEY0=0008 SDBS=0AD09210 MLTE=0A6960EC SFRX=00 SFSX=00 SFWX=00 SFZX=00 SFMX=00 PREF=0006
SCNT=00000001 SWC1=00
.....SWC2=02 SWC3=00 LOPR=82 SNAP=0A696168 EPST=0A6FC040 SCVL=01 REOP=80 FDLN=0B FDOF=0008 DEDB=80
COMP=0A4998D0 STAT=
.....WCH=00 MODU=0AD4E428 DATE=08/01/031 TIME=10.26 LCHA=
i~}}
.....MGAP....TOD=B6B61416 R0#F=00000004 00000000 0000000A 0A69602C 0A6960EC 00000001 00000001 FEE02D87 0AD091C0
0A69602C 0A6FC040 00
.....CC5B78 0AD54858 0A5BE780 8AD71906 0AD54858

```

## Get root anchor point

```

.....ERAN....TOD=B6B61417 R0#F=00000006 0AD0923C 0000000C 0A4998D0 0000000C 00000078 0A6FC140 FEE02D87 0AD091C0
0A69602C 0A6FC040 00
.....CC5B78 0AD54858 0A5BE780 8AD71906 8B6900E8 TOD=B6B61417 R0#9=00000006 0AD0923C 0000000C 0A4998D0 0000000C
00000078 0A6F
.....C140 FEE02D87 0AD091C0 0A69602C R14=8AD71906 R15=8B6900E8 EPCB=0A4998D0 FLGM=00 DMAC=00000000
ARBA=00000000 UOW0=000000
.....00 CCID=00 CCNT=00000000 UBLK=00000000 DEDB=00 MLTE=0000000C LEVL=80 SDBS=00000000 EPST=0A6FC040
SWAR=DEPTKEY =
.....

```

## Entry to randomizer

```

.....XРАН....TOD=B6B61419 WKAR=0A6FCC94 R0#F=00000000 0AD09288 0000000C 0A4998D0 0000000C 00000078 0A6FC140 FEE02D87
0AD091C0 0A6960
.....2C 0A6FC040 00CC5B78 0AD54858 0A5BE780 8AD54982 00000000 MODU=0AD54858 DATE=08/01/03P TIME=PQ7029 LCHA=96
ABCE}

```

## Entry to randomizer

```

.....NGAP....TOD=B6B6141A R0#F=00000000 0AD09288 00000000 00001004 00000004 00000000 0A6FC140 FEE02D87 0AD091C0
0A69602C 0A6FC040 00
.....CC5B78 0AD54858 0A5BE780 D9C1D5C4 00000000
.....MGRF....TOD=B6B6141B WKAR=0A6FCC94 R0#F=00000000 00000000 0000000A 0A69602C 0A6960EC 00000001 00000001 FEE02D87
0AD09288 000000
.....00 0A6FC040 00CC5B78 0AD56F70 0A5BE780 8AD71912 0AD56F70 MODU=0AD56F70 DATE=08/01/031 TIME=10.26 LCHA=
i~}}

```

## Get root

```

.....MBED...TOD=B6B6141C AREA=DEPTAR0
.....0AD09288 00000000 RE#F=8AD5730A 8A71C580

```

## Get control (CI RBA X'1000')

```

.....EXXC..TOD=B6B6141D WKAR=0A6FCC94 R0#F=00000000 00000000 0A6FC284 0A69602C 0A6960EC 8AD56FC4 00001000 00001000
0AD09288 000000
.....00 0A6FC040 00CC5B78 0AD5AB48 0A5BE810 8AD486BA 0AD5AB48 MODU=0AD5AB48 DATE=IVELOCK08 TIME=8/05/0 LCHA=03E}
i~

```

## Get EXCL CI lock (CI RBA X'1000')

```

.....NGXC..TOD=B6B6141F R0#9=0000D800 0A5BE060 00000000 09E6A040 00000008 00001000 00001000 0A960CB0 0AD09288
00000000 RE#F=00 00
.....XCRB=0A960CB0 NEXT=00000000 SHDC=00000000 OPST=0A6FC040 UOWN=00001000 FLGS=80 DMHR=00000000
.....MSRB..R0#F=FFFF4040 0AA922C0 0AD09288 0A69602C 8AD48710 0AD3FA20 00001000 0A960CB0 0AD09288 0AA922C0 0A6FC040
00CC5B78 0AD6E8
.....30 0A5BE858 8AD48B1E 0AD6E830 EPST=0A6FC040 WKAR=0A6FCC94 MODU=0AD6E830 DATE=08/05/031 TIME=13.58P
LCHA=PQ71804 0

```

## Synchronous read of CI

```

.....VSOR.TOD=B6B61424 WKAR=0A6FCC94 R0#F=FFFF4040 00000000 0AD09288 0A69602C 0AA922C0 0AD3FA20 00001000 0A960CB0
0AD09288 0AA922
.....C0 0A6FC040 00CC5B78 0AD98E88 0A5BE8A0 8AD6E902 0AD98E88 MODU=0AD98E88 DATE=08/05/031 TIME=14.00K
LCHA=KX20294 0
.....VSOR.TOD=B6B61426 WKAR=0A6FCC94 R0#F=0AB38800 00000000 0AD09288 00000001 00000000 0A95B268 00001000 00000000
0AD09288 0AA922

```

```
.....C0 0A6FC040 00CC5B78 0AD98E88 0A5BE8A0 0009A800 00000000 MODU=0AD98E88 DATE=08/05/031 TIME=14.00K
LCHA=KX20294 0
```

## CI in VSO data space

```
.....NSRB...R0#F=FFFF4040 00000000 0AD09288 0A69602C 0AA922C0 0AD3FA20 00001000 0A960CB0 0AD09288 0AA922C0 0A6FC040
00CC5B78 0AD6E8
.....30 0A5BE858 8AD6E902 00000000 EPST=0A6FC040 WKAR=0A6FCC94 MODU=0AD6E830 DATE=08/05/031 TIME=13.58P
LCHA=PQ71804 0
.....NBED...TOD=B6B61428 R0#9=FFFFFFD8 0AA922C0 FFFFFFFD7 0A69602C 8AD48710 0AD3FA20 00001000 0A960CB0 0AD09288
0AA922C0 RE#F=7A 00
.....XCRB=0A960CB0 NEXT=00000000 SHDC=00000000 OPST=0A6FC040 UOWN=00001000 FLGS=80 DMHR=0AA922C0
.....MGR9...TOD=B6B6142C WKAR=0A6FCC94 R0#F=00000004 0000000B 00000000 0A69602C 0A6960EC 8AD5703C 0AB38008 00001000
0AD09288 0AA922
.....C0 0A6FC040 00CC5B78 0AD56F70 0A5BE780 0AB38010 00000000 MODU=0AD56F70 DATE=08/01/031 TIME=10.26E LCHA=€}
i~}}}
```

## Found the root

```
.....NGRF...TOD=B6B6142D WKAR=0A6FCC94 R0#F=00000004 0000000B 00000000 0A69602C 0A6960EC 8AD5703C 0AB38008 00001000
0AD09288 0AA922
.....C0 0A6FC040 00CC5B78 0AD56F70 0A5BE780 0AB38010 00000000 MODU=0AD56F70 DATE=08/01/031 TIME=10.26E LCHA=€}
i~}}}
```

```
.....FOND...TOD=B6B6142E WKAR=0A6FCC94 R0#F=00000004 00000000 00000004 0A69602C 0A6960EC 00000001 00000001 FEE02D87
00000010 000000
.....00 0A6FC040 00CC5B78 0AD712D0 0A5BE738 0000000C 00000000 MODU=0AD712D0 DATE=08/01/031 TIME=10.28P
LCHA=PQ73448 A
.....SSA9...TOD=B6B6142F R0#9=00000004 00000000 00000004 0A69602C 0A6960EC 00000001 00000001 FEE02D87 00001004
00000000 R14=0000126
.....8 R15=00000000 EPCB=0A69602C FLGM=E0 DMAC=0AD09288 ARBA=00001004 UOWO=00000000 CCID=8C CCNT=00000002
UBLK=00000000 KUBL
.....=00000000 DEDB=10 LKFP=00000000 PRGP=00000000 CLEV=0A6960EC MLTE=0A6960EC POPT=00 SGLS=01 CLOC=0AB38008
DMHR=0AA922C0 P
.....RBA=00001004 CRBA=00001008 NRBA=00001268 GRBA=00001268 XRBA=00001004 SGCD=01 PROF=0000 LEVL=01 KEYL=0B
FLGA=88 FLGB=00
.....ACCK=000C KEYO=0008 SDBS=0AD09210 MLTE=0A6960EC SFRX=00 SFSX=00 SFWX=00 SFZX=00 SFMX=00 PREF=0006
SCNT=00000001 SWC1=20
.....SWC2=02 SWC3=00 LOPR=82 SNAP=0A696168 EPST=0A6FC040 SCVL=01 REOP=80 FDLN=0B FDOF=0008 DEDB=80
COMP=0A4998D0 STAT=
.....WCH=10 MODU=0AD712D0 DATE=08/01/031 TIME=10.28P LCHA=PQ73448 A
.....MCT3...TOD=B6B61432 WKAR=0A6FCC94 R0#F=00000004 00000000 C7C8E440 0A69602C 0A6960EC 00000001 00000010 FEE02D87
00CC4B20 000000
.....00 0A6FC040 00CC5B78 0AD4C498 0A5BE6F0 8AD4C7A4 00000000 MODU=0AD4C498 DATE=08/01/031 TIME=10.26E LCHA=€}
i~}}}
```

```
.....MCTL...R0#F=00000004 0000000C 0000000C 0A69602C 0A6960EC 00000001 0AB38010 FEE02D87 0000000B 00000000 0A6FC040
00CC5B78 0AD4C4
.....98 0A5BE6F0 0A52F35C 00000000 EPCB=0A69602C FLGM=E0 DMAC=0AD09288 ARBA=00001004 UOWO=00000000 CCID=8C
CCNT=00000003 UBL
.....K=00000000 DEDB=10 LKFP=00000000 PRGP=00000000 CLEV=0A6960EC DBPC=0A52F35C STC= A LEV=01 SFD=DEPARTMT
FD=D9F1F2F1F0F0F0F0F0F0F1C14040404040404040404040400000E2D7C3C20000000001000301 MLTE=0A6960EC POPT=00
SGLS=01 CLOC=
.....0AB38008 DMHR=0AA922C0 PRBA=00001004 CRBA=00001008 NRBA=00001268 GRBA=00001268 XRBA=00001004 SGCD=01
PROF=0000 LEVL=01
.....KEYL=0B FLGA=88 FLGB=00 ACCK=000C KEYO=0008 SDBS=0AD09210 MLTE=0A6960EC SFRX=00 SFSX=00 SFWX=00 SFZX=00
SFMX=00 PREF=00
.....06 SCNT=00000002 SWC1=20 SWC2=02 SWC3=00 LOPR=82 SNMT=0A696168 NAME=DEPARTMTY EPST=0A6FC040 SCVL=01
REOP=80 FDLN=0B FDO
.....RC04...TOD=B6B61437 R0#F=00000004 00000C17 C7C8E440 0A69602C 0A6960EC 00000004 FFFFE20 FEE02D87 00CC4B20
0A5BE060 0A6FC040 00
.....CC5B78 0AD6B640 0A5BE6A8 0000000C 00000000
```

## Found the root; moved from current position

```
.....TOGH...TOD=B6B61438 R15=00000000 DATE=01/0310.2 0#10=00000000 00000010 C7C8E440 0A69602C 0A6960EC 00000004
FFFFFFE20 FEE02D87 0
.....0CC4B20 0A5BE060 0A6FC040
.....NOPA...TOD=B6B61438 WKAR=0A6FCC94 R0#F=00000000 00000000 0A6960EC 0A69602C 0A6960EC 00000004 FFFFE20 FEE02D87
00CC4B20 0A5BE0
.....60 0A6FC040 00CC5B78 0AD6B640 0A5BE6A8 0000000C 00000000 MODU=0AD6B640 DATE=08/01/031 TIME=10.27P
LCHA=PQ69789 A
.....MRQC...TOD=B6B6143B R0#9=00000000 0AB3800E 0A6960EC 0A69602C 0A6960EC 00000004 00000000 FEE02D87 00CC4B20
0A5BE060 R14=8AD6C27
.....2 R15=00000000 EPCB=0A69602C FLGM=E0 DMAC=0AD09288 ARBA=00001004 UOWO=00000000 CCID=8C CCNT=00000003
UBLK=00000000 KUBL
.....=00000000 DEDB=10 LKFP=0A6960EC PRGP=0A6960EC CLEV=0A6960EC MLTE=0A6960EC POPT=00 SGLS=01 CLOC=0AB38008
DMHR=0AA922C0 P
.....RBA=00001004 CRBA=00001008 NRBA=00001268 GRBA=00001268 XRBA=00001004 SGCD=01 PROF=0000 LEVL=01 KEYL=0B
FLGA=88 FLGB=00
.....ACCK=000C KEYO=0008 SDBS=0AD09210 MLTE=0A6960EC SFRX=00 SFSX=00 SFWX=00 SFZX=00 SFMX=00 PREF=0006
SCNT=00000002 SWC1=60
.....SWC2=02 SWC3=00 LOPR=82 SNAP=0A696168 EPST=0A6FC040 SCVL=01 REOP=80 FDLN=0B FDOF=0008 DEDB=80
COMP=0A4998D0 STAT=
.....WCH=18 MODU=0AD6B640 DATE=08/01/031 TIME=10.27P LCHA=PQ69789 A
.....SEG4...TOD=B6B61443 R15=00000000 DATE=01/0310.2 0#10=00000000 00000C17 0A52F35C 0A69602C 0A6960EC 0AD4B67C
FFFFFFE20 FEE02D87 0
```

```

.....0CC4B20 0A5BE060 0A6FC040
...NCL0.....R0#F=00000000 00000C17 0A52F35C 0A69602C 0A6960EC 8BC14F5B FFFFE20 FEE02D87 00CC4B20 0A52F35C 0A6FC040
00CC5B78 0AD4AD
.....80 0A5BE660 00000030 00000000 EPCB=0A69602C CCID=8C LCID=8C PRGP=0A6960EC LKFP=0A6960EC DBPC=0A52F35C
STC= A LEV=01 S
.....FD=DEPARTMT
.....0080000100400010100010B8800000C00080AD0 STAS=

```

## Call ends

```

..OPMV.....TOD=B6B61447 R0#F=00000000 00000000 C7C8E440 0A69602C 0BC04F54 0A5BE060 8A6FC491 00000000 00CC4B20
0BC04F54 0A6FC040 00
.....CC5B78 0ADBA930 0A5BE618 0BC04F54 00000000 IOAR=8A6FC491 IOAL=0000
IOAD=808A6FC491000000000A71C58000000000000000 EPST=0
.....A6FC040 MOV=8A6FC491 GETL=0000

```

## Move data back to application I/O area

```

FPR3.....TOD=B6B6146E WKAR=0A6FCC94 R0#F=000121F8 0A69602C D9C5D7D3 00000000 00000001 0A5BE060 0BC14F5B C4C5D7C1
D9E3D4E3 400060
.....18 0A6FC040 0001D9F1 8ADBA272 0001EE48 0001EC80 0E48B3AA MODU=8ADBA272 DATE=08/05/03P TIME=PQ6040 LCHA=01
1IAB

```

## REPL call starts

```

IRC1.....TOD=B6B61471 R015=00000000 00000C18 D9C5D7D3 0A69602C 0BC04F54 0A5BE060 00000000 8A71C580 D9E3D4E3
0A9F1048 0A6FC040 00
.....CC5B78 8ADBA272 0A5BE618 0A6FC9FC 00000001 LCRE=0A9F1048 TOKEN=E2E8E2F3404040400000000600000000
EPCB=0A69602C PCBA=0A52F
.....35C PCBD=000121F8 ESCD=00CC5B78 MADR=8AD477B0 EPST=0A6FC040 WKAR=0A6FCC94 SVIO=0BC04F54 SVIL=D9F1 SVSN=0001
.MCL0.....TOD=B6B61472 CALL=REPL EPST=0A6FC040 WKAR=0A6FCC94 EPCB=0A69602C CCID=8C LCID=8C PRGP=0A6960EC
LKFP=0A6960EC PCB=0A52F3
.....5C SSA=DEPARTMT
MODU=0AD4AD80 DATE=08/01/031 TIME=10.25E LCHA=E}
i~}}

```

## SSA traced

```

.SSAX.....SSA=DEPARTMT

```

## First SSA

```

..SSR9.....TOD=B6B61474 R15=0AD72D18 DATE=01/0310.2 0#10=00000000 00000C18 D9C5D7D3 0A69602C 0BC04F54 0AD4B68C
FFFFFE10 FF3C46E1 0
.....0CC4B20 000000EE 0A6FC040
...VSNA.....TOD=B6B61475 R0#9=00000000 00000C18 D9C5D7D3 0A69602C 0A6960EC 0A6FC588 8BC14F5B FF3C46E1 00CC4B20
000000EE R14=8AD72DD
.....A R15=0AD74478 EPCB=0A69602C FLGM=E0 DMAC=0AD09288 ARBA=00001004 UOWO=00000000 CCID=E0 CCNT=00000003
UBLK=00000000 KUBL
.....=00000000 DEDB=10 LKFP=0A6960EC PRGP=0A6960EC CLEV=0A6960EC MLTE=0A6960EC POPT=00 SGLS=01 CLOC=0AB38008
DMHR=0AA922C0 P
.....RBA=00001004 CRBA=00001008 NRBA=00001268 GRBA=00001268 XRBA=00001004 SGCD=01 PROF=0000 LEVL=01 KEYL=0B
FLGA=88 FLGB=00
.....ACCK=000C KEYO=0008 SDBS=0AD09210 MLTE=0A6960EC SFRX=00 SFSX=00 SFWX=00 SFZX=00 SFMX=00 PREF=0006
SCNT=00000002 SWC1=60
.....SWC2=02 SWC3=00 LOPR=82 SNAP=0A696168 EPST=0A6FC040 SCVL=01 REOP=80 FDLN=0B FDOF=0008 DEDB=80
COMP=0A4998D0 STAT=
.....WCH=00 MODU=0AD74478 DATE=08/01/031 TIME=10.28E LCHA=E}
i~}}
...VSNA.....TOD=B6B61478 R0#9=00000000 00000C18 D9C5D7D3 0A69602C 0A6960EC 0A6FC588 8BC14F5B 00000000 00CC4B20
000000EE R14=0A69616
.....8 R15=00000000 EPCB=0A69602C FLGM=E0 DMAC=0AD09288 ARBA=00001004 UOWO=00000000 CCID=E0 CCNT=00000003
UBLK=00000000 KUBL
.....=00000000 DEDB=10 LKFP=0A6960EC PRGP=0A6960EC CLEV=0A6960EC MLTE=0A6960EC POPT=00 SGLS=01 CLOC=0AB38008
DMHR=0AA922C0 P
.....RBA=00001004 CRBA=00001008 NRBA=00001268 GRBA=00001268 XRBA=00001004 SGCD=01 PROF=0000 LEVL=01 KEYL=0B
FLGA=88 FLGB=00
.....ACCK=000C KEYO=0008 SDBS=0AD09210 MLTE=0A6960EC SFRX=00 SFSX=00 SFWX=00 SFZX=00 SFMX=00 PREF=0006
SCNT=00000002 SWC1=40
.....SWC2=00 SWC3=00 LOPR=00 SNAP=0A696168 EPST=0A6FC040 SCVL=01 REOP=80 FDLN=0B FDOF=0008 DEDB=80
COMP=0A4998D0 STAT=
.....WCH=00 MODU=0AD74478 DATE=08/01/031 TIME=10.28E LCHA=E}
i~}}

```

## Validating segment name

```
..SSR9.....TOD=B6B6147D R15=00000000 DATE=01/0310.2 0#10=00000000 00000C18 D9C5D7D3 0A69602C 0A6960EC 0A6FC588
8BC14F5B 00000000 0
.....0CC4B20 000000EE 0A6FC040
...MRPL.....TOD=B6B6147D R0#9=00000000 00000C18 D9C5D7D3 0A69602C 0A6960EC 0AD4B68C 8BC14F5B 00000000 00CC4B20
0A5BE060 R14=8AD4B01
.....2 R15=0AD6AD40 EPCB=0A69602C FLGM=E0 DMAC=0AD09288 ARBA=00001004 UOW0=00000000 CCID=E0 CCNT=00000003
UBLK=00000000 KUBL
.....=00000000 DEDB=10 LKFP=0A6960EC PRGP=0A6960EC CLEV=0A6960EC MLTE=0A6960EC POPT=00 SGLS=01 CLOC=0AB38008
DMHR=0AA922C0 P
.....RBA=00001004 CRBA=00001008 NRBA=00001268 GRBA=00001268 XRBA=00001004 SGCD=01 PROF=0000 LEVL=01 KEYL=0B
FLGA=88 FLGB=00
.....ACCK=000C KEY0=0008 SDBS=0AD09210 MLTE=0A6960EC SFRX=00 SFSX=00 SFWX=00 SFZX=00 SFMX=00 PREF=0006
SCNT=00000002 SWC1=40
.....SWC2=00 SWC3=00 LOPR=80 SNAP=0A696168 EPST=0A6FC040 SCVL=01 REOP=80 FDLN=0B FDOF=0008 DEDB=80
COMP=0A4998D0 STAT=
.....WCH=00 MODU=0AD6AD40 DATE=08/01/03P TIME=PQ7304 LCHA=49 A{}
```

## Replace call handler

```
...PI09.....TOD=B6B61482 WKAR=0A6FCC94 R0#F=00000000 0BC04F54 0AB38008 0A69602C 0A6960EC 0AD4B68C 0A6960EC 00000000
00CC4B20 0A5BE0
.....60 0A6FC040 00CC5B78 0AD678C0 0A5BE6F0 8AD6AE8A 0AD678C0 MODU=0AD678C0 DATE=08/01/03
...PI09.....TOD=B6B61483 WKAR=0A6FCC94 R0#F=0A499672 00000000 0AB38008 0A69602C 0A6960EC 00000000 0A499666 00000000
0000025A 0A5BE0
.....60 0A6FC040 00CC5B78 0AD678C0 0A5BE6F0 0A115556 00000000 MODU=0AD678C0 DATE=08/01/03
```

## Process I/O area for replace

```
...MUH1.....TOD=B6B61484 WKAR=0A6FCC94 R0#F=00000260 00000008 0000000B 0A69602C 0A6960EC 0AD4B68C 00000008 00000000
00000260 0AA922
.....C0 0A6FC040 00CC5B78 0AD737A8 0A5BE6F0 8AD6B0A2 0AD737A8 MODU=0AD737A8 DATE=08/01/031 TIME=10.28E LCHA={
i~}}
...MUHE.....TOD=B6B61485 WKAR=0A6FCC94 R0#F=00000018 00000025 0A1150C9 0000022B 0AB3803D 0000022B 00000008 00000025
0AB38025 0AA922
.....C0 0A6FC040 00CC5B78 0AD73158 0A5BE738 8AD73850 0AD73158 MODU=0AD73158 DATE=08/01/031 TIME=10.28E LCHA={
i~}}
...NUHE.....TOD=B6B61486 WKAR=0A6FCC94 R0#F=00000018 00000025 0000001C 0000022B 00000000 00000004 0000003C 00000001
0AB38025 0AA922
.....C0 0A6FC040 00CC5B78 0AD73158 0A5BE738 8AD73850 0AD73158 MODU=0AD73158 DATE=08/01/031 TIME=10.28E LCHA={
i~}}
...MUH1.....TOD=B6B61487 WKAR=0A6FCC94 R0#F=00000018 00000025 0A1152F4 00000000 0AB38268 00000000 00000008 00000025
0AB38025 0AA922
.....C0 0A6FC040 00CC5B78 0AD737A8 0A5BE6F0 8AD73850 00000000 MODU=0AD737A8 DATE=08/01/031 TIME=10.28E LCHA={
i~}}
```

## Record changes to buffer

```
..MRPL.....TOD=B6B61488 R0#9=00000260 00000008 0000000B 0A69602C 0A6960EC 0AD4B68C 00000008 00000000 00000260
0AA922C0 R14=8AD6B0A
.....2 R15=00000000 EPCB=0A69602C FLGM=E0 DMAC=0AD09288 ARBA=00001004 UOW0=00000000 CCID=E0 CCNT=00000003
UBLK=00000000 KUBL
.....=00000000 DEDB=10 LKFP=0A6960EC PRGP=0A6960EC CLEV=0A6960EC MLTE=0A6960EC POPT=00 SGLS=01 CLOC=0AB38008
DMHR=0AA922C0 P
.....RBA=00001004 CRBA=00001008 NRBA=00001268 GRBA=00001268 XRBA=00001004 SGCD=01 PROF=0000 LEVL=01 KEYL=0B
FLGA=88 FLGB=00
.....ACCK=000C KEY0=0008 SDBS=0AD09210 MLTE=0A6960EC SFRX=00 SFSX=00 SFWX=00 SFZX=00 SFMX=00 PREF=0006
SCNT=00000002 SWC1=40
.....SWC2=00 SWC3=00 LOPR=80 SNAP=0A696168 EPST=0A6FC040 SCVL=01 REOP=80 FDLN=0B FDOF=0008 DEDB=80
COMP=0A4998D0 STAT=
.....WCH=00 MODU=0AD6AD40 DATE=08/01/03P TIME=PQ7304 LCHA=49 A{

.SEG4.....TOD=B6B6148D R15=00000000 DATE=01/0310.2 0#10=00000000 00000C18 0A52F35C 0A69602C 0A6960EC 0AD4B68C
8BC14F5B 00000000 0
.....0CC4B20 0A5BE060 0A6FC040
...NCL0.....R0#F=00000000 00000C18 0A52F35C 0A69602C 0A6960EC 8BC14F5B 8BC14F5B 00000000 00CC4B20 0A52F35C 0A6FC040
00CC5B78 0AD4AD
.....80 0A5BE660 00000030 00000000 EPCB=0A69602C CCID=E0 LCID=8C PRGP=0A6960EC LKFP=0A6960EC DBPC=0A52F35C
STC= A LEV=01 S
.....FD=DEPARTMT
.....0080000100400010100010B8800000C00080AD0 STAS=
```

## REPL call ends

```
OPMV.....TOD=B6B61490 R0#F=00000000 00000000 D9C5D7D3 0A69602C 0BC04F54 0A5BE060 0A6FC491 00000000 00CC4B20
0BC04F54 0A6FC040 00
.....CC5B78 0ADBA930 0A5BE618 0BC04F54 00000000 IOAR=0A6FC491 IOAL=0000
IOAD=800A6FC4910A1155560A71C58000000000000000 EPST=0
.....A6FC040 MOVP=0A6FC491 GETL=0000
```

```
SYN1.....TOD=B6B6149B R15=0AD35F68 DATE=05/0314.0 0#10=00CC4B20 0A5BE060 8A390E8E 0AE1EA34 0A5BE060 0A358698
0A9F1048 00000003 0
.....A35879C 0A5BE060 0A6FC040
```

## Begin synchronization point

```
.SLOG.....TOD=B6B6149C R0#F=0A6FC040 00000001 00000000 0AE1EA34 0A5BE060 00000000 0A9F1048 00000003 0A35879C
0A5BE060 0A6FC040 00
.....CC5B78 0AD34168 0A5BE738 8AD3617E 0AD34168 MODU=0AD34168 DATE=08/05/03U TIME=UP9BDN LCHA=N0112
```

## Log 5950 CI updates

```
..SLGE.....TOD=B6B6149F R15=000007B4 DATE=05/0313.5 0#10=00000004 00001530 0A6FC040 00000010 00CC6EF0 00000070
0A6FC7F4 00CC4B20 0
.....A369920 0A5BE060 0A6FC040
..SLGE.....TOD=B6B614A0 R15=00000004 DATE=05/0313.5 0#10=3A53400A 000014C0 0A6FC040 0AB38000 0A6FD044 00000000
00000002 00CC4B20 0
.....AD09288 0AD091C0 0A6FC040
..SLGE.....TOD=B6B614A1 R15=000007B4 DATE=05/0313.5 0#10=00000000 00001498 0A6FC040 00007E99 0A88F31E 00000088
0A6FC7F4 00CC4B20 0
.....A369920 0A5BE060 0A6FC040
..SLGE.....TOD=B6B614A2 R15=00000000 DATE=05/0313.5 0#10=00000000 00001498 0A6FC040 00007E99 0A88F31E 00000088
0A6FC7F4 00CC4B20 0
.....A369920 00000000 0A6FC040
```

## Logger calls Fast Path logger exit

```
.TLOG.....TOD=B6B614A3 R0#F=001D431E 0A5BE060 00000000 0A6FC27C 00000090 00000000 0A6FC7DC 0A6FC7DC 00000000
00CC4B20 0A6FC040 00
.....CC5B78 0AD34168 0A5BE738 8AD3486C 00000000 MODU=0AD34168 DATE=08/05/03U TIME=UP9BDN LCHA=N0112
TYN1.....TOD=B6B614A4 R15=00000000 DATE=05/0314.0 0#10=0A6FC040 00800001 00000000 0AE1EA34 305BE060 8A71C580
0A9F1048 00000003 0
.....A35879C 0A5BE060 0A6FC040
```

## End phase I

```
.SYN2.....TOD=B6B614A4 R15=0AD36748 DATE=05/0314.0 0#10=00000000 0A5BE060 8A390E8E 00000000 0A5BE060 0A358698
00CC4B20 00000003 0
.....A35879C 0A5BE060 0A6FC040
```

## Start phase II

```
..SLG2.....TOD=B6B614A5 R15=00000000 DATE=05/0314.0 0#10=00800001 00000000 0A5BE060 00000000 00000000 00000000
00CC4B20 00000003 0
.....A35879C 0A5BE060 0A6FC040
...SYP2.....TOD=B6B614A7 R0#F=00000000 00000000 00000000 00000000 0A5BE060 0A358698 00CC4B20 00000003 0A35879C
0A5BE060 0A6FC040 00
.....CC5B78 0AD370D0 0A5BE738 8AD36938 0AD370D0 MODU=0AD370D0 DATE=08/05/03P TIME=PQ6949 LCHA=94 A£}

....SPIX....TOD=B6B614A8 R15=0AD35920 DATE=01/03
.....AD09288 00000000 0A6FC040
....XPIX....TOD=B6B614A9 WKAR=0A6FCC94 R0#F=00000001 0A960CB0 00000000 0A6FC264 00000000 00000000 00CC4B20 0A960CB0
00000000 000000
.....00 0A6FC040 00CC5B78 0ADA6668 0A5BE810 8AD35AA2 0ADA6668 MODU=0ADA6668 DATE=08/05/03K TIME=KZC007 LCHA=77
0£}

....NPIX....TOD=B6B614AE WKAR=0A6FCC94 R0#F=0A6FC040 00000000 8ADA7284 0ADA4C78 09E6A03C 00001000 FFFFFFFE90 00000000
0AD09288 000000
.....00 0A6FC040 00CC5B78 0ADA6668 0A5BE810 8ADA4D58 00000000 MODU=0ADA6668 DATE=08/05/03K TIME=KZC007 LCHA=77
0£}

....NPIX....TOD=B6B614AF WKAR=0A6FCC94 R0#F=00000001 00000000 00000000 0A6FC264 00000000 00000000 00CC4B20 00000000
00000000 000000
.....00 0A6FC040 00CC5B78 0AD35920 0A5BE7C8 8AD35AA2 0ADA6668 MODU=0AD35920 DATE=08/01/03
```

## Release locks

```
...TYP2.....TOD=B6B614B0 R0#F=00000000 00000000 00000000 00000000 0A5BE060 0A358698 00000000 00000003 0A35879C
0A5BE060 0A6FC040 00
.....CC5B78 0AD370D0 0A5BE738 8AD37168 00000000 MODU=0AD370D0 DATE=08/05/03P TIME=PQ6949 LCHA=94 A£}
```

```
...SHDQ.....TOD=B6B614B1 R15=0AD30B50 DATE=/01/03PQ7 0#10=00000000 00000000 00000000 00000000 0A5BE060 0A358698
00CC4B20 00000003 0
.....A35879C 0A5BE060 0A6FC040
...SHDX.....TOD=B6B614B2 R15=00000000 DATE=/01/03PQ7 0#10=00000000 00000000 00000000 0A6966A8 00000000 00000000
```

```

00CC4B20 00000003 0
.....A696018 0A5BE060 0A6FC040
...SDEQ.....TOD=B6B614B2 R0#F=00000000 00000000 00000000 00000000 0A5BE060 0A358698 00CC4B20 00000003 0A35879C
0A5BE060 0A6FC040 00
.....CC5B78 0AD2F3B8 0A5BE738 8AD369BC 0AD2F3B8 MODU=0AD2F3B8 DATE=08/01/031 TIME=10.31K LCHA=KZC0077 1
...BENQ.....TOD=B6B614B3 DATE=01/0310.1 FC=08 UDNA=00000000 UDSR=0000 QBLK=00000000000000000000000000000000
PARM=2003225F0002292773
.....
...RENQ.....TOD=B6B614B4 R0#F=0A6FC040 0A6FC040 0A6FC4EC 00000000 0A5BE060 0A358698 00CC4B20 00000003 0A35879C
00000000 0A6FC040 00
.....CC5B78 0AD2DB78 0A5BE780 8AD2F582 00000000 MODU=0AD2DB78 DATE=08/01/031 TIME=10.10K LCHA=KZC0077 0
...TDEQ.....TOD=B6B614B5 R15=00000000 DATE=01/0310.3 0#10=0A6FC040 00000000 00000008 0A6966A8 00000000 00000001
00000000 00000003 0
.....A696018 00000000 0A6FC040

```

## Dequeue other resources

```

..TYN2.....TOD=B6B614B6 R15=8A71C580 DATE=05/0314.0 0#10=00000000 00800001 00000000 00000000 0A5BE060 0A358698
00CC4B20 00000003 0
.....A35879C 0A5BE060 0A6FC040

```

## End phase II

```

..SYN2.....TOD=B6B614B7 R15=0AD36748 DATE=05/0314.0 0#10=00000000 8A5BE060 00000000 0AE1EA34 0A5BE060 0A358698
00CC4B20 00000003 0
.....A35879C 0A5BE060 0A6FC040
..TYN2.....TOD=B6B614B8 R15=0AD36748 DATE=05/0314.0 0#10=00000000 8A5BE060 00000000 0AE1EA34 0A5BE060 0A358698
00CC4B20 00000003 0
.....A35879C 0A5BE060 0A6FC040

```



# Chapter 14. IMS Connect service aids

The service aids for IMS Connect include the IMS Connect Dump Formatter and trace options.

**Related tasks**

[“Collecting data about IMS Connect problems” on page 20](#)

If a problem occurs during IMS Connect execution, you need to collect logs, data sets, and dumps to determine the source of the problem.

## IMS Connect Dump Formatter

An IBM Software Support representative might ask you to use the IMS Offline Dump Formatter to diagnose an IMS Connect problem. The IMS Connect Dump Formatter enables you to format various IMS Connect internal control blocks under the ISPF IPCS environment.

### Accessing the IMS Connect Dump Formatter

You access the IMS Connect Dump Formatter from the IMS Offline Dump Formatter ISPF panels.

**About this task**

To access the IMS Connect Dump Formatter:

**Procedure**

1. On the IPCS Component Analysis panel, select DFSAAMPR.

The IMS Dump Formatting Primary Menu panel opens.

```
DFSAAMPR ----- IMS DUMP FORMATTING PRIMARY MENU -----
OPTION  ==>

  0 INIT          - IMS formatting initialization and content summary
  1 BROWSE        - Browse Dump data set (IPCS norm)          *****
  2 HI-LEVEL      - IMS Component level formatting            *USERID  - IMSDUMP
  3 LOW-LEVEL     - IMS ITASK level formatting                 *DATE    - 00/08/23
  4 ANALYSIS      - IMS dump analysis                         *JULIAN   - 00.236
  5 USER          - IMS user formatting routines              *TIME    - 16:06
  6 OTHER COMP    - Other IMS components (BPE, CQS...)         *PREFIX   - IMSDUMP
  7 OTHER PROD    - Other IMS-related products                *TERMINAL - 3278
  E EDA           - IMS Enhanced Dump Analysis               *PF KEYS  -
  T TUTORIAL      - IMS dump formatting tutorial              *****
  X EXIT          - Exit IMS dump formatting

Enter END or RETURN command to terminate IMS component formatting.

Use PFKeys to scroll up and down if needed.
```

*Figure 72. IMS Dump Formatting Primary Menu panel*

2. On the IMS Dump Formatting Menu panel, type 6 for Other Components and press **Enter**.
3. On the Other Components panel, type 6 for IMS Connect and press **Enter**.

**Results**

# Initializing a dump by using the IMS Connect Dump Formatter

Before you can format a dump, you must initialize it. After you initialize a dump, you can use the options on the IMS Connect Dump Formatting Menu to browse the dump data set, perform high-level or low-level IMS Connect formatting, or perform IMS Connect BPE formatting.

## About this task

To initialize a dump:

## Procedure

1. Open the IMS Connect Dump Formatting Menu panel, as shown in the following figure.

```
----- IMS CONNECT DUMP FORMATTING MENU -----
OPTION ==>

  0 INIT      - Show BPE status and initialize dump
  1 BROWSE    - Browse dump data set (IPCS norm)
  2 HI-LEVEL  - IMS Connect component level formatting
  3 LOW-LEVEL- IMS Connect level formatting
  4 EXT TRACE- BPE external trace formatting
  X EXIT      - Exit IMS dump formatting

*****
*USERID  - IMSDUMP
*DATE    - 00/08/23
*JULIAN   - 00.236
*TIME     - 16:08
*PREFIX   - IMSDUMP
*TERMINAL- 3278
*PF KEYS  - 24
*****

Enter  END or RETURN command to terminate IMS Connect formatting.
```

Figure 73. IMS Connect Dump Formatting Menu panel

2. Type 0 for Show BPE status and initialize dump, and press **Enter**.

The IMS Connect Dump Initialization panel opens, as shown in the following figure, along with the message that the symbol HWSCSD was not found.

```
----- IMS CONNECT DUMP CONTENT STATUS AND CONTROL -----
COMMAND ==>
SYMBOL HWSCSD NOT FOUND
Enter the IMS Connect jobname or ASID to cause the IMS
Connect IPCS symbols to be set for this dump. Press
ENTER with blank jobname and ASID fields to list all
IMS Connect address spaces in the dump.

-----
JOBNAME      ASID      A.S. TYPE      DUMPED?
-----
IMS Connect      ???      NO

SDWA Address:      BPE Release:
CSCD Address:      ??? Release:
??? Sys Name:      ??? Product:
```

Figure 74. IMS Connect Dump Initialization panel

3. Specify either the job name or the address space ID (ASID) of the IMS Connect address you want to format and press **Enter**.

After you specify a job name or ASID, the remaining fields are completed in the initialization panel, as shown in the following figure.

```

----- IMS CONNECT DUMP CONTENT STATUS AND CONTROL -----
COMMAND ===>

Enter the IMS Connect jobname or ASID to cause the IMS
Connect IPCS symbols to be set for this dump. Press
ENTER with blank jobname and ASID fields to list all
IMS Connect address spaces in the dump.

-----
JOBNAME          ASID          A.S. TYPE      DUMPED?
-----
IMS Connect      HWS71R          0042          HWS          YES

BP SDWA Address: 00000000      BPE Release:  010600
BPE CSCD Address: 450015A0      HWS Release:  0A0100
HWS Sys Name:      HWS Product:  5635-A01

```

Figure 75. IMS Connect Dump Initialization panel after initialization

4. To obtain a list of all dumped IMS Connect address spaces, leave the job name and ASID fields blank.
5. Press **F3** to return to the IMS Connect Dump Formatting Menu.

## IMS Connect traces

You can trace two types of information about IMS Connect: information about the messages that are processed by IMS Connect and information about the IMS Connect subsystem.

The information about the messages that are processed by IMS Connect is captured by the IMS Connect Recorder Trace facility. Information about the IMS Connect subsystem is captured by the Base Primitive Environment (BPE) tracing services. The information that is captured by both types of traces is stored in BPE trace tables.

**Recommendation:** Use the BPE trace tables to store information that is captured by the IMS Connect Recorder Trace facility. The IMS Connect Recorder Trace facility can store information in either a BPE trace table or in an IMS Connect-managed Recorder Trace data set (HWSRCRDR).

### IMS Connect Recorder Trace facility

When the IMS Connect Recorder Trace facility is active, IMS Connect takes a snapshot of the first 670 bytes of messages at key points during IMS Connect processing. The first 670 bytes of most messages include the message header and the message data. Included in the information that is captured by the IMS Connect Recorder Trace facility are elements such as:

- Time stamps
- The client ID
- The user message exit routine

**Recommendation:** If network security credentials are included in IMS Connect client input messages, enable the BPE External Trace facility for the IMS Connect Recorder Trace facility. If network security credentials are passed to IMS Connect, the size of both input and output messages to and from IMS Connect might be larger than 670 bytes and the BPE External Trace facility would be required to capture the data of the entire message.

When the BPE trace level is set to MEDIUM, or when the HWSRCRDR trace data set is used to capture the trace data, IMS Connect takes a snapshot of each input and output message at the following points:

- Immediately before a message is passed to a user message exit routine by IMS Connect
- Immediately after a message is returned to IMS Connect by a user message exit routine

When the BPE trace level is set to HIGH, the trace information includes a complete snapshot of the message. The recorder trace record also includes a message type code that identifies the source or destination type. IMS Connect takes a snapshot of each complete message, including the message prefixes, at the following points:

Table 131. IMS Connect recorder trace points with BPE trace level HIGH

ID	Type of message	Trace point
TR	ISC, MSC, ODBM, OTMA, or OM	A TCP/IP message was received by IMS Connect.
TS	ISC, MSC, ODBM, OTMA, or OM	A TCP/IP message was sent by IMS Connect.
IR	OTMA	An XCF message was received from OTMA by IMS Connect.
IS	OTMA	An XCF message was sent to OTMA by IMS Connect.
SR	ISC, MSC, ODBM, or OM	An SCI message was received by IMS Connect.
SS	ISC, MSC, ODBM, or OM	An SCI message was sent by IMS Connect.

**Recommendation:** Setting the trace level to HIGH can cause a significant impact to IMS Connect performance. Do not set the trace level to HIGH except when troubleshooting problems with connectivity.

If you use the BPE External Trace facility for the IMS Connect Recorder Trace, BPE manages the trace data sets. You must perform certain setup tasks, such as defining the GDG base and specifying the EXTTRACE parameter in the BPE configuration member in the IMS.PROCLIB data set. You can then start the IMS Connect Recorder Trace facility by issuing the BPE command **UPDATE TRACETABLE** NAME(RCTR) OWNER(HWS) LEVEL(MEDIUM) EXTERNAL(YES) or by specifying TRCLEV=(RCTR,MEDIUM,HWS) in the BPECFG member of the IMS.PROCLIB data set and restarting IMS Connect. To view the results of the trace, you can use the BPE Dump Formatter.

If you use a data set that is managed by IMS Connect, it must be a single, fixed-block data set with a fixed logical record length. The data set must be identified by an HWSRCORD DD statement in the IMS Connect startup JCL. When this data set fills, the IMS Connect Recorder Trace facility is disabled. Starting a new trace overwrites any existing trace data in the data set. To change the data set, you must shut down IMS Connect. From an IMS perspective, you can manage this trace data set in the same way as the IMS Monitor Trace (IMSMON) data set.

## IMS Connect subsystem traces

The IMS Connect subsystem traces are captured by using BPE services. Subsystem traces trace events that occur within the IMS Connect address space and write them to BPE-managed trace tables. Specific types of events are captured in specific trace table types by IMS Connect or BPE. For example, TCP/IP activity is captured in the IMS Connect TCPI trace table type and BPE dispatcher events are captured in the BPE DISP trace table type.

BPE traces of the IMS Connect are enabled either by issuing the UPDATE TRACETABLE command or by including a TRCLEV parameter for IMS Connect in the BPECFG configuration member and restarting IMS Connect support.

For more information about configuring BPE and preparing for IMS Connect diagnosis, see *IMS Version 15.5 System Definition*.

## Enabling and starting all IMS Connect traces at the same time

You can enable tracing for all trace table types, excluding the Recorder Trace facility trace table (RCTR), by specifying an asterisk in the first position of the TRCLEV parameter in the BPECFG file member, for example TRCLEV=(\*,HIGH,HWS). When the TRCLEV parameter is specified in the BPECFG file, the IMS Connect traces run continuously and resume every time IMS Connect starts.

**Recommendation:** Do not run IMS Connect Recorder trace facility continuously. The IMS Connect Recorder Trace facility can affect performance and should be used only when detailed information is needed about a specific problem with IMS Connect message handling. Remember to disable the IMS Connect Recorder Trace after the problem is resolved.

To enable all traces except the Recorder Trace facility, specify the following parameters:

```
TRCLEV=(*,HIGH,HWS)
```

BPE exempts the Recorder trace (RCTR) from wildcard arguments, so it must be enabled with the following trace level configuration statement:

```
TRCLEV=(RCTR,MEDIUM,HWS)
```

**Recommendation:** Start the IMS Connect Recorder Trace with the command, rather than with the TRCLEV parameter. Only use the TRCLEV parameter when you are diagnosing problems that prevent you from issuing commands.

BPE traces can be started and stopped by using BPE commands. If you start a trace by using the UPDATE TRACETABLE command, the traces are only active for the current execution of IMS Connect. The trace tables that are used by BPE to store the trace information, such as the RCTR trace table, are also managed by using BPE commands.

### Related tasks

“Formatting a BPE trace entry” on page 12

You can format a BPE trace entry by using either the Interactive Problem Control System (IPCS) or a batch job.

## Configuring BPE for an external trace of IMS Connect

Before you can start a BPE external trace for IMS Connect, you must first configure BPE and the external trace data set.

### About this task

The following steps configure the external trace data sets and BPE for an external BPE trace of IMS Connect.

### Procedure

1. Define the generation data group (GDG) base.

```
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DEFINE GENERATIONDATAGROUP -
    (NAME(IMSTESTL.RCTR.GDG01) -
     NOEMPTY -
     SCRATCH -
     LIMIT(255))
//
```

2. In the BPECFGxx member of the IMS.PROCLIB data set, specify the following parameters for IMS Connect:

- EXTTRACE
- TRCLEV

In the EXTTRACE statement, specify the data set name of the generation data set group that you created in Step 1 and specify HWS in the COMP() parameter to identify IMS Connect as the BPE component to which this EXTTRACE statement applies.

```
EXTTRACE(GDGDEF( DSN(IMSTESTL.RCTR.GDG01)
                 UNIT(SYSDA) VOLSER(000000)
                 SPACE(1) SPACEUNIT(CYL)
                 BLKSIZE(32760) ) COMP(HWS) )
```

When the Recorder Trace (TRCLEV type parameter RCTR) trace level is set to MEDIUM, message data is recorded before a message is sent to a user message exit and after the exit returns a message.

You can set the trace level to HIGH to enable additional trace points. When these trace points are enabled, a trace record is also written when IMS Connect sends a message to or receives a message from a TCP/IP endpoint, a DRDA client, an SCI client, or OTMA via XCF. These records are not collected when the trace level is set to any value other than HIGH or when IMS Connect uses internal tracing.

**Recommendation:** Use the BPE External Trace with the TRCLEV parameter HIGH only when troubleshooting IMS Connect client or OTMA-related problems. A large amount of trace data is recorded when the trace level is set to HIGH.

In the TRCLEV parameter, you must specify EXTERNAL=YES to enable the External Trace facility:

```
TRCLEV=(RCTR,MEDIUM,HWS,EXTERNAL=YES)
```

**Tip:** If TRCLEV=(RCTR,MEDIUM,HWS,EXTERNAL=YES) is specified in the BPECFG member, the Recorder trace resumes every time IMS Connect restarts. If you do not want the Recorder trace to resume when IMS Connect restarts, issue the BPE command F hwsjobname,UPD TRTAB NAME(RCTR) LEVEL(MEDIUM) EXTERNAL(YES)

## What to do next

If BPE is correctly configured for an external IMS Connect Recorder trace, BPE issues message BPE0044I BPE EXTERNAL TRACE FUNCTION NOT ACTIVE. You can now start the trace.

### Related tasks

[Tracing BPE components \(System Administration\)](#)

[Setting up tracing for BPE-managed address spaces \(System Definition\)](#)

### Related reference

[BPE configuration parameter member of the IMS PROCLIB data set \(System Definition\)](#)

## Starting an external trace for IMS Connect

You can start an external trace of IMS Connect by modifying the IMS Connect startup procedure or by issuing the BPE **UPDATE TRACETABLE** command.

### Before you begin

Before you can start a BPE external trace for IMS Connect, you must first define a generation data group (GDG) base and code an EXTTRACE statement for IMS Connect in the BPECFGxx member of the IMS.PROCLIB data set.

### About this task

Start the IMS Connect Recorder Trace by using one of the following methods.

### Procedure

- Include BPECFG=BPECFGxx in the IMS Connect startup procedure, where BPECFGxx is the BPE PROCLIB member that contains the EXTTRACE statement for IMS Connect, and restart IMS Connect. The BPE external trace starts when IMS Connect is started and BPE issues the following message:

```
BPE0046I EXTERNAL TRACE DATA SET  
IMSTESTL.RCTR.GDG01.G0001V00 OPENED ON VOL=000000
```

- While IMS Connect is running, issue the BPE command **UPDATE TRACETABLE**. For example:  
F HWS1,UPDATE TRACETABLE NAME(RCTR) OWNER(HWS) LEVEL(MEDIUM) EXTERNAL(YES).  
When the command completes successfully, the following messages are issued:

```
BPE0032I UPDATE TRACETABLE COMMAND COMPLETED  
BPE0046I EXTERNAL TRACE DATA SET  
IMSTESTL.RCTR.GDG01.G0003V00 OPENED ON VOL=000000
```

**Note:** If EXTERNAL(NO) is specified on either the UPDATE TRACETABLE command or in the TRCLEV parameter in the BPECFGxx PROCLIB member, IMS Connect writes the Recorder Trace data to the in-core trace tables. IMS Connect writes to the in-core trace tables by default.

**Recommendation:** Use the BPE Trace facility. If the BPE Trace facility is in use, setting the trace level to HIGH captures records when messages are passed to and from TCP/IP clients and OTMA in addition to the normal trace records. If the HWSRCDR trace data set is used, these additional trace records are not captured.

#### Related tasks

[Starting and stopping BPE external tracing \(System Administration\)](#)

#### Related reference

[BPE configuration parameter member of the IMS PROCLIB data set \(System Definition\)](#)

## Stopping an external BPE trace of IMS Connect

Stop an external BPE trace of IMS Connect by issuing the BPE command **UPDATE TRACETABLE**.

### About this task

For example, F HWS1,UPDATE TRACETABLE NAME(RCTR) OWNER(HWS) LEVEL(NONE) EXTERNAL(NO).

**Note:** To deallocate the GDG data set, you must specify EXTERNAL(NO) on the **UPDATE TRACETABLE** command.

### What to do next

After the successful execution of the UPDATE TRACETABLE command, BPE issues the following messages.

```
BPE0032I UPDATE TRACETABLE COMMAND COMPLETED
BPE0046I EXTERNAL TRACE DATA SET
IMSTESTL.RCTR.GDG01.G0001V00 CLOSED ON VOL=000000
BPE0044I BPE EXTERNAL TRACE FUNCTION NOT ACTIVE
```

#### Related tasks

[Starting and stopping BPE external tracing \(System Administration\)](#)

## Displaying the status of an external trace of IMS Connect

Display the status of an external BPE trace of IMS Connect by issuing the BPE command **DISPLAY TRACETABLE**.

### Example

For example, F HWS1,DISPLAY TRACETABLE NAME(RCTR).

### What to do next

If the BPE external trace of IMS Connect is active, BPE displays the following messages:

```
BPE0030I TABLE OWNER LEVEL #PAGES EXT #ENTRIES #CYCLES
BPE0000I RCTR HWS MEDIUM 300 YES 4 0
BPE0032I DISPLAY TRACETABLE COMMAND COMPLETED
```

If the BPE external trace of IMS Connect is inactive, BPE displays the following messages:

```
BPE0030I TABLE OWNER LEVEL #PAGES EXT #ENTRIES #CYCLES
BPE0000I RCTR HWS NONE 300 NO 0 0
BPE0032I DISPLAY TRACETABLE COMMAND COMPLETED
```

## Formatting the trace data from an external trace of IMS Connect

The following example shows the JCL that can be used to format the trace data from an external trace of IMS Connect.

### Example

```
//STEP01 EXEC PGM=IKJEFT01,REGION=0M,COND=(0,LT)
//STEPLIB DD DISP=SHR,DSN=IMSB LD.I11RTS19.CRESLIB
//SYSTSPRT DD SYSOUT=*
//IPCSPRNT DD SYSOUT=*
//IPCSPARM DD DISP=SHR,DSN=SYS1.PARMLIB
//SYSTSIN DD *
  ALLOC F(IPCSDDIR) DA('IMSTESTL.DDIR') SHR REUSE
  ALLOC F(INFILE) DA('IMSTESTL.RCTR.GDG01.G0001V00') SHR REUSE
  IPCS NOPARM
  SETDEF DSN('IMSTESTL.RCTR.GDG01.G0001V00')
  SETDEF NOPROBLEM PRINT NOTERMINAL
  VERBX BPETRFM0 'TRACE(TYPE(ALL))'
  END
//
```

### Related tasks

[“Formatting a BPE trace entry” on page 12](#)

You can format a BPE trace entry by using either the Interactive Problem Control System (IPCS) or a batch job.

## Tracing to the HWSRCDR data set

If you are not using BPE to manage the output of the IMS Connect Recorder Trace facility, you can enable and start a recorder trace that is managed by IMS Connect.

### About this task

To trace to the HWSRCDR data set:

### Procedure

1. Allocate an HWSRCDR data set.
2. Include an HWSRCORD DD statement in the IMS Connect startup JCL.
3. Issue one of the following commands:
  - IMS Connect WTOR command **RECORDER OPEN**
  - IMS Connect z/OS command **UPDATE MEMBER TYPE(IMSCON) START(TRACE)**
  - IMS Connect type-2 command **UPDATE IMSCON TYPE(CONFIG) START(RECORDER)**

### Related reference

[RECORDER command \(Commands\)](#)

## Recorder log record mappings

IMS Connect trace recorder logs are mapped with the HWSUSTAT DSECT or the HWSLRCTR DSECT.

If the IMS Connect trace facility is activated with the HWSRCDR internal data set, standard record log records are written that are mapped with the HWSUSTAT DSECT.

If the IMS Connect trace facility is activated with an external, BPE-managed data set and the trace level is set to HIGH, extended log records are also written in addition to the standard log records. These additional records contain more information, including a complete snapshot of the message, and are mapped with the HWSLRCTR DSECT.

Both DSECTS are in the HWSUSTAT macro in the IMS.SDFSMA C data set.



## Standard recorder log record mapping

When the IMS Connect line trace facility is activated by the IMS Connect WTOR command **nnRECORDER OPEN**, the IMS Connect type-2 command **UPDATE IMSCON TYPE(CONFIG) START(RECORDER)**, or the z/OS modify command **UPDATE MEMBER TYPE(IMSCON) START(TRACE)**, IMS Connect writes the log records to the HWSRCDR data set. If the trace facility is activated with a BPE data set, these records are written to the specified BPE data set instead.

The following examples provide mapping information to help you navigate and interpret the contents of the HWSRCDR data set, or an external BPE data set that contains the HWSUSTAT log records. The DSECT for this mapping is located in the HWSUSTAT macro of the IMS.SDFSMA data set. The DSECT name is HWSUSTAT. When the trace level for an external data set is set to HIGH, the data set contains both standard log records (mapped by the HWSUSTAT DSECT) and extended log records (mapped by the HWSLRCTR DSECT).

**This topic contains Diagnosis, Modification, and Tuning information.**

```
*****
*          COMMON SECTION          32_BYTES
*****
USTAT_NEXT   DS  F          NEXT POINTER
USTAT_EYE    DS  CL4'ICON'    EYECATCHER
USTAT_CALLID DS  CL2          CALLER ID
*
*          CHARS "AE" - ADAPTER MSG ERROR
*          CHARS "AR" - ADAPTER MSG RECEIVE
*          CHARS "AX" - ADAPTER MSG SEND
*          CHARS "ER" - IMS TM MSG READ ERROR
*          CHARS "ME" = MSC ERROR
*          CHARS "MR" = MSC RECEIVE
*          CHARS "MS" = MSC SEND
*          CHARS "OE" - IMS DB MSG READ ERROR
*          CHARS "OR" - IMS DB MSG RECEIVE
*          CHARS "OX" - IMS DB MSG SEND
*          CHARS "RC" - IMS TM MSG RECEIVE
*          CHARS "RE" = OTMA REMOTE ALTPCB
*          ERROR
*          CHARS "RR" = OTMA REMOTE ALTPCB
*          RECEIVE
*          CHARS "RS" = OTMA ALTPCB IMS to IMS
*          SEND
*          CHARS "SN" - IMS TM MSG SEND
*          CHARS "TO" = TIMEOUT EVENT
*
USTAT_SMFHDR DS  0C          SMF HEADER
SMFITOCLEN   DS  CL2          SMF LENGTH
SMFITOCSEG   DS  CL2          INTERNAL WORK
SMFITOCFLG   DS  X            INTERNAL FLAG
SMFITOCRTY   DS  X            RECORD TYPE
SMFITOCTME   DS  CL4          TIME OF TRACE
SMFITOCDTE   DS  CL4          SEQUENCE NUMBER
SMFITOCSID   DS  CL4          RESERVED
DS  CL4          RESERVED
*****
*          UOW PROGRESSION TIME STAMP SECTION
*****
SMFITOCCID   DS  CL8          CLIENT NAME
USTAT_TSMREC DS  D            TIME HWSW MSG RECEIVED
USTAT_TSMNQ  DS  D            TIME HWSW MSG ENQUEUED
USTAT_TDMDQ  DS  D            TIME 1ST DST MSG DEQUEUED
USTAT_TCLRQ  DS  D            TIME DST CLR DENQUEUED
USTAT_TERROR DS  D            TIME ERROR OCCURRED
USTAT_NMSGX  DS  H            NUMBER OF MSGS TRANSMITTED
USTAT_NMSGR  DS  H            NUMBER OF MSGS RECEIVED
USTAT_LTOKEN DS  CL8          SOCKET LOGON TOKEN
USTAT_SMFITOCL EQU *-USTAT_SMFHDR LENGTH OF SMF
*****
*          INPUT MSG
*****
USTAT_IN_EYE DS  CL4'*IPB'    EYECATCHER
*
*          *IPB IS THE INPUT TO THE EXIT
*          FOR EITHER RECEIVE OR SEND
*          USTAT_CALLID = AE - ADAPTER ERROR
*          AR - ADAPTER RECEIVE
*          AX - ADAPTER SEND
*          ER - TM READ ERROR
*
```

```
*
*
* ME - MSC ERROR
* MR - MSC RECIEVE
* MS - MSC SEND
* OE - DB ERROR
* OR - DB RECIEVE
* OX - DB SEND
* RC - TM RECEIVE
* RE - OTMA REMOTE
*      ALTPCB ERROR
* RR - OTMA REMOTE
*      ALTPCB RECIEVE
* RS - OTMA REMOTE
*      ALTPCB SEND
* SN - TM SEND
*
* for ICONRC and *IPB
*   (USTAT_CALLID = "RC")
*
* THE LOGGED DATA STARTING AT OFFSET X'60' IS AS FOLLOWS:
*   (INPUT TO EXIT FROM CLIENT)
*       1111
*       IRM
*       11zzTRANCODEDATA
*       X'00040000'
*****
* for ICONSN and *IPB
*   (USTAT_CALLID = "SN")
*
* THE LOGGED DATA STARTING AT OFFSET X'60' IS AS FOLLOWS:
*   (INPUT TO EXIT FROM IMS APPLICATION)
*       OTMA CONTROL HEADER followed by
*       OTMA STATE DATA HEADER (if present) followed by
*       OTMA SECURITY DATA HEADER (if present) followed by
*       OTMA USER DATA HEADER (if present) followed by
*       DATA TO BE SENT
*       11zzTRANCODEDATA
*
* USTAT_MSG_I    DS CL202             MSG
*****
* for ICONAR and *IPB
*   (USTAT_CALLID = "AR")
*
* THE LOGGED DATA STARTING AT OFFSET X'60' IS AS FOLLOWS:
*   (INPUT TO EXIT FROM CLIENT)
*       1111
*       IRM
*       XML
*****
* for ICONAX and *IPB
*   (USTAT_CALLID = "AX")
*
* THE LOGGED DATA STARTING AT OFFSET X'60' IS AS FOLLOWS:
*   (INPUT TO EXIT FROM OUTPUT FROM EXIT OF APPLICATION OUTPUT DATA)
*       1111
*       11zzDATA
*       CSM
*
* USTAT_MSG_I    DS CL202             MSG
*****
*
* OUTPUT MSG
*****
*
* USTAT_OUT_EYE DS CL4'*OPB'          EYECATCHER
*
*                               *OPB IS THE OUTPUT FROM THE EXIT
*                               FOR EITHER RECEIVE OR SEND
*                               USTAT_CALLID = AE - ADAPTER ERROR
*                               AR - ADAPTER RECEIVE
*                               AX - ADAPTER SEND
*                               ER - TM READ ERROR
*                               ME - MSC ERROR
*                               MR - MSC RECIEVE
*                               MS - MSC SEND
*                               OE - DB ERROR
*                               OR - DB RECIEVE
*                               OX - DB SEND
*                               RC - TM RECEIVE
*                               RE - OTMA REMOTE
```



*			CHARS "IR" - XCF RECEIVE
*			CHARS "IS" - XCF SEND
*			CHARS "SR" - SCI RECEIVE
*			CHARS "SS" - SCI SEND
*			CHARS "TR" - TCP/IP RECEIVE
*			CHARS "TS" - TCP/IP SEND
	DS	H	RESERVED 4 FUTURE CALLER ID
LRCTR_MSG_TYPE	DS	CL4	MESSAGE TYPE
*			CHARS "ISC " - ISC MESSAGE
*			CHARS "MSC " - MSC MESSAGE
*			CHARS "ODBM" - ODBM MESSAGE
*			CHARS "OM " - OM MESSAGE
*			CHARS "OTMA" - OTMA MESSAGE
	DS	CL4	RESERVED (MSG TYPE)
	DS	F	RESERVED
LRCTR_SOURCE	DS	CL8	MESSAGE SOURCE
	ORG	LRCTR_SOURCE	OR
LRCTR_PORTID	DS	CL8	MESSAGE PORT
LRCTR_CLIENT	DS	CL8	CLIENT NAME
	ORG	LRCTR_CLIENT	OR
LRCTR_TPIPE	DS	CL8	TPIPE NAME
LRCTR_TIMSTMP	DS	D	RECORD'S TIME STAMP
LRCTR_LTOKEN	DS	CL8	ICON Socket Logon token
	DS	F	RESERVED
LRCTR_IO_EYE	DS	CL4	'*IPB' OR '*OPB' INPUT OR OUTPUT
LRCTR_IO_IPB	EQU	C'*IPB'	INBOUND MESSAGE:
*			*IPB DENOTES THE BEGINNING OF A
*			MESSAGE RECEIVED BY IMS CONNECT.
LRCTR_IO_OPB	EQU	C'*OPB'	OUTBOUND MESSAGE:
*			*OPB DENOTES THE BEGINNING OF A
*			MESSAGE SENT BY IMS CONNECT.

## IMS Connect recorder trace for TM error scenarios

For IMS Transaction Manager (TM) transactions, IMS Connect (ICON) will return error codes with either the OTMA sense code, the ICON error codes in OMUSR\_RETCODE and OMUSR\_RESCODE, or the error message in the application data.

### ICONSN Record

The ICONSN record traces messages ICON sent to the client. If an error occurs in ICON or an error is returned from OTMA, the error codes and messages can be found in the ICONSN trace.

#### Example Scenarios:

- When a security violation with userid NOT AUTHORIZED occurs, ICON sends an error response to the client with DFS1292E SECURITY VIOLATION in the application data.
- When a transaction request to a stopped transaction occurs, ICON sends an error response to the client with DFS065 TRAN/LTERM STOPPED in the application data.
- When a ResumeTpipe request to DATASTORE is stopped in ICON and is in a disconnected state, ICON sends an error response to the client with error code OMUSR\_RETCODE=4 OMUSR\_RESCODE='STP/CLSE'.
- When a ResumeTpipe request to an invalid data store is made, ICON sends an error response to the client with error code OMUSR\_RETCODE=4 OMUSR\_RESCODE='NFNDST'.
- The client issues ResumeTpipe Single Wait Forever and then the client sends data on the same connection. After sending the ResumeTpipe Wait Forever request to OTMA, ICON issues a TCP/IP read on the socket connection to check for client disconnects. TCP/IP read returns Return\_value=10, Return\_code=0, and Reason\_code=0. The positive Return\_value means the client sent data, which ICON flags as a protocol violation because ICON is in the middle of a ResumeTpipe request. ICON issues a HWSP1495E message to the MVS console (HWSP1495E OTMA PROTOCOL VIOLATION; R=20, C=CLIENT02, DS=IMS1, M=SCVC).

### ICONER Record

The ICONER record traces errors when ICON is not able to send a response to the client.

#### Example Scenarios:

- On a Send-only transaction with an invalid transaction name, OTMA sends ICON an error response with sense code x1A and reason code x1D. However, ICON cannot send the error response to the client because ICON does not send any messages back to the client when Send-only protocol is used. The OTMA response message will be captured in the ICONER record trace.

## ICONTO Record

The ICONTO record traces timeout conditions.

## Error scenarios without trace records

There are some scenarios where ICON does not capture the trace record of the error codes or error messages that occur in either IMS Connect or OTMA.

### Example Scenarios:

- The client issues ResumeTpipe Single Wait Forever and then the client application is terminated as an ungraceful shutdown. The ICON's TCP/IP read (BPX1RED) returns Return\_value=-1 which indicates a session error. TCP/IP read returns Return\_value=-1, Return\_code=1121, and Reason\_code=1990066833 (x769E0291). ICON issues a HWSP1415E message to the z/OS system console. The recorder trace captures the ICONRC record, but there are no other recorder traces because no message is sent back to the client after termination.
- The client issues ResumeTpipe Single Wait Forever and then the client closes socket connection with the TCP/IP disconnect call. After sending the ResumeTpipe Wait Forever request to OTMA, ICON issues a TCP/IP read on the socket connection to check for client disconnects. When the client closes the connection, ICON's TCP/IP read (BPX1RED) returns Return\_value=0, which indicates the client has closed the connection. TCP/IP read returns Return\_value=0, Return\_code=0, and Reason\_code=0. The recorder trace captures the ICONRC record, but there are no other trace records because no message was sent back to the client.



---

## Chapter 15. IRLM - Internal resource lock manager service aids

Several service aids can help you analyze internal resource lock manager (IRLM) problems. Additionally, IRLM generates diagnostic messages that begin with the prefix DXR.

### About this task

---

## IRLM dumps

IRLM uses the SDUMP system services of z/OS when failures occur.

IRLM uses the SDUMP system services of z/OS in the following situations:

- Within the IRLM address space
- While executing IRLM code or IMS code within the IMS address space
- While executing IRLM code for exits from SLM within the IMS address space

SDUMP dumps the IRLM address space to a SYS1.DUMPxx data set without formatting it. When dump processing completes, you can format the dump offline by specifying IRLM on the VERBEXIT subcommand in IPCS. If more than one IRLM is active in the system at the time the dump was taken, you must also specify the z/OS subsystem name (IRLMNM in the IRLM procedure). IRLM dump formatters that are shipped with a particular release of IRLM are specific to that release. If multiple IRLM releases are active, IPCS must be configured to have access to the appropriate IRLM PDS. The release of the IRLM in the dump must match the release of the IRLM in the IPCS STEPLIB.

To access z/OS component trace entries for IRLM, use the IPCS CTRACE or VERBX command. To see the syntax of the VERBX command for displaying traces, enter: IPCS VERBX IRLM 'help'.

### Examples:

- If only one IRLM is in the dump, this command formats the IRLM address space:

```
VERBX IRLM 'SUBsys=IRLM'  
or  
VERBX IRLM  
or  
VERBX IRLM 'SUB=IRLM'
```

- If more than one IRLM is in the dump, this command formats the KRLM address space:

```
VERBX IRLM 'SUBsys=KRLM'  
or  
VERBX IRLM 'SUB=KRLM'
```

If you want to format dumps online during the abnormal termination process, you must change the FMTO parameter to request a SNAP dump.

**Tip:** Under the direction of IBM Software Support, you can use the Modify DIAG command to take diagnostic dumps.

---

## SYS1.LOGREC records

The internal resource lock manager (IRLM) generates a software LOGREC record when the IRLM detects a program error.

You can use the z/OS IFCEREP1 service aid to obtain a listing of the SYS1.LOGREC data set that contains the LOGREC entries for the IRLM.

## z/OS component trace

Use the z/OS **TRACE CT** command to start, stop, or modify an internal resource lock manager (IRLM) diagnostic trace.

### About this task

IRLM does not support all the options that are available on the TRACE command.

You can use the **TRACE CT** command to trace interactions with a DBMS, member and group events, z/OS locking components, and so on.

The following example shows trace output for a lock request using the DBM and SLM sublevel traces.

The z/OS Interactive Problem Control System (IPCS) subcommand that produced this output is: **CTRACE COMP(IRLE) SUB((DBM)) FULL**

#### **COMPONENT TRACE FULL FORMAT COMP (IRLE) SUBNAME ((DBM))**

```
COMPONENT TRACE FULL FORMAT
COMP(IRLE) SUBNAME((DBM))
**** 02/10/09
MNEMONIC ENTRY ID TIME STAMP DESCRIPTION
-----
DBM 00000002 18:42:05.816178 RLPL format
+0000 ID..... DXRRL100-01: START A REQUEST
+0020 TLA1..... 000100C8 07166220
+0028 RLPL..... 00000000 06545768 00000000 80000000 00000000
+003C 00000000 006B12C8 008FBBC0 0090B000 00906048
+0050 00316545 06545060 00000000 00316545 06545060
+0064 00000000 00000000 00000000 0423AD20 09000058
+0078 C8806D01 D7000000 00000000 00000000 00000000
+008C 00000000 00000000 80000000 00000000 00000000
+00A0 006B12C8 008FBBC0 02060000 8A000000 00000000
+00B4 00000000 006B5BE4 00000000 00000000 00000000
+00C8 00000000 00000000 00000000 00000000 00000000
+00DC 00000000 00000000 00000000 00000000 00000000
DBM 00000002 18:42:05.816406 RLPL format
+0000 ID..... DXRRL100-02: REQUEST COMPLETED
+0020 TLA1..... 000100C8 07166220
+0028 RLPL..... 00000000 06545768 00000000 80000000 00000000
+003C 00000000 006B12C8 008FBBC0 0090B000 00906048
+0050 00316545 06545060 00000000 00316545 06545060
+0064 00000000 00000000 00000000 0423AD20 09000058
+0078 C8806D01 D7000000 00000000 00000000 00000000
+008C 00000000 00000000 80000000 00000003 00000000
+00A0 006B12C8 008FBBC0 02060000 8A000000 00000000
+00B4 00000000 006B5BE4 00000000 00000000 00000000
+00C8 00000000 00000000 00000000 0067027C A743B4E5
+00DC 09010080 00000000 00080000 00000000 00000000
```

The z/OS IPCS subcommand that produced this output is: **CTRACE COMP(IRLE) SUB((SLM)) FULL**

#### **COMPONENT TRACE FULL FORMAT COMP (IRLE) SUBNAME ((SLM))**

```
COMPONENT TRACE FULL FORMAT
COMP(IRLE) SUBNAME((SLM))
**** 02/10/09
MNEMONIC ENTRY ID TIME STAMP DESCRIPTION
-----
SLM 00000010 18:42:05.816193 RNA, RTE and UDB format
+0000 ID..... DXRRL120-01: IXLOCK OBTAIN
+0020 TLA1..... 00060020 00670238
+0028 RNA..... 09000058 C8806D01 D7000000 00000000 00000000
+003C 00000000 00000000 00000000
+0048 TLA2..... 000C0040 07166418
+0050 RTE..... 0423AD20 09000058 C8806D01 D7000000 00000000
+0064 00000000 00000000 00000000 00000000 00000008
+0078 C9D4E2C5 40404040 0423AD20 00000000 00000000
+008C 00000000
+0090 TLA3..... 000B0040 071663D8
+0098 UDB..... C9D4E2C5 40404040 00000000 00000000 00080000
+00AC 00000000 00000000 00000000 00000000 40000000
+00C0 08000000 00000000 A8D1A743 B4D7B281 A8D1A743
+00D4 B4D7B281
```



```

SLM      00000020  18:42:05.816397  RNA and reason code
+0000    ID..... DXRRL120-03: IXLLOCK RETURN
+0020    TLA1..... 00060020  00670238
+0028    RNA..... 09000058  C8806D01  D7000000  00000000  00000000
+003C    TLA2..... 00000000  00000000  00000000
+0048    TLA2..... 00060004  0716637C
+0050    REAS..... 00000000

```

### Related reference

[z/OS: MVS interactive problem control system \(IPCS\) CTRACE subcommand](#)

[TRACE CT command \(Commands\)](#)



---

## Chapter 16. MSC - Multiple Systems Coupling service aids

Various types of traces are available to help you diagnose Multiple Systems Coupling (MSC) problem in a non-Database Control (DBCTL) environment.

### About this task

---

## Multiple Systems Coupling communication task trace

The flow through an MSC communication task is similar to that through the terminal communication task. The register 0 trace is read in the same manner, and most of the MSC analyzer and MSC DDM entry points provide the same functions as the terminal communications analyzer and DDMs.

The entry points for the MSC analyzer and device-dependent modules (DDMs) are:

#### **DDM entry point**

##### **Analyzer**

#### **AM01**

Process input from a link

#### **AM02**

Perform read or read of the link

#### **AM03**

Determine what to do next on the link

#### **AM04**

Not used

#### **AM05**

Perform write or send to the link

#### **AM06**

Dequeue the message after a good write or send

#### **AM07**

Not used

#### **AM08**

Return a message to the message queues for later transmission

#### **AM09**

Generate an error message

#### **AM10**

Quiesce the link

#### **AM11**

Not used

#### **AM12**

Wait for the completion of asynchronous I/O or the enqueue of a message

---

## Multiple Systems Coupling device-dependent module

A Multiple Systems Coupling (MSC) device-dependent module performs all of the functions unique to a type of link.

The functions the device-dependent module performs at each entry point are:

#### **DDM entry point**

##### **MSC**

**DM01**

Setup output buffer for a write or send operation

**DM02**

Error check last output operation

**DM03**

Setup to obtain input from the link

**DM04**

Error check an input operation

**DM05**

Not used

**DM06**

Not used

**DM07**

Connect or disconnect the link

**DM01**

An access method is entered from the device-dependent module

Several entry points are not used to preserve a commonality between coupling communication and terminal communication functions.

The following table summarizes the MSC communication task trace.

*Table 132. Multiple Systems Coupling communication task trace*

<b>Traced by</b>	<b>Entry point</b>	<b>Function</b>	<b>Trace indent</b>
DFSCMS00	DFSCMA01	Process Input	AM01
DFSCMS00	DFSCMA02		AM02
DFSCMS00	DFSCMA03	What's Next?	AM03
DFSCMS00	DFSCMA05		AM05
DFSCMS00	DFSCMA06	After Good Write	AM06
DFSCMS00	DFSCMA08	Wash Message	AM08
DFSCMS00	DFSCMA09	Generate Message	AM09
DFSCMS00	DFSCIO10	Quiesce Link	AM10
DFSCMS00	DFSCIO12	Wait for I/O or Message Enqueue	AM12
DFSCMS00	DFSCIOC0	Get Work Buffer	CM00
DFSCMS00	DFSCIOC0	Reposition Queue Buffer	CM01
DFSCMS00	DFSCIOC0	Get Next	CM02
DFSCMS00	DFSCIOC0	Dequeue Message	CM03
DFSCMS00	DFSCIOC0	Wash Output	CM04
DFSCMS00	DFSCIOC0	Find Output	CM05
DFSCMS00	DFSCIOC0	Get New Output	CM06
DFSCMS00	DFSCIOC0	Free Input Queue Buffer	CM07
DFSCMS00	DFSCIOC0	Free Work Buffer	CM08
DFSCMS80	DFSCMS80	Abort Processing (First LTB)	MSS1

Table 132. Multiple Systems Coupling communication task trace (continued)

Traced by	Entry point	Function	Trace indent
DFSCMS80	DFSCMS80	Abort Processing (Second LTB)	MSS2
DFSCMS81	DFSCMS81	Prior to DDM I/O	DM01
DFSCMS00	DFSCIO03;06	Write Setup	DM01
DFSCMS00	DFSCIO00	Write Interrupt	DM02
DFSCMS00	DFSCIO01;03	Read Setup	DM03
DFSCMS00	DFSCIO00	Read Interrupt	DM04
DFSCMS00	DFSCIO00;03	Connect/Disconnect I/O Interrupt	DM07
DFSCMEIO	DFSCMEIO	Message Control/Error exit processing	CMEI
DFSCMEIO	DFSCMEIO	Before calling Message Control/Error exit DFSCMUX0	CMEA
DFSCMEIO	DFSCMEIO	After calling Message Control/Error exit DFSCMUX0	CMEB

## Multiple Systems Coupling traces

Multiple Systems Coupling traces include MSC Message Processing trace, Main storage-to-main storage access method trace, and Main storage-to-main storage save set trace.

### MSC Message Processing Trace—BUFMSTRA

The MSC message processing trace records the SYSIDs of the last four IMS systems that processed the MSC message (that is, a BMP or MPP issued a GET UNIQUE to the message queue). The trace is located in the MSC message prefix at label BUFMSTRA within the BUFMS DSECT. The trace contains up to four 1-byte SYSID entries. The low-order byte contains the most recent entry. The initial entry contains the SYSID of the system to which the inputting terminal is attached. Each additional entry results in a shift left (the high-order byte is shifted out).

The SYSID is increased to 2 bytes and it is traced in field MSGMETRA of the MSC extension in DSECT MSGMSCE. If the SYSID is less than 256, it is traced both in field BUFMSTRA and MSGMETRA for compatibility. If the SYSID is greater than 255, it is only traced in MSGMETRA; field BUFMSTRA contains zeros.

### Main storage-to-main storage access method trace

The main storage-to-main storage access method trace records information related to the main storage-to-main storage access method, DFSMTMA0, and the main storage-to-main storage device-dependent module, DFSDN540. The trace is located in global storage pointed to by the "MTMWINDOW" and copied to module DFSMTMTR during abend processing. The following locates the trace:

- TTOP—Table beginning
- TPTR—Next entry to be used
- TBOT—Table end

The trace is a wraparound trace. Each entry is 192 bytes long and contains information such as function, return code, and control blocks. The TRACEMAP DSECT contains further details on entry contents. TRACEMAP is embedded in macro INTFMTMA. Trace operation is controlled by a global SETC labeled within DFSMTMA0. The default assembly value is ON.

### Main storage-to-main storage save set trace

DSECT SAVWORK describes a key work area used by DFSMTMA0. This work area is chained into the standard IMS save set chain with a SAVE ID of MTMWORKAREA. The trace appears in the save set chain even when the trace is set. The SAVWORK DSECT is embedded within macro INTFMTMA.

## Diagnosing link problems

---

Turn traces on for the appropriate lines from either the IMS master terminal or, if you are using type-2 commands, from the Operations Manager (OM) API. Trace all terminals on a line.

### About this task

For example, use either of the following commands:

- `/TRACE SET ON LEVEL 4 MODULE ALL LINK or UPDATE MSLINK NAME(linkname|linkname*|*) START(TRACE)`

**Note:** The type-2 command `UPDATE MSLINK NAME(linkname) START(TRACE)` uses the same level and module settings that were used the last time the `/TRACE SET (ON) LINK` command was issued. If a `/TRACE SET (ON) LINK` command has not been issued since the last cold start, this command defaults to `MODULE=ALL` and `LEVEL=4`.

- `/TRACE SET OFF LINK x or UPDATE MSLINK NAME(linkname) STOP(TRACE)`

Two report types are generated:

#### Summary

Contains the average response time, in milliseconds (msec), of the total number of send and receive data values for each link trace.

#### Detail

Contains the individual response times, in milliseconds (msec), for every send and receive data value for each MSC link that has been traced.

The formatting of this record is available only if the IMS input log contains X'6701' records that are generated by the `/TRACE SET ON LINK` command or the type-2 command `UPDATE MSLINK NAME(linkname)`.

For diagnosing link problems, the trace records with the following identifiers are helpful.

#### *AM01 RECEIPT OF DATA FROM PARTNER SYSTEM*

This entry is invoked because the link is stopped (using either the `/PSTOP` command, the `UPDATE MSLINK linkname STOP` command, or because of an I/O error).

Assemble a copy of DFSADSCT and refer to the BUFMS DSECT in the listing.

#### **I TP BUF**

Contains the segments received.

#### **BUFTFLAG**

Indicates more about what was received (that is, first segment).

#### **O TP BUF**

Contains the data set last sent to the partner.

#### **Q BUF**

Contains the segments received so far.

#### **I WP BUF**

Contains the MSC prefix/work buffer.

#### **O WP BUF**

Contains the MSC prefix/work buffer.

#### *AM02 ERROR - CHECK LAST OUTPUT OPERATION*

**I WP BUF**

Contains the MSC prefix/work buffer.

**O WP BUF**

Contains the MSC prefix/work buffer.

*AM03 MSC ANALYZER 'WHAT NEXT'*

If this entry is invoked from a device-dependent module, it is because the device-dependent module has nothing else to do.

Example: EOT received to ACK. Neither side sending; therefore, let the analyzer decide what to do.

Example: A data block containing only the message prefix was received (no segment could fit in the remaining buffer space). The device-dependent module goes to AM03 because there might be output that can be sent. Data response to data is okay.

If this entry is invoked from another analyzer entry point, it is because that function is complete.

Example: After the dequeue of an output message, ENTRY 6 goes to AM03 to see if more output can be initiated.

**CLBCNTQB**

Is a QCB for a destination that has messages queued to be sent across the link.

**CLB3INP and/or CTBAINP**

Indicates that the device-dependent module is not able to send any output data.

**CTBAERR**

Indicates that an error message is to be sent to the partner.

**I WP BUF**

Contains the MSC prefix/work buffer.

**O WP BUF**

Contains the MSC prefix/work buffer.

*AM05 MSC ANALYZER ENTRY 5*

This entry is invoked from the device-dependent module to send a message.

**O TP BUF**

Contains the data last sent to the partner.

**I WP BUF**

Contains the MSC prefix/work buffer.

**O WP BUF**

Contains the MSC prefix/work buffer.

*AM06 LAST OUTPUT OPERATION SUCCESSFUL*

This entry is invoked from the device-dependent module when the previous output was successful.

**CTBAEOM=1**

Indicates that the previous output included the last piece of the message, and that the message is to be dequeued.

**CTBAEOM=0**

Indicates that the last piece of the message has not been sent. No dequeue is to take place. The device-dependent module is dispatched at DM01 to attempt to continue transmitting.

*AM08 CANCEL MESSAGE ENQUEUE OPERATION*

There is a probable contention situation, and this partner must yield. The output message in progress is returned ("washed back") to the queues to be sent later.

**O TP BUF**

Contains the data that the device-dependent module was attempting to transmit.

*AM09 GENERATE AN ERROR MESSAGE*

## **I WP BUF**

Contains the MSC prefix/work buffer.

## **O WP BUF**

Contains the MSC prefix/work buffer.

### *AM10 LINK SHUTDOWN: OPERATOR INTERVENTION REQUIRED*

This entry is invoked because the link is PSTOPPED (either using the /PSTOP command, the UPDATE MSLINK NAME(*linkname*) STOP(COMM) command, or because of an I/O error). Find the previous device-dependent module interrupt entry (DM02, DM04 or DM07) to determine why the device-dependent module went to AM10.

General cleanup is performed: Queue buffers and I/O buffers are released.

### *AM12 NORMAL 'LINK IDLE' CONDITION*

This entry is invoked when the device-dependent module has nothing else to do under normal conditions.

Example: MTM link is attention driven. There is no outstanding READ as with BSC. When the device-dependent module has no more data to send and no pending acknowledgment, it becomes idle to wait for a POST by either the enqueue of output or an attention from the partner. This entry is different from AM10 because the analyzer does not complete a general cleanup.

### *CM00 GET A WORK BUFFER*

This analyzer entry is called when the device-dependent module needs additional space to perform message editing. An example is the collecting of all pieces of a SPA.

### *CM01 REPOSITION QUEUE BUFFER*

This analyzer entry is called when the device-dependent module wants to ensure that the queue buffer is in storage. This entry is currently not used.

### *CM02 GET NEXT*

This analyzer entry is called when the device-dependent module needs the next output segment of a message.

### *CM03 DEQUEUE MESSAGE*

This analyzer entry is called when the device-dependent module wants to dequeue a message (rather than let the analyzer do it). An example is the emergency restart of a link. The device-dependent modules exchange message sequence numbers. If one device-dependent module determines that a message in its queues has already been received by the partner, the message is dequeued to prevent it from being sent twice.

### *CM04 WASH OUTPUT MESSAGE*

This analyzer entry is called when the device-dependent module wants to return an in-process message to the queues. An example is a permanent I/O error. The device-dependent module washes any output in progress and is sent again after the error recovery sequence completes.

### *CM05 DETERMINE IF QUEUED OUTPUT IS PRESENT ON A LINK*

This analyzer entry is called when it must be determined if there is any (more) queued output to be sent across the link emergency restart processing. If one device-dependent module determines that a message in its queue has already been received by the partner, the device-dependent module issues a get unique (GU) call (for positioning) followed by a DEQUEUE (CM03) to get rid of the message.

### *CM06 GET NEW MESSAGE*

The system issues a get unique (GU) call to get a new output message.

### *CM07 FREE INPUT QUEUE BUFFER*

This analyzer entry is called when the device-dependent module wants to cancel an input queue buffer. An example is permanent I/O error. The device-dependent module discards all input segments that, up



to the point of failure, have been collected in queue buffers. The message is lost on this system, and the ABORT sequence sent to the partner tells the partner that the message must be sent again later.

#### *CM08 FREE A WORK BUFFER*

This analyzer entry is called when the device-dependent module wants to free an extra work buffer. This entry is currently not used because the buffer mentioned in the CM00 description is automatically freed by the analyzer.

#### *CM09 GET A PREFIX/WORK BUFFER*

The system obtains a prefix or work buffer.

#### *CM10 FREE A PREFIX/WORK BUFFER*

The system frees a prefix or work buffer.

#### *CM11 QUEUE ERROR*

The system processes a QMGR message queue error and issues message DFS082.

#### *CM12 GLOBAL WASH*

The system issues a CANCEL OUTPUT call to clear the global queue in a shared-queues environment.

#### *CM13 INSERT MESSAGE*

The system inserts an input message to the message queue.

#### *CM14 ENQUEUE A MESSAGE*

The system enqueues an input message to the message queue.

#### *CM15 REREAD MESSAGE*

The system reads an output message from the shared queues again.

#### *CM16 GET MESSAGE BY DRRN*

The system gets the message with the specified device relative record number (DRRN).

#### *CM17 GET LINK INPUT/OUTPUT BUFFERS*

The system obtains link I/O buffers.

#### *CM18 FREE LINK INPUT/OUTPUT BUFFERS*

The system frees link I/O buffers.

#### *DM01 WRITE SETUP*

The device-dependent module is entered here when the MSC analyzer finds output to be sent and the link is available (CLB3INP off).

Assemble a copy of DFSADSCT and refer to the BUFMS DSECT in the listing.

#### **Q BUF**

Contains the segments to be sent.

#### **O TP BUF**

Contains the data stream ready to be sent.

#### **I TP BUF**

Contains any data received from the partner.

#### *DM02 WRITE INTERRUPT*

The device-dependent module is entered here at the completion of a logical write operation.

#### **DECSDECB**

Contains the completion code.

#### **BUFTYPE**

Contains more information about the type of completion (MTM).

**O TP BUF**

Contains the data stream sent to the partner.

**I TP BUF**

Contains any data received from the partner.

**I WP BUF**

Contains the MSC prefix/work buffer.

**O WP BUF**

Contains the MSC prefix/work buffer.

*DM03 READ SETUP*

The device-dependent module is entered here when the MSC analyzer determines there is no output that can be sent. MTM and CTC are attention driven, and no I/O is initiated here.

*DM04 READ INTERRUPT*

The device-dependent module is entered here at the completion of a logical read operation.

**DECSDECB**

Contains completion code.

**BUFTYPE**

Contains more information about the type of completion (MTM).

**DECTYPE**

Indicates the type of the last operation.

**I TP BUF**

Contains the data just read.

**O TP BUF**

Contains any data sent to the partner in response to a previous read completion.

**I WP BUF**

Contains the MSC prefix/work buffer.

**O WP BUF**

Contains the MSC prefix/work buffer.

*DM07 RESTART*

The device-dependent module is entered here from the MSC analyzer whenever the link is not active (CRB1ACT is not equal to X'11').

**DECTYPE**

Indicates the type of the last operation attempted.

**DECSDECB**

If I/O is completed, this indicates status.

**I TP BUF**

Contains the last data read.

**O TP BUF**

Contains the data to write or the data last written.

**I WP BUF**

Contains the MSC prefix/work buffer.

**O WP BUF**

Contains the MSC prefix/work buffer.

*DM0I ENTRY TO ACCESS METHOD*

This record is traced at entry to the access method from the device-dependent module.

**DECTYPE**

Indicates the type of operation.

## O TP BUF

If output, contains data to be written.

## Diagnosing link problems by using MSC link statistics

You can use MSC link statistics to diagnose link performance problems; determine the message sizes that are sent and received on an MSC link and use this information to determine the link and message queue buffer sizes to use; run benchmarks to determine how many parallel MSC links are needed, and to provide comparisons to use for diagnosing performance problems; and determine the amount of MSC work (such as messages and I/Os) and resources that are being used.

### About this task

MSC link statistics are kept in the MSC work area, and are pointed to by field CLBMSCWA of the link LLB. This work area is called the DFSMSCWA area. Link statistics from the DFSMSCWA area are logged at checkpoint with type X'4513' log records. Each X'4513' log record is for a separate link. To assemble DSECTs of these three areas (the LLB, the DFSMSCWA work area, and the X'4513' log records), assemble the following source:

```
CSECT
ICLI CLBBASE=0          Map CLB/LLB
DFSMSCWA FUNC=DSECT     MAP DFSMSCWA work area
DFS4513 FUNC=DSECT      Map 4513 log record
END
```

The DFSMSCWA area contains two areas for statistics:

#### MAIN statistics area

A continuous record of link statistics for logging. Statistics start recording at IMS restart (cold or warm). This area is not reset unless a statistics field overflows. When a statistics field overflows, the field is reset to zero, and a flag is set to indicate that the counter overflowed. The flag remains on until the next IMS checkpoint (and statistics records are logged) and then is reset.

#### COPY area

A mirror of the main area. This area is for the QUERY MSLINK SHOW STATISTICS command. The copy area is copied from the main area whenever the query statistics are reset. Query statistics are reset at IMS checkpoints if the reset mode is RESET,CHKPT, or by the command UPDATE MSLINK NAME() START(STATISTICS OPTION(RESET)). The QUERY command always uses the difference of the MAIN statistics area minus the COPY area, so copying MAIN to COPY resets the QUERY (COPY) statistics without affecting the logging (main area).

The following table shows the key labels for statistics in the DFSMSCWA area.

Table 133. Descriptions of key labels for statistics in the DFSMSCWA area

Area	Label	Description
LLB - Logical link control block	CLBMSCWA	Points to DFSMSCWA
	CLBNAME2	Logical link name
DFSMSCWA - MSC work area for the logical link	MSCWA_ID	Beginning, = characters = DFSMSCWA
	STAT_FLAG1	Start of main statistics
	MSCWA_STARTTIM	Start main general statistics
	MSCWA_SEND#	Start main send statistics
	MSCWA_REC#	Start main receive statistics
	MSCPY_STARTTIM	Start copy general statistics
	MSCPY_SEND#	Start copy send statistics

Table 133. Descriptions of key labels for statistics in the DFSMSCWA area (continued)

Area	Label	Description
DFSL4513 - Log X'4513' area	MSCPY_REC#	Start copy receive statistics
	ST4513	Beginning
	ST4513_TYPE	Log record code = X'4513'
	ST4513_LNKNAME	Logical link name
	ST4513_LNKNUM	Logical link number
	ST4513_FLAG1	Main statistics reset flags from DFSMSCWA STAT_FLAG1
	ST4513_STARTTIM	Start main general statistics from DFSMSCWA MSCWA_STARTTIM
	ST4513_SEND#	Start main send statistics from DFSMSCWA MSCWA_SEND#
	ST4513_REC#	Start main receive statistics from DFSMSCWA MSCWA_REC#

Three categories of link statistics are kept for each logical link:

- General statistics, such as statistics start time, ITASK dispatch counts, ITASK processing times, and the rate and number of logger check writes.
- Send statistics, such as messages sent, byte count sent, send message sizes, Queue Manager get counts and times, and send I/O times.
- Receive statistics, such as messages received, byte count received, receive message sizes, Queue Manager insert counts and times, and receive I/O times.

## MSS1 and MSS2 records

MSS1 and MSS2 records are created as a result of ABORT processing when an I/O error (correctable or not) occurs. All available control blocks are snapped, regardless of any trace options in effect on the link involved.

These records are followed by a type X'03' record containing the message that was sent to the master terminal as a result of the error.

The following table shows the significant control blocks snapped in MSS1 and MSS2 records.

Table 134. Significant fields in MSS1 and MSS2 records

Link type	Control blocks
CTC	<p>POST code (first word of LLB)  DECTYPE  LLB, LTB(S), LCB, LXB, CRB, CTT, LNB control blocks  IOSB  MSCWA (MSC work area)  I_WP_BUF, O_WP_BUF</p> <p>I/O buffers (input and output). In non-bandwidth mode, the I/O buffers include I_TP_BUF and O_TP_BUF. In bandwidth mode, the input and output I/O buffers include:</p> <ul style="list-style-type: none"> <li>• I_BUFHDR, O_BUFHDR Send and receive buffer headers</li> <li>• BUFRSP Response data</li> <li>• BUFMSG Message data</li> <li>• BUFERR Error message data</li> <li>• BUFRST Restart data</li> <li>• BUFQUI Shutdown data</li> <li>• BUFUNK Unknown data</li> </ul>
MTM	<p>POST code (first word of LLB)  DECTYPE  LLB, LTB(S), LCB, LXB, CRB, CTT, LNB control blocks  MSCWA (MSC work area)  I_WP_BUF, O_WP_BUF</p> <p>I/O buffers (input and output). In non-bandwidth mode, the I/O buffers include I_TP_BUF and O_TP_BUF. In bandwidth mode, the input and output I/O buffers include:</p> <ul style="list-style-type: none"> <li>• I_BUFHDR, O_BUFHDR Send and receive buffer headers</li> <li>• BUFRSP Response data</li> <li>• BUFMSG Message data</li> <li>• BUFERR Error message data</li> <li>• BUFRST Restart data</li> <li>• BUFQUI Shutdown data</li> <li>• BUFUNK Unknown data</li> </ul>

Table 134. Significant fields in MSS1 and MSS2 records (continued)

Link type	Control blocks
TCP/IP	<p>POST code (first word of LLB)            LLB, LTB(S), LCB, LXB, CRB, CTT, LNB control blocks            MSCWA (MSC work area)            I_WP_BUF, O_WP_BUF            SCIWORK (SCI work area)</p> <p>I/O buffers (input and output). In non-bandwidth mode, the I/O buffers include I_TP_BUF and O_TP_BUF. In bandwidth mode, the input and output I/O buffers include:</p> <ul style="list-style-type: none"> <li>• I_BUFHDR, O_BUFHDR Send and receive buffer headers</li> <li>• BUFRSP Response data</li> <li>• BUFMSG Message data</li> <li>• BUFERR Error message data</li> <li>• BUFRST Restart data</li> <li>• BUFQUI Shutdown data</li> <li>• BUFUNK Unknown data</li> </ul> <p>DFSWE, the input AWE from the CSL SCI Input message exit. This AWE chain is initially queued to the LXBAWEHDR, then moved over to CLBQE by the TCPIP post handler DFSTCP20 for processing by the DDM. The AWE is mapped by the DFSWEI macro. The AWEI_MBRPLPTR field contains the DFSMSDIR parameter list from SCI.</p> <p>DFSMSDIR, the parameter list of doubleword parameters, containing the length and address of the parameters that are passed between MSC and IMS Connect through the SCI interface.</p> <p>ICONDATA, the data from IMS Connect, pointed to by the DFSMSDIR parameter MDIR_ICONMSGPTR. This data is mapped by the BUFMS macro, starting at BUFMSHDR.</p>
VTAM	<p>POST code (first word of LLB)            LLB, LTB(S), LCB, LXB, CRB, CTT, LNB control blocks            MSCWA (MSC work area)            I_WP_BUF, O_WP_BUF            RPLs (request parameter list)</p> <p>I/O buffers (input and output). In non-bandwidth mode, the I/O buffers include I_TP_BUF and O_TP_BUF. In bandwidth mode, the input and output I/O buffers include:</p> <ul style="list-style-type: none"> <li>• I_BUFHDR, O_BUFHDR Send and receive buffer headers</li> <li>• BUFRSP Response data</li> <li>• BUFMSG Message data</li> <li>• BUFERR Error message data</li> <li>• BUFRST Restart data</li> <li>• BUFQUI Shutdown data</li> <li>• BUFUNK Unknown data</li> </ul>

## Related reference

[“MSS3 and MSS4 records” on page 457](#)

The MSS3 and MSS4 records capture link control blocks when links are started and stopped. MSS3 captures information when links are stopped. MSS4 captures information when the links are started.

## MSS3 and MSS4 records

---

The MSS3 and MSS4 records capture link control blocks when links are started and stopped. MSS3 captures information when links are stopped. MSS4 captures information when the links are started.

MSC always logs the MSS3 and MSS4 records for diagnostics, even when a link trace is not active.

### MSS3

The MSS3 record is a snapshot of the MSC link incore trace buffer for CTC, VTAM, and TCP/IP links taken each time a link stops, either normally or due to an error.

The MSC link incore trace buffer, which is a wrap-around type buffer, contains data from selected fields in the link and I/O control blocks. MSC writes the data to the buffer each time a link is dispatched to do work and each time the link exits after completing the work. When the buffer is full, MSC wraps around and overwrites the oldest data in the buffer. Module DFSCMS80 contains the DSECTS of the selected fields that are written to the MSC link incore trace buffer.

The MSS3 record captures data from the following control blocks:

- LLB (logical link block)
- LTBs (link terminal blocks), input and output
- CRB (communication restart block)
- LXB (link extension block)
- I/O control blocks:
  - For CTC-type links: IOSB, IOB, and IEDB
  - For TCP/IP type links: SCIWORK
  - For VTAM type-links: input and output RPLs (request parameter lists)
- R0 trace word

### MSS4

The MSS4 record is a snapshot of link control blocks taken each time a link is started. The MSS4 record captures the following control blocks:

- LLB
- MSCWA
- LTBs, input and output
- I/O buffers, input and output
- LXB
- CRB
- I/O control blocks:
  - For CTC-type links: IOSB
  - For TCP/IP type links: SCIWORK
  - For VTAM type-links: RPLs, input and output
- LNB (link name block for the MSNAME)
- CTT (communication terminal table block)

## Related reference

[“MSS1 and MSS2 records” on page 454](#)

MSS1 and MSS2 records are created as a result of ABORT processing when an I/O error (correctable or not) occurs. All available control blocks are snapped, regardless of any trace options in effect on the link involved.

## Channel-to-channel access method trace stack (LXB trace)

---

The LXB trace stack is designed to be used in conjunction with the module listings to provide a detailed trace of instruction flow through the channel-to-channel (CTC) access method.

The trace stack is located in the LXB at label LXBCTRAC, 288 (X'E4') bytes into the LXB, and is 50 bytes long. The only modules that manipulate the LXB trace stack are the CTC access method modules, DFSCMC00, DFSCMC10, DFSCMC40, and DFSCMC50. The code that manipulates the LXB trace stack is unconditionally operative. (That is, it is not conditionally assembled and the function is not controlled by the operator command.) If level 3 or 4 of the IMS trace command is in effect, the LXB is included among the areas traced to the log.

Most LXB trace stack entries are 2 bytes long; a few are 1 byte long. Usually, each invocation of one of the access method modules causes a trace entry to be placed in the LXB trace stack. In order to create a trace entry, the module first moves (pushes) the trace stack 2 (or 1) bytes backward (toward low storage), thereby deleting the oldest portions of the trace stack. The module then inserts the new entry at the high (storage address) end of the trace stack. In rare instances, when the asynchronous modules DFSCMC40 and DFSCMC10 interrupt execution of another CTC access method module, the trace entries might overlap and thus might not be meaningful.

The format and meaning of the possible LXB trace entries follow:

### Byte 1, bit 0

If on, this is a 2-byte entry; otherwise it is a 1-byte entry.

### Byte 1, bits 1-3

This identifies the module and, if applicable, the routine within the module that made the entry in the LXB.

#### Value

#### Meaning

- |          |                                       |
|----------|---------------------------------------|
| <b>1</b> | DFSCMC40, attention DIE routine       |
| <b>2</b> | DFSCMC10, channel-end appendage       |
| <b>3</b> | DFSCMC10, abnormal-end appendage      |
| <b>4</b> | DFSCMC40, I/O request DIE routine     |
| <b>5</b> | DFSCMC10, shutdown appendage          |
| <b>6</b> | DFSCMC50, shutdown processing routine |
| <b>7</b> | DFSCMC00, MSC analyzer                |

### Byte 1, bits 4-7

This identifies what processing was performed. The meaning of the bits, as shown below, is dependent on the routine that made the entry in the LXB.

### Byte 2

This is an input byte that the routine keys on. This is also dependent on the routine and is described below.



## DFSCMC00 (MSC analyzer)

The DFSCMC00 module manipulates the LXB trace stack.

### Byte 1, bits 4-7

Value	Meaning
-------	---------

0	No I/O operation was queued; contention exists for the CTC adapter
1	WRITE channel program was queued
2	ACK channel program was queued
3	WRACK channel program was queued
4	READ channel program was queued; contention exists for use of the CTC adapter
5	STARTUP channel program was modified to be a WRITE channel program
6	Old STARTUP channel program was modified to be a WRITE channel program
7	WRITE channel program was not queued; write-pending switch was set
8	Error return was given

### Byte 2

This contains the operation code (found in DECTYPE+1).

## DFSCMC50 (shutdown processing routine)

The module DFSCMC50 manipulates the LXB trace stack.

### Byte 1, bits 4-7

Value	Meaning
-------	---------

1	Normal STACK operation was performed
2	Normal SHUTDOWN operation was performed
3	Abnormal SHUTDOWN occurred

### Byte 2

This contains the operation code (found in DECTYPE+1).

## DFSCMC40 (attention DIE routine)

The only modules that manipulate the LXB trace stack are the CTC access method modules, DFSCMC00, DFSCMC10, DFSCMC40, and DFSCMC50. The values and meaning of DFSCMC40 are listed.

### Byte 1, bits 4-7

IOSB was passed to IOS to perform a read.

Value	Meaning
-------	---------

- 0** Error was previously posted
- 1** IOSB was passed to IOS
- 2** IOSB on queue was modified to perform a read
- 3** LLB was posted with ACK received
- 4** LXB was posted with STARTUP complete; the link is available for a WRITE operation
- 5** LXB was posted with an error
- 6** LLB was posted with an error
- 7** During STARTUP processing, a control command was received after this routine used a no-operation command
- 8** Attention interrupt was received during SHUTDOWN processing; UCB was already cleared
- 9** Attention interrupt was received during SHUTDOWN processing; this routine did not reset UCBQISCE switch
- A** Attention interrupt was received during SHUTDOWN processing; this routine did not reset UCBQISCE switch
- B** Attention interrupt was received during SHUTDOWN processing; this routine scheduled an IOSB
- C** Attention interrupt was received during SHUTDOWN processing; this routine set LXBC2XS switch
- D** LXBC2SD switch was set after an attention interrupt because a WRITE command was received; READ operation was not done
- E** Read-pending or response-received switch was set
- F** Attention interrupt was received during SHUTDOWN processing; SHUTDOWN channel program was aborted

**Byte 2**

The command byte is sensed from the channel-to-channel adapter (found at IOSCTCMD), except when an I/O error prevented retrieval of the command byte, in which case byte 2 is absent.

## DFSCNC40 (I/O request DIE routine)

The only modules that manipulate the LXB trace stack are the CTC access method modules, DFSCMC00, DFSCMC10, DFSCMC40, and DFSCMC50. The values and meaning of DFSCNC40 are listed.

**Byte 1, bits 4-7**

Value	Meaning
-------	---------

- |          |  |
|----------|--|
| <b>0</b> | Second entry into this routine was taken; nothing was done |
|----------|--|

- 1 LXBCLIB switch was reset
- 2 IOSB on queue was modified to perform a WRITE operation (this is always a 1-byte entry)

## DFSCMC10 (channel-end appendage)

The only modules that manipulate the LXB trace stack are the CTC access method modules, DFSCMC00, DFSCMC10, DFSCMC40, and DFSCMC50. The values and meaning of DFSCMC100 are listed.

### Byte 1, bits 4-7

Value	Meaning
0	Nothing was done
1	LXB was posted with STARTUP complete; the link is available for a WRITE operation
2	LXB was posted with STARTUP complete; STARTUP message was received
3	During STARTUP processing, no-operation command was scheduled
5	LXB was posted; message received
6	LLB was posted; message received
8	During STARTUP processing, control command was scheduled
9	LLB was posted; an error occurred on message that was written
A	LLB was posted; an error occurred on message that was received
B	LXB was posted; an error occurred on message that was received

### Byte 2

This contains the first command code in the just-completed channel program (pointed to by IOSVST).

## DFSCMC10 (abnormal-end appendage)

The only modules that manipulate the LXB trace stack are the CTC access method modules, DFSCMC00, DFSCMC10, DFSCMC40, and DFSCMC50. The abnormal values and meaning of DFSCMC100 are listed.

### Byte 1, bits 4-7

Value	Meaning
2	Not a permanent error; control is given to an ERP
3	Error was declared permanent
4	Serial channel error
5	MIH detected error before retry

## Byte 2

This contains the value in IOSCOD.

## DFSCMC10 (shutdown appendage)

The only modules that manipulate the LXB trace stack are the CTC access method modules, DFSCMC00, DFSCMC10, DFSCMC40, and DFSCMC50. The shutdown values and meaning of DFSCMC100 are listed.

### Byte 1, bits 4-7

#### Value

#### Meaning

1

Completion was normal; a new I/O operation was scheduled

2

Completion was normal; LLB was posted

3

Completion was abnormal; UCB was already cleared

4

Completion was abnormal; this routine has cleared UCB and posted LLB

5

Completion was abnormal; this routine will restart I/O

6

Completion was abnormal; this routine has restarted I/O

7

Completion was normal; UCB was already cleared

### Byte 2

This contains the first command code in the just-completed channel program (pointed by IOSVST).

## LXB trace stack example

This example shows a printout of the LXB portion of an internal trace record.

The LXB trace stack begins at AE90E8 and contains 29 entries. Following the figure is a list of the meanings of the routines that made each entry.

### Printout of the LXB trace stack

```
DFSERA30 — FORMATTED LOG PRINT
:
: INTERNAL TRACE RECORD
:
: LXB
AE9004 000000 807F0BC9 00093660 00AE9350 00AE92B0 00091E90 00AE991C 17000000 7F0C0000
AE9024 000020 80000000 520821CE 0008229C 000820C6 80082194 012141CE 60000054 0A000000
AE9044 000040 30000005 022140C6 600000CE 09000000 30000005 47000000 20000001 00000000
AE9064 000060 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
AE9084 000080 TO AE90C4 0000C0 SAME AS ABOVE
AE90E4 0000E0 00000000 0C419317 F1044193 17F10441 9337E218 D243F510 A314A8C3 419101A2
AE9104 000100 02F30C41 93179101 A502F004 F30C4193 17F10441 93170000 00000000 00B66218
```

#### Entry

#### Meaning

X'0C'

The first byte of this entry, the oldest entry in the trace stack, has been pushed off the trace stack. Ignore this entry.

X'41'

DFSCMC40 (I/O request DIE). LXBCLIB was reset.

**X'9317'**

DFSCMC40 (attention DIE). Operation code X'17' (ACK) was received from the other system. The LLB was posted X'7F1C0000' (ACK received).

**X'F104'**

DFSCMC00. Operation code X'04' (WRITE) was received. The WRITE channel program was queued.

**X'41'**

DFSCMC40. (I/O request DIE). LXBCLIB was reset. WRITE operation was completed.

**X'9317'**

DFSCMC40 (attention DIE). Operation code X'17' (ACK) was received from the other system. The LLB was posted X'7F1C0000' (ACK received).

**X'F104'**

DFSCMC00. Operation code X'04' (WRITE was received). The WRITE channel program was queued.

**X'41'**

DFSCMC40 (I/O request DIE). LXBCLIB was reset. WRITE operation was completed.

**X'9337'**

DFSCMC40 (attention DIE). Operation code X'37' (STACK) was received from the other system. The LLB was posted X'7F1C0000' (ACK received).

**X'E218'**

DFSCMC50 (SHUTDOWN processing). Operation code X'18' (SHUTDOWN) was received. Normal SHUTDOWN was performed.

**X'D243'**

DFSCMC10 (SHUTDOWN appendage). Channel command X'43' (enable compatibility) completed normally. The LLB was posted.

**X'F510'**

DFSCMC00. Operation code X'10' (STARTUP) was received. The start-link channel program was queued.

**X'A314'**

DFSCMC10 (channel-end appendage). Channel command X'14' (sense command byte) of the start-link channel program completed normally. The disable compatibility no-operation command was scheduled.

**X'A8C3'**

DFSCMC10 (channel-end appendage). Channel command C'X3' (disable compatibility no-operation) completed normally. The startup control command was scheduled.

**X'41'**

DFSCMC40 (I/O request DIE). LXBCLIB was reset. Channel end was received from the startup control.

**X'9101'**

DFSCMC10 (attention DIE). Operation code X'01' (WRITE) was received from the other system. The IOSB was passed to IOS to initiate a READ.

**X'A202'**

DFSCMC10 (channel-end appendage). Channel command X'02' (read) completed normally. The LXB was posted X'7F080000' (startup complete, startup message received).

**X'F30C'**

DFSCMC00. Operation code X'0C' (WRACK) was received. ACK with data (WRACK) channel program was queued.

**X'41'**

DFSCMC40 (I/O request DIE). LXBCLIB was reset. WRACK operation has completed.

**X'9317'**

DFSCMC40 (attention DIE). Operation code X'17' (ACK) was received from the other system. The LLB was posted X'7F0C0000' (ACK received).

**X'9101'**

DFSCMC40 (attention DIE). Operation code X'01' (WRITE) was received from the other system. The IOSB was passed to IOS to initiate a READ operation.

**X'A502'**

DFSCMC10 (channel-end appendage). Channel command X'02' (read) was completed. The LXB was posted X'7F0C0000' (message received).

**X'F004'**

DFSCMC00. Operation code X'04' (WRITE) was received. No I/O was scheduled. Contention exists between this WRITE operation and the WRITE operation received from the other system in the preceding 9101 entry. The device-dependent module has not yet received control in response to the LXB post traced by the preceding A502 entry.

**X'F30C'**

DFSCMC00. Operation code X'0C' (WRACK) was received. ACK with data (WRACK) channel program was queued.

The ACK acknowledges the data received from the other system in the preceding 9101 entry. The data is the data that was not sent in the preceding F004 entry.

**X'41'**

DFSCMC40 (I/O request DIE). LXBCLIB was reset.

**X'9317'**

DFSCMC40 (attention DIE). Operation code X'17' (ACK) was received from the other system. The LLB was posted X'7F1C0000' (ACK received).

**X'F104'**

DFSCMC00. Operation code X'04' (WRITE) was received. The WRITE channel program was queued.

**X'41'**

DFSCMC40 (I/O request DIE). LXBCLIB was reset. WRITE operation was completed.

**X'9317'**

DFSCMC40 (attention DIE). Operation code X'17' (ACK) was received from the other system. The LLB was posted X'7F1C0000' (ACK received).

## MSC routing trace - BUFMSVID

The MSC routing trace is located in the MSC message prefix at label BUFMSVID within the BUFMS DSECT. The low-order byte in the trace contains the most recent entry, and each additional entry results in a shift left (the high-order byte is shifted out).

This trace records the MSVID (as specified in the IMSCTRL macro during system definition) of the last eight IMS systems through which messages were routed. It is initialized when a terminal sends a message or when an application program does an ISRT of a message, and it is updated for each intermediate system and the destination system.

This trace records the primary MTO's local SYSID of the last eight IMS systems through which messages were routed. It is initialized when a terminal sends a message or when an application program does an ISRT of a message, and it is updated for each intermediate system and the destination system. The MSC routing trace is located in the MSC message prefix extension at label MSGMEVID in DSECT MSGMSCE. The low-order byte in the trace contains the most recent entry, and each additional entry results in a shift left (the high-order byte is shifted out). If the SYSID is equal to or greater than 255, it is traced both in field BUFMEVID and MSGMEVID. If the SYSID is less than 255, it is only traced in MSGMEVID; BUFMEVID contains zeros.

## Chapter 17. ODBA - Diagnosing hung threads and UORs

Use the following steps to configure your system to gather data that IBM Software Support can use to determine why IMS threads or UORs are hanging on your system.

### Procedure

1. Turn on the following IMS table traces by using the following IMS commands:

```
/TRACE SET ON TABLE RRST OPTION LOG
/TRACE SET ON TABLE DISP
/TRACE SET ON TABLE SCHD
```

These traces are sent to the IMS OLDS, unless IMS external trace data sets are used. For a test environment, data sent to the OLDS files is appropriate. For production systems, use IMS external tracing. For more information about IMS external tracing, see [Diagnostic setup recommendations for IMS \(System Definition\)](#).

2. If it is not already active with the following options, turn on the MVS SYSRRS CTRACE:

- a) Place the following lines in the CTIRRSxx PARMLIB member:

```
TRACEOPTS
ON
BUFSIZE(1000M)
OPTIONS('EVENTS(URSERVS,LOGGING,CONTEXT,EXITS,STATECHG,RRSAPI,RE
START)')
```

- b) Issue the following MVS Command:

```
TRACE CT,ON,COMP=SYSRRS,PARM=CTIRRSxx
```

- c) Issue the following MVS Command to display the trace setting:

```
DISPLAY TRACE,COMP=SYSRRS
```

- d) The display output for the SYSRRS portion appears as follows:

```
IEE843I 13.17.07 TRACE DISPLAY 479 SYSTEM STATUS INFORMATION
ST=(ON,1000K,09000K) AS=ON BR=OFF EX=ON MO=OFF MT=(ON,500K)
COMPONENT MODE BUFFER HEAD SUBS
-----
SYSRRS ON 1000M
ASIDS *NONE*
JOBNAMES *NONE*
OPTIONS EVENTS(URSERVS,LOGGING,CONTEXT,EXITS,STATECHG,
RRSAPI,RESTART)
```

**Recommendation:** The MVS CTRACE should always be active on any system using IMS with RRS. The MVS trace command should be added to the MVS COMMNDxx SYS1.PARMLIB member to ensure that it is always activated at IPL time. No performance impacts have been reported with the SYSRRS CTRACE using the above options.

After this trace is activated on your system, future operational dumps of IMS must include the RRS address space in the list of jobs to be dumped along with the RRS data spaces (DSPNAME=('rrsname'\*)). Doing so ensures that the RRS control blocks and traces can be accessed for diagnostic purposes.

3. Turn GTF trace on with the TRACE=SYS,DSP,JOBNAMEP options.

When GTF Trace prompts for the JOBNAME, specify the JOBNAME for the stored procedure address space and IMS CTL Region. Set the GTF trace dataset to 1200 cylinders.

The GTF Messages should appear similar to the following:

```

START GTFxx.GTF
HASP100 GTFxx.GTF ON STCINRDR
HASP373 GTFxx.GTF STARTED
*01 AHL100A SPECIFY TRACE OPTIONS
R 01,TRACE=SYS,DSP,JOBNAMEP
IEE600I REPLY TO 01 IS;TRACE=SYS,DSP,JOBNAMEP
*02 ALH101A SPECIFY TRACE EVENT KEYWORDS - JOBNAME=
R 02,JOBNAME=(spasname,ctlname)
IEE600I REPLY TO 02 IS;JOBNAME=(spasname,ctlname)
*03 ALH102A CONTINUE TRACE DEFINITION OR REPLY END
R 03,END
END
IEE600I REPLY TO 03 IS;END
AHL103I TRACE OPTIONS SELECTED-SYS, DSP
AHL103I JOBNAME=(spasname,ctlname)
*04 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U
R 04,U
U
IEE600I REPLY TO 04 IS;U

```

4. Set the following SLIP trap to capture dumps for RRS related events:

```

SLIP SET,C=U0711,
JL=(imsctl,imsdli,rrsjname,???SPW2),
SDATA=(CSA,PSA,RGN,SQA,SUM,TRT,LPA,ALLNUC),
ID=U711,DSPNAME=('rrsjname'.*),END

```

The variables in the preceding example are defined in the following list.

***imsctl***

The IMS control region jobname

***imsdli***

The IMS DLI region jobname

***rrsjname***

The RRS region jobname

5. Use IMS Display commands for hung threads:

```

- DIS ACTIVE
- DIS CCTL
- DIS UOR

```

6. If a hung ODBA thread or UOR occurs, take a dump of the system so that IBM Software Support can see the related control blocks and access the RRS traces. Use a single dump command as indicated below and not separate dumps for each job.

```

DUMP TITLE=(DUMP IMS/SPAS/RRS)
JOBNAME=(ctlname,spasname,dlijname,rrsjname),
SDATA=(CSA,PSA,RGN,SQA,SUM,TRT,LPA,ALLNUC,GRSQ),
DSPNAME=('rrsjname'.*),END

```

The variables in the preceding example are defined in the following list.

***ctlname***

The IMS CTL Region jobname

***tspasname***

The stored procedure address space jobname

***dlijname***

The IMS DLI region jobname

***rrsjname***

The RRS region jobname

Tune DUMP MAXSPACE and the SYS1.DUMP space allocation to ensure that your system has sufficient space for the dump.



7. It is important for IBM Software Support to receive all of the requested diagnostic information as a complete unit from a single occurrence with all data centered around the time of the thread or UOR hang. This information typically includes:

- Dump data sets obtained
- IMS OLDS or SLDS
- MVS SYSLOG
- LOGREC data set
- JES JOBLOG for cltname,spasname

After taking the dump, you can free the hung thread using the recommendations in [Stopping Db2 for z/OS stored procedure threads \(Application Programming\)](#).



## Chapter 18. SYS - System service aids

Various tools, utilities, and traces can help you analyze IMS system problems.

### Related reference

“Sequential buffering service aids” on page 227

When you receive a message or abend that indicates a problem with sequential buffering (SB), several diagnostic tools are available. Some of these tools are useful for diagnosing other IMS database-related problems.

## Log records

To diagnose some problems, you need to examine the content of log records to determine what was going on in the system before the problem occurred. By knowing the layout of the log records, you can set up a DFSERA10 job that will produce the specific log records that you need to examine.

In addition, the content of the log records frequently contains information that you can use in your keyword string or when reviewing existing APAR descriptions and comparing them to your own situation.

You can obtain individual log record DSECTs by assembling the ILOGREC macro, and including a RECID format statement for the log record that you want. For example:

- ILOGREC RECID=01 - to format the DSECT for an IMS TYPE01 log record
- ILOGREC RECID=56 - to format the DSECT for an IMS TYPE56 log record

For Fast Path log record formats, you can assemble mapping macros DBFLSRT, DBFLGRQ, DBFLGRIM, DBFLGROM, DBFLGRSD, DBFLGSYN, and DBFBMSDB.

The following table describes the IMS log records, including the conditions that cause the records to be written.

Table 135. IMS log records used to analyze IMS problems

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'01'	QLOGMSGP	QLOGMSGP	Data was put in a message queue buffer. Caller is data communication. (DFSQLOG0)
X'02'	DFSLOG02	CMLOG	A <b>/LOG</b> command or a command that alters data required for restart was successfully completed. (DFSICLP0)
X'03'	QLOGMSGP	QLOGMSGP	Data was put in a message queue buffer. Caller is DL/I. (DFSQLOG0)
X'06'	DFSLOG06	ACLOGREC	IMS was started or stopped, or FEOV was issued. The VTAM TPEND exit routine was entered or the IRLM failed in an IMS XRF complex. A <b>/SWITCH</b> command was processed in an IMS XRF complex. A <b>/START</b> command connected IMS to VTAM. Data sharing capability was quiesced. (DFSFLG0, DFSFDM0, DFSICA20, DFSICL40, DFSRDSH0)
X'07'	DFSLOG07	DLREC	An application program terminated. (DFSRBLB0, DFSRBOI0, DFSSABN0, DFSDABN0, DFSDLA30, DFSTMAD0)
X'08'	DFSLOG08	LINTREC	An application program was scheduled. (DFSSMSC0, DFSSBMP0, DFSDASPO, DFSDLA30, DFSTMAD0)

Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'09'	SBLOGREC	SBLOGREC	An application potentially using sequential buffering terminated. The following subcodes, contained within the log record, identify the type of statistics written in the log record. (DFSSBTD0)  <b>X'01'</b> Sequential buffering summary statistic for the PST. <b>X'02'</b> Sequential buffering detailed statistics for each SDSG.
X'0A07'	DFSLOG0A	LOAREC	A CPI-communications driven application program terminated. (DFSSABN0)
X'0A08'	DFSLOG0A	LOAREC	A CPI-communications driven application program was scheduled. (DFSSMSC0)
X'10'	DFSLOG10	SCREC	A security violation occurred. (DFSICIO0, DFSCMD30, DFSICLZ0, DFSTMAD0)
X'11'	LCONVERS	LCONVERS	A conversational program started. (DFSCON00)
X'12'	LCONVERS	LCONVERS	A conversational program terminated. (DFSCON20)
X'13'	DFSLOG13	LOG13	This log record contains conversational CCBs at logon for static non-ISC terminal, signon for ETO user, or static ISC allocation. (DFSRMD00)
X'16'	DFSLOG16	LOG16	A <b>/SIGN</b> command successfully completed. (DFSICLZ0, DFSCBDL0)
X'18'	DFSLOG18	XLOG18	A user program established intent to use extended checkpoint and then issued a CHKP call. The user program issued a CHKP by issuing an XRST call with eight blank characters as a checkpoint ID value. (DFSZSC00)
X'20'	DFSLOG20	ILRDOC	A database was opened. (DFSDLOC0)
X'21'	DFSLOG20	ILRDOC	A database was closed. (DFSDLOC0)
X'22'	DFSLOG22	DFSLOG22	A type-2 command completed successfully and is recoverable across a restart. (DBFCPRC0, DBFRTS00, DFSCPD00, DFSCPSM0, DFSDBS00, DFSIC020, DFSIC160, DFSID020, DFSID160, DFSIU070, DFSIU090, DFSIU100, DFSIU110, DFSIU120, DFSIU160, DFSMLS00, DFSMNS00, DFSMPS00, DFSMRS00, DFSPGS00, DFSTRS00)
X'23'	DFSLOG23	LOG23DSC	The listed databases were open when a batch application requested a checkpoint. (DFSRDBL0)
X'24'	DFSLOG24	ERLGD SCT	The buffer handler detected an I/O error. (DFSDVSM0, DBFMER00)
X'25'	DFSLOG25	EEQLOG	An EEQE was created or deleted. (DFSTOLG0)
X'26'	DFSLOG26	IOTBUF	An I/O toleration buffer was created. (DFSTOLG0)
X'27'	DFSLOG27	DBXLOG	A data set was extended, according to these subcodes:  <b>X'01'</b> Data set extend phase 1. (DFSDVSM0)  <b>X'02'</b> Data set extend phase 2. (DFSDBHIO)

Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'28'	DFSLOG28	PH1DC	The IMS restart facility updated the sequence numbers of input messages for response mode non-Fast Path transactions from STSN devices. (DFSFXC40)
X'29'	DFSLOG29	DFSLOG29	<p>The progress of a HALDB online reorganization is represented in the following subcodes:</p> <p><b>X'00'</b> The OLR command was received. (DFSORC00, DFSORC10)</p> <p><b>X'03'</b> IMS catalog activity record. (DFS3SMF)</p> <p><b>X'10'</b> Ownership of the reorganization for a partition was established through DBRC. (DFSORP60)</p> <p><b>X'11'</b> Conditional ownership established. An attempt to establish ownership will be made. (DFSORP60)</p> <p><b>X'20'</b> The <b>UPDATE OLREORG</b> command updated either the RATE option or the [NO]DEL option for a HALDB partition. (DFSORC00, DFSORC10)</p> <p><b>X'30'</b> The output data sets were successfully validated or created. One record includes all output data sets. (DFSORA00, DFSRDBL0)</p> <p><b>X'40'</b> Cursor active. Initialization of the reorganization of the partition was completed successfully, two sets of data sets exist, and copying is about to begin. The partition is now in cursor-active status. (DFSORP60, DFSORP70)</p> <p><b>X'50'</b> The cursor was updated, but the unit of reorganization was not committed. (DFSORP20)</p> <p><b>X'70'</b> Cursor inactive. Copying from the input to the output data sets has completed. The output data sets become active, and the input data sets become inactive. (DFSORP60, DFSORP70)</p> <p><b>X'71'</b> Conditional cursor inactivate. An attempt to reset the cursor will be made. (DFSORP60)</p> <p><b>X'90'</b> Ownership of the reorganization for a partition was relinquished. This is followed by the X'07' log record for OLR ITASK termination. (DFSORP60)</p>
X'30'	QLOGMSGI	QLOGMSGI	A message prefix was changed. (DFSQLOG0)
X'31'	QLOGGETU	QLOGGETU	A GU was issued for a message. (DFSQLOG0)
X'32'	QLOGREJE	QLOGREJE	A message was rejected. It was presumed to have been the cause of an application program abend. (DFSQLOG0)
X'33'	QLOGFREE	QLOGFREE	The queue manager released a record. (DFSQLOG0)

Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'34'	QLOGCANC	QLOGCANC	A message was canceled. (DFSQLOG0)
X'35'	QLOGENQU	QLOGENQU	A message was enqueued or re-enqueued. (DFSQLOG0)
X'36'	QLOGDEQS	QLOGDEQS	A message was dequeued or saved or deleted. (DFSQLOG0)
X'37'	DFSXFER QLOGXFER	DFSXFER QLOGXFER	<p>Records marked as NO INPUT and NO OUTPUT are written by the sync point coordinator when all resource managers have completed Phase 1. (DFSFXC30, DBFSLG20)</p> <p>Records marked as NO INPUT and NO OUTPUT (for example, X'3730') are also written by the DBCTL sync point processor after receiving a phase 2 commit request. (DFSDSC00)</p> <p>Phase 2 DC processing. One or more output messages were transferred from a queue block anchored off the PST temporary output queue to a permanent destination. There is a X'37' record for each destination that has messages transferred. (DFSQLOG0)</p>
X'38'	QLOGRELI	QLOGRELI	<p>An input message was put back on the input queue when the application abnormally terminated. (DFSQLOG0)</p> <p>Records marked as "Release with no input message" (for example, X'3801') are written by the DBCTL sync point processor (DFSDSC00) after receiving an abort request.</p> <p>A Protected Conversation has been put in doubt, and the input message has been moved to an RRE until the unit of work is aborted or committed.</p> <p>This record is logged for each message returned to its original anchor block (SMB or CNT) after QCF has abnormally terminated.</p>
X'39'	QLOGRELO	QLOGRELO	The output queue was freed during cleanup processing of a RELEASE call. (DFSQLOG0)
X'3A'	QLFXFREE	QLFXFREE	A bitmap record was replaced after a queue record was freed at the end of DFSQFIX0 processing. (DFSQFIX0)
X'3B'	QLFXRERR	QLFXRERR	An invalid message record or a nonrecoverable message response was detected during queue validation. (DFSQFIX0)
X'3C'	QLFXBERR	QLFXBERR	A control block was changed during validation by DFSQFIX0. (DFSQFIX0)
X'3D'	QLFXQBLK	QLFXQBLK	A QBLK record was altered during DFSQFIX0 processing. (DFSQFIX0)

Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'40'	DFSCHKPT	LOG01	<p>A checkpoint was taken. The following subcodes in the log record precede and identify each type of information written in the log record.</p> <p><b>X'01'</b> Checkpoint information begins here. (DFSRCPO0)</p> <p><b>X'02'</b> Message queue checkpoint record. (DFSQCP00)</p> <p><b>X'03'</b> CNTs or LNBs, or both, follow. (DFSRCP30)</p> <p><b>X'04'</b> SMBs follow. (DFSRCP30)</p> <p><b>X'05'</b> Non-VTAM CTBs follow. (DFSRCP30)</p> <p><b>X'06'</b> DMBs follow. (DFSRCP40)</p> <p><b>X'07'</b> PSB follows. (DFSRCP40)</p> <p><b>X'08'</b> Non-VTAM CLB, LLB, or both, follow. (DFSRCP30)</p> <p><b>X'0C'</b> CVB follows. (DFSRCP30)</p> <p><b>X'0D'</b> CCBs follow. (DFSRCP30)</p> <p><b>X'0F'</b> Message queues TTR and LCB follow. (DFSRCP30)</p> <p><b>X'10'</b> Non-VTAM CRBs follow. (DFSRCP30)</p> <p><b>X'12'</b> Tran input edit routine table. (DFSRCP30)</p> <p><b>X'14'</b> SPQBs and related CNTs follow. (DFSRCP30)</p> <p><b>X'20'</b> Non-VTAM CIBs follow. (DFSRCP30)</p> <p><b>X'21'</b> VTAM VTCBs follow. (DFSRCP30)</p> <p><b>X'22'</b> Subcode for Queue Anchor Block (QAB). (DFS6CKP0)</p> <p><b>X'23'</b> Subcode for LU 6.2 descriptors modified by <b>/CHANGE DESCRIPTOR</b> command. (DFS6CKP0)</p> <p><b>X'24'</b> Subcode for LU 6.2 TIB. (DFS6CKP0)</p>

Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'40' (cont'd)	DFSCHKPT	LOG01	<p><b>X'25'</b> EEQE follows. (DFSTOLG0)</p> <p><b>X'26'</b> I/O toleration buffer follows. (DFSTOLG0)</p> <p><b>X'27'</b> Contains database updates for an in-doubt unit of recovery. (DFSRC40)</p> <p><b>X'28'</b> Error queue elements (EQEL) for recovery in-doubt structure (RIS). (DFSRC40)</p> <p><b>X'30'</b> RREs follow. (DFSRC50)</p> <p><b>X'31'</b> SIDXs follow. (DFSRC50)</p> <p><b>X'32'</b> TPIPE/YQAB follow. (DFSYCKP0)</p> <p><b>X'33'</b> MTE follow. (DFSYCKP0)</p> <p><b>X'34'</b> TIB follow. (DFSYCKP0)</p> <p><b>X'35'</b> OTMA (DFSYCKP0)</p> <p><b>X'36'</b> MSC logical link blocks (LLB, LTB1, LTB2, CRB) follow. (DFSRC30)</p> <p><b>X'37'</b> MSC physical link blocks (LCB, CTT) follow. (DFSRC30)</p> <p><b>X'38'</b> MSC PCB for bandwidth follows. (DFSRC30)</p> <p><b>X'39'</b> MSC remote LTERM (RCNT). (DFSRC30)</p> <p><b>X'40'</b> Remote LTERM follows. (DFSRC30)</p> <p><b>X'70'</b> MSDB record follows. (DBFHDMPO)</p> <p><b>X'71'</b> ECNT follows. (DBFHDMPO)</p> <p><b>X'72'</b> MSDB header follows. (DBFHDMPO)</p>



Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'40' (continued)	DFSCHKPT	LOG01	<p><b>X'73'</b> Page fixed MSDBs follow. (DBFHDMPO)</p> <p><b>X'74'</b> Pageable MSDBs follow. (DBFHDMPO)</p> <p><b>X'79'</b> MSDB record ends. (DBFHDMPO)</p> <p><b>X'80'</b> Fast Path checkpoint information begins here. (DBFCHKP0)</p> <p><b>X'81'</b> Snaps the current subpool information for use during warm and emergency restart.</p> <p><b>X'82'</b> EMHB follows. (DBFCHKP0)</p> <p><b>X'83'</b> RCTE follows. (DBFCHKP0)</p> <p><b>X'84'</b> DMCB and DMAC follow. (DBFCHKP0)</p> <p><b>X'85'</b> MTO buffer follows. (DBFCHKP0)</p> <p><b>X'86'</b> DMHR and DEDB buffers follow. (DBFCHKP0)</p> <p><b>X'87'</b> ADSC follows. (DBFCHKP0)</p> <p><b>X'88'</b> Fast Path IEEQEs. (DBFCHKP0)</p> <p><b>X'89'</b> Fast Path checkpoint information ends here. (DBFCHKP0)</p> <p><b>X'98'</b> Checkpoint information ends here. (DFSRC10)</p> <p><b>X'99'</b> The message queue checkpoint information ends here. (DFSQCP00)</p>
X'41'	DFSLOG41	LOG41DSC	A batch program or BMP program issued a checkpoint. (DFSRLBL0)
X'42'	DFSLOG42	ATLOGREC	IMS switched from one OLDS to another, or a checkpoint was taken, or a shutdown checkpoint was taken. (DFSFDLS0, DFSRDS00, DFSRC100)
X'43'	DFSLOG43	ADSETLOG	<p>The log manager or the Log Archive utility created this log record. The following subcodes identify each type of record:</p> <p><b>X'01'</b> Record contains status of current online log data set. (DFSFDLS0)</p> <p><b>X'02'</b> Dummy record created by Log Archive utility. This record is created as a substitute for a record that is omitted because of control statement specifications. (DFSUARPO)</p>

Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'45'	DFSLOG45	STLOGREC	<p>Checkpoint statistics were gathered, including statistics from the 64-bit storage pool. The following subcodes within the log record mark the start of various types of statistics written in the log record (DFSSTAT0).</p> <p><b>X'01'</b> Dynamic database log statistics.</p> <p><b>X'02'</b> Queue buffer statistics.</p> <p><b>X'03'</b> Format pool statistics.</p> <p><b>X'04'</b> DL/I buffer pool statistics.</p> <p><b>X'05'</b> Variable storage pool statistics.</p> <p><b>X'06'</b> Application scheduling statistics.</p> <p><b>X'07'</b> Logging statistics.</p> <p><b>X'08'</b> VSAM buffer pool statistics.</p> <p><b>X'09'</b> Program isolation statistics.</p> <p><b>X'0A'</b> Latch management statistics.</p> <p><b>X'0B'</b> Selected dispatcher statistics.</p> <p><b>X'0C'</b> Storage pool statistics. (DFSCBT00)</p> <p><b>X'0D'</b> Receive Any (RECA) Buffer statistics.</p> <p><b>X'0E'</b> Fixed storage pool usage statistics.</p> <p><b>X'0F'</b> Dispatcher statistics.</p> <p><b>X'10'</b> RCF multi-TCB statistics.</p> <p><b>X'11'</b> General storage statistics.</p> <p><b>X'12'</b> IMS Storage Manager (DFSMODU0) statistics.</p> <p><b>X'15'</b> 64-bit Cache Manager statistics.</p>

Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'45' (continued)	DFSLOG45	STLOGREC	<p><b>X'16'</b> Statistics for the storage pool of the 64-bit Storage Manager. Four control blocks contain information about the Fast Path 64-bit buffer manager. The information about data bound in these control blocks is documented in the control blocks. Each control block has a starting eye catcher to identify the control blocks in the log record.</p> <ul style="list-style-type: none"> <li>• <b>DBFBPNO2:</b> Contains general information about Fast Path 64-bit buffer manager.</li> <li>• <b>DBFBPNO3:</b> Contains information about a subpool.</li> <li>• <b>DBFBPNO4:</b> Contains information about a subpool extent. The DBFBPNO4 (extent) blocks for a subpool follow the subpool block (DBFBPNO3) until the next subpool block is found.</li> <li>• <b>DBFBPNO9:</b> Contains information about all buffers of the same buffer size. All buffers for a specific buffer size are chained off this block.</li> </ul> <p><b>X'17'</b> Information about user exit routines defined in the DFSDFxxx member of the IMS.PROCLIB data set. Mapped by macro DFSL4517.</p> <p><b>X'18'</b> Individual TCB dispatcher statistics. Mapped by macro DFSL4518.</p> <p><b>X'19'</b> Information about the pools managed by the IMS 64-bit storage manager. Mapped by macro DFSL4519.</p> <p><b>X'1A'</b> OTMA global statistics. (DFSL451A)</p> <p><b>X'1B'</b> OTMA member statistics. Each record is for an individual OTMA member. (DFSL451B)</p> <p><b>X'21'</b> IRLM subsystem statistics. (DXRRSTAT)</p> <p><b>X'22'</b> IRLM system statistics. (DXRRSTAT)</p> <p><b>X'FF'</b> End of statistics records.</p>
X'47'	DFSLOG47	CAPLOG	A checkpoint was just taken. This log record contains all the PSTs that were in the system. (DFSRCPI0)

Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'48'	DFSPALOG	PALOGREC	<p>This is a variable-length padding log record. A X'48' log record at the end of a block contains log block descriptive information. (DFSFLG0)</p> <p><b>X'00'</b> OLDS padding X'48' record.</p> <p><b>X'01'</b> X'4301' record space holder.</p> <p><b>X'02'</b> Archived OLDS X'48' record.</p> <p><b>X'03'</b> Batch SLDS padding X'48' record.</p> <p><b>X'04'</b> Archived batch SLDS X'48' record.</p>
X'4C'	DFSLOG4C	STDBLOG	<p>Activity related to database processing, according to these subcodes:</p> <p><b>X'01'</b> A backout for token was done. (DFSRBOIO)</p> <p><b>X'02'</b> A backout error occurred. (DFSRBOIO)</p> <p><b>X'04'</b> First update flag was reset. (DFSDBDR0)</p> <p><b>X'08'</b> A share level or held state was changed. (DFSDBAU0, DFSDBLOC0)</p> <p><b>X'10'</b> A write error occurred. (DFSDBH40, DFSDEVSM0)</p> <p><b>X'20'</b> A program was stopped. (DFSRBOIO)</p> <p><b>X'40'</b> A database was started. (DFSDBDR0)</p> <p><b>X'80'</b> A database was stopped. (DFSDBDR0)</p> <p><b>X'82'</b> A database backout failure occurred. (DFSRESP0)</p>
X'4E'	DFSLOG4E	DFSLOG4E	An event occurred during monitoring. This record is in the monitor log and contains statistical information about the system. (DFS MNTR0)
X'50'	DFSLOG	DLOGDB	The database was updated. This log record contains the new data on an insert and update call as well as the old data and FSE updates on a delete call. (DFS RDBL0)
X'52'	DFSLOG	DLOGDB	IMS is about to do an ISRT operation for a new root in a key sequence data set. This record contains a copy of the data before it was changed. (DFS RDBL0)
X'53'	DFSLOG53	SPLLOG	Bitmap is written for a log record for an alternate IMS that is tracking a CI split on an active IMS. (DFS RCHB0, DFSGGSP0, DFSFRSP0, DFSDEVSM0)
X'55'	DFSETPCP	DFSETPCP	Record reserved for external subsystem information. (DFS ESS30)

Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'56'	DFSETPCP	DFSETPCP	<p>IMS external subsystem support recovery log record ID. The following subcodes, contained within the record, precede information in the log record. X'56' records are written by three IMS components. These components can represent the status of IMS external subsystem transactions, the status of the connection between IMS and the CCTL, or the stages of IMS sync point processing. The subcodes listed below represent the X'56' record components and their purposes. The subcodes are contained in the record and precede data in the log record.</p> <p><b>X'000001'</b> IMS began the commit process. (DFSESP10)</p> <p><b>X'000002'</b> IMS finished the commit process. (DFSESP20)</p> <p><b>X'000003'</b> IMS signed on to an external subsystem. (DFSESS00)</p> <p><b>X'000004'</b> IMS created a thread for external subsystem. (DFSESCT0)</p> <p><b>X'000005'</b> IMS resolved an in-doubt transaction. (DFSESI60)</p> <p><b>X'000006'</b> An IMS dependent region abended. (DFSFPSP0)</p> <p><b>X'000007'</b> IMS deleted a residual recovery element (RRE) through the <b>/CHA</b> command. (DFSESI70)</p> <p><b>X'000008'</b> IMS deleted a residual recovery element (RRE) by a restart or start command. (DFSIESI0)</p> <p><b>X'000009'</b> An external subsystem disconnected. (DFSESI30)</p> <p><b>X'00000A'</b> Commit found no work to do.</p> <p><b>X'07'</b> Start of a unit of recovery (UOR).</p> <p><b>X'08'</b> A CCTL connected to DBCTL. (DFSDASI0) Mapping macro is DFSETPCP.</p>

Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'56' (cont'd)	DFSETPCP	DFSETPCP	<p><b>X'09'</b> A CCTL disconnected from DBCTL. (DFSDASD0) Mapping macro is DFSETPCP.</p> <p><b>X'10'</b> Phase 1 commit processing started. (DFSDSC00, DFSTMS00)</p> <p><b>X'11'</b> Phase 1 commit processing ended. (DFSDSC00, DFSTMS00)</p> <p><b>X'12'</b> Phase 2 commit processing ended. (DFSDSC00, DFSFXC30, DFSSMSC0, DFSTMS00)</p> <p><b>X'13'</b> Recoverable in-doubt structure (RIS) created. (DFSDRIS0)</p> <p><b>X'14'</b> Recoverable in-doubt structure (RIS) deleted. (DFSDRID0)</p> <p><b>X'15'</b> IMS restarted with z/OS Resource Recovery Services. (DFSRRSI0)</p> <p><b>X'16'</b> Interest has been registered with RRS for this UOW. (DFSRRSI0)</p> <p><b>X'37'</b> Phase 2 commit processing started by a resynchronization request. (DFSDRID0)</p> <p><b>X'38'</b> Phase 2 abort processing started by a resynchronization request. (DFSDRID0)</p> <p><b>X'FA'</b> Transaction-level statistics. (DFSDSC00, DFSFXC30, DFSTMS00)</p>
X'59'	DBFL59X	L59X	<p><b>X'10'</b> I/O from a data space has started (DBFVXOC0, DBFVOCIO)</p> <p><b>X'12'</b> A group of CIs (control intervals) from a data space has been written to DASD (DBFVXOC0, DBFVOCIO, DBFERS21)</p> <p><b>X'45'</b> Contains information about buffer use and buffer waits (using FP 64 bit buffer manager) for each UOW).</p> <p><b>X'60'</b> Records all creations, expansions, and compressions of the subpools.</p>

Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'59' (continued)	DBFLS9FF	L59FF	<p><b>X'51'</b> To indicate that nonrecoverable suppression has taken place.</p> <p><b>X'FF'</b> To track internal IMS FP information in various modules.</p> <p><b>Mapping Macro</b> This is a Fast Path log record. The following subcodes, contained within the record, precede information in the log record:</p>
	DBFLGRIM	FLIM	<p><b>X'01'</b> An input message was received. (DBFSHSP0)</p>
	DBFLGROM	FLOM	<p><b>X'03'</b> An output message was sent. (DBFSHSP0)</p>
	DBFSQRIM	DBFSQRIM	<p><b>X'11'</b> An input message was inserted on an EMHQ structure. (DBFHIEL0, DBFSYN20)</p>
	DBFSQROM	DBFSQROM	<p><b>X'16'</b> An output message was inserted on an EMHQ structure. (DBFATRM0, DBFHCTR0, DBFHCA0, DBFERMG0, DBFSYN20)</p>
	DBFBMSDB	MSUPLOG	<p><b>X'20'</b> An MSDB was updated. (DBFSLOG0, DBFBMSDB)</p>
	DBFDOCL	DOCL	<p><b>X'21'</b> DEDB area data set was opened. (DBFMOCL0)</p> <p><b>X'22'</b> DEDB area data set was closed. (DBFMOCL0)</p> <p><b>X'23'</b> DEDB area data set status was changed. (DBFMOCL0)</p>
	DBFEQE	EQE	<p><b>X'24'</b> An ADS error queue element (EQE) was created. (DBFMEQE0)</p>
	DBFLGRDQ	FLDQ	<p><b>X'36'</b> An output message was dequeued. This log record also contains information that is necessary to run the Fast Path Log Analysis utility in a shared EMH environment. (DBFHQMI0, DBFHTMG0)</p>

Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'59' (cont'd)	DBFLGSYN	SYNC	<b>X'37'</b> A synchronization point operation completed. (DBFSLG20) <b>X'38'</b> A synchronization point operation was unsuccessful. (DBFSLG20)
	DBFLGRIC	HICL5947	<b>X'47'</b> Contains a bit map of CIs that have updates in an HSSP image copy data set. (DBFSLGE1)
	DBFLSRT	LSRT	<b>X'50'</b> A DEDB was updated—DMAC status log record for DMACOCNT or DMACNXTS. (DBFSLOG0, DBFARDB0, DBFMLOP0) <b>X'53'</b> An online utility updated a DEDB. (DBFUMAL0, DBFUMAI0) <b>X'54'</b> A log record is created each time an area containing sequential dependent buffers was opened. (DBFMLOG0)
	DBFLFRSD	FLSD	<b>X'55'</b> A new buffer for sequential dependent segments was obtained. (DBFSYP20)
	DBFLSRT	LSRT	<b>X'56'</b> Indoubt SDEP buffer from the resynchronization process. (DBFMLOG0) (DBFSYP20) <b>X'57'</b> Local/Global portion of DMAC logged. (DBFARDB0, DBFUMAL0)
	DBFL56X	L56X	<b>X'58'</b> An SDEP buffer was successfully written. (DBFSYP20)
	DBFLGRDA	LGRDA	<b>X'61'</b> DEDB Alter log records.
	DBFLGRRE	FLRE	<b>X'70'</b> The MSDB relocation factor for XRF is shown. (DFSRLP00)
X'5E'	DFSLOG5E	SBLI	Sequential buffer image capture record. A sequential buffer-handler function has been called, according to these subcodes (DFSSBIC0): <b>X'00'</b> Application start record. <b>X'04'</b> Search/Read. <b>X'0C'</b> OSAM buffer-handler crossed a buffer boundary. <b>X'18'</b> New logical position. <b>X'1C'</b> Application stop record.



Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'5F'	DFSLOG5F	DLTRLOGR	A DL/I call was completed. This record contains DL/I call image capture trace data. (DFSDDLTO)
X'62'	DFSLOG62	LIOEBLK	<p>OSAM I/O error, according to these subcodes:</p> <p><b>X'01'</b> ALWAYS LOGGED; CONTAINS DECB, DCB, IOSB for a legacy OSAM I/O error. (DFSLOS80)</p> <p><b>X'02'</b> ALWAYS LOGGED; CONTAINS DEB for a legacy OSAM I/O error. (DFSLOS80)</p> <p><b>X'03'</b> OPTIONALLY LOGGED; CONTAINS IOMA OR DYNAMIC WORKAREA for a legacy OSAM I/O error. (DFSLOS80)</p> <p><b>X'04'</b> ALWAYS LOGGED; CONTAINS DECB, DCB, MMRQ for an OSAM LDS I/O error. (DFSLOS8M)</p>
X'63'	LOGCSQ	S3REC63	Log session initiation and termination. When X'02' is on in the second byte, the X'63' record represents only the deletion of a VTCB. (DFSCVLG0)
X'64'	DFSMSREC	SMREC	An inconsistency was found in processing associated with MSC. (DFSCMS00)
X'65'	DFSLOG65	SSREC	A message is about to be enqueued. (DFSCRSV0)
X'66'	LOG3600	SXREC	A message is about to be enqueued or dequeued (applicable for 3614, FINANCE, and SLU P nodes, MSC links, or ISC sessions). (DFSCVFD0, DFSCVFI0, DFSCVFN0, DFSCVLG0, DFSCMSV0, DFSCMSF0)
X'67'	DFSL6701	CTLDESC	<p>This log record is a service trace record. The following subcodes, contained within it, identify what conditions caused a particular part of the log record to be written:</p> <p><b>X'01'</b> There are three situations in which X'6701' is written:</p> <ul style="list-style-type: none"> <li>• A <b>/TRACE</b> command was issued. This record can also indicate that error blocks were written unconditionally by device-dependent code when a major error condition was detected (DFSCFEZ0).</li> <li>• Errors were detected in AOI module DFSAQUE0.</li> <li>• Errors were detected in DFSAUE00 or another AOIE type exit routine.</li> </ul> <p><b>X'03'</b> A 3270 error was detected. More information about this condition is contained in <a href="#">“Terminal communication task trace” on page 277</a>. (DFSCFEZ0)</p> <p><b>X'04'</b> An IMS notification exit failed to obtain an AWE for restart processing. IMS was unable to post the deferred unit of recovery with z/OS Resource Recovery Services.</p> <p><b>X'06'</b> A normal record that is generated when a Fast Path region ends abnormally.</p>

Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'67'			<b>X'05'</b> A thread terminated abnormally. The data portion of the log record contains diagnostic information for dependent regions. (Note that data that is contained in the DTRM work area of the XPST block, such as the abend code, might be residual data from a previous abnormal termination in the same region.) All blocks logged have eye catchers preceding them. Normal IMS DSECTs map the logged information. (DFSASK00, DFSDTTA0, DFSSDA20)
X'67'	DFSL6701	CTREC	<b>X'07'</b> An HSSP PCB is repositioned backward rather than to the next UOW. The control blocks EPST, EPCB, and SPCB are snapped in this log record for diagnostic purposes. (DBFSHDQ0)
X'67'	DFSL6740	DFS6740	<b>X'40'</b> This log record represents an IMS UOW that was placed on the Common Queue Server's (CQS) cold queue because CQS found UOWs on its private queues on a cold start of either TM (COLDSYS or COLDCOMM) or CQS. CQS moves these UOWs to the CQS cold queue and passes the UOW values to IMS. IMS logs these UOWs in the type X'6740' log record for audit purposes. The customer can then process these log records to determine what action to take for these UOWs. (DFSSQ030, DBFSQ030)

Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'67'	DFS67D0	DFS67D0	<p><b>X'D0'</b> Indicates the diagnostic record of a failed service request.</p> <p><b>X'01'</b> Failure during a DB DL/I call.</p> <p><b>X'02'</b> Failure during a DC DL/I call. (DFSCPY00, DFSDLA30, DBFHGU10, DFSTMAPO)</p> <p><b>X'03'</b> Failure during a SYS DL/I call.</p> <p><b>X'04'</b> An exit failure occurred. (DFSRRSI0)</p> <p><b>X'05'</b> Failure during SPOOL API processing. (DFSIAFP0)</p> <p><b>X'06'</b> Failure during Transaction Manager schedule processing. (DFSTMAS0, DFSTMCD0)</p> <p><b>X'07'</b> Failure during Service Logical Unit Manager (SLUM) processing.</p> <p><b>X'08'</b> Failure during Asynchronous Logical Unit Manager processing.</p> <p><b>X'09'</b> Failure during coupling facility processing. (DFSDCFR0, DFSDMAW0)</p> <p><b>X'0A'</b> Failure during queue manager processing.</p> <p><b>X'0B'</b> Failure during shared queues interface processing. (DBFIPQS0, DFSITQS0, DBFILQS0, DFSILQS0)</p> <p><b>X'0C'</b> Failure during NDM user exit interface processing. (DFSNDMIO)</p> <p><b>X'0D'</b> Failure during shared queues CQSINFRM processing.</p> <p><b>X'0E'</b> Failure during shared queues request processing. (DBFHCA50, DBFHGU10, DBFHQS0)</p> <p><b>X'0F'</b> Failure during UOWE resync processing. (DBFHGU10, DBFHCA50)</p> <p><b>X'10'</b> Shared EMH z/OS cross-system coupling facility communication error. (DBFHXC50)</p> <p><b>X'11'</b> An unsolicited output message was detected. (DBFHQS0)</p> <p><b>X'12'</b> In-flight input message deleted. (DBFHCA50)</p> <p><b>X'1C'</b> DBFDSRP0 Failed to write in-doubt CI.</p>

Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'67' (cont'd)	DFS67D0	DFS67D0	<p><b>X'13'</b> Fast Path Queue Manager Diagnostics. (DBFHQMIO).</p> <p><b>X'14'</b> System Termination Diagnostics. (DFSSDA20, DFSTRM00).</p> <p><b>X'15'</b> System Service Error. (DFSOCMD0).</p> <p><b>X'16'</b> Unexpected return or reason code from RM, OM, SCI, or CQS request.</p> <p><b>X'17'</b> Failure during RM update, query, or delete processing.</p> <p><b>X'19'</b> VSAM JRNAD ERROR. (DFSDVSM0)</p> <p><b>X'1A'</b> Sysplex Serial Program Manager (SSPM) encountered an error.</p> <p><b>X'1B'</b> Abend U3310 or statusBD caused by an IRLM long lock timeout.</p>
X'67'	DFSL67FD	SNELDESC	<p><b>X'ED'</b> Sequential buffering SNAP, created during a periodic evaluation of the sequential buffering process by the SBESNAP option. (DFSSBSN0)</p> <p><b>X'EE'</b> SNAP of a call to the sequential buffering buffer-handler created by the SBSNAP option. (DFSSBSN0)</p> <p><b>X'EF'</b> SNAP created when the sequential buffering COMPARE option detects a mismatch between the results of a call to the buffer handler and the DASD block as stored on DASD. (DFSSBSN0)</p> <p><b>X'FB'</b> An invalid AWE was detected. Some of the possible causes of the invalid AWE include conflicting parameters, missing addresses or bad pointers. The log record indicates which of the processing modules detected the invalid AWE.</p> <p><b>X'FD'</b> A SNAP call was issued. (DFSERA20)</p> <p><b>X'FF'</b> A pseudoabend or dependent region abnormal termination occurred. Further information of this condition is contained in <a href="#">“SNAP call facility”</a> on page 572. (DFSERA20)</p>
X'67'	DFSL67FA	DFSTRHD	<p><b>X'FA'</b> Contains images of the in memory trace tables. These tables are written to the log when requested by the OPTIONS statement in the VSPEC=parameter member or the <b>/TRACE</b> command. (DFSTRA20)</p>
X'69'	DFSLOG69	JM	An unauthorized 3275 terminal dialed into a line specified as VERIFY=YES. (DFSDS060)
X'6C'	DFSMSCRC	CMSCREC	MSC partner systems were started. (DFSCMSW0)

Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'6D'	DFSLOG6D	SURVLOG	<p>This log record is used in an XRF environment when:</p> <ul style="list-style-type: none"> <li>• XRF surveillance was started or stopped.</li> <li>• A write error occurred on the active subsystem.</li> <li>• The interval or timeout values on the active subsystem were changed by a <b>/CHANGE</b> command. (DFSHIC40, DFSHSRV0, DFSISL60)</li> </ul> <p><b>X'04'</b> Fast DB recovery creates this log record to indicate which TASK or ITASK received a TIMEOUT or is in a wait or loop for more than one second.</p> <p><b>X'40'</b> Diagnostic information for FDR. This record is written without any data in a one-second interval to the log.</p>
X'6E'	DFSLOG6E	LUMLOG	<p>One of the following SNA commands was processed: QEC, QC, RELQ, RSHUT, SHUTD, SHUTC, LUS. (DFSHCLG0)</p>

Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'70'	DFSLOG70	OLCREC	<p><b>X'00'</b> An online change <b>/MODIFY</b> command sequence completed successfully. The IMS.MODSTAT data set is being updated. (DFSICV80)</p> <p><b>X'01'</b> Allows the XRF primary to signal the alternate that the transaction has been stopped (PSTOP) by module DFSSMSC0. (DFSICV90)</p> <p><b>X'02'</b> Begin write of the new versions of the updated members log record. The log records will contain the names of all the members that are affected by the member online change. If more member names exist than can fit in one X'7002' log record, multiple X'7002' log records are logged.</p> <p><b>X'03'</b> Write complete ACB library member online change log record.</p> <p><b>X'04'</b> Commit start ACB library member online change log record.</p> <p><b>X'05'</b> Commit complete ACB library member online change log record.</p> <p><b>X'06'</b> Restart abort ACB library member online change record.</p> <p><b>X'07'</b> Begin update of the staging data set to delete members already committed log record.</p> <p><b>X'08'</b> Complete update of the staging data set to delete members already committed log record.</p> <p><b>X'10'</b> Terminate ACB library member online change process log record.</p> <p><b>X'12'</b> Begin rename old directory version catalog import/DDI activation log record.</p> <p><b>X'13'</b> End rename old directory version catalog import/DDI activation log record.</p> <p><b>X'14'</b> Begin delete old directory version catalog import/DDI activation log record.</p> <p><b>X'15'</b> End delete old directory version catalog import/DDI activation log record.</p> <p><b>X'29'</b> Prepare phase has locked the OLCSTAT data set log record.</p> <p><b>X'30'</b> Commit phase before it attempts to update the OLCSTAT data set log record.</p>

Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'70' (cont'd)	DFSLOG70	OLCREC	<p><b>X'30'</b> Commit phase before it attempts to update the OLCSTAT data set log record.</p> <p><b>X'31'</b> Commit phase after it attempted to update the OLCSTAT data set log record.</p> <p><b>X'32'</b> Commit phase or terminate phase after unlocking the OLCSTAT data set log record.</p> <p><b>X'40'</b> Cleanup to abort import/DDDL activation with no updates to directory log record.</p> <p><b>X'41'</b> Cleanup at abort phase 1 to start delete new directory version catalog import/DDDL activation log record.</p> <p><b>X'42'</b> Cleanup at abort phase 1 to end delete new directory version catalog import/DDDL activation log record.</p> <p><b>X'43'</b> Cleanup at abort phase 2 to start delete new directory version catalog import/DDDL activation log record.</p>

Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'70' (cont'd)	DFSLOG70	OLCREC	<p><b>X'44'</b> Cleanup at abort phase 2 to end delete new directory version catalog import/DDL activation log record.</p> <p><b>X'45'</b> Cleanup at abort phase 2 to rename directory member catalog import/DDL activation log record.</p> <p><b>X'46'</b> Cleanup at abort phase 3 to start delete new directory version catalog import/DDL activation log record.</p> <p><b>X'47'</b> Cleanup at abort phase 3 to end delete new directory version catalog import/DDL activation log record.</p> <p><b>X'48'</b> Cleanup at abort phase 3 to start rename directory member catalog import/DDL activation log record.</p> <p><b>X'49'</b> Cleanup at abort phase 3 to end rename directory member catalog import/DDL activation log record.</p> <p><b>X'4A'</b> Cleanup at commit to start delete old directory version catalog import/DDL activation log record.</p> <p><b>X'4B'</b> Cleanup at commit to end delete old directory version catalog import/DDL activation log record.</p> <p><b>X'4C'</b> Cleanup at commit to update online change timestamp catalog import/DDL activation log record.</p> <p><b>X'4D'</b> Cleanup at commit to delete staging members catalog import/DDL activation log record.</p> <p><b>X'4E'</b> Cleanup at abort phase complete import/DDL activation log record.</p> <p><b>X'50'</b> Cleanup at commit to delete directory work members log import/DDL activation log record.</p> <p><b>X'51'</b> Cleanup at commit phase complete import/DDL activation log record.</p>



Table 135. IMS log records used to analyze IMS problems (continued)

Type	Mapping macro name	DSECT name	Why written (issuing module)
X'72'	DFSLOG72	USRREC	<p>Used by dynamic terminals during sign on create, sign off delete, and sign on modification. The following subcodes identify the conditions that caused a particular log record to be written and the content of the log record:</p> <p><b>X'01'</b> ETO user structure dynamically created. Contains the SPQB name and one or more CNTs.</p> <p><b>X'02'</b> ETO user structure dynamically deleted. Contains only the SPQB name.</p> <p><b>X'03'</b> ETO user structure modified. Contains the SPQB name and one or more CNTs.</p> <p><b>X'04'</b> One or more CNTs added to an ETO user structure. Contains the SPQB name and the CNTs that were added.</p> <p><b>X'06'</b> Dynamic logical link block (LNB) created.</p> <p><b>X'07'</b> Transaction SMB changed by shared queues SID exchange.</p> <p><b>X'08'</b> TPIPE for OTMA in an XRF environment deleted.</p>
X'99'	DFSDXBLK	DFSDXBLK	<p>Created by the logging option on the EXIT= parameter on the DBDGEN. This allows a user to capture database changes that can then be propagated to another environment (for example, Db2 for z/OS). The subcodes indicate the type of record being logged:</p> <p><b>X'04'</b> Changed data</p> <p><b>X'28'</b> End of job (EOJ)</p> <p><b>X'30'</b> SETS call</p> <p><b>X'34'</b> ROLS call</p> <p>This log record is mapped by the macro, DFSDXBLK, which is not shipped.</p>

#### Related concepts

[“Determining a value for the LOCKMAX parameter” on page 220](#)

To decide what value to use for the LOCKMAX parameter, analyze over a period of time the X'37', X'41', and X'5937' commit log records to determine the maximum number of locks being held per unit of work by the application.

#### Related reference

[“Terminal communication task trace” on page 277](#)

When an output device (such as a terminal, line, or node) hangs, you can use the terminal communication task trace to diagnose the problem.

[“SNAP call facility” on page 572](#)

The SNAP call facility (DFSERA20) produces SNAP dumps of DL/I control blocks and identifies calling routines that generate SNAP dumps. Supervisor call (SVC) dumps are generated only for the intended abend codes or status codes, and for unknown calling routines.

[“DEDB control interval \(CI\) problem assistance aids” on page 405](#)

When you print portions of the DEDB, the control intervals (CIs) have these identifying characteristics.

## Format of X'29' log record

The format of the X'29' log records, including offset (ex.), length, field name, and field description are listed.

### X'2900': OLR command received

A X'2900' log record is written to indicate the receipt of a HALDB OLR command.

Only one X'2900' log record is written for each command. The following table describes the layout of the X'2900' log record.

Table 136. X'2900' log record layout

Offset (hex.)	Length (decimal)	Field name	Field description
00	2	HORLENG	Length of this record, including this length field and the sequence number
02	2	HORRSV1	X'0000' Reserved
04	1	HORTYPE	X'29' Record type
05	1	HORSTYPE	X'00' Record sub-type
06	2	HORRSV2	X'0000' Reserved
08	8	HORRSENM	RSE name or IMS ID
10	16	HOROMCT	OM command token if from a type-2 command, or zeros if from a type-1 command
20	1	HORCTYPE	Command type flags: "1000 ...." INITIATE "0100 ...." UPDATE "0010 ...." QUERY "0001 ...." TERMINATE
21		HOROCMD	OM command instance block (OCMD) if from a type-2 command, or zeros if from a type-1 command

### X'2910': ownership established

Ownership of the online reorganization for a partition was established through DBRC.

The following table describes the X'2910' log record layout.

Table 137. X'2910' log record layout

Offset (hex.)	Length (decimal)	Field name	Field description
00	2	HORLENG	Length of this record, including this length field and the sequence number

Table 137. X'2910' log record layout (continued)

Offset (hex.)	Length (decimal)	Field name	Field description
02	2	HORRSV1	X'0000' Reserved
04	1	HORTYPE	X'29' Record type
05	1	HORSTYPE	X'10' Record sub-type
06	2	HORPSTNO	PST number
08	8	HORRSENM	RSE name or IMS ID
10	8	HORDBD	DBD name
18	8	HORPSB	PSB name
18	1	HORPSB0	C'0'
19	7	HORPART	Partition name
20	1	HOROFLG1	Flag bit definitions: <b>B'10.. ....'</b> The <b>INITIATE OLREORG</b> command was issued with OPTION(NODEL) <b>B'01.. ....'</b> The <b>INITIATE OLREORG</b> command was issued with or defaulted to OPTION(DEL) <b>B'..1. ....'</b> The <b>INITIATE OLREORG</b> command was issued with the RATE option <b>B'...0 ....'</b> A new reorganization started <b>B'...1 ....'</b> The reorganization restarted <b>B'.... ..1.'</b> The <b>INITIATE OLREORG</b> command was issued with or defaulted to OPTION(NOREL) <b>B'.... ....1'</b> The <b>INITIATE OLREORG</b> command was issued with OPTION(REL)
21	1	HORORATEV	RATE value (1 through 100 percent)

### X'2911': conditional ownership established

An attempt to establish ownership of OLR in the RECON will be made. The ownership may or may not be recorded in the RECON for this partition. When OLR has established ownership, this log record contains the data set retention (DEL/NODEL/REL/NOREL) and rate options specified on the INIT OLREORG command.

The following table describes the X'2911' log record layout.

Table 138. X'2911' log record layout

Offset (hex.)	Length (decimal)	Field name	Field description
00	2	HORLENG	Length of this record, including this length field and the sequence number
02	2	HORRSV1	X'0000'
04	1	HORTYPE	X'29' Record type
05	1	HORSTYPE	X'10' Record sub-type
06	2	HORPSTNO	PST number
08	8	HORSENM	RSE name or IMS ID
10	8	HORDBD	DBD name
18	8	HORPSB	PSB name
18	1	HORPSB0	C'0'
19	7	HORPART	Partition name
20	1	HORCOFLG1	Flag bit definitions: <b>B'10.. ....'</b> The <b>INITIATE OLREORG</b> command was issued with <b>OPTION(NODEL)</b> <b>B'01.. ....'</b> The <b>INITIATE OLREORG</b> command was issued with <b>OPTION(DEL)</b> <b>B'..1. ....'</b> The <b>INITIATE OLREORG</b> command was issued with <b>OPTION(NOREL)</b> <b>B'...1 ....'</b> The <b>INITIATE OLREORG</b> command was issued with <b>OPTION(REL)</b>
21	1	HORCORATEV	RATE value

### X'2920': UPDATE OLREORG command

The UPDATE OLREORG command was processed. The X'2920' log record is written once for each HALDB partition affected by the UPDATE OLREORG command.

The following table describes the X'2920' log record layout.

Table 139. X'2920' log record layout

Offset (hex.)	Length (decimal)	Field name	Field description
00	2	HORLENG	Length of this record, including this length field and the sequence number.
02	2	HORRSV1	X'0000' Reserved.
04	1	HORTYPE	X'29' Record type.
05	1	HORSTYPE	X'20' Record sub-type.

Table 139. X'2920' log record layout (continued)

Offset (hex.)	Length (decimal)	Field name	Field description
06	2	HORPSTNO	PST number.
08	8	HORRSENM	RSE name or IMS ID.
10	8	HORDBD	DBD name.
18	8	HORPSB	PSB name.
18	1	HORPSB0	C'0'
19	7	HORPART	Partition name
20	1	HORUFLG1	Flags: <b>B'10.. ....'</b> The NODEL option is now in effect. <b>B'01.. ....'</b> The DEL option is now in effect. <b>B'..1. ....'</b> The NOREL option is now in effect. <b>B'...1 ....'</b> The REL option is now in effect.
21	1	HORURATEV	RATE value (1 through 100 percent) that is now in effect.

## X'2930': output data set information

The output data sets have been successfully validated or created. This X'2930' log record contains various characteristics of all of the output data sets, both those that were preexisting and those that were automatically created.

There is enough information to recreate any of these output data sets. The following table describes the X'2930' log record layout.

Table 140. X'2930' log record layout

Offset (hex.)	Length (decimal)	Field name	Field description
00	2	HORLENG	Length of this record, including this length field and the sequence number.
02	2	HORRSV1	X'0000' Reserved.
04	1	HORTYPE	X'29' Record type.
05	1	HORSTYPE	X'30' Record sub-type.
06	2	HORPSTNO	PST number.
08	8	HORRSENM1	RSE name or IMS ID.
10	8	HORDBD	DBD name.
18	8	HORPSB	PSB name.
18	1	HORPSB0	C'0'

Table 140. X'2930' log record layout (continued)

Offset (hex.)	Length (decimal)	Field name	Field description
19	7	HORPART	Partition name
20	4	HORDUSN	Update sequence number (USN).
24	4	HORDUSID	Update set ID (USID).
28	1	HORDFLG1	Flags: ". . . . . 0." The A-thru-J and X data sets are the output data sets. ". . . . . 1." The M-thru-V and Y data sets are the output data sets. ". . . . . 0" PHDAM database. ". . . . . 1" PHIDAM database.
29	1	HORDDSECT	Number of following entries.
2A		HORDDSE	The group of fields shown in Table 141 on page 496 is repeated for each output data set. There are two entries for the primary index data set of a PHIDAM database.

The group of fields shown in the following table is repeated for each output data set. There are two entries for the primary index data set of a PHIDAM database.

Table 141. X'2930' log record layout — repeated data set fields

Offset (hex.)	Length (decimal)	Field name	Field description
00	2	HORDENTL	Entry length, including this length field.
04	1	HORDUCBDCL	UCB device class.
05	1	HORDUCBDCC	UCB device class.
06	1	HORDDCBN	DCB number, with A-thru-J, X or M-thru-V, Y indicator: "0. . . . ." One of the A-thru-J or X data sets. "1. . . . ." One of the M-thru-V or Y data sets. ". . . . . nnnn" DCB number.
07	8	HORDDDNAM	The DD name used for allocation.

Table 141. X'2930' log record layout — repeated data set fields (continued)

Offset (hex.)	Length (decimal)	Field name	Field description
0F	1	HORDDFL1	<p>Data set flags:</p> <p>"0... .." OSAM data set.</p> <p>"1... .." VSAM data set.</p> <p>".0.. .." Data set existed before INITIATE OLREORG command.</p> <p>".1.. .." Data set was created automatically.</p> <p>".0. ...." Non-SMS-managed data set.</p> <p>".1. ...." SMS-managed data set.</p> <p>"1..0 ...." VSAM ESDS.</p> <p>"1..1 0..." VSAM KSDS data component (DCB number X'05' or X'85').</p> <p>"1..1 1..." VSAM KSDS index component (DCB number X'04' or X'84').</p> <p>".... .0..." NOREPLICATE. Do not replicate index records (or replication not applicable).</p>
10	1	HORDDFL2	<p>Data set space allocation flags for Input data set primary space allocation unit</p> <p>"1000 ...." Records (VSAM) or blocks (OSAM).</p> <p>"0100 ...." Bytes</p> <p>"0010 ...." Kilobytes.</p> <p>"....0001" Megabytes.</p> <p>".... 1000" Cylinders</p> <p>".... 0100" Tracks</p>

Table 141. X'2930' log record layout — repeated data set fields (continued)

Offset (hex.)	Length (decimal)	Field name	Field description
11	1	HORDDFL3	Data set space allocation flags for Input data set secondary space allocation unit: <b>"1000...."</b> Records (VSAM) or blocks (OSAM) <b>"0100...."</b> Bytes <b>"0010...."</b> Kilobytes <b>"....00001"</b> Megabytes <b>"....1000"</b> Cylinders <b>"....0100"</b> Tracks
12	4	HORDRCSZ	VSAM record size or OSAM block size.
16	4	HORDCISZ	For VSAM, control interval size. For OSAM: 0.
1A	8	HORDDATC	For SMS-managed, data class if present. Otherwise, blanks.
22	8	HORDSTGC	For SMS-managed, storage class. Otherwise, blanks.
2A	8	HORDMGTC	For SMS-managed, management class if present. Otherwise, blanks.
32	4	HORDPRIA	Primary allocation amount. See HORDDFL2.
36	4	HORDPRIACV	Primary space converted amount in blocks/records/megabytes.
3A	4	HORDSECA	Secondary allocation amount. See HORDDFL2.
3E	4	HORDSECACV	Secondary space converted amount in blocks/records/megabytes.
42	1	HORDFSCI	For VSAM KSDS data component, free space percentage in each control interval.
43	1	HORDFSCA	For VSAM KSDS data component, free space percentage in each control area.
44	1	HORDKYLN	For VSAM KSDS data component, key length.
45	2	HORDKYOF	For VSAM KSDS data component, key offset.
47	1	HORDSHRR	For VSAM, SHAREOPTIONS value, cross-region.
48	1	HORDSHRS	For VSAM, SHAREOPTIONS value, cross-system.
49	1	HORDVOLR	Number of existing or requested DASD volumes.
4A	1	HORDVOLC	Number of volume serial numbers following.
4B	<i>n</i>	HORDVOLS	Volume serial numbers.
4C	6	HORDVOL	Volume serial number (repeated). HORDVOLC contains the number of these volume serial number entries.



## X'2931': conditional output data set information

The type X'2931' log record contains conditional output data set information based on the contents of the 2930 log record.

The output data sets have been successfully validated or created but the OLR start time has not been set and Cursor Active has not been set in the RECON. The 2931 log record is a copy of the 2930 log record and contains the same information as the 2930 except for the USID and the USN which are excluded.

## X'2940': cursor-active status set

The cursor is active. Initialization of the reorganization of the partition was completed successfully, two sets of data sets exist, and copying is about to begin. The reorganization was recorded through DBRC as being in a cursor-active status.

The following table describes the X'2940' log record layout.

Table 142. X'2940' log record layout

Offset (hex.)	Length (decimal)	Field name	Field description
00	2	HORLENG	Length of this record, including this length field and the sequence number
02	2	HORRSV1	X'0000' Reserved
04	1	HORTYPE	X'29' Record type
05	1	HORSTYPE	X'40' Record sub-type
06	2	HORPSTNO	PST number
08	8	HORRSENM	RSE name or IMS ID
10	8	HORDBD	DBD name
18	8	HORPSB	PSB name
18	1	HORPSB0	C'0'
19	7	HORPART	Partition name
20	4	HORDUSN	Update sequence number (USN)
24	4	HORDUSID	Update set ID (USID)
28	1	HORAFLG1	Flags: ".... 0..." This log record was created at the IMS doing the HALDB Online Reorganization. ".... 1..." This log record was created by an IMS doing data sharing with the IMS doing the HALDB Online Reorganization. ".... ..0." The A-thru-J and X data sets are the input data sets. ".... ..1." The M-thru-V and Y data sets are the input data sets. ".... ...0" PHDAM database. ".... ...1" PHIDAM database.

Table 142. X'2940' log record layout (continued)

Offset (hex.)	Length (decimal)	Field name	Field description
29	12	HORARATIM	Time stamp of reorganization active. This is the time of the DBRC ALLOC for the first output data set.

### X'2950': cursor movement

The cursor was updated. The X'2950' log record appears before the X'3730' log record that indicates that a unit of reorganization was committed.

The following table describes the X'2950' log record layout.

Table 143. X'2950' log record layout

Offset (hex.)	Length (decimal)	Field name	Field description
00	2	HORLENG	Length of this record, including this length field and the sequence number
02	2	HORRSV1	X'0000' Reserved
04	1	HOR TYPE	X'29' Record type
05	1	HORSTYPE	X'50' Record sub-type
06	2	HORPSTNO	PST number
08	8	HORRSENM	RSE name or IMS ID
10	8	HORDBD	DBD name
18	8	HORPSB	PSB name
18	1	HORPSB0	C'0'
19	7	HORPART	Partition name
20	8	HORMUORS	Segments moved in this UOR
28	8	HORMUORZ	Size moved in this UOR
30	8	HORMRSEG	Total segments moved before this UOR
38	8	HORMRSIZE	Total size moved before this UOR
40	4	HORMORSA	Number of roots moved this UOR
44	4	HORMLOCK	Lock count for this OLR
48	4	HORMSTT	UOR start time, in unsigned binary format
4C	12	HORMUTST	UOR start time, in UTC format
58	4	HORMTIME	Execution time
5C	4	HORMWAIT	Wait time
60	4	HORMORSZ	UOR size calculated
64	4	HORMORW1	Not used
68	4	HORMORW2	Not used

Table 143. X'2950' log record layout (continued)

Offset (hex.)	Length (decimal)	Field name	Field description
6C	8	HORMCHNG	DFSORP20 CHANGEID
74	1	HORMFLG1	Flags: ". . . . . 0." The A-thru-J or X data sets are the input data sets ". . . . . 1." The M-thru-V or Y data sets are the input data sets ". . . . . 0" PHDAM database ". . . . . 1" PHIDAM database
75	4	HORMRBA	PHDAM cursor RBA
75	1	HORMKLN	Length of root key for PHIDAM
76	<i>n</i>	HORMKEY	PHIDAM cursor root key. The length <i>n</i> is the length of the root key.

### X'2970': cursor-active status reset

The cursor is inactive. Copying from the input to the output data sets has completed. The reorganization was recorded through DBRC as no longer being in a cursor-active status.

The following table describes the X'2970' log record layout.

Table 144. X'2970' log record layout

Offset (hex.)	Length (decimal)	Field name	Field description
00	2	HORLENG	Length of this record, including this length field and the sequence number
02	2	HORRSV1	X'0000' Reserved
04	1	HORATYPE	X'29' Record type
05	1	HORSTTYPE	X'70' Record sub-type
06	2	HORPSTNO	PST number
08	8	HORRSENM	RSE name or IMS ID
10	8	HORDBD	DBD name
18	8	HORPSB	PSB name
18	1	HORPSB0	C'0'
19	7	HORPART	Partition name
20	4	HOREUSN	Update sequence number (USN)
24	4	HOREUSID	Update set ID (USID)

Table 144. X'2970' log record layout (continued)

Offset (hex.)	Length (decimal)	Field name	Field description
28	1	HOREFLG1	Flags: <b>B'10.. ....'</b> The NODEL option is now in effect. <b>B'01.. ....'</b> The DEL option is now in effect. <b>B'.... 0...'</b> This log record was created at the IMS doing the HALDB Online Reorganization. <b>B'.... 1...'</b> This log record was created by an IMS doing data sharing with the IMS doing the HALDB Online Reorganization. <b>B'.... ..0.'</b> The A-thru-J or X data sets are the input data sets. <b>B'.... ..1.'</b> The M-thru-V or Y data sets are the input data sets. <b>B'..1. ....'</b> The NOREL option is now in effect. <b>B'...1 ....'</b> The REL option is now in effect.
29	12	HORCITIM	Time stamp of cursor inactive

### X'2971': conditional reset of the cursor active

OLR has completed copying all data from the input data sets to the output data sets. Following this log record an attempt to reset cursor active in the RECON will be made.

The following table describes the X'2971' log record layout.

Table 145. X'2990' log record layout

Offset (hex.)	Length (decimal)	Field name	Field description
00	2	HORLENG	Length of this record, including this length field and the sequence number
02	2	HORRSV1	X'0000'
04	1	HORTYPE	X'29' Record type
05	1	HORSTYPE	X'71' Record sub-type
06	2	HORPSTNO	PST number
08	8	HORRSENM	RSE name or IMS ID
10	8	HORDBD	DBD name
18	8	HORPSB	PSB name
18	1	HORPSB0	C'0'
19	7	HORPART	Partition name

## X'2990': ownership relinquished

Ownership of the reorganization for a partition was relinquished through DBRC.

See [Table 145 on page 502](#) for X'2990' log record layout.

Table 146. X'2990' log record layout

Offset (hex.)	Length (decimal )	Field name	Field description
00	2	HORLENG	Length of this record, including this length field and the sequence number
02	2	HORRSV1	X'0000' Reserved
04	1	HORTYPE	X'29' Record type
05	1	HORSTYPE	X'90' Record sub-type
06	2	HORPSTNO	PST number
08	8	HORRSENM	RSE name or IMS ID
10	8	HORDBD	DBD name
18	8	HORPSB	PSB name
18	1	HORPSB0	C'0'
19	7	HORPART	Partition name
22	1	HORREAS	Reason for relinquishing ownership: <b>X'80'</b> Normal completion of the online reorganization. <b>X'40'</b> Pseudo-abend during an online reorganization. <b>X'20'</b> TERM command during an online reorganization.
21	4	HORABTRM	Pseudo-abend code
25	8	HORSEGCT	Number of segments copied

## Format of X'67' log record

A physical log record consists of one or more subrecords. Each subrecord is followed by its associated data.

The following figure shows the layout of the X'67' log record.

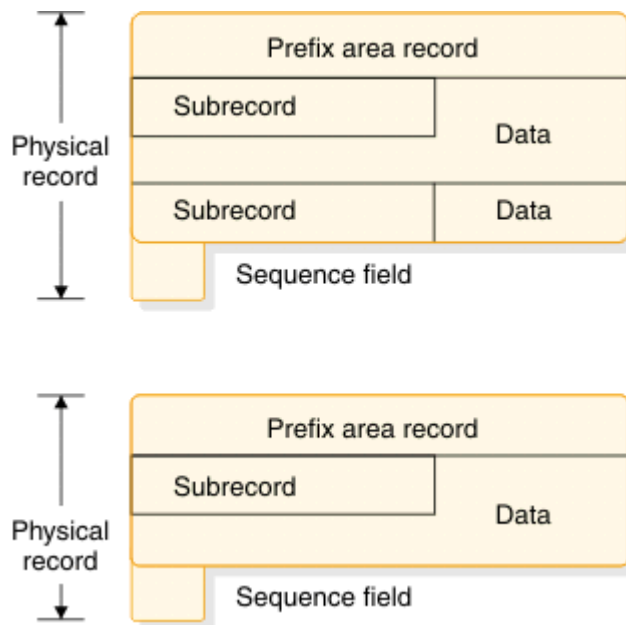


Figure 76. X'67' log record layout

### Related tasks

“Program isolation trace” on page 220

The program isolation (PI) trace traces all calls to the IMS enqueue/dequeue module (DFSFXC10) and writes the trace entries to the system log as type X'67FA' records.

### Log record prefix area format

The log record prefix area formats for X'67' and X'67FA' records are shown, including the offset, length, and a brief description.

Table 147. Log record prefix area format for X'67'

Offset (hexadecimal)	Length (decimal)	Description
00	2	Length of record, including sequence number
02	2	Reserved
04	1	X'67' record type
05	1	X'FB' X'FD' X'FF'
06	2	Reserved
08	4	Requestor identification
0C	2	Record segment number
0E	2	Reserved
10	4	Time
14	4	Date
18	4	Reserved
1C	4	Condition indicator

For X'67FA' records, the order of the fields from offset X'08' through X'14' is shown in the following table.

Table 148. Log record prefix area format for X'67FA' records

Offset (hexadecimal)	Length (decimal)	Description
08	4	Date
0C	4	Time
10	2	Table identification
12	2	Flag bytes

## Log subrecord and data area

The X'67' log subrecord uses the data format given in the following table.

Table 149. Log subrecord area format

Offset (hexadecimal)	Length (decimal)	Description
00	8	Element identification
08	2	Reserved
0A	2	Element data length, excluding descriptor
0C	4	Main storage address of data when logged; zero when continued from previous element

Table 150. Log data area format

Offset (hexadecimal)	Length	Description
10	variable	Logged data

## Log sequence field format

The log sequence field format is described in the following table.

Table 151. Log sequence field format

Offset (hexadecimal)	Length (decimal)	Description
n	8	STCK time stamp representing the time the log record was written. The time stamp is not necessarily on a word boundary.
n+8	8	Sequence number within the IMS control region.

## Format of OTMA X'6701' log records

OTMA uses the X'6701' log record to document the sending and receiving of certain types of messages, such as messages associated with the processing of synchronous callout requests.

In shared queues environments, OTMA and APPC also use X'6701 log records to document the receipt of certain messages in a front-end IMS system that are sent from a back-end IMS system via a z/OS cross-system coupling facility (XCF).

### ***Format of synchronous callout log records***

Synchronous callout messages are non-recoverable OTMA messages that are initiated by an ICAL call through the AIBTDLI interface. These messages do not go through the IMS message queue structure.

The processing of a synchronous callout request and its corresponding response involves the exchange of a number of different message types, such as a resume tpipe request message, the callout request message, the ACK or NAK message to the callout request, the response message, and so forth.

For each possible type of message that is exchanged in the processing of a synchronous callout request and its response, a simplified X'6701' log record is written that contains an ID that identifies the type of message the log record documents.

The message-type identifier for each message type is displayed in the ID= field in the log records. Message types are identified by the following ID= values that are shown in alphabetical order:

**YACK**

ACK that OTMA received from the client for a callout request

**YAKO**

ACK that OTMA sent to the client for a callout response

**YCRT**

CANCEL RESUME TPIPE request that OTMA received from the client, usually as a result of the timeout of a RESUME TPIPE request

**YNAK**

NAK that OTMA received from the client for a callout request

**YNKO**

NAK that OTMA sent to the client for a callout response

**YOUT**

Synchronous callout request that OTMA sent to the client

**YPSI**

ACK or NAK that OTMA received from the client is sent to a wrong IMS system, or delayed ACK or NAK that is received by OTMA

**YRSP**

Response to the synchronous callout that OTMA received from the client

**YRTR**

RESUME TPIPE request that OTMA received from the client

In the output log records, all the messages for a particular synchronous callout request share the same correlation ID and the message types are recorded in the order in which they occur in the flow during synchronous callout processing.

In the flow of a typical synchronous callout interaction, the message types occur in the following order:

1. YRTR
2. YOUT
3. YACK or YNAK
4. YRSP
5. YAKO or YNKO
6. YCRT

### **Examples of formatted X'6701' log records**

The X'6701' log records for each message type in OTMA callout processing have a similar format. The format shown in following examples is representative of the format of all OTMA X'6701' log records for synchronous callout processing.



## YOUT example of the OTMA X'6701' log record

The following example shows an X'6701' log record with ID = YOUT. The YOUT record is written when a synchronous callout message is sent to an OTMA client, such as IMS Connect.

```
INTERNAL TRACE RECORD          ID = YOUT  SEGNO=00  RECNO = 0000082F  TIME  12:48:22.189  DATE
2008.234?
MCI
PREF

  0992D84E 000000                                0180  20000020 E3D7C9D7  C5F14040 A0F00000
*.....TPIPE1 .0..*
0992D860 000012 00030000 00000000 00000001 0000
*.....STATE .....0 ..*
STATE

  0C29F040 000000 00482048 01004040 40404040 40400000 00000000 00B60000 00000000 00000000
*.....*
0C29F060 000020 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0C29F080 000040 00000000 00000000
*.....SECURITY.....0H.....FP*
SECURITY

  0C29F088 000000                                000E0000 0902C6D7  C5F0F0F0 F140                                *      .....FPE0001
USER SGM..*
USER
SGM

  0C29F096 000000                                0100 00000000 00000000
*.....*
0C29F0A0 00000A 0000E3D7 C9D7C5F1 40400000 00000000 00000000 00000000 00000000 00000000
*..TPIPE1 .....*
0C29F0C0 00002A 00000000 00000000 00000000 00000000 00000000 40000000 00000000 00000000
*.....*
0C29F0E0 00004A 00000000 00000100 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0C29F100 00006A 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0C29F120 00008A                                SAME AS
ABOVE
0C29F140 0000AA 00000000 00000000 00000000 00280000  C9D4E2F1 01000017 00220002 C2E0DE33
*.....IMS1.....B...*
0C29F160 0000CA 9B08FE7C E3D7C9D7  C5F14040 C6D7C5F0  F0F0F140 00000000 00000000 00000000  *...@TPIPE1
FPE0001 .....*
0C29F180 0000EA 00000000 00000000 00000000 00000000 00000000 0000
*.....APPL SGM..*
APPL
SGM

  0C29F196 000000                                0068 0000C6C9 D9E2E340
*.....FIRST *
0C29F1A0 00000A D4C5E2E2 C1C7C540  C9E240E2 C5D5E340  E3D640C3 D3C9C5D5  E3404040 40404040  *MESSAGE IS SENT TO
CLIENT *
0C29F1C0 00002A 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040
*.....*
0C29F1E0 00004A 40404040 40404040 40404040 40404040 40404040 40404040 40404040 4040
*.....B.*
```

## YACK example of the X'6701' log record

The following example shows an X'6701' log record with ID = YACK. The YACK record is written when a synchronous callout message is sent to an OTMA client, such as IMS Connect.

After a synchronous callout message is received and an acknowledgment is sent back to OTMA, OTMA writes the following X'6701' log record with ID = YACK:

```
INTERNAL TRACE RECORD          ID = YACK  SEGNO=00  RECNO = 00000830  TIME  12:48:22.303  DATE
2008.234?
MCI
PREF

  7F2D8E28 000000                                01208000 0000E3D7  C9D7C5F1 4040A0E0 0000000F 00000000
*.....TPIPE1 .....*
7F2D8E40 000018 00000000 00010000
*.....STATE .....*
STATE

  7F2D8E48 000000                                00482040 01000000 00000000 00000000 00000000 00B60000
*.....*
7F2D8E60 000018 00000000 00000000 00000000 0000C2E0  DE2137C9 6A780000 00000000 00000000
*.....B.....I.....*
7F2D8E80 000038 00000000 0000E3C5  E2E3D3E3 D9D40000
*.....TESTLTRM..SECURITY...."....*
SECURITY
```

```

7F2D8E90 000000          006AC614 0902E4E2 D9E3F0F0 F1400903
*          ..F...USRID01 ...*
7F2D8EA0 000010      40404040 40404040 00000000 00000000 00000000 00000000 00000000
*          .....*
7F2D8EC0 000030      00000000 00000000 00000000 00000000 00000000 00000000
*          .....*
7F2D8EE0 000050      00000000 00000000 00000000 00000000 0000
*          .....USER S*
USER
SGM

7F2D8EFA 000000          0100 0000C9D4
*          ....IM*
7F2D8F00 000006      E2F14040 4040E3D7 C9D7C5F1 4040F9F9 F9F94040 4040C2E0 DE2137C8 E2B80000 *S1      TPIPE1  9999
B....HS...*
7F2D8F20 000026      00000000 00000000 00000BB7 ED480000 00000000 00001000 40640000 00004040
*          .....*
7F2D8F40 000046      40404040 40400000 00000200 00000000 00000000 00004040 40404040 40400000
*          .....*
7F2D8F60 000066      00000000 00000000 00000000 00000000 00000000 00000000 00000000
*          .....*
7F2D8F80 000086 TO 7F2D8FC0 0000C6 SAME AS
ABOVE
7F2D8FE0 0000E6      00000000 00000000 00000000 00000000 0000
*          .....B.....*

```

## YRSP example of the X'6701' log record

After OTMA receives a response for a synchronous callout message, the following X'6701' log record with ID = YRSP is written:

```

INTERNAL TRACE RECORD          ID = YRSP  SEGNO=00  RECNO = 00000831  TIME  12:48:22.305  DATE
2008.234?
DFSERA30 - FORMATTED LOG
PRINT
MCI
PREF

7F2D8E10 000000          01800800 0000E2E8 D5C3D9C5 E2D7A0F0
*          .....SYNCRESP.0*
7F2D8E20 000010      00000010 00000000 00000000 00010000
*          .....STATE      ...."....*
STATE

7F2D8E30 000000          00480040 01000000 00000000 00000000
*          .....*
7F2D8E40 000010      00000000 00B60000 00000000 00000000 00000000 0000C2E0 DE33B714 1FBA0000
*          .....B.....*
7F2D8E60 000030      00000000 00000000 00000000 0000E3C5 E2E3D3E3 D9D40000
*          .....TESTLTRM..SECURITY*
SECURITY

7F2D8E78 000000          006AC614 0902C7D6
*          ..F...G0*
7F2D8E80 000008      C6C9E2C8 C9D50903 40404040 40404040 00000000 00000000 00000000 00000000
*FISHIN..
7F2D8EA0 000028      00000000 00000000 00000000 00000000 00000000 00000000 00000000
*          .....*
7F2D8EC0 000048          SAME AS
ABOVE
7F2D8EE0 000068      0000
SGM....".S....IMS1      SY*          *..USER
USER
SGM

7F2D8EE2 000000          0100 0000C9D4 E2F14040 4040E2E8 D5C3D9C5 E2D7F9F9 F9F94040 4040C2E0 *      ....IMS1
SYNCRESP9999      B.*
7F2D8F00 00001E      DE33B713 C5BA0000 00000000 00000000 00000BB7 EAB00000 00000000 00000000
*          .....E.....*
7F2D8F20 00003E      40E90000 00004040 40404040 40400000 00000200 00000000 00000000 00004040 *
Z....
7F2D8F40 00005E      40404040 40400000 00000000 00000000 00000000 00000000 00000000
*          .....*
7F2D8F60 00007E      00000000 00000000 00000000 00000000 00000000 00000000 00000000
*          .....*
7F2D8F80 00009E      00000000 00000000 00000000 00000000 00000000 00000000 00280000 C9D4E2F1
*          .....IMS1*
7F2D8FA0 0000BE      01000017 00220002 C2E0DE33 9B08FE7C E3D7C9D7 C5F14040 C6D7C5F0 F0F0F140
*          .....B.....@TPIPE1 FPE0001 *
7F2D8FC0 0000DE      00000000 00000000 00000000 00000000 00000000 00000000 00000000
*          .....*
7F2D8FE0 0000FE      0000
SGM....".S....RESPONSE T*          *..APPL
APPL
SGM

7F2D8FE2 000000          001E 0000D9C5 E2D7D6D5 E2C540E3 D640E2E8 D5C3C840 C3C1D3D3 D6E4E340 *      ....RESPONSE TO
SYNCH CALLOUT *

```

## YNAK example of the X'6701' log record

If OTMA receives a negative acknowledgment (NAK) for the synchronous callout message from its client, the following X'6701' log record with ID = YNAK is written:

```
INTERNAL TRACE RECORD          ID = YNAK  SEGN0=00  RECNO = 000008AC  TIME  12:50:52.645  DATE
2008.234?
MCI
PREF

 7F363E28 000000                01204000 0000E3D7  C9D7C5F1 4040A0E0 00000014 00000DAC
* ..TPIPE1 .....*
 7F363E40 000018 00000000 00010000
*.....STATE .....*
STATE

 7F363E48 000000                00482040 01000000  00000000 00000000 00000000 00B60000
* .....*
 7F363E60 000018 00000000 00000000  00000000 00000000 00000000 00000000
*.....*
 7F363E80 000038 00000000 0000E3C5  E2E3D3E3 D9D40000
*.....TESTLTRM..SECURITY....*
SECURITY

DFSERA30 - FORMATTED LOG
PRINT
 7F363E90 000000                006AC614 0902E4E2  D9E3F0F0 F1400903
* ..F...USRID01 ...*
 7F363EA0 000010 40404040 40404040  00000000 00000000 00000000 00000000
* .....*
 7F363EC0 000030 00000000 00000000  00000000 00000000 00000000 00000000
*.....*
 7F363EE0 000050 00000000 00000000  00000000 00000000 00000000 0000
*.....USER S*
USER
SGM

 7F363EFA 000000                0100 0000C9D4
* .....IM*
 7F363F00 000006 E2F14040 4040E3D7  C9D7C5F1 4040F9F9  F9F94040 4040C2E0  DEB11E60 BEB50000 *S1 TPIPE1 9999
B.....*
 7F363F20 000026 00000000 00000000  00000BB7 ED480000  00000000 00001000  40640000 00004040
* .....*
 7F363F40 000046 40404040 40400000  00000200 00000000  00000000 00004040  40404040 40400000
* .....*
 7F363F60 000066 00000000 00000000  00000000 00000000  00000000 00000000  00000000 00000000
* .....*
 7F363F80 000086 TO 7F363FC0 0000C6 SAME AS
ABOVE
 7F363FE0 0000E6 00000000 00000000  00000000 00000000  00000000 00000000  0000
*.....B...C...*
```

### ***Format of the TIB3 X'6701' log record for OTMA and APPC messages in a shared queues environment***

When AOSLOG=Y is specified in the DFSDCxxx member of the PROCLIB data set of a front-end IMS system in a shared queues group, the front-end IMS system writes an X'6701 log record for certain messages that are received from a back-end IMS via a z/OS cross-system coupling facility (XCF). The log records for these messages are identified by a value of TIB3 in the ID= field of the log record for the message.

A front-end system writes an X'6701' log record for OTMA or APPC messages in the following cases:

- Response message returned from the back-end system by XCF for transactions with synchronization levels of NONE, CONFIRM, and SYNCPT.
- Error message returned from the back-end system by XCF for transactions with all synchronization levels of NONE, CONFIRM, and SYNCPT.

### **Example of a formatted X'6701' log record with ID = TIB3**

The following example shows an X'6701' log record with ID = TIB3.

```
INTERNAL TRACE RECORD          ID = TIB3  SEGN0=00  RECNO = 0000077D  TIME  19:44:57.509  DATE 2010.120
MSG PREF
 0C6DF040 000000 027A0000 0040D581  85D50000 F9F9F9F9  40404040 C1D7D6D3  F1F14040 C5E8FA11 *..... NAEN..9999
APOL11 EY...*
DFSERA30 - FORMATTED LOG
PRINT
 0C6DF060 000020 E47F3023 C0000000  0C1AB060 C8E6E2F1  40404040 40404040  40404040 C9D4E2F1 *U".....-
```

```

HWS1          IMS1*
0C6DF080 000040 40404040 C5E8FA11 E4759D23 121000A8 C5E8FA11 E47F3023 00000000 00000000 *
EY..U.....Y.EY..U".....*
0C6DF0A0 000060 00000000 00000000 0C1AB060 00000000 00000000 C9D4E2F1 00000000 00000000
*.....IMS1.....*
0C6DF0C0 000080 00000000 00000000 00000000 C5E8FA11 E45D4C63 00000000 00000000 00000000
*.....EY..U).....*
0C6DF0E0 0000A0 00000000 00000000
*.....*
MCI PREF
0C6DF0E8 000000 01400000 0000F9F9 F9F94040 4040A0F0 0000000A 00000000 * . . . . .9999
..*
0C6DF100 000018 00000000 00010000
*.....*
STATE
0C6DF108 000000 00481020 00000000 00000000 00000000 00000000 00000000
*.....*
0C6DF120 000018 00000000 00000000 00000000 0000C5E8 FA11E452 72630000 00000000 00000000
*.....EY..U.....*
0C6DF140 000038 00000000 00004040 40404040 40400000
*.....*
SECURITY
0C6DF150 000000 006AC614 0902E4E2 D9E3F0F0 F3400903
*.....F...USRT003...*
0C6DF160 000010 E2E8E2F1 40404040 00000000 00000000 00000000 00000000 00000000
*SYS1.....*
0C6DF180 000030 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0C6DF1A0 000050 00000000 00000000 00000000 00000000 00000000 0000
*.....*
USER SGM
0C6DF1BA 000000 0100 0000C9D4
*.....IM*
0C6DF1C0 000006 E2F14040 4040C3D3 C9C5D5E3 F0F1F9F9 F9F94040 4040C5E8 FA0FC71D 20EB0000 *S1 CLIENT019999
EY..G.....*
0C6DF1E0 000026 00000000 00000000 00000C67 FC300000 00000000 00001010 20280000 00004040
*.....*
0C6DF200 000046 40404040 40400000 00000203 00000000 00000000 00004040 40400000
*.....*
0C6DF220 000066 00000000 00000000 00000000 00000000 00000000 00000000 00000000
*.....*
0C6DF240 000086 TO 0C6DF280 0000C6 SAME AS ABOVE
0C6DF2A0 0000E6 00000000 00000000 00000000 00000000 00000000 0000
*.....*
APPL SGM
0C070500 000000 002A0300 E3C8C1E3 E240C1D3 D340C6D6 D3D2E240 D4C5E2E2 C1C7C540 F1406BE2 *...THATS ALL FOLKS
MESSAGE 1 ,S*
0C070520 000020 C5C7D4C5 D5E340F1 404B *EGMENT
1.
UDATA
0C5F8D1C 000000 003E0000
*.....*
0C5F8D20 000004 0C1AB060 0C1AC330 C9D4E2F2 40404040 0000000D 00000000 C9D4E2F1 40404040
*.....C.IMS2.....IMS1*
0C5F8D40 000024 C5E8FA11 E48FC3A3 C9D4E2F1 40404040 C5E8FA11 E48FC3A3 4000 *EY..U.CT.TIMS1
EY..U.CT.
YTIB
0C1AB330 000000 E8E3C9C2 E8E3C9C2 00000000 0C1AB060 *
YTIBYTIB.....-
0C1AB340 000010 0C6D90A0 F9F9F9F9 40404040 0A5398D0 00000000 00000000 E4E2D9E3 F0F0F340
*...9999...Q.....USRT003*
0C1AB360 000030 E2E8E2F1 40404040 00000000 00000000 00000000 00000000 00000000
*SYS1.....*
0C1AB380 000050 00000000 00400100 004A0023 00010100 006A0023 00024840 D5810000 00000000
*.....NA.....*
0C1AB3A0 000070 00000000 C5E8FA11 E47F3023 0C6DF040 0C66F048 C1D7D6D3 F1F14040 0000
*.....EY..U"....0..0.APOL11..*
INTERNAL TRACE RECORD ID = TIB3 SEGN0=01 RECNO = 0000077E TIME 19:44:57.509 DATE 2010.120
CONTINUE
0C1AB3BE 00008E 0000
*.....*
0C1AB3C0 000090 00000000 00000000 00000000 00000000 00000000 00000000 C0000000
*.....*
0C1AB3E0 0000B0 00000000 00000000 00000000 00000000 0000037A 0C070448 0C6DF0E8 0C6DF108
*....._OY..1.*
0C1AB400 0000D0 0C6DF150 0C6DF1BA 7F2BBFEA 00000000 00028000 00000000 00000000 00000000
*..1&..1.".....*
0C1AB420 0000F0 00000000 00000000 000100EF 00000000 C9D4E2F2 40404040 00000000 00000000
*.....IMS2.....*
0C1AB440 000110 40404040 40404040 00000168 20000000 00000000 00000000 C5E8FA11 E4527263
*.....EY..U.....*
0C1AB460 000130 C9D4E2F2 40404040 0000000D 00000000 00000000 00000000 00000000
*IMS2.....*

```

## Printing log records and trace table entries

You can use the File Select and Formatting Print utility (DFSERA10) to print both log records from the IMS log data set and the externalized trace table entries that are recorded in the DFSTRAXx data set.

Formatting of the DFSTRAXx trace entries is similar to formatting trace records that are contained on the IMS log; however, the external trace data set contains only records with an ID of X'67FA'.

Figure 77 on page 511 and Figure 78 on page 512 show examples of unformatted and formatted log records. Unformatted log records include the prefix area record, the subrecord, data, and a table offset in hexadecimal. The formatted record contains the data area with its actual offset address and the table offsets.

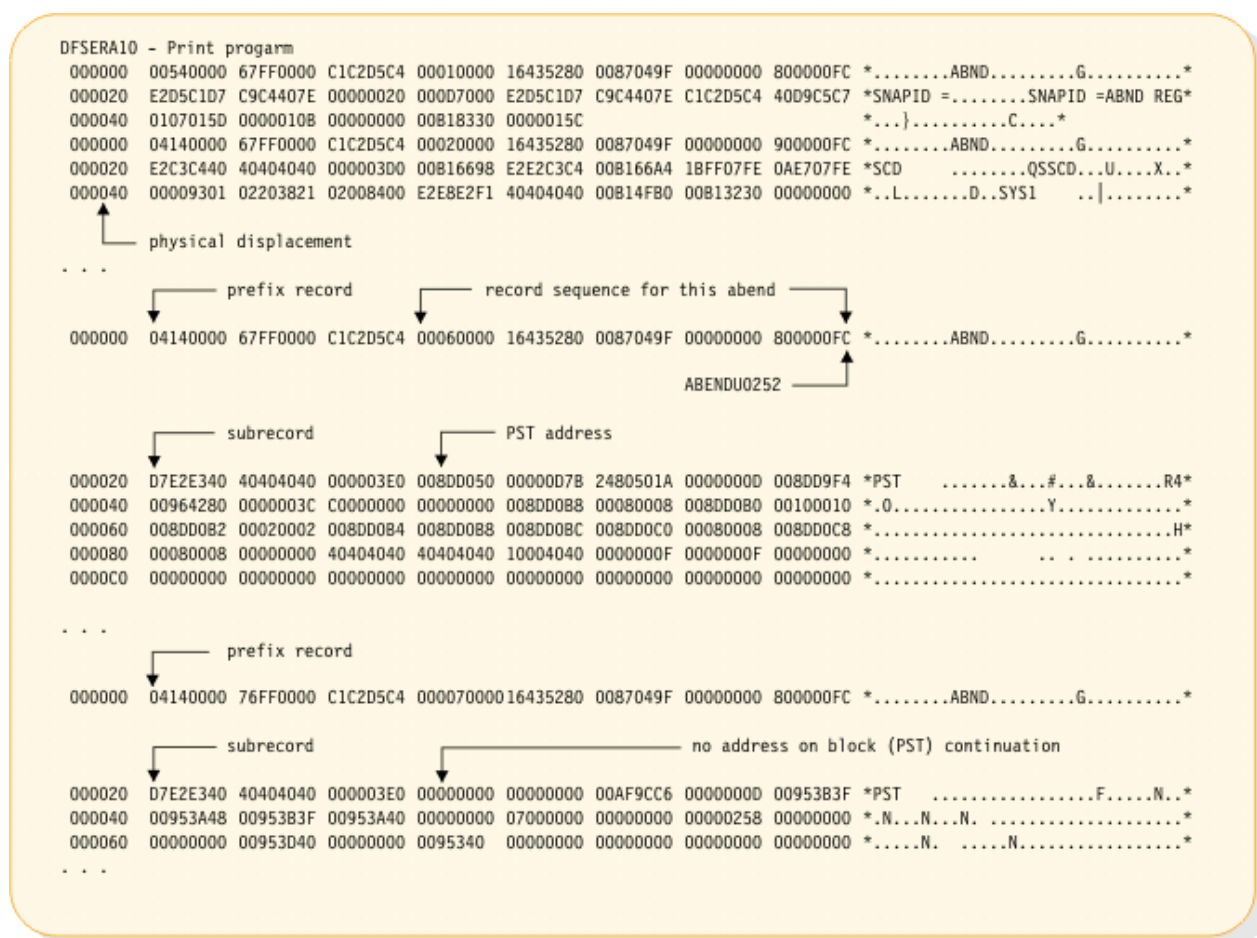


Figure 77. Unformatted output using DFSERA10

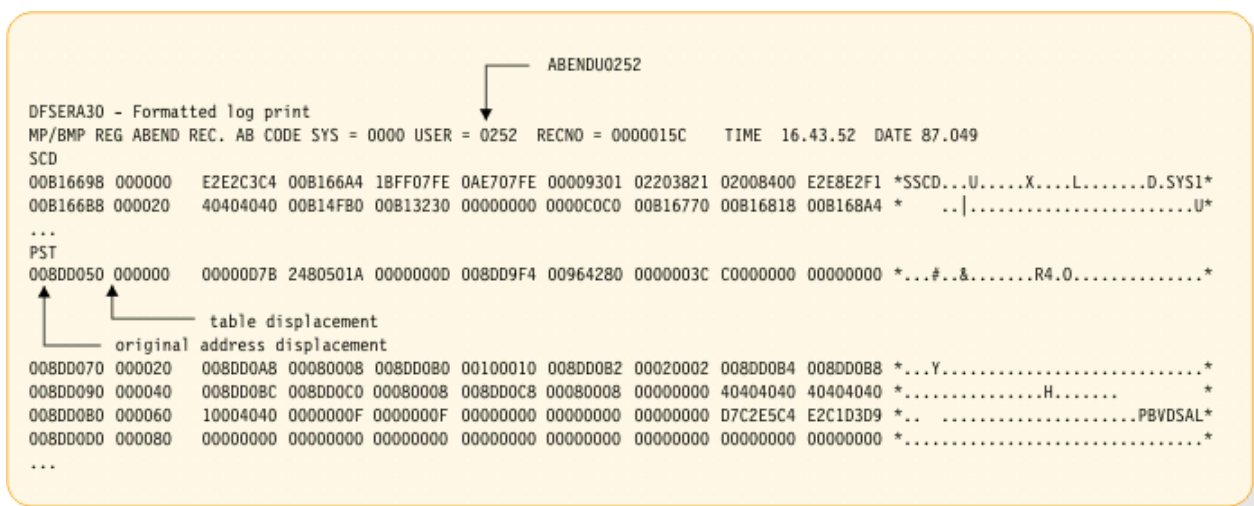


Figure 78. Formatted output using DFSERA10 with option statement, Exit=DFSERA30

### Related concepts

“SNAPs on exceptional conditions” on page 158

IMS produces SNAPs of DL/I control blocks on the IMS log (or the CICS system log).

## Log Merge utility

The Log Merge utility can merge up to nine IMS system logs. Each log is the output of a uniquely identified IMS system that is running during the same time span.

The Log Merge utility (DFSMTMG0) produces one data set that is used as input to the Log Transaction Analysis utility by merging the system log data sets (SLDS) from two or more IMS systems.

## Formatting IMS dumps offline

Two methods are available for formatting IMS dumps offline: interactive formatting, performed through a series of panels which provide formatting choices, and formatting by using JCL.

### Related concepts

“Starting the OTMA trace” on page 365

The **/TRACE SET ON TABLE OTMT** command activates the trace and sends the entries to an internal table.

### Related tasks

“ABENDU1026 analysis” on page 401

To analyze ABENDU1026 failures, you determine the documentation to obtain, and how to find and interpret diagnostic data from the documentation. Gather the necessary data before searching an IBM Software Support database or calling the IBM Software Support.

“Starting the Online Recovery Manager trace” on page 213

The **/TRACE SET ON TABLE ORTT** command activates the trace and sends the entries to an internal table.

“Starting the LU manager trace” on page 340

The **/TRACE SET ON TABLE LUMI** command activates the trace and sends the entries to an internal table.

### Related reference

“Storage manager trace” on page 629

The storage manager trace writes a record each time it is called to allocate a pool, get a buffer, or release a buffer. The storage manager traces requests from the following pools: AOIP, CESS, CIOP, CMDP, DYNP, EMHB, FPWP, HIOP, LUMC, LUMP, and SPAP.

## Overview of the IMS Offline Dump Formatter

The IMS Offline Dump Formatter (ODF) is a formatting option that reduces IMS control region abnormal termination processing.

During abend processing, IMS calls the SDUMP system service of z/OS to create a dump data set. Because SDUMP dumps the requested address spaces without formatting them, the processing time of an abnormal termination is shortened. After abend processing finishes, you can use the IMS Offline Dump Formatter to format (and print) either the complete dump or only those sections needed to analyze the problem.

One advantage of the IMS ODF is that you can make multiple formatting passes at the dump. This means you can first format a summary and then go back one or more times to format the control blocks that will help you most to analyze the problem IMS encountered.

Other advantages of the Offline Dump Formatter include:

- An integrated IMS dump that contains the address spaces of the IMS control region, DBRC, DL/I, and IRLM address spaces.

Also, the formatting modules are included in the dump data set. This ensures that the modules used for formatting the dump match the level of the dumped IMS control blocks. If you specify the REFRESH parameter on the user control statement for IPCS, you will get a new copy of the modules from the program library.

- You can use a z/OS stand-alone dump, SVC dump, or SYSMDUMP to produce the dump data set for the ODF to format.
- After formatting, you can either print the dump or use interactive aids such as IPCS and ISPF browse to view the dump.

Formatting dumps offline is the recommended option. If you want to format dumps online during abnormal termination, you must change the FMTO parameter to request a SNAP dump.

You cannot use the ODF to format z/OS trace and control block areas, the IRLM control blocks, or the VSAM modules.

## Input for the IMS Offline Dump Formatter

The dump data set you use for input to the IMS Offline Dump Formatter must include Key 0 and Key 7 CSA, the CVT, and SQA. CSA is not required for batch or CICS-local DL/I. The dump must be machine readable.

The most common input data sets are taken by SDUMPs, because the IMS control region automatically takes an SDUMP when one of its address spaces fails.

Even if a primary SDUMP request fails, the data dumped to the point of failure can still allow successful dump formatting. Some of this information might not be included in the data sets from a secondary SDUMP request, because on the secondary request only the address space that abends is dumped.

SYSMDUMPs, stand-alone dumps (SADMP), and sumps that are taken by the z/OS DUMP command usually produce acceptable input data sets.

For details of the SDUMP support job stream, refer to *IMS Version 15.5 System Definition*.

### Related tasks

[“ABENDU1026 analysis” on page 401](#)

To analyze ABENDU1026 failures, you determine the documentation to obtain, and how to find and interpret diagnostic data from the documentation. Gather the necessary data before searching an IBM Software Support database or calling the IBM Software Support.

## **Invoking the IMS Offline Dump Formatter**

To use the IMS Offline Dump Formatter, you must have an acceptable dump in a data set, a proper IMSDUMP entry in the IPCS exit control table and the IMS execution library with the dump formatting modules needed to be allocated to IPCS with the ddname ISPLLIB.

### **About this task**

You then invoke the IMS Offline Dump Formatter by executing a VERBX control statement from IPCS, or through the interactive panels.

## **Solving IMS problems by using the IMS Offline Dump Formatter**

You can learn about how to approach IMS problems using IPCS and the IMS Offline Dump Formatter and how to choose FMTIMS parameters to format a system dump.

### **Approaching the problem**

The recommended diagnostic approach with the IMS Offline Dump Formatter is:

1. Use IEBGENER or IPCS COPYDMP to transfer the dump from the SYS1.DUMPxx data set to your own data set.
2. Get an overview of the problem by formatting the dump with the subset option SUMMARY.
3. Use the abend code or reason for abnormal termination, the CALLER=id, and the TCB=id from the dump title to determine the needed subset options.
4. Format the dump again with the subset options you determined in the previous step. Use the MIN qualifier (where possible) to reduce the output size. You can always format the data again if you need more information.

You might also need to format the z/OS trace and control block areas, the IRLM control blocks, or the VSAM modules. These blocks cannot be formatted with the IMS Offline Dump Formatter.

5. The formatted output is spooled. You can either print the output or use ISPF to browse it.
6. Do additional IMS subset formatting on following jobs if necessary.
7. If you still cannot locate or fix the problem, keep the dump data set because you need it when discussing the problem with an IBM Software Support representative.

## **Using IPCS and the IMS offline dump formatter**

You can interactively format IMS dumps offline by using a series of panels that provide formatting choices.

### **Method 1**

Run the IMS Offline Dump Formatter as an IPCS verb exit to format and print the dump. You can then use IPCS to view unformatted dump storage referenced in your printed dump.

### **Method 2**

Format, but do not print the dump. Invoke split screen mode on your terminal. On one half of the screen, use ISPF browse to view the formatted control blocks. On the other half, use IPCS to view any unformatted storage referenced in the formatted control blocks.

## **Choosing FMTIMS parameters**



Identify the general problem area before you attempt to choose FMTIMS parameters. If you are unsure of the problem area, format the dump with the SUMMARY option.

The following table shows the FMTIMS parameters that are recommended for general types of problems. For example, if you suspect the problem is with your logger, specify the DISPATCH, LOG, and SYSTEM parameters on the FMTIMS statement.

To use the following table, locate your problem area in the top row. Then read that column to find the suggested formatting options (marked with an X) for that problem.

Table 152. FMTIMS parameters for general problems

Parameters	Problem area							
	Checkpoint/ Restart	DB	DC	FP	Log	System/other	Batch	CICS
CBT		X	X			X	X	X
CBTE			X					
DB		X					X	X
DBRC		X				X	X	X
DC			X				2	
DEDB		X		X				
DISPATCH	X	X	X	X	X	X	3	
EMH		X	X	X				
LOG					X		X	
MSDB		X		X				
QM			X				2	
RESTART	X						2	
SAP			X					
SAVEAREA <sup>1</sup>	X	X	X	X	X	X	2	
SB		X				X	X	X
SCD <sup>1</sup>	X	X	X	X	X	X	X	X
SPST	X			X			2	
SUBS						X	2	
SUMMARY <sup>1</sup>	X	X	X	X	X	X	X	X
UTIL			X	X			2	

Table 152. FMTIMS parameters for general problems (continued)

Parameters	Problem area							
	Checkpoint/ Restart	DB	DC	FP	Log	System/other	Batch	CICS

**Notes:**

1. You can use the single parameter (SYSTEM) to get the three areas (SAVEAREA, SCD, SUMMARY).
2. This parameter is ignored for batch.
3. (DISPATCH, MIN) is ignored for batch.

See [“Solving IMS problems by using the IMS Offline Dump Formatter”](#) on page 514 for a list of the modules formatted with each of the parameters. See [“Syntax restrictions on the FMTIMS statement”](#) on page 530 to understand the syntax rules for FMTIMS statements.

## Using the dump title to choose FMTIMS parameters

When you are deciding which areas to format for your problem, you can use the CALLER and TCB fields of the dump title (described in [“Dump title”](#) on page 541) as a guide. Unless one or both of these fields specify "unknown," they should indicate why a dump was taken.

The following table shows the options you can choose based on valid CALLER and TCB information in the dump title.

Table 153. FMTIMS parameters based on CALLER and TCB fields

CALLER=	TCB=	Recommended FMTIMS options <sup>1</sup>
CTL	CTL LOG ESS LSD LSM RDS RST STC STM	DC <sup>2</sup> , Dispatch <sup>2</sup> , QM <sup>2</sup> , Summary, System <sup>2</sup> Dispatch <sup>2</sup> , SPST, System <sup>2</sup> , SUBS, Summary Dispatch, Log, Restart, Summary, System Dispatch <sup>2</sup> , MSDB, Savearea, SCD <sup>2</sup> , Summary Dispatch <sup>2</sup> , MSDB, Savearea, SCD <sup>2</sup> , Summary Restart, Savearea, SCD <sup>2</sup> , Summary Restart, Savearea, SCD <sup>2</sup> , Summary CBT, Dispatch <sup>2</sup> , Savearea, SCD <sup>2</sup> , Summary CBT, Dispatch <sup>2</sup> , Savearea, SCD <sup>2</sup> , Summary
CURR <sup>3</sup>	DYA	Dispatch <sup>2</sup> , System <sup>2</sup>
DBRC	DBR	DBRC <sup>2</sup> , System <sup>2</sup>
DL/I	DLI STC	DB <sup>2</sup> , Dispatch <sup>2</sup> , SB <sup>2</sup> , System <sup>2</sup> CBT, Dispatch <sup>2</sup> , Savearea, SCD <sup>2</sup> , Summary
DP	BMP DEP	DB <sup>2</sup> , System <sup>2</sup> DB <sup>2</sup> , System <sup>2</sup>
FP	BMP DEP <sup>4</sup> XFP	DB <sup>2</sup> , DEDB, MSDB, System <sup>2</sup> DB <sup>2</sup> , DEDB, MSDB, System <sup>2</sup> DB <sup>2</sup> , SPST, System <sup>2</sup>
LOG	LOG	Log <sup>2</sup> , System <sup>2</sup>

**Notes:**

1. When you have a WAIT or LOOP problem, add SAVEAREA to your list of FMTIMS options.
2. Use the MIN qualifier for these options.
3. Normally dynamic allocation.
4. Can be either the MPP or the BMP region.

If CALLER=CURR, the current address space and IMS control region are dumped. This occurs when no CALLER parameter is provided or no IMSDUMP parameter list is passed and DFSFDMP0 cannot match the caller's TCB address and address space ID (ASID) with the TCBs in the IMS TCB table. You can still format the dump data set using the abend number and PSW as a guide in solving the problem. Dynamic allocation also causes CURR to be placed in the CALLER= field. In this case, format the areas listed in the above table.

If CALLER=DP, the abend occurred under the task of a dependent region address space.

If CALLER=IRLM, you need to use the IRLM Offline Dump Formatter to format the IRLM modules.

If CALLER=TRAP, a diagnostic trap for an address space abended.

## Offline Dump Formatter parameters

The Offline Dump Formatter provides the option of choosing an 80-column output format in addition to the default value of 120/132 columns. This option allows viewing of formatter output on an 80-column-width screen without needing to shift left or right.

The 80-column format mode is normally selected when the IMS Offline Dump Formatter is run under IPCS and the IPCS default is set to TERMINAL NOPRINT or TERMINAL PRINT. This allows dump and z/OS formatting to be similar under IPCS. To select the 80-column format mode, add an "H" to the IMSDUMP formatter verb parameter string between the IMS job name and the FMTIMS keyword. The following are examples of 80-column format option requests under IPCS.

```
VERBX IMSDUMP 'imsname,R,H,D'
```

```
VERBX IMSDUMP 'imsname,H,FMTIMS SCD'
```

```
VERBX IMSDUMP 'imsname,D,H,R,FMTIMS (AUTO,MIN)'
```

## Sample FMTIMS statements

You might be able to identify a problem area more precisely by using the CALLER= and TCB= identification from the dump title along with the abend number and explanation. For example, you might see CALLER=CTL in the dump title and have an abend code that shows an error in the checkpoint restart processing. In this case, you can try specifying the statement:

```
FMTIMS (RESTART,SAVEAREA,(SCD,MIN),SUMMARY)
```

Following is a list of possible subsets you could format for specific error situations. This list is not exhaustive and is not meant to represent every possible error situation.

## IMS control region problems (CALLER=CTL)

An IMS control region address space task abended. A common definition is SYS—System Services.

### **SYS/CHKPT**

System Service Checkpoint Restart Processing

```
FMTIMS (SUMMARY,SAVEAREA,(SCD,MIN),RESTART)
```

### **SYS/CNTRL**

System Service Control

```
FMTIMS (SUMMARY,SAVEAREA,(SCD,MIN),(DISPA,MIN))
```

### **SYS/ESS**

System Service External Subsystem Support

```
FMTIMS ((SYSTEM,MIN),SPST,(DISPA,MIN),SUBS)
```

**SYS/INIT**

System Service Initialization

FMTIMS (SUMMARY,SAVEAREA,(SCD,MIN))

**SYS/QMGR**

System Service Message Queue Management

FMTIMS (SUMMARY,SAVEAREA,(SCD,MIN),(DISPA,MIN),QM)

**SYS/SCHD**

System Service Scheduling

FMTIMS ((SYSTEM,MIN),SPST,(DISPA,MIN))

**SYS/SMGR**

System Service Storage Management

FMTIMS ((SYSTEM,MIN),SPST,CBT)

**DBRC problems (CALLER=DBRC)**

A DBRC address space task abended. You would use the same FMTIMS statement for all of the following problems with Database Recovery Control:

**DBRC/CMD**

Database Recovery Control Command Processing

**DBRC/CNTRL**

Database Recovery Control Processor

**DBRC/EXIT**

Database Recovery Control Exit Processing

**DBRC/SER**

Database Recovery Control Services

FMTIMS ((SYSTEM,MIN),(DBRC,MIN))

**Data communication problems (CALLER=CTL)**

An IMS data communication task abended under the CTL TCB.

**DC/CMD**

Data Communication Command Processing

FMTIMS ((SYSTEM,MIN),DC)

**DC/CNTRL**

Data Communication Control

FMTIMS ((SYSTEM,MIN),(DC,MIN),(DISPA,MIN),(QM,MIN))

**DC/CONV**

Data Communication Conversational Processing

FMTIMS ((SYSTEM,MIN),(DC,MIN))

**DC/LMGR**

Data Communication Line Manager

FMTIMS ((SYSTEM,MIN),(DC,MIN))

**DC/MFS**

Data Communication Message Format Services

FMTIMS ((SYSTEM,MIN),(DC,MIN))

**DC/TPCALL**

Data Communication DL/I Telecommunications

Call Processing

FMTIMS ((SYSTEM,MIN), (DC,MIN), (DB,MIN))

**DL/I problems (CALLER=DL/I or CALLER=DP)**

A DL/I address space task abended.

**DB/ACSMTH**

Database Access Method Interface

FMTIMS ((SYSTEM,MIN), (DB,MIN))

**DB/ANAL**

Database Call Analyzer

FMTIMS ((SYSTEM,MIN), (DB,MIN))

**DB/CMGR**

Database Call Resource Management

FMTIMS ((SYST,MIN), (DB,MIN), (DISPA,MIN), (SB,MIN))

**DB/DBCALL**

Database Call Action Processing

FMTIMS ((SYSTEM,MIN), (DB,MIN))

**DB/INTRF**

Database Application/Scheduling Interface

FMTIMS ((SYSTEM,MIN), (DB,MIN), (DISPATCH,MIN))

**Fast Path problems (CALLER=FP)**

A Fast Path task abended.

**FP/CNTRL**

Fast Path Control

FMTIMS ((SYSTEM,MIN), (DB,MIN), SPST)

**FP/DEDB**

Fast Path Data Entry Database Processing

FMTIMS ((SYSTEM,MIN), (DB,MIN), (DEDB,MIN))

**FP/EMH**

Fast Path Expedited Message Handling Call Analyzer

FMTIMS ((SYSTEM,MIN), (DB,MIN), (EMH,MIN))

**FP/MSDB**

Fast Path Main Storage Database Call Analyzer

FMTIMS ((SYSTEM,MIN), (DB,MIN), (MSDB,MIN))

**Log problems (CALLER=LOG)**

An IMS control region address space log TCB task abended. Log is part of SYS—System Services.

**SYS/LOG**

System Service Logging

FMTIMS ((SYSTEM,MIN), (LOG,MIN))

## Other problems

If you suspect that the failure was in VSAM, you do not need to run AMBLIST to secure a listing of VSAM modules IDA019L1 and IDA0192A of the failing system. Data Facility Products (DFP) formats the entry points for these modules. IMS includes LPA modules in offline dump data sets only if LPALIB is listed in the SDUMP options for your system. However, this is not recommended because the LPA modules occupy so much space in the dump data sets.

*z/OS MVS Diagnosis: Tools and Service Aids* describes how to generate a z/OS trace.

## Dump contents returned for different FMTIMS options

You can specify different FMTIMS options to add different types of information to a formatted IMS system dump.

The options are listed in alphabetical order. FMTIMS options can be specified on the FMTIMS statement in any order. The requested options are printed in the order stated under [“Formatted dump output order”](#) on page 545.

Some options state they "are ignored for batch." If the dump was taken because batch processing (IMS DB or CICS) failed, the control blocks for these options are either meaningless or not included in the dump data set. Therefore, the control blocks are not formatted even if you specify that option on the FMTIMS statement.

Most options can be specified with the MIN qualifier. Whenever possible, specify this qualifier to reduce the number of control blocks formatted. You can always format the dump data set again if you need the additional information.

### ALL

Causes a full, formatted dump.

(ALL,MIN) formats the dump as if each option were specified with the MIN qualifier.

### AOI

Formats the storage for the Type 2 Automated Operator Control blocks.

### AUTO

Provides an optimal subset of the IMS dump formatting options without having to first analyze the dump and without having to understand the content or use of all of the IMS dump formatting options.

This option uses the failing ITASK type information to choose one of the formatter's functional areas, and selects the appropriate dump formatter options.

### CBT

Formats storage management area control blocks, including:

- Control Block Table Header
- Individual Control Block Table entries

Output is the same if (CBT,MIN) is specified.

### CBTE,cbteid

Formats all the IPAGEs for the identified CBTE type (cbteid), including:

- Individual Control Block Table entries
- All IPAGE storage of the requested CBTE type

For example, if you specify (CBTE,DPST), all DPST IPAGEs are formatted.

This option can be repeated as needed and has no defaults. The requested IPAGEs must be part of the dump data set. MIN is not valid for the CBTE option.

### CLB/LLB

Permits formatting of an individual Communication Line Block or Link Line Block and its subordinate blocks. Select this option by the following:

- Address
- Node name
- LTERM name
- Communication ID

Select the LLB by address or link number.

The CLB/LLB format creates eye catchers and index entries similar to the following:

**CLB/LLB	REQUESTED CLB/LLB
-----------	-------------------

## DB

Formats areas and control blocks used for IMS Database functions. The following table shows the areas formatted under the (DB) and (DB,MIN) FMTIMS options.

<i>Table 154. Formatted areas under the FMTIMS options DB and DB,MIN</i>	
(DB)	(DB,MIN)
PSB Directory	same
DMB Directory	same
Intent List	not formatted
BFSP	same
DL/I Trace	same
Fast Path Trace (if Fast Path is active)	same
OSAM Pool Control Blocks and buffers	OSAM Pool Control Blocks only
Program Isolation blocks	same
All PSTs and related control blocks, including PCBs, SDBs, Savearea set, alternate DL/I DECB, DSGLRKEY, hierarchical holder, delete work area, RPLI, VSAM PLH, and retrieve trace	Active PSTs, with the same related control blocks
If Fast Path is present: EPSTs and related control blocks, including EPCBs, ESRTs, EMHBs, message buffers, XCRBs, DMHRs, and DEDB buffers	If Fast Path is present: EPSTs and related control blocks, including EPCBs, ESRTs, EMHBs, XCRBs, and DMHRs
VSAM buffer pool control blocks	same
RLPL for IRLM requests	same

In a DL/I–SAS environment, DPST formatting does not format related control blocks if the DL/I address space was not included in the dump data set.

## DBRC

Formats records used by DBRC in its processing, including:

- DFSRCWKB block
- DFSBRLSB block
- Dump Router storage
- Global Data block
- GDBDLTAR block
- GDBDSAAR block
- GDBRECAR block
- GDBLISAR block

- DSPEXIAG block
- DSPEXOPM block
- VFYWSPAC block
- DSPOCPAG block
- DSPJCLAR block
- GDBGPDAR block
- GDBRUPAR block
- GDBOLCAR block
- GDBMNPTR block
- GDBESAVE block
- GDBISAVE block
- GDBCSAVE block
- GDBRSAVE block
- DSPCMPAG block
- DSPVFILE block
- DBRC Internal Trace

Output is the same if (DBRC,MIN) is specified. DBRC blocks must be present in the dump data set to be formatted.

## DC

Formats the data communication areas listed in the following table. This option is skipped if the CTL address space is not included in the dump data set.

*Table 155. Data communication areas formatted by DC and DC,MIN*

(DC)	(DC,MIN) <sup>1</sup>
All CLBs, LXBs, and LCBs, with subordinate control blocks: <ul style="list-style-type: none"> <li>• Current CTB or LTB, and CNT</li> <li>• Allocated I/O buffers</li> <li>• CIB, if using MFS processing</li> <li>• CCB, if using conversational processing</li> <li>• MFS work buffers</li> <li>• ECNT, EMHB, and message buffer, if the CTB shows a Fast Path terminal</li> </ul>	Active CLBs, LXBs, and LCBs, with the same subordinate control blocks except that current CTB or LTB and CNT are not formatted.
SMB table	not formatted
CTT table	not formatted
SPQBs and the CNTs chained off unallocated SPQBs	not formatted



Table 155. Data communication areas formatted by DC and DC,MIN (continued)

(DC)	(DC,MIN) <sup>1</sup>
------	-----------------------

**Note:**

1. (DC,MIN) formats control blocks only for those lines, nodes, and links that meet at least one of the following criteria:
  - a. MSC links
  - b. Nodes in OPNDST or CLSDST processing
  - c. Lines or nodes with allocated input, output, or receive any buffers
  - d. CLBs that have an active SAP

Both DC options are ignored for batch.

**DEDB**

Formats the DEDB control blocks and areas. The areas included are listed in the following table.

Table 156. DEDB control block areas formatted by DEDB and DEDB,MIN

(DEDB)	(DEDB,MIN)
ALDS	same
DMCBs, SGTs, FDTs, and MRMBs for open DEDBs	same
DMACs and ADSC for open DEDB areas	same
XCRBs, DMHRs, and buffers	XCRBs and DMHRs only
SRBs and ESRBs	same

**DISPATCH**

Formats areas relating to the IMS Dispatcher and its functions. The following table shows the areas formatted under this FMTIMS option.

Table 157. Areas formatted by DISPATCH and DISPATCH,MIN

(DISPATCH)	(DISPATCH,MIN)
Dispatcher work areas	not formatted
Dispatcher Trace	same
Scheduler Trace	not formatted
Latch Trace	same

(DISPATCH,MIN) is ignored for batch.

**DPST,jobname**

**DPST,N,dependent region number**

**DPST,A,address**

Permits formatting of an individual Dependent Region Partition Specification Table and its subordinate blocks for PSTs related to MPPs, BMPs, IFPs, and batch DL/I. You can specify one of the following choices:

- job name
- Dependent region number
- DPST address

Output follows the DB formatting output in the dump formatter. The eye catchers and index entries appear as follows:

## EMH

Formats the Expedited Message Handler areas that are used by IMS Fast Path, as shown in the following table. The CTL address space must be included in the dump data set for this option to be formatted.

*Table 158. Areas formatted by EMH and EMH,MIN*

EMH	EMH,MIN
RCTEs	same
BALGs, EMHBs, and message buffers	BALGs and EMHBs only

## LOG

Formats control blocks and areas used by the IMS logger. The areas included shown in the following table. These areas, except for the WADS and the DLOG trace, are repeated in the dump when the IMS Monitor is active.

*Table 159. Areas formatted by LOG and LOG,MIN*

LOG	LOG,MIN
LCD	same
Restart Log Work Area	same
WADS and the data necessary to manage it	WADS only
OLDS prefix and the buffer associated with it	OLDS prefix only
Log DSET, which defines all OLDS currently available for use	same
Message work areas and Logger message areas	same
DLOG trace	same

## MSDB

Formats the Main Storage Databases used by IMS Fast Path. The areas included are listed in the following table.

*Table 160. Main storage databases formatted by MSDB and MSDB,MIN*

MSDB	MSDB,MIN
MSDB headers	same
all MSDBs	not formatted

## POOL, NAME, poolid

Invokes formatting of the storage manager control blocks and the pool storage for any of the following pools:

- ALL
- CESS
- CIOP
- DBWP
- DLDP
- DLMP
- DPSB

- EMHB
- EPCB
- FWP
- HIOP
- MFBP
- PSBW
- QBFL
- QBUF
- SPAP
- LUMC
- LUMP

NAME is an optional keyword indicating the pool name parameter. If NAME is omitted, the first parameter is assumed to be the pool name.

The poolid is a required 4-character pool name of an existing storage manager pool or the keyword ALL. If ALL is specified, the following storage pools are formatted:

- HIOP
- CIOP
- CESS
- SPAP
- EMHB
- FWP
- QBUF
- QBFL
- DLMP
- DPSB
- DBWP
- MFBP
- EPCB
- LUMP
- LUMC

ALL triggers the formatting of any storage manager trace table entries along with the storage manager control blocks and pool storage.

MIN is an optional keyword. If MIN is specified for one of the dynamic pools (HIOP, CIOP, EMHB, FWP, CESS, SPAP, LUMC, LUMP) only the storage manager pool header and block headers are formatted. If MIN is omitted, the pool header control block is formatted along with the blocks and block headers representing the dynamic storage pool.

## QM

Formats the control blocks and areas of the IMS queue manager. The formatter skips this option if the CTL address space is not included in the dump data set. The areas included are shown in the following table.

*Table 161. Areas formatted by QM and QM,MIN*

QM	QM,MIN
Qpool Prefix	same
Qpool Buffer Prefix	same

Table 161. Areas formatted by QM and QM,MIN (continued)

QM	QM,MIN
Qpool Buffer	not formatted

Both QM options are ignored for batch.

## RESTART

Formats the IMS restart control blocks and related areas, including:

- Checkpoint ID table
- SIDXs and their subordinate blocks:
  - All LCREs for the SIDX entry being processed
  - All RREs for the SIDX entry being processed
- All RPSTs for the SIDX entry being processed
- FRB, if present

Output is the same if (RESTART,MIN) is specified. Both RESTART options are ignored for batch.

## SAP, ECBADR, ecbaddr

### SAP, ADDRESS, sapaddr

The SAP option can be invoked using either the SAP address or the SAP's ECB address (providing that the ECB is a valid ITASK and has a prefix pointing to a SAP). The SAP option request can be placed either on the IMSDUMP verb line after FMTIMS or in the DFSFRMAT data set. The following examples show SAP option requests:

```
VERBX IMSDUMP 'imsjname,II,N,FMTIMS (SAP,ADDRESS,20864C0) '
```

```
VERBX IMSDUMP 'imsjname,FMTIMS SCD,(SAP,ECBADR,3064250) '
```

For compatibility reasons, the MIN qualifier is allowed, but the output is the same. Individual SAP option formatting is also available on the IMS Low Level panel of the IMS IMS Dump Formatter dialog. The ADDRESS parameter can be omitted because ADDRESS is the default TYPE for the SAP option.

Individual SAP/save area formatting allows complete formatting of SAP/save areas when additional information is required. The output from individual SAP formatting is the same as the SAVEAREA option output. Individual SAP formatting provides the following eye catcher/index entry:

```
**SAPS      REQUESTED  SAPS
```

## SAVEAREA

Formats the save area information, including:

- Formatted SAPs and any UEHBs anchored off the SAPs.

**Restriction:** The UEHBs cannot be formatted if the CTL address space is not included in the dump data set.

- Formatted Save Area Sets associated with each SAP.
- Unformatted dump of the IPAGEs containing the SAPs.

If the DL/I address space is not in the data set, then the DL/I SAPs are not formatted. If the CTL address space is not in the data set, then the non-DL/I SAPs are not formatted. Output is the same if (SAVEAREA,MIN) is specified. Both SAVEAREA options are ignored for batch.

The SAVEAREA also comes with a summary option that allows a faster overview scan of the IMS ITASK status within a dump. The SAVEAREA SUMmary output reduces the SAP/Savearea formatting to minimal data while adding keyword scan capability and automatic computation of the exit offsets.

This reduces keystroke resources required to overview the ITASK status and ITASK module flow. The SAVEAREA SUMmary and individual SAP formatting provides the following eye catcher/index entry:

**SSS	SAP/SAVE CONDENSED SUMMARY
-------	----------------------------

SAVEAREA SUMmary formatting contains the following scannable keywords with their associated meanings:

#### **RUN**

ITASKs that are active are given a RUN indicator. Abend and loop analysis is usually concerned only with running ITASKs.

#### **LATCHREQ**

ITASKs that are waiting for an IMS SLX latch (not checkpoint restart LATE latches) are given a LATCHREQ indicator. Enabled wait problem analysis often requires analyzing ITASKs that are waiting for latches.

#### **LATCHOWN**

ITASKs that own an IMS SLX latch (not checkpoint restart LATE latches) are given a LATCHOWN indicator. Enabled wait problem analysis often requires analyzing ITASKs that own SLX latches.

#### **ITASK type**

The ITASK type is in the summary and is scannable. The ITASK type names are not at the end of the scan list, however. The ITASK type is preceded by the label "type". The possible type names can be gotten from the DFSCIR macro prolog.

### **SB**

Formats the control blocks, areas, and buffers of the Sequential Buffering function (SB) of IMS. This option also formats those DL/I control blocks which are important for debugging the SB function.

The SB information is divided into four sections. The following table shows which sections are formatted with the SB and SB,MIN options. .

<i>Table 162. Sections formatted by SB and SB,MIN</i>	
<b>SB</b>	<b>SB,MIN</b>
Subsystem overview	same
PST overview <sup>1</sup>	same <sup>2</sup>
Sorted blocks <sup>1</sup>	same <sup>2</sup>
Sorted buffers <sup>1</sup>	not formatted

#### **Note:**

1. The DL/I address space must be included in the dump data set for these areas to be formatted.
2. Formatted only if you requested a conditional SB activation for that application or PST.

The SB information is divided into the following sections:

1. Subsystem Overview of SB—provides an overview of SB control blocks from an IMS subsystem point-of-view. The SDCBs appear in the order in which they are anchored in the SBSCD. Each SDCB is followed by its SDSGs. The section contains the following information:
  - SB section of the SCD
  - SBSCD, including the SBHE blocks
  - SDCBs
  - SDSGs
2. PST Overview of SB—formats the SB control blocks (and other IMS control blocks significant to SB) for each active PST. These blocks are sorted in hierarchical order. For example, the first DBPCB and

its JCB, DSGs, EDSGs, and SDSGs; then the second DBPCB with its subordinate blocks, and so on. The section contains the following information:

- SB and buffer-handler sections of the PST
  - PST DECB prefix
  - SB extensions to the PST
  - SB work area
  - SBPARMS
  - DBPCBs and their JCBs, DSGs, EDSGs, and SDSGs
3. Sorted SB Blocks—contains SB control blocks (and other IMS control blocks significant to SB) sorted according to their virtual storage address. The section contains the following information:
- DBPCBs
  - DCB with its OSAM extensions
  - DSGs
  - EDSGs
  - JCBs
  - OV-IO DECB prefix
  - PST DECB prefix
  - SB extensions to DCBs
  - SB extensions to DSGs
  - SB extensions to the PST
  - SB work area
  - SBPARMS
  - SBUFs
  - SCARs
  - SRANs
4. Sorted SB Buffers—contains the SB buffers of each SB buffer pool. The SB buffers of one SB buffer pool are contiguous in storage and are formatted as one entity. The buffer pools are then sorted by virtual storage address.

## SCD

Formats the IMS SCD and related areas. The areas included are listed in the following table.

*Table 163. Areas formatted by SCD and SCD,MIN*

<b>(SCD)</b>	<b>(SCD,MIN)</b>
SCD	same
Latch Extensions	same
Scheduler Sequence Queues	not formatted
Synchronous APPC/OTMA Shared Message Queue SCD Extension	same
Fast Path SCD Extension, if Fast Path is active	same
Formatted dumps of the batch key 7 SCD	same
LU 6.2 SCD extension	same

## SPST

Formats the system PSTs, which are ITASKs used by IMS. This includes:

- Global system PSTs
- Local control region address space PSTs
- Local DL/I address space PSTs
- Areas related to the above PSTs, including LWA and IRLMA

Some SPSTs are not formatted if the CTL address space is not in the dump data set. Output is the same if (SPST,MIN) is specified. Both SPST options are ignored for batch.

## SUBS

Formats the areas and control blocks that IMS uses to manage subsystems, including:

- Subsystem trace
- Global ESET block

Output is the same if (SUBS,MIN) is specified. Both SUBS options are ignored for batch.

## SUMMARY

Formats the current diagnostic section.

The SUMMARY data areas are not formatted if the SDWA address space is not part of the dump data set. (For abends and batch processing, the SDWA address is saved by the ESTAE module. For online processing, the dump must be taken by DFSOFMD0, and the SDWA parameter must be passed at DFSDUMP time.)

The areas formatted with this option include:

- Failing PSW
- Abend code
- Module name
- Registers at time of abend
- 256 byte instruction area—128 bytes above and below the failing PSW
- 16 register storage areas—512 bytes above and 256 bytes below the registers at time of abend
- SDWA address space of IMS
- Failing SAP and its UEHB
- Failing ITASK when the ITASK is a DPST, system PST, CLB, or LLB (dependent region errors, some systems services errors, terminal process errors, and MSC errors)

The SUMMARY option names the ITASK type when it is determined, even if it is not one of the ITASK types that provide for additional formatting. The ITASK type name is two to four characters. If it is unknown, the type name is "UNKN".

Output is the same if (SUMMARY,MIN) is specified.

## SYSPST

Permits formatting of an individual system partition specification table and some of its subordinate blocks. Select this option by address or system PST name. This option creates eye catchers and index entries similar to the following:

**SYSPSTS	REQUESTED SYSTEM PSTS
-----------	-----------------------

## SYSTEM

Formats the SUMMARY, SAVEAREA, and SCD areas as one group. The areas and control blocks formatted are the same as if each of the options were invoked separately.

(SYSTEM,MIN) is formatted as though each of the options were specified with MIN.

See the individual options for a list of the areas formatted.

## TRACE, NAME, table-id

Gets a new search module that invokes the normal trace format control module (DFSATRA0) to format trace tables separately. This option enables viewing of trace table data without having to format the

entire option that usually includes the formatted trace table. The TRACE option request uses the 2-character trace table EBCDIC ID code from the Trace Selection panel. The IMS Dump Formatter ISPF panels also accept an option of "ALL" to format all IMS trace table traces. The IMS Dump Formatter dialog TRACE SELECTION panel provides a selectable list of IMS trace tables with the trace name, internal ID, and description. The following examples are TRACE format requests, followed by comments for each. In each case, the NAME keyword can be omitted because NAME is the default TYPE parameter. The following example is a request for the DL/I trace table.

```
FMTIMS... (TRACE,NAME,DL) , ...
```

The following is a request for the dispatcher trace table and the DL/I trace table with a MIN option that is ignored.

```
FMTIMS... , (TRACE,NAME,DL,MIN) , (TRACE,NAME,DS) ...
```

## UTIL

Formats the control blocks for the IMS Partial Database Reorganization utility, including:

- Common area
- Database table
- Segment table
- Action table

The output is the same if (UTIL,MIN) is specified. Both UTIL options are ignored for batch.

## Related concepts

[“Dump formatting options” on page 551](#)

General formatting options are available for IMS and z/OS memory dumps.

## Invoking the IMS Offline Dump Formatter under IPCS

You can invoke the Offline Dump Formatter under IPCS by using a VERBX command or by using menus.

## About this task

*Using a VERBX command*

## Procedure

Enter FMTIMS and the valid IMS format options after the job name and any refresh, debug, half line, and nonheader options. The following is an example.

```
VERBX IMSDUMP, 'imsname,D,H,R,FMTIMS (SAP,ADDRESS,1234580) '
```

## Syntax restrictions on the FMTIMS statement

The format control data set requires adherence to ten syntax rules, but the sequence in which you apply the rules does not matter.

The control statements in the format control data set must follow the following syntax rules:

- The first record must contain "FMTIMS".
- A comma (,) must separate parameters from their qualifiers (MIN or cbteid).
- The number of leading blanks on both the initial record and on subsequent records is not limited.
- The last 8 bytes of all records are ignored by the formatter; you can use them for sequence numbers or any other purpose.
- A comma after the last parameter on any record indicates continuation to the next record. You can split a parameter and its qualifier, but you cannot split the spelling of a parameter over two records. For example:



```
FMTIMS ((SYSTEM,MIN), (LOG,
MIN))
```

is acceptable, but the following is not:

```
FMTIMS ((SYS      TEM,MIN), (LOG,MIN))
```

Notice that you can insert blanks between the last parameter in a record and the end of that record.

- The order in which the options are specified in the control statement data set has no effect on the dump formatting output order.
- Blanks imbedded within the parameters on a given record cause the formatter to assume the control statement is ended.
- The options can be uppercase or lowercase EBCDIC; they are translated to uppercase before being processed.
- Options can be specified by any unique number of the option's lead characters. If a non-unique abbreviation is passed, the first matching option is chosen. The FMTIMS verb cannot be abbreviated.
- Enclose an option that has a qualifier in parentheses.

## Contents formatted for FMTIMS options

FMTIMS options can be specified on the FMTIMS statement in any order.

### Contents formatted for FMTIMS options

The options are listed in alphabetical order.

Some options state they "are ignored for batch." If the dump was taken because batch processing (IMS DB or CICS) failed, the control blocks for these options are either meaningless or not included in the dump data set. Therefore, the control blocks are not formatted even if you specify that option on the FMTIMS statement.

Most options can be specified with the MIN qualifier. Whenever possible, specify this qualifier to reduce the number of control blocks formatted. You can always format the dump data set again if you need the additional information.

#### **ALL**

Causes a full, formatted dump.

(ALL,MIN) formats the dump as if each option were specified with the MIN qualifier.

#### **AOI**

Formats the storage for the Type 2 Automated Operator Control blocks.

#### **AUTO**

Provides an optimal subset of the IMS dump formatting options without having to first analyze the dump and without having to understand the content or use of all of the IMS dump formatting options.

This option uses the failing ITASK type information to choose one of the formatter's functional areas, and selects the appropriate dump formatter options.

#### **CBT**

Formats storage management area control blocks, including:

- Control Block Table Header
- Individual Control Block Table entries

Output is the same if (CBT,MIN) is specified.

#### **CBTE,cbteid**

Formats all the IPAGEs for the identified CBTE type (cbteid), including:

- Individual Control Block Table entries

- All IPAGE storage of the requested CBTE type

For example, if you specify (CBTE,DPST), all DPST IPAGEs are formatted.

This option can be repeated as needed and has no defaults. The requested IPAGEs must be part of the dump data set. MIN is not valid for the CBTE option.

### CLB/LLB

Permits formatting of an individual Communication Line Block or Link Line Block and its subordinate blocks. Select this option by the following:

- Address
- Node name
- LTERM name
- Communication ID

Select the LLB by address or link number.

The CLB/LLB format creates eye catchers and index entries similar to the following:

**CLB/LLB	REQUESTED CLB/LLB
-----------	-------------------

### DB

Formats areas and control blocks used for IMS Database functions. The following table shows the areas formatted under the (DB) and (DB,MIN) FMTIMS options.

<i>Table 164. Formatted areas under the FMTIMS options DB and DB,MIN</i>	
(DB)	(DB,MIN)
Data Base Tracker Blocks	same
Coupling Facility Block	same
PSB Directory	same
DMB Directory	same
Intent List	not formatted
BFSP	same
DL/I Trace	same
Fast Path Trace (if Fast Path is active)	same
OSAM Pool Control Blocks and buffers	OSAM Pool Control Blocks only
Program Isolation blocks	same
All PSTs and related control blocks, including PCBs, SDBs, Savearea set, alternate DL/I DECB, DSGLRKEY, hierarchical holder, delete work area, RPLI, VSAM PLH, and retrieve trace	Active PSTs, with the same related control blocks
If Fast Path is present: EPSTs and related control blocks, including EPCBs, ESRTs, EMHBs, message buffers, XCRBs, DMHRs, and DEDB buffers	If Fast Path is present: EPSTs and related control blocks, including EPCBs, ESRTs, EMHBs, XCRBs, and DMHRs
VSAM buffer pool control blocks	same
RLPL for IRLM requests	same

In a DL/I–SAS environment, DPST formatting does not format related control blocks if the DL/I address space was not included in the dump data set.

## DBRC

Formats records used by DBRC in its processing, including:

- DFSRCWKB block
- DFSBRLSB block
- Dump Router storage
- Global Data block
- GDBDLTAR block
- GDBDSAAR block
- GDBRECAR block
- GDBLISAR block
- DSPEXIAG block
- DSPEXOPM block
- VFYWSPAC block
- DSPOCPAG block
- DSPJCLAR block
- GDBGPDAR block
- GDBRUPAR block
- GDBOLCAR block
- GDBMNPTR block
- GDBESAVE block
- GDBISAVE block
- GDBCSAVE block
- GDBRSAVE block
- DSPCMPAG block
- DSPVFILE block
- DBRC Internal Trace

Output is the same if (DBRC,MIN) is specified. DBRC blocks must be present in the dump data set to be formatted.

## DC

Formats the data communication areas listed in the following table. This option is skipped if the CTL address space is not included in the dump data set.

*Table 165. Data communication areas formatted by DC and DC,MIN*

(DC)	(DC,MIN) <sup>1</sup>
All CLBs, LXBs, and LCBs, with subordinate control blocks: <ul style="list-style-type: none"><li>• Current CTB or LTB, and CNT</li><li>• Allocated I/O buffers</li><li>• CIB, if using MFS processing</li><li>• CCB, if using conversational processing</li><li>• MFS work buffers</li><li>• ECNT, EMHB, and message buffer, if the CTB shows a Fast Path terminal</li></ul>	Active CLBs, LXBs, and LCBs, with the same subordinate control blocks except that current CTB or LTB and CNT are not formatted.
SMB table	not formatted

Table 165. Data communication areas formatted by DC and DC,MIN (continued)

(DC)	(DC,MIN) <sup>1</sup>
CTT table	not formatted
SPQBs and the CNTs chained off unallocated SPQBs	not formatted

**Note:**

1. (DC,MIN) formats control blocks only for those lines, nodes, and links that meet at least one of the following criteria:
  - a. MSC links
  - b. Nodes in OPNDST or CLSDST processing
  - c. Lines or nodes with allocated input, output, or receive any buffers
  - d. CLBs that have an active SAP

Both DC options are ignored for batch.

**DEDB**

Formats the DEDB control blocks and areas. The areas included are listed in the following table.

Table 166. DEDB control block areas formatted by DEDB and DEDB,MIN

(DEDB)	(DEDB,MIN)
ALDS	same
DMCBs, SGTs, FDTs, and MRMBs for open DEDBs	same
DMACs and ADSC for open DEDB areas	same
XCRBs, DMHRs, and buffers	XCRBs and DMHRs only
SRBs and ESRBs	same

**DISPATCH**

Formats areas relating to the IMS Dispatcher and its functions. The following table shows the areas formatted under this FMTIMS option.

Table 167. Areas formatted by DISPATCH and DISPATCH,MIN

(DISPATCH)	(DISPATCH,MIN)
Dispatcher work areas	not formatted
Dispatcher Trace	same
Scheduler Trace	not formatted
Latch Trace	same

(DISPATCH,MIN) is ignored for batch.

**DPST,jobname**

**DPST,N,dependent region number**

**DPST,A,address**

Permits formatting of an individual Dependent Region Partition Specification Table and its subordinate blocks for PSTs related to MPPs, BMPs, IFPs, and batch DL/I. You can specify one of the following choices:

- job name

- Dependent region number
- DPST address

Output follows the DB formatting output in the dump formatter. The eye catchers and index entries appear as follows:

**DPSTS	REQUESTED DPSTS
---------	-----------------

## EMH

Formats the Expedited Message Handler areas that are used by IMS Fast Path, as shown in the following table. The CTL address space must be included in the dump data set for this option to be formatted.

*Table 168. Areas formatted by EMH and EMH,MIN*

EMH	EMH,MIN
RCTEs	same
BALGs, EMHBs, and message buffers	BALGs and EMHBs only

## LOG

Formats control blocks and areas used by the IMS logger. The areas included shown in the following table. These areas, except for the WADS and the DLOG trace, are repeated in the dump when the IMS Monitor is active.

*Table 169. Areas formatted by LOG and LOG,MIN*

LOG	LOG,MIN
LCD	same
Restart Log Work Area	same
WADS and the data necessary to manage it	WADS only
OLDS prefix and the buffer associated with it	OLDS prefix only
Log DSET, which defines all OLDS currently available for use	same
Message work areas and Logger message areas	same
DLOG trace	same

## MSDB

Formats the Main Storage Databases used by IMS Fast Path. The areas included are listed in the following table.

*Table 170. Main storage databases formatted by MSDB and MSDB,MIN*

MSDB	MSDB,MIN
MSDB headers	same
all MSDBs	not formatted

## POOL, NAME, poolid

Invokes formatting of the storage manager control blocks and the pool storage for any of the following pools:

- AIOP
- ALL
- CESS

- CIOP
- CMDP
- DBWP
- DLDAP
- DLMP
- DPSB
- DYNP
- EMHB
- EPCB
- FWP
- HIOP
- MFBP
- PSBW
- QBFL
- QBUF
- SPAP
- LUMC
- LUMP

NAME is an optional keyword indicating the pool name parameter. If NAME is omitted, the first parameter is assumed to be the pool name.

The poolid is a required 4-character pool name of an existing storage manager pool or the keyword ALL. If ALL is specified, the following storage pools are formatted:

- AIOP
- HIOP
- CIOP
- CMDP
- CESS
- DYNP
- SPAP
- EMHB
- FWP
- QBUF
- QBFL
- DLMP
- DPSB
- DBWP
- MFBP
- EPCB
- LUMP
- LUMC

ALL triggers the formatting of any storage manager trace table entries along with the storage manager control blocks and pool storage.

MIN is an optional keyword. If MIN is specified for one of the dynamic pools (AOIP, CESS, CIOP, CMDP, DYNP, EMHB, FWP, HIOP, LUMC, LUMP, or SPAP) only the storage manager pool header and block

headers are formatted. If MIN is omitted, the pool header control block is formatted, along with the blocks and block headers that represent the dynamic storage pool.

## QM

Formats the control blocks and areas of the IMS queue manager. The formatter skips this option if the CTL address space is not included in the dump data set. The areas included are shown in the following table.

*Table 171. Areas formatted by QM and QM,MIN*

QM	QM,MIN
Qpool Prefix	same
Qpool Buffer Prefix	same
Qpool Buffer	not formatted

Both QM options are ignored for batch.

## RESTART

Formats the IMS restart control blocks and related areas, including:

- Checkpoint ID table
- SIDXs and their subordinate blocks:
  - All LCREs for the SIDX entry being processed
  - All RREs for the SIDX entry being processed
- All RPSTs for the SIDX entry being processed
- FRB, if present

Output is the same if (RESTART,MIN) is specified. Both RESTART options are ignored for batch.

## SAP, ECBADR, ecbaddr

### SAP, ADDRESS, sapaddr

The SAP option can be invoked using either the SAP address or the SAP's ECB address (providing that the ECB is a valid ITASK and has a prefix pointing to a SAP). The SAP option request can be placed either on the IMSDUMP verb line after FMTIMS or in the DFSFRMAT data set. The following examples show SAP option requests:

```
VERBX IMSDUMP 'imsjname,II,N,FMTIMS (SAP,ADDRESS,20864C0) '
```

```
VERBX IMSDUMP 'imsjname,FMTIMS SCD,(SAP,ECBADR,3064250) '
```

For compatibility reasons, the MIN qualifier is allowed, but the output is the same. Individual SAP option formatting is also available on the IMS Low Level panel of the IMS IMS Dump Formatter dialog. The ADDRESS parameter can be omitted because ADDRESS is the default TYPE for the SAP option.

Individual SAP/save area formatting allows complete formatting of SAP/save areas when additional information is required. The output from individual SAP formatting is the same as the SAVEAREA option output. Individual SAP formatting provides the following eye catcher/index entry:

```
**SAPS      REQUESTED SAPS
```

## SAVEAREA

Formats the save area information, including:

- Formatted SAPs and any UEHBs anchored off the SAPs.

**Restriction:** The UEHBs cannot be formatted if the CTL address space is not included in the dump data set.

- Formatted Save Area Sets associated with each SAP.
- Unformatted dump of the IPAGEs containing the SAPs.

If the DL/I address space is not in the data set, then the DL/I SAPs are not formatted. If the CTL address space is not in the data set, then the non-DL/I SAPs are not formatted. Output is the same if (SAVEAREA,MIN) is specified. Both SAVEAREA options are ignored for batch.

The SAVEAREA also comes with a summary option that allows a faster overview scan of the IMS ITASK status within a dump. The SAVEAREA SUMmary output reduces the SAP/Savearea formatting to minimal data while adding keyword scan capability and automatic computation of the exit offsets. This reduces keystroke resources required to overview the ITASK status and ITASK module flow. The SAVEAREA SUMmary and individual SAP formatting provides the following eye catcher/index entry:

```
**SSS      SAP/SAVE CONDENSED SUMMARY
```

SAVEAREA Summary formatting contains the following scannable keywords with their associated meanings:

#### **RUN**

ITASKs that are active are given a RUN indicator. Abend and loop analysis is usually concerned only with running ITASKs.

#### **LATCHREQ**

ITASKs that are waiting for an IMS SLX latch (not checkpoint restart LATE latches) are given a LATCHREQ indicator. Enabled wait problem analysis often requires analyzing ITASKs that are waiting for latches.

#### **LATCHOWN**

ITASKs that own an IMS SLX latch (not checkpoint restart LATE latches) are given a LATCHOWN indicator. Enabled wait problem analysis often requires analyzing ITASKs that own SLX latches.

#### **ITASK type**

The ITASK type is in the summary and is scannable. The ITASK type names are not at the end of the scan list, however. The ITASK type is preceded by the label "type". The possible type names can be gotten from the DFSCIR macro prolog.

### **SB**

Formats the control blocks, areas, and buffers of the Sequential Buffering function (SB) of IMS. This option also formats those DL/I control blocks which are important for debugging the SB function.

The SB information is divided into four sections. The following table shows which sections are formatted with the SB and SB,MIN options.

*Table 172. Sections formatted by SB and SB,MIN*

<b>SB</b>	<b>SB,MIN</b>
Subsystem overview	same
PST overview <sup>1</sup>	same <sup>2</sup>
Sorted blocks <sup>1</sup>	same <sup>2</sup>
Sorted buffers <sup>1</sup>	not formatted

#### **Note:**

1. The DL/I address space must be included in the dump data set for these areas to be formatted.
2. Formatted only if you requested a conditional SB activation for that application or PST.

The SB information is divided into the following sections:

1. Subsystem Overview of SB—provides an overview of SB control blocks from an IMS subsystem point-of-view. The SDCBs appear in the order in which they are anchored in the SBSCD. Each SDCB is followed by its SDSGs. The section contains the following information:
  - SB section of the SCD
  - SBSCD, including the SBHE blocks



- SDCBs
  - SDSGs
2. PST Overview of SB—formats the SB control blocks (and other IMS control blocks significant to SB) for each active PST. These blocks are sorted in hierarchical order. For example, the first DBPCB and its JCB, DSGs, EDSGs, and SDSGs; then the second DBPCB with its subordinate blocks, and so on. The section contains the following information:
    - SB and buffer-handler sections of the PST
    - PST DECB prefix
    - SB extensions to the PST
    - SB work area
    - SBPARMS
    - DBPCBs and their JCBs, DSGs, EDSGs, and SDSGs
  3. Sorted SB Blocks—contains SB control blocks (and other IMS control blocks significant to SB) sorted according to their virtual storage address. The section contains the following information:
    - DBPCBs
    - DCB with its OSAM extensions
    - DSGs
    - EDSGs
    - JCBs
    - OV-IO DECB prefix
    - PST DECB prefix
    - SB extensions to DCBs
    - SB extensions to DSGs
    - SB extensions to the PST
    - SB work area
    - SBPARMS
    - SBUFs
    - SCARs
    - SRANs
  4. Sorted SB Buffers—contains the SB buffers of each SB buffer pool. The SB buffers of one SB buffer pool are contiguous in storage and are formatted as one entity. The buffer pools are then sorted by virtual storage address.

## SCD

Formats the IMS SCD and related areas. The areas included are listed in the following table.

*Table 173. Areas formatted by SCD and SCD,MIN*

(SCD)	(SCD,MIN)
SCD	same
Latch Extensions	same
Scheduler Sequence Queues	not formatted
Synchronous APPC/OTMA Shared Message Queue SCD Extension	same
Fast Path SCD Extension, if Fast Path is active	same
Formatted dumps of the batch key 7 SCD	same

Table 173. Areas formatted by SCD and SCD,MIN (continued)

(SCD)	(SCD,MIN)
LU 6.2 SCD extension	same

### SPST

Formats the system PSTs, which are ITASKs used by IMS. This includes:

- Global system PSTs
- Local control region address space PSTs
- Local DL/I address space PSTs
- Areas related to the above PSTs, including LWA and IRLMA

Some SPSTs are not formatted if the CTL address space is not in the dump data set. Output is the same if (SPST,MIN) is specified. Both SPST options are ignored for batch.

### SUBS

Formats the areas and control blocks that IMS uses to manage subsystems, including:

- Subsystem trace
- Global ESET block

Output is the same if (SUBS,MIN) is specified. Both SUBS options are ignored for batch.

### SUMMARY

Formats the current diagnostic section.

The SUMMARY data areas are not formatted if the SDWA address space is not part of the dump data set. (For abends and batch processing, the SDWA address is saved by the ESTAE module. For online processing, the dump must be taken by DFSOFMD0, and the SDWA parameter must be passed at DFSDUMP time.)

The areas formatted with this option include:

- Failing PSW
- Abend code
- Module name
- Registers at time of abend
- 256 byte instruction area—128 bytes above and below the failing PSW
- 16 register storage areas—512 bytes above and 256 bytes below the registers at time of abend
- SDWA address space of IMS
- Failing SAP and its UEHB
- Failing ITASK when the ITASK is a DPST, system PST, CLB, or LLB (dependent region errors, some systems services errors, terminal process errors, and MSC errors)

The SUMMARY option names the ITASK type when it is determined, even if it is not one of the ITASK types that provide for additional formatting. The ITASK type name is two to four characters. If it is unknown, the type name is "UNKN".

Output is the same if (SUMMARY,MIN) is specified.

### SYSPST

Permits formatting of an individual system partition specification table and some of its subordinate blocks. Select this option by address or system PST name. This option creates eye catchers and index entries similar to the following:

```
**SYSPSTS          REQUESTED SYSTEM PSTS
```

## SYSTEM

Formats the SUMMARY, SAVEAREA, and SCD areas as one group. The areas and control blocks formatted are the same as if each of the options were invoked separately.

(SYSTEM,MIN) is formatted as though each of the options were specified with MIN.

See the individual options for a list of the areas formatted.

## TRACE, NAME, table-id

Gets a new search module that invokes the normal trace format control module (DFSATRA0) to format trace tables separately. This option enables viewing of trace table data without having to format the entire option that usually includes the formatted trace table. The TRACE option request uses the 2-character trace table EBCDIC ID code from the Trace Selection panel. The IMS Dump Formatter ISPF panels also accept an option of "ALL" to format all IMS trace table traces. The IMS Dump Formatter dialog TRACE SELECTION panel provides a selectable list of IMS trace tables with the trace name, internal ID, and description. The following examples are TRACE format requests, followed by comments for each. In each case, the NAME keyword can be omitted because NAME is the default TYPE parameter. The following example is a request for the DL/I trace table.

```
FMTIMS... (TRACE,NAME,DL) , ...
```

The following is a request for the dispatcher trace table and the DL/I trace table with a MIN option that is ignored.

```
FMTIMS... , (TRACE,NAME,DL,MIN) , (TRACE,NAME,DS) ...
```

## UTIL

Formats the control blocks for the IMS Partial Database Reorganization utility, including:

- Common area
- Database table
- Segment table
- Action table

The output is the same if (UTIL,MIN) is specified. Both UTIL options are ignored for batch.

## Related reference

[“Formatted dump contents” on page 541](#)

The title for a formatted dump varies, depending on the parameters provided. The output includes eye catchers and an index to help you locate individual control blocks.

[“Table of control block definitions” on page 72](#)

All of the control blocks are listed, and for each control block the acronyms, macros and descriptions are given.

## Formatted dump contents

The title for a formatted dump varies, depending on the parameters provided. The output includes eye catchers and an index to help you locate individual control blocks.

### Dump title

The contents of the dump titles that are created by the dump assist module (DFSFDMP0) and the initialization routines vary, depending on the internal DFSDUMP parameters provided and the SDUMP errors met.

The following examples show five possible dump title formats.

## Title format 1

DFSFDMP0 issued the SDUMP and passed the SDWA parameter. The CALLER parameter was either passed to DFSFDMP0 or the routine generated the parameter using the IMS TCB table.

```
ljjjjjjjj ABEND SYS sss USER uuuu-rrr, DATE.TIME: ddd.tttttt,  
CALLER=cccc, TCB=xxx, MODULE=mmmmmmmm,i
```

***l***

Length of title in hexadecimal - here 91 decimal.

***jjjjjjj***

Job name.

***sss***

System abend code.

***uuuu***

User abend code.

***rrr***

Optional user abend reason code.

***ddd***

Julian day of year.

***ttttt***

Time, in the form HHMMSS.

***cccc***

DFSFDUMP caller parameter or blanks.

***xxx***

Abending TCB or 'UNK'.

***mmmmmmmm***

Abending module or 'UNKNOWN', using the SDWA.

***i***

Indicator if primary (P) or secondary (S) request.

## Title format 2

DFSFDMP0 issued the SDUMP, but did not have an SDWA. The CALLER parameter was either passed to DFSFDMP0 or the routine generated the parameter by using the IMS TCB table.

```
ljjjjjjjj DATE.TIME: ddd.tttttt, IMS DUMP REQUESTED,  
CALLER=cccc, TCB=xxx, REASON=rrr,i
```

***l***

Length of title in hexadecimal - here 80 decimal.

***jjjjjjj***

Job name.

***ddd***

Julian day of year.

***ttttt***

Time, in the form HHMMSS.

***cccc***

DFSFDUMP caller parameter or blanks.

***xxx***

Abending TCB or 'UNK'.

***rrr***

Optional user reason code.



**DFSPINIO,**  
FAILURE ESTABLISHING ESTAE

**DFSPAT00,**  
GETMAIN FAILURE

**DFSPINIO,**  
SSI FAILURE DURING SONCRT

**DFSPINIO,**  
DBCTL FAILURE DURING SONCRT

**DFSPSCH0,**  
SSI FAILURE DURING SCHED

**DFSPSCH0,**  
DBCTL FAILURE DURING SCHED

**DFSPUSC0,**  
SSI FAILURE DURING UNSCHED

**DFSPUSC0,**  
DBCTL FAILURE DURING UNSCHED

**DFSPSYN0,**  
DBCTL FAILURE DURING SYNC

**DFSPDLIO,**  
DBCTL FAILURE DURING DLI

**DFSPPTK0,**  
DBCTL FAILURE DURING PRIME

**DFSPTTH0,**  
SSI FAILURE DURING TERMTHD

**DFSPTTH0,**  
DBCTL FAILURE DURING TERMTHD

**DFSPRA40,**  
PQE CANNOT BE PROCESSED

**DFSPRRA0,**  
PQE OR PAPL IS INVALID

**DFSFPRA0,**  
CONTROL TCB ESTAE INVOKED

**DFSFPAT0,**  
THREAD TCB ESTAE INVOKED

**DFSFPRA0,**  
DRA ESTAE FAILED TO ESTABLISH ESTAE

**NO OTHER DRA MESSAGE**

## Title format 4

The SNAP call facility identifies calling routines that generate snap dumps. Supervisor call (SVC) dumps are generated only for the intended abend codes or status codes, and for unknown calling routines.

This dump is created by DFSERA20 when a SNAP dump is requested. The title is the format:

```
nnnnnnnn IMS USER ABEND uuuu, DATE.TIME: ddd.tttttt,i
```

**nnnnnnnn**  
IMS name.

**uuuu**  
The user abend code or UNK if a SNAP was requested, but there was no abend set.

**ddd**

Julian day of year.

**ttttt**

Time, in the form HHMMSS

**i**

Indicator if primary (P) or secondary (S) requested.

## Title format 5

This dump is created by DFSERA20 when a SNAP dump is requested. The format is generated for dumps that are taken when an unexpected DL/I status code is returned during HALDB Online Reorganization. The title is the format:

```
nnnnnnnn UNEXPECTED STATUS CODE cc, DATE.TIME: ddd.tttttt,i
```

**nnnnnnnn**

IMS name.

**cc**

The unexpected status code returned during HALDB Online Reorganization.

**ddd**

Julian day of year.

**ttttt**

Time in the form HHMMSS.

**i**

Indicator if primary (P) or secondary (S) requested.

cc is the unexpected status code returned during HALDB Online Reorganization.

## Eye catchers

To help you locate areas that are dumped, eye catchers are printed near the major control blocks in the formatted dump. Eye catchers are also useful when you are using IPCS to view the formatted dump. Examples of eye catchers are:

**\*\*SCD**

System Contents Directory Area

**\*\*SSA**

SAP and Save Area

**\*\*SB-1**

Subsystem Overview for Sequential Buffering

Eye catchers are also listed at the top of the formatted dump.

## Index

The formatted dump also contains an index at the end that is created by the z/OS Index Service routine. Index entries are created each time an eye catcher is processed during formatting, and after the Offline Dump Formatter finishes processing.

Entry length is limited to 40 decimal characters.

## Formatted dump output order

The following list shows the order in which the Offline Dump Formatter prints control blocks. If you specify **FMTIMS ALL** and all necessary data is available to the formatter, you get all of the areas listed. The order does not change when you specify subset options, but only the areas you specify are formatted. Descriptive information has been added for some control blocks where it is useful.

**ODF Initialization Messages**

These messages appear when the formatter is unable to find particular address spaces in the dump data set.

**Eye catchers**

Eye catchers of the areas that you requested to be formatted on this pass of the formatter.

An eye catcher could be included in this list even if the Offline Dump Formatter was unable to format the control block, because the list is built from the parameters you include in the FMTIMS statement.

**Diagnostic Area**

Contains the PSW, system and user completion codes, save area ID of the module that was executing, and registers in use when abnormal termination occurred.

**Instruction Area**

Contains the area of storage from 128 bytes before to 128 bytes after the address of the failing instruction in the PSW.

**Register Area**

This area contains 512 bytes above and 256 bytes below each register value in the passed SDWA. The address space ID (ASID) used is the one passed in the SDWA.

**System Diagnostic Work Area**

The mapping DSECT is IHASDWA.

**Referenced SAP**

The mapping DSECT is ISAP.

**System Contents Directory**

The mapping DSECT is ISCD.

**SCD Latch Extension**

The mapping DSECT is ISCD.

**Scheduler Sequence Queues**

Controls the status of each region. The mapping DSECT is ISCD.

**Synchronous APPC/OTMA Shared Message Queue SCD Extension**

The mapping DSECT is DFSCSCD.

**FP ESCD**

The mapping DSECT is DBFESCD.

**Control Block Table**

Contains entries of control blocks that macro DFSCBTS uses for tracking. The mapping DSECT is DFSCBTS.

**Control Block Table Pools**

All IPAGEs for CBTE types requested with the (CBTE,cbteid) option.

**Save Area Trace****SAPs with their Active UEHBs****Save Area Prefix**

All SAPs are SNAPed. Each SAP is followed by its save area set. At the end of this section, all of the SAP IPAGEs are dumped.

**IMS Task Dispatch Work Area**

The mapping DSECT is IDSPWRK.

**DBRC Task Dispatch Work Area**

If present in the system, it is mapped.

**IMS Control Task Dispatch Work Area**

Contains the same information as the IMS log task dispatch work area.

**Dependent Region Dispatch Work Area**

For every dependent region in IMS, the dispatcher work area is mapped.



**Dispatcher Trace Data**

DSECT IDSPWRK contains the function codes associated with the dispatcher and an explanation of each code.

**Scheduler Trace Data**

Scheduler trace data is mapped by DFSSCHED. The trace entries contain scheduler function codes.

**Latch Trace Data**

The trace entries contain latch and unlatch function codes. The mapping DSECT is IDLIVSAM TRACENT.

**Timer Work Areas**

These are control blocks used by the internal IMS timers.

**System PSTs**

These are system work areas for any online or batch region. The mapping DSECT is IPST.

**Restart Work Areas**

See RESTART [“Solving IMS problems by using the IMS Offline Dump Formatter” on page 514](#) for a list of these areas.

**Log Control Directory**

Contains information about the IMS log. The mapping DSECT is LCDSECT.

**Log Work Areas****Log Buffers**

Each log buffer contains buffer information and the log control DECB. The mapping DSECT is LCDSECT.

**Open Record**

Contains the type 06 log record. The mapping DSECT is ILOGREC.

**Control Record**

Contains the type 42 log record. The mapping DSECT is ILOGREC.

**Monitor Log Directory**

Contains the same information as the log control directory.

**DLOG Trace Data**

Trace table used to show IMS logging activity. The mapping DSECT is ILOGREC (67FA).

**Subsystem Control Table****Attach Work Areas****PSB Directory**

A SNAP of the PSB directory. The mapping DSECT is PDIR.

**DMB Directory**

A SNAP of the DMB directory. The mapping DSECT is DDIR.

**Intent List**

The DL/I address space must be in the dump data set for this list to be formatted.

**Fast Path Trace****Dependent Region PST formatting**

For each DPST:

- PST
- Savearea
- PDIR
- Intent List
- PSB prefix
- PSB Index Maintenance, Index I/O, I/O, SSA, and User Parms work areas

- SMB
- DB PCB blocks
- Delete work area
- Retrieve Trace
- HD Space Trace
- FLDS
- RPL
- IRLM area
- PST log work area
- Fast Path EPST and chain addresses, ECNTs, EMH message, EPCBs, XCRBs, and DMHR

#### **BFSP**

Formats the buffer pool prefix. The mapping DSECT is BFSP.

#### **BFUS**

Formats the subpool prefix. The mapping DSECT is BFUS. The mapping DSECT is RPLI.

#### **DL/I Data**

A dump of the DL/I lock activity and program isolation trace table. The mapping DSECT is IDLIVSAM TRACENT.

#### **Lock Activity Trace Data**

See DL/I Data.

#### **Program Isolation Data**

Includes the QEL, QCB and REQ areas. The mapping DSECT is XC00.

#### **OSAM Control Blocks**

The system attempts to follow the main pool, the subpool header, and the buffer prefix, and to dump the buffer. However, if an error is encountered during formatting, the entire buffer pool is SNAPed from the last valid subpool address.

#### **DL/I Trace Table**

#### **Sequential Buffering Blocks**

Sequential Buffering information is grouped into the following four sections. (See the explanation of the (SB) FMTIMS option [“Solving IMS problems by using the IMS Offline Dump Formatter”](#) on page 514 for a complete list of the blocks dumped in each section.)

1. Subsystem Overview for Sequential Buffering
2. PST Overview of Sequential Buffering control blocks
3. Formatted Sequential Buffering control blocks
4. Sequential Buffering buffers

#### **DEDB Formatting**

#### **Fast Path EMH Formatting**

#### **Fast Path MDSB Formatting**

#### **Communication Line Blocks and Subordinate Blocks <sup>2</sup>**

For each CLB line, all the control blocks associated with that line are formatted.

#### **CTB <sup>2</sup>**

The mapping DSECT is ICLI CTBBASE=0.

#### **Input Buffer <sup>2</sup>**

A SNAP of the input buffer, if input is active.

**Output Buffer <sup>2</sup>**

A SNAP of the output buffer, if output is active.

**CCB <sup>2</sup>**

Present if a conversation is active or held. The mapping DSECT is ICLI CCBASE=0.

**CIB <sup>2</sup>**

Present if MFS is in use. The mapping DSECT is ICLI CIBBASE=0.

**Communication Terminal Table <sup>2</sup>**

Defines terminal characteristics. The mapping DSECT is ICLI CTTBASE=0.

**SPQB Entries <sup>2</sup>**

Entries on the subpool queue block chain. Unallocated CNTs are also formatted here.

**SMB Table <sup>2</sup>**

This table defines transaction characteristics in the IMS system. The mapping DSECT is IAPS SMBBASE=0.

**Queue Manager Pool Prefix and Buffers <sup>2</sup>**

The mapping DSECTs are ICLI POOLBASE=0, ICLI BFRBASE=0, and QPOOL. The buffer prefix list contains the address of each buffer's prefix, status byte, and first and last pending and current device relative record number (DRRN).

**Batch Utility Areas****DBRC Work Areas****LUM Trace**

Allows LU 6.2 activities to be analyzed with the MVS/ESA APPC trace entries by the LU manager.

**Related reference**

“Contents formatted for FMTIMS options” on page 531

FMTIMS options can be specified on the FMTIMS statement in any order.

## Edited command buffer format

---

The edited command buffer is logged in the X'02' log record and is passed to the AOI user exit. You can use the edited command buffer to determine if any recoverable commands were issued for the resource you are analyzing.

**Edited command buffer examples**

For example, if you are analyzing a hung terminal problem, look at any log records, including X'02' records, that apply to that terminal.

However, finding the applicable log records might be difficult. If the problem is repeatable, you can use the /LOG command to mark the log when certain activities are started or stopped. The /LOG command writes a comment to a X'02' log record. This narrows the range of log records you need to examine.

**Example:** If transaction XYZ results in a hung terminal, use the /LOG command to write a comment to a X'02' log record before the transaction is started and after the terminal is hung, as follows:

```
/LOG START XYZ TRAN THAT RESULTED IN HUNG TERMINAL .
/LOG TERMINAL IS NOW HUNG.
```

Look for these comments in the X'02' log record edited command buffers to determine the range of log records to examine.

The following figure shows the layout of the edited command.

---

<sup>2</sup> These areas are not dumped in a DBCTL environment.

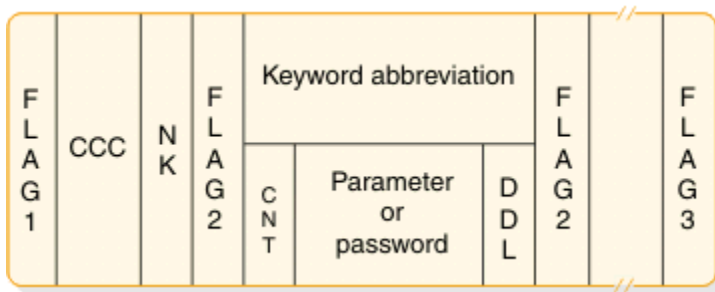


Figure 79. Edited command layout

## Figure Number Description

### FLAG1

X'FE' to denote the beginning of the edited command. If any parameter contains an error, the command action modules set this byte to X'FC'. An exception is DFSICL40 processing of "ALL" expanded parameters.

### CCC

First 3 characters of entered command.

### NK

Hexadecimal value of number of keywords in the condensed buffer.

### FLAG2

One of the following:

#### X'FC'

Parameter that follows found in error.

#### X'FF'

3-byte keyword abbreviation follows.

#### X'FE'

Count (CNT) field and parameter follow.

#### C'('

Count (CNT) field and password follow.

### Keyword Abbreviation

First 3 characters of entered command. Consult DFSCKWDO to obtain the abbreviation; it is sometimes the first 3 characters of any keyword.

### CNT

Count of number of characters in parameter or password immediately following the CNT. It can be a comma, period, blank, or left parenthesis.

### Parameter or Password

Exists exactly as entered from the terminal.

### DDL

The delimiter entered after the parameter or password. It may be X'80' if the keyword "ALL" was expanded to individual parameters.

### FLAG3

Period indicating end of command.

**Exception:** Only parameter passwords are present in the condensed buffer; command passwords are not present.

## IMS Dump Formatter

The IMS Dump Formatter provides ISPF dialog support for IMS Offline Dump Formatter requests, and simplifies the process of making requests by providing menus for format option selection, help members

for online options, automatic terminal and spool output control, and a configuration panel to provide interactive assistance in defining the IMS environment.

The IMS Dump Formatter menu is available from the component analysis section of the IPCS dialogs (IPCS ISPF selection 2.6).

## Dump formatting options

General formatting options are available for IMS and z/OS memory dumps.

- Offline IMS or z/OS formatting from a SYS1.DUMPxx (SDUMP) data set
- Offline IMS or z/OS formatting from a SYSMDUMP data set
- Online IMS or z/OS formatting directed to either a SYSABEND or SYSUDUMP SYSOUT class
- Online z/OS formatting directed to either a SYSABEND or SYSUDUMP spinoff dump SYSOUT class

The dumping options that are in effect in an IMS environment depend on:

- Whether the execution is an online execution or a batch execution.
- Whether the error might terminate IMS.
- Whether the FMTO EXEC parameter is specified.
- Whether the SOD EXEC parameter is specified for an online execution.

**Note:** This parameter is not supported in a batch execution.

- Whether a SYSABEND, SYSUDUMP, or SYSMDUMP is chosen.
- The z/OS dump options chosen for SDUMP, SYSABEND, SYSUDUMP, and SYSMDUMP.
- Whether the Dump Override Table (DFSFDOT0) contains any entries. DFSFDOT0, and its relationship to the various dumps, is explained in *IMS Version 15.5 Exit Routines*.

## FMTO options

The FMTO options that you can specify for the **FMTO=** parameter in a procedure are listed in [Table 174 on page 551](#).

Note the following points when you want IMS to produce memory dumps:

- The default and recommended value is FMTO=D.
- SYSABEND, SYSUDUMP, and SYSMDUMP are mutually exclusive.
- z/OS formatting always accompanies IMS online formatting.
- Spinoff dumps can occur in addition to SDUMPs.

Table 174. FMTO options and their effect on memory dumps produced											
FMTO value	IMS online formatting			z/OS online formatting		IMS offline formatting		IMS offline formatting		IMS spinoff dump	
	SYSABEND SYSUDUMP			SYSABEND SYSUDUMP		SDUMP		SYSMDUMP		SYSABEND SYSUDUMP	
	TE	NT	FDDL	TE	NT	TE	NT	TE	NT	TE	NT
D	S	—	S	S	—	Y	Y	S	—	—	Y
X	S	—	S	S	—	Y	N	S	—	—	Y
M	N	—	N	S	—	Y	Y	S	—	—	Y
R	N	—	N	S	—	Y	N	S	—	—	Y
T	Y	—	Y	Y	—	N	Y	Y	—	—	Y

Table 174. FMTO options and their effect on memory dumps produced (continued)											
FMTO value	IMS online formatting SYSABEND SYSUDUMP			z/OS online formatting SYSABEND SYSUDUMP		IMS offline formatting SDUMP		IMS offline formatting SYSMDUMP		IMS spinoff dump SYSABEND SYSUDUMP	
P	Y	—	N	Y	—	N	Y	Y	—	—	Y
F	Y	—	N	Y	—	N	N	Y	—	—	Y
N	N	—	N	Y	—	N	Y	Y	—	—	Y
Z	N	—	N	Y	—	N	N	Y	—	—	Y

The codes indicated in the column headers in [Table 174 on page 551](#) are as follows:

**TE**

Terminating error. Error will terminate IMS.

**NT**

Non-terminating error. Error will not terminate IMS.

**FDDL**

Formatted dump delete list processing.

The codes indicated in the table cells in [Table 174 on page 551](#) are as follows:

**Y**

Yes, the dump will be produced.

**S**

Yes, the dump will be produced only if the SDUMP fails.

**N**

No, the dump will not be produced.

**—**

Not applicable, or not attempted to produce a dump.

## SDUMPs

If SDUMP is used, be aware of these additional considerations:

- If SDUMP is requested by the FMTO option, it is attempted first. If SDUMP is successful, no other dump (except a spinoff dump) is generated.
- If SDUMP fails, if IMS online formatting is requested, and if a SYSABEND or SYSUDUMP DD statement is present, a dump with IMS and z/OS online formatting is generated for terminating errors.

The formatted dump delete list (FDDL) consists of IMS module and control block names, and a dump or delete action indicator. For a module, the delete process removes storage occupied by the module. This causes the module to be omitted from the storage image portion of the dump. However, the module/save ID is printed in the dump. For a control block, the delete process similarly removes storage and storage image output if the block was successfully formatted.

- If SDUMP fails, IMS online formatting is not requested, and a SYSABEND or SYSUDUMP DD statement is present, a dump with z/OS online formatting is generated for terminating errors.
- If SDUMP fails and a SYSMDUMP DD statement is present, a machine readable dump is generated for terminating errors.

## SYSMDUMPs

If SYSMDUMP is used, be aware of these additional considerations:

- SYSMDUMP DD statements should be present in the IMS, DBRC, and DLISAS procedures.

- The SYSMDUMP DD statement must specify DISP=MOD, because it is possible for multiple tasks to generate dumps during the processing of an error.
- The SYSMDUMP data set must be refreshed (for example, scratched and reallocated) before it can be reused. If this is not done, new dumps are appended to the end of the data set and may not be accessible to the Offline Dump Formatting utility.
- Do not use SYSMDUMP in place of SYSABEND or SYSUDUMP.

## Spinoff dumps

If spinoff dumps are used, be aware of these additional considerations:

- If a spinoff dump SYSOUT class is specified (SOD EXEC parameter in the IMS control region), a spinoff dump is generated for nonterminating errors.
- Spinoff dumps allow the dump data set to be released to JES for immediate printing. (This can be useful when external subsystems are supported.)
- Spinoff dumps are only generated for those tasks that do not terminate IMS.
- If a SYSABEND or SYSUDUMP DD statement is not provided, the first dump is not printed. For subsequent dumps, the dump data set is dynamically allocated using the spinoff dump class.
- The spinoff dump class (SOD execute parameter) should be the same as the SYSABEND or SYSUDUMP SYSOUT class.

If the classes are different, the first dump goes to the class specified on the SYSABEND or SYSUDUMP DD statement. The dump data set is then dynamically deallocated to free the dump for printing and then dynamically reallocated using the spinoff dump class. Therefore, all subsequent dumps go to the spinoff dump class.

- The only valid SYSABEND or SYSUDUMP DD statement parameter used for spinoff dump is the SYSOUT class. Other parameters are ignored.
- Spinoff dumps use the default output limit established during JES definition. Incomplete spinoff dumps can result if the limit specified is too low.

## z/OS dump options

To create IMS dumps that are useful in diagnosing problems, specify the proper dump options for each type of z/OS dump used for IMS.

Use the z/OS operator command DISPLAY DUMP, OPTIONS to determine the dump options currently in effect on your system.

If the current dump options do not include the options listed in this section, the defaults that are obtained from SYS1.PARMLIB (members IEAABD00, IEADMP00, and IEADMR00) must be altered. Use the z/OS operator command CHNGDUMP to change dump options.

Table 175. z/OS dump options	
Type of dump	z/OS dump options
SDUMP	SDUMP=(CSA,PSA,RGN,SQA,SUM,TRT,GRSQ)
SYSABEND	<ul style="list-style-type: none"> <li>• SYSABEND,SDATA=(CB,DM,ENQ,ERR,IO,LSQA,SUM,TRT)</li> <li>• SYSABEND,PDATA=(JPA,LPA,PSW,REGS,SA,SPLS)</li> </ul>
SYSUDUMP	<ul style="list-style-type: none"> <li>• SYSUDUMP,SDATA=(CB,ERR,SUM)</li> <li>• SYSUDUMP,PDATA=(JPA,LPA,PSW,REGS,SA,SPLS)</li> </ul>
SYSMDUMP (online system only)	SYSMDUMP=(CSA,LSQA,RGN,SQA,SUM,SWA,TRT)

Table 175. z/OS dump options (continued)	
Type of dump	z/OS dump options
SYSMDUMP (batch system only)	SYSMDUMP=(LSQA,RGN,SQA,SUM,SWA,TRT)

## IMS online system dump options

In an online IMS environment, the following dumps are possible:

- SDUMP to a SYS1.DUMPxx data set. SDUMP can then be formatted using the Offline Dump Formatter utility.

This dump can be generated both for errors that cause IMS to terminate (*terminating errors*) and for errors that do not cause IMS to terminate (*non-terminating errors*).

- SYSABEND or SYSUDUMP with IMS or z/OS formatting performed online.

This dump is generated for terminating errors only if SDUMP fails or was not requested.

- SYSMDUMP to a data set. SYSMDUMP can then be formatted using the Offline Dump Formatting utility.

This dump is generated for terminating errors only if SDUMP fails or was not requested.

- Spinoff dump to SYSABEND or SYSUDUMP with z/OS formatting performed.

This dump is generated for non-terminating errors and can be generated in addition to SDUMP.

## Online system dependent region dump options

Except for SDUMPs, the dumping options for the dependent regions are controlled by the following:

- The spinoff dump SYSOUT class (SOD EXEC parameter in the DFSMPR and IMSFP procedures)
- The presence or absence of a SYSABEND or SYSUDUMP DD statement in the dependent region JCL. These DD statements are mutually exclusive; provide only one.

Do not use SYSMDUMP in IMS dependent regions.

## MPP, IFP, and BMP dependent regions

In the online IMS environment, the following dumps are useful for MPP, IFP, and BMP dependent regions:

- SYSABEND or SYSUDUMP (with z/OS formatting performed online).
- Spinoff dump to SYSABEND or SYSUDUMP (with z/OS formatting). This dump is available only for MPP and IFP dependent regions.
- SDUMP to a SYS1.DUMPxx data set. (SDUMP can then be formatted using the Offline Dump Formatter utility.) An SDUMP is produced in the dependent region for certain IMS system errors involving DL/I or Fast Path. In this instance, the SDUMP is controlled by the FMTO specification for the control region.

## Spinoff dumps in dependent regions

If spinoff dumps are used, some additional considerations are necessary:

- Spinoff dumps allow the dump data set to be released to JES for immediate printing. (This can be useful for "never-ending" MPP or IFP regions.)
- If a SYSABEND or SYSUDUMP DD statement is not provided, the first dump is not be printed. For subsequent dumps the dump data set is dynamically allocated using the spinoff dump class.
- The spinoff dump class (SOD EXEC parameter) should be the same as the SYSABEND or SYSUDUMP SYSOUT class.



If the classes are different, the first dump goes to the class specified on the SYSABEND or SYSUDUMP DD statement. The dump data set is then dynamically deallocated to free the dump for printing and then dynamically reallocated using the spinoff dump class. Therefore, all subsequent dumps go to the spinoff dump class.

- The only valid SYSABEND or SYSUDUMP DD statement parameter used for a spinoff dump is the SYSOUT class. Other parameters are ignored after the first dump.
- Spinoff dumps use the default output limit established during JES definition. Incomplete spinoff dumps can result if the limit specified is too low.
- If the COBOL DEBUG option is used, dumps cannot be generated.

## Batch system dumps

In the batch IMS environment, the following dumps are possible:

- SYSMDUMP to a data set. (SYSMDUMP can then be formatted using the Offline Dump Formatting utility.) This dump is only generated for terminating errors.
- SYSABEND or SYSUDUMP (with IMS or z/OS formatting performed online). This dump is only generated for terminating errors.

## FMTO options for batch dumps

The FMTO options for batch dumps, and their effect on the dumps produced, are different from the FMTO options described in [“FMTO options” on page 551](#). The FMTO options for batch dumps are summarized in [Table 174 on page 551](#).

When using Batch FMTO options, remember the following:

- If a SYSMDUMP DD statement is present, a machine readable dump is generated for terminating errors.
- If IMS online formatting is requested and a SYSABEND or SYSUDUMP DD statement is present, an IMS online formatted dump is generated for terminating errors.
- If IMS online formatting is not requested and a SYSABEND or SYSUDUMP DD statement is present, a z/OS online formatted dump is generated for terminating errors.
- The default is FMTO=D.
- SYSABEND, SYSUDUMP, and SYSMDUMP are mutually exclusive.
- z/OS formatting always accompanies IMS online formatting.

In the following table, the following codes are used:

### Option

#### Effect on dump produced

#### TE

Error would terminate IMS.

#### NT

Error would not terminate IMS.

#### FDDL

Formatted dump delete list processing.

#### Y

Yes

#### N

No

#### —

Not applicable, or not attempted.

Table 176. Batch FMTO options

FMTO Value	IMS Online Formatting SYSABEND SYSUDUMP			z/OS Online Formatting SYSABEND SYSUDUMP		IMS Offline Formatting SYSMDUMP	
	TE	NT	FDDL	TE	NT	TE	NT
D	Y	—	Y	Y	—	Y	—
X	Y	—	Y	Y	—	Y	—
M	N	—	N	Y	—	Y	—
R	N	—	N	Y	—	Y	—
T	Y	—	Y	Y	—	Y	—
P	Y	—	N	Y	—	Y	—
F	Y	—	N	Y	—	Y	—
N	N	—	N	Y	—	Y	—
Z	N	—	N	Y	—	Y	—

If SYSMDUMP is used, be aware of these additional considerations:

- The SYSMDUMP DD statement should specify DISP=MOD.
- If DISP=MOD is used, then the SYSMDUMP data set must be refreshed (for example, scratched and reallocated) before it can be reused. If this is not done, new dumps are appended to the end of the data set and might not be accessible to the Offline Dump Formatting utility.

#### Related concepts

[“Dump contents returned for different FMTIMS options” on page 520](#)

You can specify different FMTIMS options to add different types of information to a formatted IMS system dump.

#### Related reference

[“Sequential buffering service aids” on page 227](#)

When you receive a message or abend that indicates a problem with sequential buffering (SB), several diagnostic tools are available. Some of these tools are useful for diagnosing other IMS database-related problems.

## IMS Dump Formatter menus

You can use the options on the IMS Dump Formatting Menu to browse the dump data set, and perform high-level or low-level IMS formatting.

To use the menus:

1. Go to the IPCS Component Analysis panel.
2. Select DFSAAMPR. The IMS Dump Formatting Primary Menu panel appears.

```

DFSAAMPR ----- IMS DUMP FORMATTING PRIMARY MENU -----
OPTION ===>

 0 INIT          - IMS formatting initialization and content summary
 1 BROWSE        - Browse dump data set (IPCS norm)          *****
 2 HI-LEVEL      - IMS Component level formatting            *USERID  - SK0N0
 3 LOW-LEVEL     - IMS ITASK level formatting                 *DATE    - 00/01/06
 4 ANALYSIS      - IMS dump analysis                         *JULIAN   - 00.006
 5 USER          - IMS user formatting routines               *TIME    - 15:00
 6 OTHER COMP    - Other IMS components (BPE, CQS...)         *PREFIX   - SK0N0
 7 OTHER PROD    - Other IMS-related products                *TERMINAL - 3278
 8 EDA           - IMS Enhanced Dump Analysis                *PF KEYS  - 24
 9 TUTORIAL      - IMS dump formatting tutorial
 X EXIT          - Exit IMS dump formatting

Enter END command to terminate IMS component formatting

```

Figure 80. IMS Dump Formatting Primary Menu panel

- If this is the first time you are reading the dump, select 0 (Initialization). The IMS Dump Content Status panel opens.

```

DFSAAEI0 ----- IMS DUMP CONTENT STATUS -----
COMMAND ===>

Enter the IMS CTL/BATCH or DL/I jobname to cause the IMS symbols to
be set for this dump. Request subsystem list for possible IMS names.

IMS SUBSYSTEM LIST DESIRED? (Y or N)===> N

-----
JOBNAME      ID      ASID      DUMPED?
-----
CTL
DL/I
DBRC
IRLM

ABEND CODE = SYS      USER
MODULE      =

IMS SDWA ADDRESS - 00672918  IMS RELEASE - 1010
IMS SCD ADDRESS  - 00C449E8  IRLM LEVEL  - N/A
ABENDED ASID    - 0027

```

Figure 81. IMS dump formatting initialization/content panel - inactive

- Type the IMS job name in the row marked CTL, or the DL/I job name in the row marked DL/I, and press Enter. If you do not know the job name, type Y next to the IMS SUBSYSTEM LIST DESIRED prompt to scan for dumped IMS address spaces. When valid information has been supplied, several fields are filled in the panel, as shown in the following figure. Press PF3 to return to the primary menu.

```

Enter the IMS CTL/BATCH or DL/I jobname to cause the IMS symbols to
be set for this dump. Request subsystem list for possible IMS names.

N <==== IMS SUBSYSTEM LIST DESIRED? (Y or N)
N <==== FORMATTER REFRESH? (Y or N)

-----
JOBNAME      ID      ASID      DUMPED?
-----
CTL          IMS1      IMS1      0041      YES
DL/I         HPCICSAK      003F      YES
DBRC         DBRICSAK      0029      YES
IRLM         IRLME2N      IRLM      003D      YES
TMS

ABEND CODE = SYS 0C4      USER 0
MODULE      = UNKNOWN

IMS SDWA ADDRESS - 006728D0  IMS RELEASE - 1110
IMS SCD ADDRESS  - 00C549B8  IRLM LEVEL  - IRLM22
ABENDED ASID    - 0041

```

Figure 82. IMS dump formatting initialization/content panel - active

- IMS formatting is invoked from the high-level, low-level, and analysis option menus. Each menu contains a list of selectable entries. Type S or M next to an entry to request formatting, and press Enter

to process your selections. Examples of the high-level and low-level options menus are shown in the following panels.

### IMS High-Level Dump Formatting panel

```

----- IMS HIGH LEVEL DUMP FORMATTING OPTIONS ----- ROW 1 OF 23
Command ==>                                           Scroll ==> PAGE

N <====SPOOL OUTPUT? (Y or N)      N <====REFRESH FORMATTER? (Y or N)
      S = select      M = select,min      select choices and hit enter
                                           to process or UP/DOWN to scroll

Additional IMS format requests==>

Cmd  Option      Description
-----
-    AUTO        Internally determined options (by failing ITASK type)
-    ALL         All high level IMS dump formatting options
-    SUMMARY     PSW, regs, SAP, failing ITASK blocks at time of abend
-    SCD         SCD, SLX, FP ESCD, scheduler sequence queues
-    SAVEAREA    SAP, savearea, ECB prefix, UEHB (sorted by DSPNO)
-    DISPATCH    Dispatcher work areas, Dispatcher and Latch traces
-    SPST        System PSTs and subordinate blocks
-    RESTART     CHKPT ID table, SIDX, LCRE, RPST, RRE, EQEL, IEEQE, FRB
-    LOG         LCD, log buffer prefixes, log buffers (OLDS and MON)
-    DB          DDIRs, PDIRs, intent list, DLI and LOCK traces, DPSTs
-    DEDB        ALDS, DMCB, DMAC, XCRB, SRB, ESRB
-    MSDB        BHDR, Main storage databases
-    DC          CLB, LLB, VTCB, CTB, CNT, CTT, SMB, SPQB, LGND, USRD
-    EMH         RCTE, BALG, EMHB
-    QM          QPOOL, QSCD, QMGR hash table, QBFPFR, Queue buffers
-    UTIL        Partial reorg blocks
-    SUBS        External subsystem blocks and trace
-    CBT         Control block table
-    SDE         Storage Descriptor Element Blocks and Storage
-    SB          Sequential buffering control block formatting
-    DBRC        DBRC control blocks and trace
-    IRLM        IRLM control block formatting
-    LUM         LUM trace and control blocks
-    LR          Log router trace and control blocks
-    TMS         Transport manager control blocks
-    TMSC        Transport manager component dump formatting
-    AOI         Automated Operator Interface (Directed Message Manager)
-    OTMA        Open TM Access
-    DBRM        Database Recovery Manager
-    SMBS        All SMBs
***** Bottom of data *****

```

The IMS high-level formatter request panel facilitates the selection of IMS formatting areas. The MIN qualifier and spooling and terminal outputs can be selected as well.

## IMS Low-Level Formatting selection panel

```
DFSAAAL0 ----- IMS LOW LEVEL DUMP FORMATTING OPTIONS ----- ROW 1 OF 17
COMMAND ==>                                           Scroll ==> PAGE

N <===== SPOOL OUTPUT? (Y or N)    N <===== REFRESH FORMATTER? (Y or N)
      S or M at left plus required ARGument value to select option.
      (Items marked *P* will prompt if ARG blank). UP/DOWN to scroll

Additional IMS formatter requests==>

Cmd Option  Type      ARG      Argument description
v-----vvvvvvvv-----
-   CLB      ADDRESS    CLB/LLB address (hexadecimal)
-   CLB      NODE       VTAM node name
-   CLB      LTERM      IMS logical terminal name (CNT)
-   CLB      CID        VTAM communication ID (hexadecimal)
-   LLB      LINK       MSC link number (decimal)
-   DPST      ADDRESS    Dependent region PST address (hexadecimal)
-   DPST      NUMBER     Dependent region PST number (hexadecimal)
-   DPST      NAME       Dependent region PST jobname
-   SYSPST    ADDRESS    System PST address
-   SYSPST    NAME       *P* System PST name
-   TRACE     NAME       *P* Trace table ID (2 characters)
-   SAP       ADDRESS    Savearea block address (hexadecimal)
-   SAP       ECBADR     SAP's ECB address (hexadecimal)
-   POOL      NAME       *P* IMS storage pool name
-   CBTE      NAME       Control Block Table name
-   LUB       NAME       LU name
-   DTT       ADDRESS    DTT address
-   SMB       NAME       SMB by transaction name
-   SMB       ADDRESS    SMB address
***** Bottom of data *****
```

## IMS Dump Analysis selection panel

```
DFSAAAL0 ----- IMS DUMP ANALYSIS -----
COMMAND ==>

N <=====SPOOL OUTPUT? (Y or N)    N <=====REFRESH FORMATTER? (Y or N)

      Put an S left of desired option to select. Additional FMTIMS
      strings may be entered after "ADDITIONAL REQUESTS". Press Enter to
      process.

Additional formatting requests ==>

      analysis      output
CMD  option        description
v-----
-   SAPS           savearea set overview analysis
```

## IMS components formatting panels

The IMS components Common Queue Server (CQS), Database Recovery Facility (DBRC), IMS Connect, Open Database Manager (ODBM), Operations Manager (OM), Repository Server (REPO), Resource Manager (RM), and Structured Call Interface (SCI), run under the Base Primitive Environment (BPE) system services, rather than the IMS system services. These components use the BPE formatter, and their format options are selected separately from the IMS Dump Formatter.

### About this task

BPE direct external trace lets any component that is running under the BPE system services write ad hoc data of variable length directly to the BPE data set. To use BPE direct external trace, you must enable

it when you create the trace table definition. See "Setting up IMS for diagnostics" in *IMS Version 15.5 System Definition*.

### To access the BPE External Trace Formatting panels:

#### Procedure

1. Select Other IMS components formatting from the IMS Dump Formatting Primary menu panel, type 6 and press **Enter**.  
The IMS Components panel appears.
2. Select the component that you are formatting or select B for the BPE formatting panel, and press **Enter**.
3. Type 4 to access the BPE external trace formatting panel, and press **Enter**.  
The BPE External Trace Formatting panel opens. From this panel, you can select the specific component formatting to be done (for example, BPE or CQS). Dump initialization for these components is done through the BPE status and initialization panel under option 0 after step 2.

```
-- IMS BPE EXTERNAL TRACE FORMATTING MENU --
OPTION ==>

Input BPE trace DSN _____
Specify any BPE component. . . . . (component type or ALL)
Specify any BPE trace type . . . . . (trace type or ALL)

Select the BPE external trace record types to format or type ALL above:
Asynchronous work element (AWE) . . . . . (Y/N)
Control block storage/services (CBS) . . . . . (Y/N)
Commands trace (CMD) . . . . . (Y/N)
Dispatcher trace (DISP) . . . . . (Y/N)
Error trace table (ERR) . . . . . (Y/N)
Hash table services (HASH) . . . . . (Y/N)
Latch management and serialization (LATC) . . . . . (Y/N)
System service calls (SSRV) . . . . . (Y/N)
Storage service requests (STG) . . . . . (Y/N)
Activity related to exit routines (USRX) . . . . . (Y/N)

Optional parameters:
Start Date/Time . . . . . (Example 2002190 133500)
Stop Date/Time . . . . . (Example 2002190 133500)
UTC or Local. . . . . (U/L) Convert trace STCK. . . . . (Y/N)
```

Figure 83. BPE External Trace Formatting panel

```
-- CQS BPE EXTERNAL TRACE FORMATTING MENU --

Input BPE trace DSN _____
Specify any CQS trace type . . . . . (trace type or ALL)

Select the BPE external trace record types to format or type ALL above:
Common Queue Server Trace (CQS) . . . . . (Y/N)
CQS Error Trace (ERR) . . . . . (Y/N)
CQS Interface Trace (INTF) . . . . . (Y/N)
CQS Overflow Trace (OFLW) . . . . . (Y/N)
CQS Structure Event Trace (SEVT) . . . . . (Y/N)
CQS Structure Trace (STR) . . . . . (Y/N)

Optional parameters:
Start Date/Time . . . . . (Example 2002190 133500)
Stop Date/Time . . . . . (Example 2002190 133500)
UTC or Local. . . . . (U/L) Convert trace STCK. . . . . (Y/N)
```

Figure 84. CQS BPE External Trace Formatting panel

```

----- DBRC BPE EXTERNAL TRACE FORMATTING MENU -----
OPTION ==> -----

Input BPE trace DSN _____

Specify any DBRC trace type . . . . . _____ (trace type or ALL)

Select the BPE external trace record types to format or type ALL above:
Error Trace (ERR) . . . . . _ (Y/N)
DBRC Request Trace (RQST) . . . . . _ (Y/N)
DBRC Module Flow Trace (MODF) . . . . . _ (Y/N)
DBRC Group Service Trace (GRPS) . . . . . _ (Y/N)

Optional parameters:
Start Date/Time . . . _____ (Example 2002190 133500)
Stop Date/Time. . . _____ (Example 2002190 133500)
UTC or Local. . . . _ (U/L) _____ Convert trace STCK. . . Y (Y/N)

```

Figure 85. DBRC BPE External Trace Formatting panel

```

----- ODBM BPE EXTERNAL TRACE FORMATTING MENU -----
OPTION ==> -----

Input BPE trace DSN _____

Specify any ODBM trace type . . . . . _____ (trace type or ALL)

Select the BPE external trace record types to format or type ALL above:
Open Database Manager (ODBM) . . . . . _ (Y/N)

Optional parameters:
Start Date/Time . . . _____ (Example 2002190 133500)
Stop Date/Time. . . _____ (Example 2002190 133500)
UTC or Local. . . . _ (U/L) _____ Convert trace STCK. . . Y (Y/N)

```

Figure 86. ODBM BPE External Trace Formatting panel

```

-- OM BPE EXTERNAL TRACE FORMATTING MENU --
OPTION ==>

Input BPE trace DSN _____

Specify any OM trace type . . . . . _____ (trace type or ALL)

Select the BPE external trace record types to format or type ALL above:
Common Service Layer (CSL) . . . . . _ (Y/N)
Error Trace (ERR) . . . . . _ (Y/N)
Trace (OM) . . . . . _ (Y/N)
IMSpIplex Trace for OM (PLEX) . . . . . _ (Y/N)

Optional parameters:
Start Date/Time . . . _____ (Example 2002190 133500)
Stop Date/Time. . . _____ (Example 2002190 133500)
UTC or Local. . . . _ (U/L) _____ Convert trace STCK. . . _ (Y/N)

```

Figure 87. OM BPE External Trace Formatting panel

```

-- RM BPE EXTERNAL TRACE FORMATTING MENU --
OPTION ==>

Input BPE trace DSN _____

Specify any RM trace type . . . . . _____ (trace type or ALL)

Select the BPE external trace record types to format or type ALL above:
Common Service Layer (CSL) . . . . . _ (Y/N)
Error Trace (ERR) . . . . . _ (Y/N)
Trace (RM) . . . . . _ (Y/N)
IMSpIplex Trace for RM (PLEX) . . . . . _ (Y/N)

Optional parameters:
Start Date/Time . . . _____ (Example 2002190 133500)
Stop Date/Time. . . _____ (Example 2002190 133500)
UTC or Local. . . . _ (U/L) _____ Convert trace STCK. . . _ (Y/N)

```

Figure 88. RM BPE External Trace Formatting panel

```
-- SCI BPE EXTERNAL TRACE FORMATTING MENU --
OPTION ==>

Input BPE trace DSN _____
Specify any SCI trace type . . . . . (trace type or ALL)

Select the BPE external trace record types to format or type ALL above:
Common Service Layer (CSL). . . . . (Y/N)
Error Parameter List (ERPL) . . . . . (Y/N)
Error (ERR). . . . . (Y/N)
Interface (INTF). . . . . (Y/N)
Interface Parameter (INTP). . . . . (Y/N)
Structured Call Interface (SCI) . . . . (Y/N)
IMSpIplex Trace for SCI (PLEX). . . . . (Y/N)

Optional parameters:
Start Date/Time . . . . . (Example 2002190 133500)
Stop Date/Time. . . . . (Example 2002190 133500)
UTC or Local. . . . . (U/L) Convert trace STCK. . . _ (Y/N)
```

Figure 89. SCI BPE External Trace Formatting panel

```
-- IMS CONNECT BPE EXTERNAL TRACE FORMATTING MENU --
OPTION ==>

Input BPE trace DSN _____
Specify any IMS Connect trace type . . . . . (trace type or ALL)

Select the BPE external trace record types to format or type ALL above:
Common Trace (CMDT) . . . . . (Y/N)
Interface (ENVT). . . . . (Y/N)
IMS Connect to OTMA driver (HWSI) . . . . . (Y/N)
IMS Connect to local option driver (HWSN) . . . . (Y/N)
IMSpIplex driver and IMS Connect events (HWSO). . . (Y/N)
IMS Connect to TCP/IP driver (HWSW) . . . . . (Y/N)
SCI calls activity (OMDR) . . . . . (Y/N)
OTMA Communication driver (OTMA). . . . . (Y/N)
Local option driver (PCDR). . . . . (Y/N)
TCP/IP communication driver (TCPI). . . . . (Y/N)

Optional parameters:
Start Date/Time . . . . . (Example 2002190 133500)
Stop Date/Time. . . . . (Example 2002190 133500)
UTC or Local. . . . . (U/L) Convert trace STCK. . . _ (Y/N)
```

Figure 90. IMS Connect BPE External Trace Formatting panel

```
----- REPOSITORY SERVER BPE EXTERNAL TRACE FORMATTING MENU -----
OPTION ==>

Input BPE trace DSN _____
Specify any RS trace type. . . . . (trace type or ALL)

Select the BPE external trace record types to format or type ALL above:
Repository Server Diag (DIAG) . . . (Y/N)

Optional parameters:
Start Date/Time . . . . . (Example 2010195 164224)
Stop Date/Time. . . . . (Example 2010195 164242)
UTC or Local. . . . . U (U/L) Convert trace STCK. . . N (Y/N)
```

Figure 91. REPO BPE External Trace Formatting panel

## Related concepts

[Base Primitive Environment overview \(System Administration\)](#)



## Using the other IMS-related products formatting panels

IMS provides a selection for calling the dump formatters for products that are separate from IMS, but are still related to IMS.

### Procedure

Select Other IMS-related products formatting from the IMS dump formatting primary menu panel, type 7, and press **Enter**

You are then presented with a list of all possible products. However, you can only use the formatters of those products that are installed on your system. Each product's formatter will provide a dump initialization panel; you should not use the panel from option 0 on the primary menu.

```
DFSAAMPR ----- IMS DUMP FORMATTING PRIMARY MENU -----
OPTION  ==>

  0 INIT          - IMS formatting initialization and content summary
  1 BROWSE        - Browse Dump data set (IPCS norm)          *****
  2 HI-LEVEL      - IMS Component level formatting            *USERID  - IMSDUMP
  3 LOW-LEVEL     - IMS ITASK level formatting                 *DATE    - 00/08/23
  4 ANALYSIS      - IMS dump analysis                         *JULIAN   - 00.236
  5 USER          - IMS user formatting routines              *TIME    - 16:06
  6 OTHER COMP    - Other IMS components (BPE, CQS...)         *PREFIX   - IMSDUMP
  7 OTHER PROD    - Other IMS-related products                *TERMINAL - 3278
  E EDA           - IMS Enhanced Dump Analysis               *PF KEYS  -
  T TUTORIAL      - IMS dump formatting tutorial              *****
  X EXIT          - Exit IMS dump formatting

Enter END or RETURN command to terminate IMS component formatting.

Use PFKeys to scroll up and down if needed.
```

Figure 92. IMS Dump Formatting Primary Menu panel

## IMS IPCS symbols

IMS offline dump formatting creates IPCS symbols for selected key IMS control blocks. IMS creates and lists the IPCS symbols when the job name of an address space using BPE is supplied in the BPE initialization panel (for example, a CQS, OM, RM, or SCI address space).

The Interactive Formatter helps create these symbols and then uses them to make Offline Dump Formatter requests easier by providing known starting points, including starting points for CLISTs. The dump formatter also sets symbols for the registers (R0-R15) and PSW (DFSPSW) at abend for abend dumps. This allows you to quickly locate areas in storage pointed to by the registers and PSW when you are in IPCS browse mode.

## Using IMS enhanced dump analysis

You use the IMS Enhanced Dump formatting menu to browse and select formatting options for database, Fast Path, Transaction Manager and System dumps. There is also a tutorial available about the formatter and how to use the filtering tool.

### Procedure

1. Select option E from the IMS dump formatting primary menu.

The IMS Enhanced Dump Formatting Menu displays, as shown in the following figure.

```

----- IMS ENHANCED DUMP FORMATTING MENU
OPTION  ===>

  1 BROWSE      - Browse dump dataset (IPCS norm)
  2 DB          - Full Function Data Base
  3 FP          - Fast Path Data Base
  4 TM          - Transaction Management and DC
  5 SYS         - Systems
  6 DBRC        - Database Recovery Control
  T TUTORIAL    - IMS Dump Formatter Tutorial
  X EXIT        - Exit EDA dump formatting menu

Enter END or RETURN command to terminate IMS component

```

*Figure 93. IMS Enhanced Dump Formatting Menu*

In this panel, the control blocks are organized by function for ease of use. For example, EPST (the extended partition specification table) is located under option 3 for Fast Path.

2. To review tutorial information about the formatter and about how to use the filtering tool, type T and press **Enter**.
3. To use a filtering tool to identify filtering criteria, type on of the options 2, 3, 4, or 5 and press **Enter**.

An example of a filtering panel is shown in the following figure.

```

----- Generic Filtering Panel
-----
Explanation of the fields:
Offset  (required)  - Offset of the field in the block.
                    (hex)
Length  (default = 1) - Length of field in the control
                    block. (decimal)
Cond    (default = EQ) - Type of compare to be done. (EQ,NE,
                    GT,GE,LT,LE)
Bit     (default = N) - Should comparison be a bit mask?
                    (Y or N)
Type    (default = X) - Is the value type decimal, hex, or
                    char (D,X,A)?
Value   (required)  - Value of the field to be compared
                    at given offset.
Qual    - Qualify filter to search in
                    sub-blocks.
AND/OR  - How to combine multiple conditions.
                    If blank, only the first condition
                    will be executed.
                    (up to four conditions
allowed).
```

*Figure 94. Sample filtering panel*

4. You can overwrite the generic filtering panel default values. For example, you can select criteria that presents two separate conditions. By selecting AND you indicate that both conditions must be true
  - a) You want all the blocks starting at OFFSET 1C that have a value of X'08.'
  - b) You want all the blocks starting at OFFSET A4 that have a non-zero value.

These values are shown in the following figure.

```

<=====  AND/OR  (A/O)          QUAL =====>

```

*Figure 95. Sample filtering criteria*

## Formatting log records for use with log analysis tools

You can use the IMS Dump Formatter to recreate the final part of an IMS log from the information that is available in an IMS dump.

## About this task

The log records in the log buffers of a dump are copied to a VB file that can be analyzed by the File Select and Formatting Print utility (DFSERA10) or other log analysis tools. Log records can then be put in increasing log sequence number (LSN) order and be manipulated with the DFSERA10 utility.

## Procedure

1. Select option E (EDA - IMS Enhanced Dump Analysis) from the IMS Dump Formatting primary menu.

```

      IMS DUMP FORMATTING PRIMARY MENU
OPTION  ==> E

0  INIT          -  IMS formatting initialization and content summary
1  BROWSE        -  Browse Dump dataset
2  HI-LEVEL      -  IMS Component level formatting
3  LOW-LEVEL     -  IMS ITASK level formatting
4  ANALYSIS      -  IMS dump analysis
5  USER          -  IMS user formatting routines
6  OTHER COMP    -  Other IMS components (BPE, CQS...)
7  OTHER PROD    -  Other IMS-related products
E  EDA           -  IMS Enhanced Dump Analysis
T  TUTORIAL      -  IMS dump formatting tutorial
X  EXIT          -  Exit IMS dump formatting

Enter END or RETURN command to terminate IMS component formatting.
Use PFKeys to scroll up and down if needed.

* This product contains "Restricted Materials of IBM". 5655-C56 (C) *
* Copyright IBM Corp. 1991,2000 Licensed Materials - Property of IBM. *
* All rights reserved. U.S. government users restricted rights - use, *
* duplication, or disclosure restricted by GSA ADP schedule contract *
* with IBM Corp. Refer to copyright instructions form number G120-2083. *

```

2. Select option 5 (SYS - Systems) from the IMS Enhanced Dump Formatting menu.

```

----- IMS ENHANCED DUMP FORMATTING MENU -----
OPTION ==> 5

1 BROWSE      - Browse dump dataset (IPCS norm)          *****
2 DB          - Full Function Data Base                  *USERID   - BETHRM
3 FP          - Fast Path Data Base                     *DATE     - 08/03/05
4 TM          - Transaction Management and DC           *JULIAN    - 08.065
5 SYS         - Systems                                  *TIME     - 17:05
6 DBRC        - Database Recovery Control               *PREFIX   - BETHRM
T TUTORIAL    - IMS Dump Formatter Tutorial             *TERMINAL  - 3278
X EXIT        - Exit EDA dump formatting menu            *PF KEYS   -
                                                         *****

Enter END or RETURN command to terminate IMS component formatting.
```

3. Select **WRITE** from the **Systems Formatting** options menu.

```

. ----- SYSTEMS FORMATTING OPTIONS ----- Row 1 to 14 of 14 .
.
.      S = SELECT                               Select choice and hit enter to process.
.                                         Use UP/DOWN to scroll.
.
.
.
.  Cmd  Type      Description
.  v-----vvvvvvvv-----
.  -      BCB      BCB statistics summary
.  -      BPECSCD   BPE LFS CSCD in the IMS control region
.  -      BPEHASH   BPE LFS hash tables in the IMS control region
.  -      CDE       CDE/SDE storage List
.  -      CDECOMM   CDE/SDE storage list (common only)
.  -      CSLA      CSL anchor block formatting
.  -      DFA       Definition Anchor Block and sub-blocks
.  -      GRMB      Global RESMGR Block, trace table, and SSCTs
.  -      OCMD      OM command instance block formatting
.  -      PDIR      PSB Directory Formatting
.  -      PST       PST formatting menu
.  -      SMB       SMB formatting menu
.  -      TRC       Trace Control Blocks
.  -      WRITE     Write data to output dataset
.  ***** Bottom of data *****
.
.
.
.  COMMAND ==>                               Scroll ==> PAGE

```

The WRITE option writes the contents of the log records that are within the buffers of the dump to a data set.

## Formatting a dump for analysis

You can use the IMS Dump Formatter to format a dump for analysis.

### About this task

To format a dump for analysis:

### Procedure

1. Start IPCS with the IMS Dump Formatter.
2. Select the dump you want to browse.
3. From the IPCS Primary Option Menu, select Option 2.6, ANALYSIS.COMPONENT.
4. From the Component Analysis Panel, select DFSAAMPR.
5. From the IMS Dump Formatting Primary Option Menu, select Option 0, INIT.
6. From the IMS Dump Contents Status Panel, enter the job name of either the IMS control/batch region or the IMS DLISAS region.
7. Press the Enter key and then PF3 to return to the IMS Dump Formatting Primary Option Menu.
8. Select an option from the Primary Option Menu. HILEVEL is a good starting point because it provides a broad diagnostic overview of the problem.

## Formatting IMS dumps online

One of the tools available for problem diagnosis is the IMS formatted dump, which formats the control blocks and data areas in an IMS region.

### Abnormal Termination

When an abnormal termination occurs and dumping is to be performed, CSECT DFSABND0 gets control from the SCP and gives control to IMS routines to do the dumping. To assist you in locating areas that are dumped, eye catchers are supplied in the formatted dump. See [“Eye catchers” on page 545](#) for eye catcher examples.

**Exception:** Address spaces using BPE (for example, CQS, OM, RM, and SCI) do not provide any online dump formatting output.

## Formatted dump for the CTL address space

You can use DSECT mapping macros (when applicable) to analyze control address space areas when they are dumped. The various CTL areas are listed and described.

### CTL address space areas

The following is a list of the control address space areas that are dumped (in the order in which they are dumped) and, where applicable, the DSECT mapping macros that are most useful in analyzing them. For a list of the areas dumped when LSO=S, see [“Formatted dump for the DL/I address space” on page 570](#). Descriptive information has been added for some control blocks where it would be useful.

#### Diagnostic Area

Contains the PSW, system and user completion codes, save area ID of the module that was executing, and registers in use when abnormal termination occurred.

#### Instruction Area

Contains the area of storage from 128 bytes before to 128 bytes after the address of the failing instruction in the PSW.

#### System Diagnostic Work Area

The mapping DSECT is IHASDWA.

#### U0113 Area

Present when an abend caused the dump.

#### Referenced Sap

The mapping DSECT is ISAP.

#### System Contents Directory

The mapping DSECT is ISCD.

#### SCD Extension

The mapping DSECT is DBFESCD.

#### SCD Latch Extension

The mapping DSECT is ISCD.

#### Scheduler Sequence Queues

Controls the status of each region. The mapping DSECT is ISCD.

#### FP ESCD

The mapping DSECT is DBFESCD.

#### Control Block Table

Contains entries of control blocks that macro DFSCBTS uses for tracking. The mapping DSECT is DFSCBTS.

#### Save Area Prefix

All SAPs are SNAPed except those owned by the DL/I address space. Each SAP is followed by its save area set. At the end of this section, all of the SAP IPAGES are dumped.

#### IMS Task Dispatch Work Area

The mapping DSECT is IDSPWRK.

#### DBRC Task Dispatch Work Area

If present in the system, it is mapped.

#### IMS Control Task Dispatch Work Area

Contains the same information as the IMS log task dispatch work area.

#### Dependent Region Dispatch Work Area

For every dependent region in IMS, the dispatcher work area is mapped.

**Dispatcher Trace Data**

DSECT IDSPWRK contains the function codes associated with the dispatcher and an explanation of each code.

**Scheduler Trace Data**

Scheduler trace data is mapped by DFSSCHED. The trace entries contain scheduler function codes.

**Latch Trace Data**

The trace entries contain latch and unlatch function codes. The mapping DSECT is IDLIVSAM TRACENT.

**System PSTs**

These are system work areas for any online or batch region. The mapping DSECT is IPST.

**Checkpoint ID Table**

The mapping DSECT is BCPT.

**LCRE**

The mapping DSECT is DFSLCRE.

**SIDX**

The mapping DSECT is DFSSSIE.

**RRE**

The mapping DSECT is DFSRRE.

**Log Control Directory**

Contains information about the IMS log, for example:

- DCB1—the primary log DCB
- DCB2—the secondary log DCB (if dual logs were specified)
- Log ITASK—the status information

The mapping DSECT is LCDSECT.

**Log Buffers**

Each log buffer contains buffer information and the log control DECB. The mapping DSECT is LCDSECT.

**Log Trace**

Contains entries which show IMS internal logging activity if the log trace is active. The trace entries are described by the "IDLIVSAM TRACENT" macro.

**Open Record**

Contains the type 06 log record. The mapping DSECT is ILOGREC.

**Control Record**

Contains the type 42 log record. The mapping DSECT is ILOGREC.

**Monitor Log Directory**

Contains the same information as the log control directory and is used for logging data to the IMS Monitor data set.

**DLOG Trace Data**

Trace table used to show IMS logging activity. The mapping DSECT is ILOGREC (67FA).

**SUBS Trace Data**

Trace table used by IMS to show IMS activity in attaching or detaching subsystems. The mapping DSECT is ILOGREC (67FA).

**Global ESET Block**

The mapping DSECT is DFSGESE.

**PSB Directory**

A SNAP of the PSB directory. The mapping DSECT is PDIR.

**DMB Directory**

A SNAP of the DMB directory. The mapping DSECT is DDIR.

**Fast Path Trace**

### **Dependent Region PST**

See Dependent Region PST Formatting [“Formatted dump contents” on page 541](#) for a list of the areas formatted here.

### **OSAM I/O Control Blocks**

The system attempts to dump the IOSB and IOMA blocks.

### **Sequential Buffering Blocks**

Sequential Buffering information is grouped into the following three sections. (See the explanation of the (SB) FMTIMS option [“Solving IMS problems by using the IMS Offline Dump Formatter” on page 514](#) for a complete list of the blocks dumped in each section.)

1. Subsystem Overview for Sequential Buffering
2. PST Overview of Sequential Buffering control blocks
3. Formatted Sequential Buffering control blocks

### **DEDB Formatting**

### **Fast Path EMH Formatting**

### **Fast Path MDSB Formatting**

### **Data Communication Control Blocks**

These areas are noted in a DBCTL environment. For each CLB (line), all the control blocks associated with that line are formatted.

#### **CLB**

These areas are noted in a DBCTL environment. The mapping DSECT is ICLI CLBBASE=0.

#### **CTB**

These areas are noted in a DBCTL environment. The mapping DSECT is ICLI CTBBASE=0.

#### **Input Buffer**

These areas are noted in a DBCTL environment. A SNAP of the input buffer, if input is active.

#### **Output Buffer**

These areas are noted in a DBCTL environment. A SNAP of the output buffer, if output is active.

#### **CCB**

These areas are noted in a DBCTL environment. Present if a conversation is active or held. The mapping DSECT is ICLI CCBBASE=0.

#### **CIB**

These areas are noted in a DBCTL environment. Present if MFS is in use. The mapping DSECT is ICLI CIBBASE=0.

### **Communication Terminal Table**

These areas are noted in a DBCTL environment. Defines terminal characteristics. The mapping DSECT is ICLI CTTBASE=0.

### **SPQB Entries**

These areas are noted in a DBCTL environment. Entries on the subpool queue block chain. Unallocated CNTs are also formatted here.

### **SMB Table**

These areas are noted in a DBCTL environment. This table defines transaction characteristics in the IMS system. The mapping DSECT is IAPS SMBBASE=0.

### **Queue Manager Pool Prefix and Buffers**

These areas are noted in a DBCTL environment. The mapping DSECTs are ICLI POOLBASE=0 and ICLI BFRBASE=0.

### **Buffer Prefix List**

These areas are noted in a DBCTL environment. Contains the address of each buffer's prefix, status byte, and first and last pending and current device relative record number (DRRN).

**QPOOL Prefix**

These areas are noted in a DBCTL environment. Contains the main QPOOL prefix formatted. The mapping DSECT is QPOOL.

**IRLM Control Blocks**

The IRLM Subsystem RLMCB block are formatted here if the IMS system is running with IRLM.

**Format/Dump/Delete List**

Contains module names, module IDs, and module dump data that are not in the storage dump listing.

## Formatted dump for the DL/I address space

One of the tools available for problem diagnosis is the IMS formatted dump, which formats the control blocks and data areas in an IMS region. Dumped areas within the DL/I address space are listed and descriptive information has been added for some control blocks where it would be useful.

### Dumped areas within the DL/I address space

**System Contents Directory**

The mapping DSECT is ISCD.

**SCD Latch Extension**

The mapping DSECT is ISCD.

**Scheduler Sequence Queues**

Controls the status of each region. The mapping DSECT is ISCD.

**Save Area Trace****Save Area Prefix**

All SAPs belonging to the DL/I address space are SNAPed. A SAP is marked "ACTIVE" if the ITASK associated with it is active. Each SAP is followed by its save area set. At the end of this section, all of the SAP IPAGES are dumped.

**DLS Task Dispatch Work Areas**

The mapping DSECT is IDSPWRK.

**DBRC Task Dispatch Work Area**

If present in the system, it is mapped.

**Dependent Region Dispatch Work Area**

For every dependent region in IMS, the dispatcher work area is mapped.

**Dispatcher Trace Data**

DSECT IDSPWRK contains the function codes associated with the dispatcher and an explanation of each code.

**Latch Trace Data**

The trace entries contain latch and unlatch function codes. The mapping DSECT is IDLIVSAM TRACENT.

**System PSTs**

These are system work areas for any online or batch region. The mapping DSECT is IPST.

**PSB Directory**

A SNAP of the PSB directory. The mapping DSECT is PDIR.

**DMB Directory**

A SNAP of the DMB directory. The mapping DSECT is DDIR.

**Intent List**

This is a SNAP of the intent list.

**Partition Specification Table**

Formats the PST. The mapping DSECT is IPST.

**PDIR**

Formats the PDIR, whose address is in the PST. The mapping DSECT for PDIR is PDIR.



**PSB Prefix**

A SNAP of the PSB prefix, which contains the following:

- Index Maintenance Work Area
- Index I/O Work Area
- Segment Work Area
- I/O Work Area
- SSA Work Area
- User PARMS Area

**Buffer Handler Pool**

The system attempts to format buffer handler blocks in the order in which they are chained on the queue. However, if an error is encountered during the formatting, the entire pool is dumped as is (unchained).

The pool contains the following:

**BFSP**

Formats the buffer pool prefix. The mapping DSECT is BFSP.

**BFUS**

Formats the subpool prefix. The mapping DSECT is BFUS.

**RPLI**

Formats the DL/I RPL block. The mapping DSECT is RPLI.

**DL/I Data**

A dump of the DL/I lock activity and program isolation trace table. The mapping DSECT is IDLIVSAM TRACENT.

**Lock Activity Trace Data**

See DL/I DATA.

**Program Isolation Data**

Includes the QEL, QCB, and REQ areas. The mapping DSECT is XC00.

**OSAM Control Blocks**

The system attempts to follow the main pool, the subpool header, and the buffer prefix, and to dump the buffer. However, if an error is encountered during formatting, the entire buffer pool is SNAPed from the last valid subpool address.

The pool contains the following:

**MAINPOOL**

Formats the main pool header. The mapping DSECT is IBPOOL.

**SUBPOOL**

Formats the subpool header. The mapping DSECT is ISUBPL.

**Buffer Prefix**

Formats the buffer prefix. The mapping DSECT is IBFPRF.

**Buffer**

Physical data not mapped.

**OSAM I/O Control Blocks**

The system attempts to dump the IOSB and IOMA control blocks. The mapping DSECT is QPOOL.

**Sequential Buffering Blocks**

Sequential Buffering information is grouped into the following three sections. (See the explanation of the (SB) FMTIMS option [“Solving IMS problems by using the IMS Offline Dump Formatter”](#) on page 514 for a complete list of the blocks dumped in each section.)

1. Subsystem Overview for Sequential Buffering
2. PST Overview of Sequential Buffering control blocks
3. Formatted Sequential Buffering control blocks

## **Fast Path DEDB Formatting**

## **Fast Path EMH Formatting**

## **Fast Path MDSB Formatting**

### **IRLM Control Blocks**

The IRLM Subsystem RLMCB block is formatted here if the IMS system is running with IRLM.

### **Format/Dump/Delete List**

Contains module names, module IDs, and module dump data that are not in the storage dump listing.

## **SNAP call facility**

---

The SNAP call facility (DFSERA20) produces SNAP dumps of DL/I control blocks and identifies calling routines that generate SNAP dumps. Supervisor call (SVC) dumps are generated only for the intended abend codes or status codes, and for unknown calling routines.

The SNAP call facility produces SNAP dumps of DL/I control blocks for the following items:

- External DL/I SNAP calls. The DL/I test program, DFSDDLTO, issues SNAP calls when it detects unequal conditions based on compare statements.
- Exceptional conditions, such as pseudoabends in DL/I modules and message or batch region abends.
- Internal SNAP requests from DL/I modules.
- SNAP specific requests from other IMS modules.

GSAM modules issue SNAP calls for GSAM databases. See [“GSAM control block dump - DFSZD510” on page 230](#) for a description of the GSAM SNAP.

When a SNAP call is performed for a Fast Path region abend, module DFSERA20 bypasses some dumps:

- For a Fast Path database (an MSDB or DEDB), module DFSERA20 bypasses the DMB dump.
- For a DB PCB that refers to a Fast Path database, module DFSERA20 bypasses the DMB, DB PCB, JCB, and SDB dumps.

SNAP output consists of buffer pools and all PSB-related control blocks. Optionally, you can request subpools 0-127 in addition to the buffers and blocks.

SNAP output for exceptional conditions is always directed to the IMS log. In all other cases, IMS sends SNAP output to a data set identified on the PRINTDD DD statement. If this data set is not already open, it is opened and closed for each SNAP request. If you do not supply a PRINTDD statement, IMS sends the SNAP output to the IMS log as X'67FD' log records. When neither a SNAP data set nor the IMS log can be used for SNAPs, all SNAP actions are bypassed.

The File Select and Formatting Print utility (DFSERA10) extracts X'67FD' log records, and the exit routine (DFSERA30) formats them.

Status codes are not set for SNAP calls.

### **Related reference**

[“Log records” on page 469](#)

To diagnose some problems, you need to examine the content of log records to determine what was going on in the system before the problem occurred. By knowing the layout of the log records, you can set up a DFSERA10 job that will produce the specific log records that you need to examine.

## **/DIAGNOSE command SNAP function**

---

The **/DIAGNOSE** command SNAP function provides a non-intrusive alternative to creating a console dump. Use of this command can decrease the time that is required to generate problem determination data for IBM Software Support.

The **/DIAGNOSE** command SNAP function takes a current snapshot of system resources at any time without negatively impacting IMS. This system resource information is displayed on the issuing LTERM. Optionally, the resource information can be sent to one of the following data sets:

- Online data set (OLDS)
- SYSOUT data set
- Trace data sets (as type X'6701' log records)

The **/DIAGNOSE** command SNAP function captures information for the following resources:

- A specific IMS control block. For example, the command **/DIAGNOSE SNAP BLOCK(CSCD)** captures storage information for the APPC/OTMA SMQ SCD Extension control block.
- One of the following user-defined resources:
  - User-defined database
  - Communication line
  - Logical link
  - Logical link path
  - Node
  - Program
  - Transaction
  - Logical terminal (LTERM)
  - User
- Primary control blocks for a dependent region.
- Any area of storage within the control region address space (by specifying the address of that storage area).
- Prolog information for an IMS load module. The command **/DIAGNOSE SNAP MODULE(modname)** identifies the entry point address and captures prolog information for the specified IMS module. The prolog information contains the current maintenance level for a module on your system, which can help you to determine whether any maintenance is missing.
- A user-defined shared queues structure. The command **/DIAGNOSE SNAP STRUCTURE(structurename)** captures storage information for the DFSSQS control block storage for the specified shared queues structure.

You might also use the **/DIAGNOSE** command SNAP function to:

- Show filtered resource information captured by the SNAP function.
- Specify a limit for the number of lines of formatted SNAP data to display in response to the command.
- Specify the control blocks to be captured by the SNAP function.

The **/DIAGNOSE** command is a standard type-1 command.

### **Related reference**

[/DIAGNOSE SNAP command \(Commands\)](#)

## Type-1 trace table interface

For each trace, you can learn about the trace identifier, the events that are traced, and, if the trace is documented in this information, the topic where you can find more information. You use the trace identifier as an eye catcher to locate a trace in a dump.

The common trace table interface consists of the traces shown in the following table.

*Table 177. Trace tables in the common trace interface*

Trace	Table type	ID	Traced events	Where described
	ALL	ALL	All IMS table traces	
Common Service Layer trace	CSLT	CS	The interaction of IMS with the CSL	<a href="#">“Common Service Layer trace” on page 582</a>
DASD log trace	DLOG	DG	DASD logging	See “DASD log” in <a href="#">“IMS type-1 trace function codes” on page 577</a>
Dispatcher trace (online only)	DISP	DS	Dispatcher activities	<a href="#">“Dispatcher trace” on page 587</a>
DL/I and lock	DLI/I and LOCK	DL	DL/I calls, DL/I buffer handler, DL/I OPEN/CLOSE, Delete/Replace, HD space management, lock activity using PI or IRLM, OSAM, DFP interface, ABENDU0427	<a href="#">“DL/I trace” on page 165</a>
Enhanced Command Trace	OCMD	OC	Activity related to commands that originate from OM	Not available
External subsystem trace (online only)	SUBS	SU	Subsystem activities	<a href="#">“External subsystem trace” on page 596</a>
Fast Path Trace		FT	Fast Path activity	<a href="#">“Fast Path trace” on page 639</a>
Force trace	FORCE	FO	Internal trace for IMS initialization	Not in use
Intercommunications trace	IDCO	IC	VTAM exit activity	<a href="#">“Starting the trace” on page 279</a>
Latch trace (online only)	LATC	LA	Latch activities	<a href="#">“Latch trace” on page 630</a>
LU trace	LUMI	LU	LU 6.2 activity	<a href="#">“LU manager trace” on page 339</a>
Multiple Systems Coupling Trace	MSCT	MS	MSC activities. Not yet used.	Not applicable
ODBA trace	ODBA	OD	ODBA events	Not available
Online Recovery Manager trace	ORTT	OR	ORS activity	<a href="#">“Online Recovery Manager trace” on page 212</a>
Operations Manager (OM) commands	OCMD	OC	Activity related to commands received from OM	Not available

Table 177. Trace tables in the common trace interface (continued)

Trace	Table type	ID	Traced events	Where described
OTMA trace	OTMT	OA	OTMA activity	<a href="#">“OTMA trace” on page 365</a>
z/OS Resource Recovery Services (RRS)	RRST	RR	RRS activity in dependent region(s)	<a href="#">“Resource Recovery Services trace” on page 611</a>
Queue manager trace	QMGR	QM	Queue manager activity	<a href="#">“Queue manager trace” on page 634</a>
Security trace	SECT	SE	Security events	Not available
Scheduler trace (online only)	SCHD	SC	Scheduler activities	<a href="#">“Scheduler trace” on page 626</a>
Shared queues interface trace	SQTT	SQ	Shared queues interface activities.	<a href="#">“Shared queues interface trace” on page 638</a>
Storage Manager trace	STRG	SM	Storage Manager activities	<a href="#">“Storage manager trace” on page 629</a>

#### Related concepts

[“DL/I trace” on page 165](#)

The DL/I trace table is a combined trace consisting of entries from DL/I calls, the DL/I buffer handler, DL/I OPEN/CLOSE, HD space management, lock activity (using PI or IRLM), OSAM, DFP interface, HALDB OLR trace, and ABENDU0427.

## Finding type-1 trace tables in a dump

If you do not write the trace to the log data set, IMS formats trace tables as part of an IMS dump.

### How to locate trace tables

The following figure shows where to find the location of each of the traces in a dump.

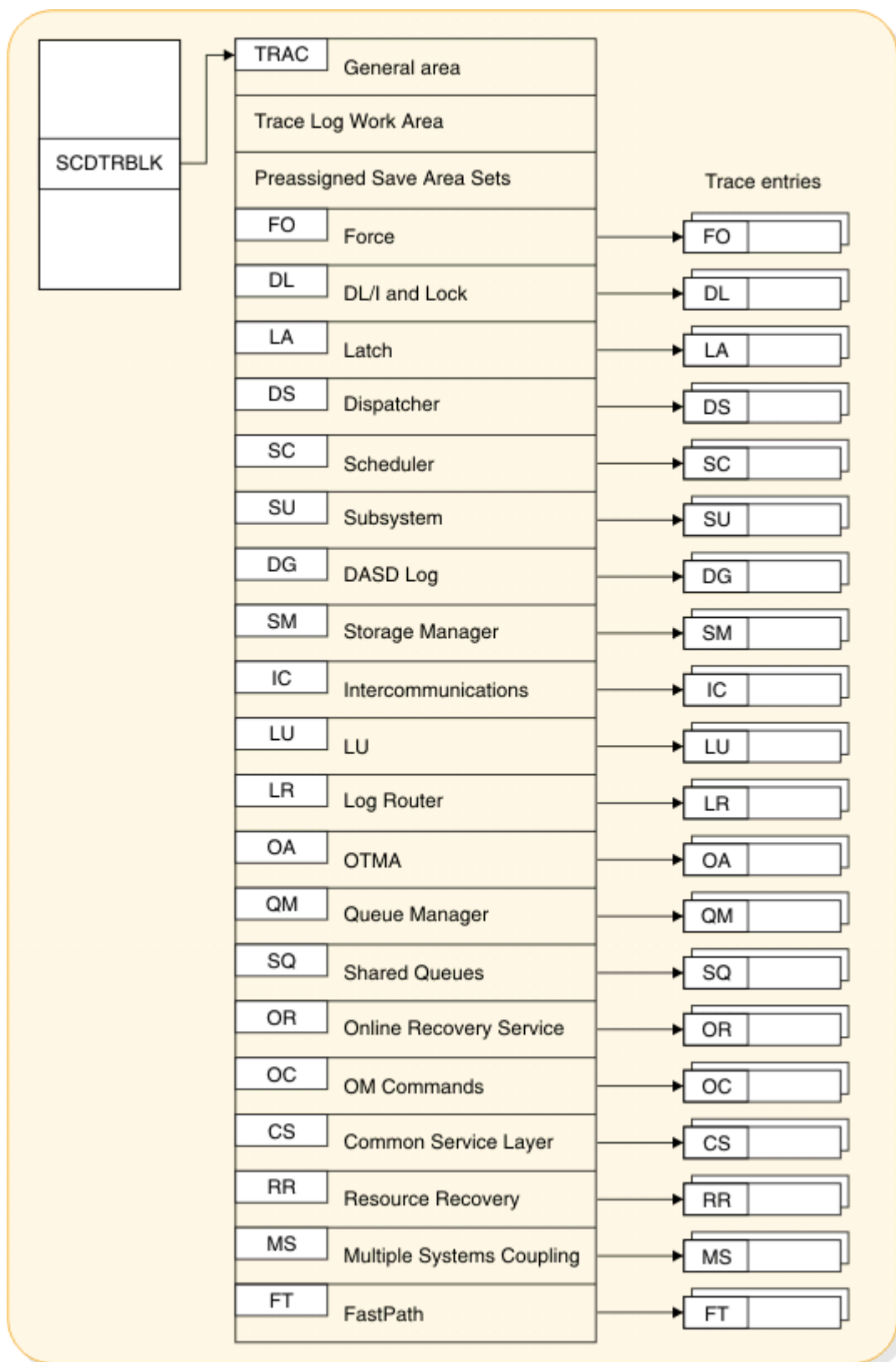


Figure 96. How to locate trace tables

# Format of type-1 trace records

By examining the trace records, you can determine the function that was being traced as well as the order in which a series of system operations took place.

In the example trace record in the following figure, the number in the trace sequence field in each entry identifies where that trace entry fits in the sequence of system operations. In addition, each trace entry provides pertinent information about that function.

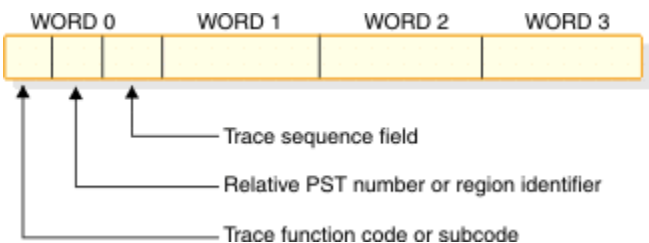


Figure 97. General trace record format

You can find the format of the trace entries by assembling macro IDLIVSAM TRACENT. Assembling IDLIVSAM after each system definition ensures that you have a current mapping of the trace record formats.

# IMS type-1 trace function codes

You use trace function codes to help you diagnose performance problems or other problems with IMS.

The following table shows some of the important functions that are traced by the common trace interface and their location in the trace tables. These function codes are a subset of all trace function codes.

You can also find a one-line description of each trace code in macro DFSTRAE0.

Table 178. Trace function codes		
Trace table	Function code	Description
DL/I and lock	X'0C'	DL/I OPEN/CLOSE for each data set
	X'30'	IWAIT called with IXCTL=YES option
	X'31'	Get space for the segment
	X'32'	Free space for the segment
	X'34'	Get space close to root anchor
	X'35'	HD space management GET /ERE local serialization lock
	X'36'	HD space management release local serialization lock /ERE
	X'60'	(OSAM) I/O operation initiated
	X'61'	(OSAM) I/O operation posted
	X'62'	(OSAM) OPEN/CLOSE/EOV complete
	X'69'	Sequential buffering: invalidate SB buffers
	X'6A'	Sequential buffering: buffering evaluation
	X'6B'	Sequential buffering: description why SB was/was not used
	X'6C'	Sequential buffering: refresh SB buffers after a write
	X'6F'	Sequential buffering: search/read call issued by OSAM Buffer Handler

Table 178. Trace function codes (continued)

Trace table	Function code	Description
	X'80'	Database authorization request
	X'81'	Database change authorization request
	X'82'	Database re-authorization request
	X'AA'	DL/I call analyzer entry for each database call
	X'AB'	(VSAM) ABEND U0427
	X'B1'	Demand space set by backout or DELETE/REPLACE
	X'B2'	Free space for backout
	X'C4'	DELETE/REPLACE
	X'C7'	(PI) Exclusive control deadlock detection
	X'C8'	Lock request manager (DFSLMGR0) entry
	X'C9'	Lock request manager (DFSLMGR0) exit
	X'CA'	(PI) request trace entry
	X'CA'—X'08'	(PI) DL/I call trace entry
	X'CB'	(PI) lock elapsed time entry
	X'CC'	Lock request handler (DFSLRH00)
	X'CF'	I/O Toleration (DFSTOPR0)
	X'D0'	IRLM NOTIFY sent
	X'D1'	IRLM NOTIFY received
	X'D2'	IRLM status exit
	X'D3'	IRLM deadlock exit
	X'D5'	Sysplex data sharing
	X'D9'	HALDB online reorganization trace entry
	X'DA'	VSAM JRNAD or UPAD exit
	X'DB'	Search pool for record in range (buffer handler)
	X'DD'	Release record ownership (buffer handler)
	X'DE'	Retrieve buffer pool statistics (buffer handler)
	X'DF'	VSAM verify
	X'E0'	VSAM PUT
	X'E1'	Block locate (buffer handler)
	X'E2'	Byte locate (buffer handler)
	X'E4'	Create new ESDS/OSAM LRECL (buffer handler)
	X'E5'	Write LRECLs for user (purge) (buffer handler)
	X'E6'	Mark record altered (buffer handler)
	X'E9'	Free space in buffer pool (BFPL) (buffer handler)
	X'EA'	Perform background write function (buffer handler)
	X'EB'	Byte locate and mark altered (buffer handler)



Table 178. Trace function codes (continued)

Trace table	Function code	Description
Dispatcher	X'EC'	Mark buffers empty (BFPL) (buffer handler)
	X'ED'	Checkpoint (buffer handler)
	X'EE'	Batch STAE purge at ABEND (buffer handler)
	X'EF'	OSAM buffer forced write (buffer handler)
	X'F0'	Retrieve first LRECL by key (buffer handler)
	X'F1'	Erase logical record (buffer handler)
	X'F2'	Retrieve by key EQ or GT (buffer handler)
	X'F3'	Retrieve key EQ or GT—Repair CI (buffer handler)
	X'F4'	Retrieve by key record to chain from insert logical record (KSDS) (buffer handler)
	X'F8'	Retrieve next sequential root by key (buffer handler)
	X'F9'	Position by key for image copy (buffer handler)
	X'FA'	Get next record for image copy (buffer handler)
	X'01'	FRR driven attempting to SCHEDULE a RESUME SRB in IPOST common (DFSIPOTC)
	X'02'	ITASK started (created)
	X'03'	ITASK terminated
	X'04'	IWAIT called
	X'05'	ITASK reinstated
	X'06'	IPOST called
	X'07'	IXCTL called
	X'08'	ISWITCH 'TO' invoked
	X'09'	Un-initialize ECB called
	X'0A'	Dependent region dispatch reattach
	X'0B'	Process IMS TCB signoff
	X'0C'	Reserved — used by DL/I Open Close
	X'0D'	INITECB called
	X'0E'	Memory change done using PC/PT
	X'0F'	Dispatcherabend issued
	X'10'	Cross memory ISWITCH TO=XM or TO=HOME
	X'11'	Cross memory state change
	X'12'	DFSKPXT store POST code in ECB
	X'13'	DFSKPXT called (z/OS branch-entry local POST)
	X'14'	DFSCIR called to create an ITASK
	X'15'	DFSKPXT issued z/OS branch-entry local POST
	X'16'	Post exit posted ECB enqueue
	X'17'	Post exit resume target IMS TCB

Table 178. Trace function codes (continued)

Trace table	Function code	Description
	X'18'	IPOST common store post code in ECB
	X'19'	IPOST common posted ECB enqueue
	X'1A'	IPOST common resume target IMS TCB
	X'1B'	INITECB ECB store results
	X'1C'	INITECB posted ECB enqueue
	X'1D'	Suspend back out resume issued
	X'1E'	SRB scheduled for alternate IPOST
	X'1F'	IPOST called ('SAP=')
	X'20'	Dependent region shutdown ISWITCH
	X'21'	Entry to POST-Exit routine
	X'22'	Reserved
	X'23'	ISERWAIT called
	X'24'	ISWITCH 'TO' with stack invoked
	X'25'	Reserved
	X'26'	Branch entry SCP post
	X'27'	Suspend IMS TCB
	X'28'	Dependent region open dispatcher — sign on
	X'29'	ISWITCH TO=UNSTACK
	X'2A'	IMS list post called
	X'2B'	SCP WAIT issued
	X'2C'	SCP WAIT completed
	X'2D'	ISWITCH 'RET' invoked
	X'2E'	Shutdown ISWITCH reinstated
	X'2F'	Dependent region open dispatcher — TCB switch
z/OS Resource Recovery Services (RRS)	X'A5'	RRS calls
Scheduler	X'41'	Scheduling starts
	X'42'	Block mover
	X'43'	Scheduling ends
	X'44'	IRC started
	X'45'	TMS00 started
	X'46'	TMS00 finished
	X'47'	APPC extract call made
	X'48'	Scheduling failed
Queue Manager	X'4E'	Information related to the queue manager
DASD log	X'50'	Logical logger trace entry.

Table 178. Trace function codes (continued)

Trace table	Function code	Description
	X'51'	Physical logger master ITASK trace entry
	X'52'	Physical logger buffer ITASK trace entry
	X'53'	Physical logger setup ITASK trace entry
	X'54'	Physical logger WADS ITASK trace entry
	X'55'	Physical logger READ ITASK trace entry
External	X'57'	Created by the module that operates in the IMS control region
Subsystem	X'58'	Created by the module that operates in the IMS dependent region
Storage Manager	X'5F'	Storage Manager trace entry written on pool allocation Buffer Get and Buffer release (AOIP, CESS, CIOP, CMDP, DYNP, EMHB, FPWP, HIOP, SPAP, LUMC, LUMP)
Latch	X'70'	Information related to the latch manager and the use manager
	X'76'	Reserved
Fast Path	X'9C'	The FP Notify trace code
	X'9D'	The FP General trace code

#### Related concepts

[“Dispatcher trace” on page 587](#)

When you use the **/TRACE SET ON TABLE DISP** command, IMS enables the dispatcher trace to an internal table. This internal table is formatted in any dump that is formatted by IMS.

[“Latch trace” on page 630](#)

When you use the **/TRACE SET ON TABLE LATC** command, IMS traces events related to its internal serialization services (latch manager, use manager, and system locate control function) to an internal table.

#### Related tasks

[“Resource Recovery Services trace” on page 611](#)

The Resource Recovery Services trace (RRST) provides information about relevant z/OS Resource Recovery Services (RRS) events in the IMS dependent region. Use the trace under direction of IBM Software Support when problems are suspected in the RRS area.

#### Related reference

[“External subsystem trace” on page 596](#)

You enable the external subsystem trace by using the **/TRACE SET ON TABLE SUBS** command. When you specify **OPTION LOG**, IMS writes the trace externally as type X'67FA' records.

[“Scheduler trace” on page 626](#)

When you use the **/TRACE SET ON TABLE SCHD** command, IMS enables the scheduler trace. When you specify **OPTION LOG**, IMS sends these entries to the log as type X'67FA' records.

[“Storage manager trace” on page 629](#)

The storage manager trace writes a record each time it is called to allocate a pool, get a buffer, or release a buffer. The storage manager traces requests from the following pools: AOIP, CESS, CIOP, CMDP, DYNP, EMHB, FPWP, HIOP, LUMC, LUMP, and SPAP.

[“Queue manager trace” on page 634](#)

The queue manager trace provides information about relevant queue manager functional and exceptional events. Use the trace under the direction of IBM support personnel when problems are suspected in the queue manager area.

[“Shared queues interface trace” on page 638](#)

The shared queues interface trace provides information about errors associated with the interface between IMS and CQS.

[“Fast Path trace” on page 639](#)

When you use the **/TRACE SET ON TABLE FPTT** command, IMS enables the Fast Path trace. The Fast Path trace resides in the internal IMS trace tables, with the **OPTION LOG** parameter causing the trace to also be written to the IMS logs.

## Common Service Layer trace

The Common Service Layer trace (CSLT) provides information about the interactions between IMS and the Common Service Layer, including how IMS interacts with the Operations Manager (OM), the Resource Manager (RM), and the Structured Call Interface (SCI).

You can turn on the Common Service Layer trace during online operation by using the **/TRACE** command. Each trace entry is X'20' bytes long. You can specify trace output destination and tracing volume on the **/TRACE** command.

If you send the output to the common trace table, you can format the table using the Offline Dump Formatter under IPCS, using either the VERBX command or the IMS Dump Formatter panels. If you send the output to an external data set, you can use the File Select and Formatting Print utility (DFSERA10) with exit routine DFSERA60 to format the trace entries.

To locate the common service layer trace in a dump, look for eye catcher \*\*CSTR.

## Format of Common Service Layer trace records

The Common Service Layer trace function and subfunction codes are listed, and the trace (low level) record format these functions with these subfunction codes (SC) are shown.

*Table 179. Common Service Layer trace function and subfunction codes*

Subfunction codes (SC)	Function
X'01'	Process flow
X'02'	Storage error
X'03'	Load or delete error
X'05'	Parameter validation error
X'09'	AWE error
X'0A'	Latch error
X'0F'	Miscellaneous MVS service error
X'11'	CQS service error
X'12'	SCI service error
X'13'	RM service error
X'14'	OM service error
X'15'	Unknown function
X'16'	Parameter list version error
X'50'	SCI Input exit

Table 179. Common Service Layer trace function and subfunction codes (continued)

Subfunction codes (SC)	Function
X'51'	SCI Notify exit
X'80'	Terminal logon
X'81'	Terminal logoff
X'82'	User signon
X'83'	User signoff
X'84'	DFSRMAM0 query interface
X'85'	DFSRMUP0 update interface
X'86'	RM resource entry
X'90'	Miscellaneous RM directive processing errors

### Subfunction code trace record format

The following diagram shows the format of the trace records for each of the subfunction codes listed above. Each trace record has a trace function code of X'A2' and is X'20' bytes long.

```

Word 0    - byte 1 - Trace function code
           - byte 2 - Trace function subcode
           - byte 3-4 - Trace record sequence number
Words 1-5 - Contains information about the activity being traced.
           The information recorded in this part of the trace record
           depends on the trace function subcode of the trace record.
Words 6-7 - Timestamp (STCK value)

```

The data in words 1-5, which is specific to each trace entry, is described in the following diagrams:

```

Trace function subcode = X'01'
Description:  Process flow (Begin Process and Normal Process)

```

```

Word 1      - byte 1 - Service code
             - byte 2 - Object type
             - bytes 3-4 - Module identifier
Word 2      - Not used
Word 3      - Not used
Word 4      - Not used
Word 5      - Thread ECB address

```

```

Trace function subcode = X'01'
Description:  Process flow (End Process)

```

```

Word 1      - byte 1 - Service code
             - byte 2 - Object type
             - bytes 3-4 - Module identifier
Word 2      - Not used
Word 3      - Not used
Word 4      - Return code
Word 5      - Reason code

```

```

Trace function subcode = X'02'
Description:  Storage Request Error

```

```

Word 1      - byte 1 - Service code
             - byte 2 - Object type
             - bytes 3-4 - Module identifier
Word 2      - Return code
Word 3      - Storage length
Word 4      - Storage address
Word 5      - Thread ECB address

```

```

Trace function subcode = X'03'
Description:  Module LOAD/DELETE Error

```

Word 1	- byte 1 - Service code
	- byte 2 - Object type
	- bytes 3-4 - Module identifier
Word 2	- Return code
Word 3	- bytes 1-2 - Target module identifier
	- bytes 3-4 - Not used
Word 4	- Not used
Word 5	- Thread ECB address

Trace function subcode = X'04'  
Description: Proclib/Execute Parameter Error

Word 1	- byte 1 - Service code
	- byte 2 - Object type
	- bytes 3-4 - Module identifier
Word 2	- Return code
Word 3	- Not used
Word 4	- Not used
Word 5	- Thread ECB address

Trace function subcode = X'05'  
Description: Parameter Validation Error

Word 1	- byte 1 - Not used
	- byte 2 - Object type
Words 2-5	- Parameter Value

Trace function subcode = X'07'  
Description: TCB/Thread Error

Word 1	- byte 1 - Service code
	- byte 2 - Object type
	- bytes 3-4 - Module identifier
Word 2	- Return code
Word 3	- Not used
Word 4	- Not used
Word 5	- Thread ECB address

Trace function subcode = X'09'  
Description: AWE Error (Create AWE Queue Server, Get AWE, Enq AWE)

Word 1	- byte 1 - Service code
	- byte 2 - Object type
	- bytes 3-4 - Module identifier
Word 2	- Return code
Word 3	- Not used
Word 4	- Not used
Word 5	- Thread ECB address

Trace function subcode = X'09'  
Description: AWE Error (Invalid AWE)

Word 1	- byte 1 - Service code
	- byte 2 - Object type
	- bytes 3-4 - Module identifier
Word 2	- byte 1 - Function code
	- bytes 2-4 - Not used
Word 3	- Address of invalid AWE
Word 4	- Enqueueer's ECB
Word 5	- Thread ECB address

Trace function subcode = X'0A'  
Description: LATCH Error

Word 1	- byte 1 - Service code
	- byte 2 - Object type
	- bytes 3-4 - Module identifier
Word 2	- Return code
Word 3	- Not used
Word 4	- Not used
Word 5	- Thread ECB address

Trace function subcode = X'0F'  
Description: Miscellaneous MVS Service Error

Word 1	- byte 1 - Service code
	- byte 2 - Object type
	- bytes 3-4 - Module identifier
Word 2	- Return code
Word 3	- Reason code
Word 4	- Not used
Word 5	- Thread ECB address

Trace function subcode = X'11'  
Description: CQS Service Error

Word 1	- byte 1 - Service code
	- byte 2 - Object type
	- bytes 3-4 - Module identifier
Word 2	- Return code
Word 3	- Reason code
Word 4	- Not used
Word 5	- Thread ECB address

Trace function subcode = X'12'  
Description: SCI Service Error

Word 1	- byte 1 - Service code
	- byte 2 - Object type
	- bytes 3-4 - Module identifier
Word 2	- Return code
Word 3	- Reason code
Word 4	- Not used
Words 4-5	- Target member name or zero

Trace function subcode = X'13'  
Description: RM Service Error

Word 1	- byte 1 - Service code
	- byte 2 - Object type
	- bytes 3-4 - Module identifier
Word 2	- Return code
Word 3	- Reason code
Word 4	- Not used
Words 4-5	- Target member name or zero

Trace function subcode = X'14'  
Description: OM Service Error

Word 1	- byte 1 - Service code
	- byte 2 - Object type
	- bytes 3-4 - Module identifier
Word 2	- Return code
Word 3	- Reason code
Word 4	- Not used
Words 4-5	- Target member name or zero

There are two formats used for Trace Subcode X'15':

Trace function subcode = X'15'  
Description: Unknown Function Exit Errors

Word 1	- bytes 1-2 - Function Code
	- bytes 3-4 - Module identifier
Words 2-5	- SCI Token

Trace function subcode = X'15'  
Description: Unknown Function Exit Errors

Word 1	- bytes 1-2 - Function Code
	- bytes 3-4 - Module identifier
Words 2-3	- Subject member name
Words 4-5	- Subject member type and subtype

Trace function subcode = X'16'  
Description: Parameter list version errors

Word 1	- byte 1 - Not used
--------	---------------------

Words 2	- byte 2 - Object type
Words 3-4	- bytes 3-4 - Module identifier
Word 5 -	- Parameter version
	- Member name
	- Member version
Trace function subcode = X'50'	
Description: SCI Input Exit	
Word 1	- byte 1 - Service code
	- byte 2 - Flag
	- bytes 3-4 - Source member type
Word 2	- Function code
Word 3	- Subfunction code
Words 4-5	- Source member name
Trace function subcode = X'51'	
Description: SCI Notify Exit	
Word 1	- byte 1 - Service code
	- byte 2 - Flag
	- bytes 3-4 - Source member type
Word 2	- Source member type
Word 3	- Event
Words 4-5	- Source member name
Trace function subcode = X'80'	
Description: Logon Process	
Word 1	- bytes 1-2 - Return code
	- byte 3 - CLBSRM1
	- byte 4 - CLBSRM2
Words 2-3	- Node name
Word 4	- Not used
Word 5	- Thread ECB address
Trace function subcode = X'81'	
Description: Logoff Process	
Word 1	- bytes 1-2 - Return code
	- byte 3 - CLBSRM1
	- byte 4 - CLBSRM2
Words 2-3	- Node name
Word 4	- Not used
Word 5	- Thread ECB address
Trace function subcode = X'82'	
Description: Signon Process	
Word 1	- bytes 1-2 - Return code
	- byte 3 - CLBSRM1
	- byte 4 - CLBSRM2
Words 2-3	- User structure name
Word 4	- Not used
Word 5	- Thread ECB address
Trace function subcode = X'83'	
Description: Signoff Process	
Word 1	- bytes 1-2 - Return code
	- byte 3 - CLBSRM1
	- byte 4 - CLBSRM2
Words 2-3	- User structure name
Word 4	- Not used
Word 5	- Thread ECB address
Trace function subcode = X'84'	
Description: DFSRMAM0 query interface	
Word 1	- byte 1 - RMAP flag 1
	- byte 2 - RMAPE flag 1
	- byte 3 - RMAPE flag 2
	- byte 4 - RMAPE flag 3
Words 2-3	- Resource name



Word 4	- Data pointer
Word 5	- Return code

Trace function subcode = X'85'	
Description: DFSRMUP0 update interface	
Word 1	- byte 1 - RMAP flag 1
	- byte 2 - RMAPE flag 1
	- byte 3 - RMAPE flag 2
	- byte 4 - RMAPE flag 3
Word 2	- Resource pointer
Word 3	- Not used
Word 4	- Data pointer
Word 5	- Return code

Trace function subcode = X'86'	
Description: RM Resource Entry trace	
Word 1	- byte 1 - Service code
	- byte 2 - Condition code
	- bytes 3-4 - Module identifier
Words 2-3	- Resource name
Word 4	- byte 1 - Resource Type
	- byte 2 - Not used
	- byte 3 - Input version number (last byte)
	- byte 4 - Output version number (last byte)
Word 5	- Thread ECB address

Trace function subcode = X'90'	
Description: Miscellaneous Directive Processing errors	
Word 1	- byte 1 - Service code
	- byte 2 - Not used
	- bytes 3-4 - Module identifier
Words 2-3	- Process name
Word 4	- Process type
Word 5	- Not used

## Dispatcher trace

When you use the **/TRACE SET ON TABLE DISP** command, IMS enables the dispatcher trace to an internal table. This internal table is formatted in any dump that is formatted by IMS.

When you use **OPTION LOG**, IMS sends the entries to the log as type X'67FA' records. You can select and format these log entries by using the utility **DFSERA10** with exit **DFSERA30**.

The following table shows the general format of a dispatcher trace entry.

*Table 180. Dispatcher trace record format*

WORD 0			WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7
I	T	SEQ NUM							TIME STAMP

where

**represents**

**I**

One-byte trace ID field. This byte indicates the type of the trace entry.

**T**

One-byte TCB ID. This byte indicates the IMS TCB type which made the trace entry.

The dispatcher trace formatting usually includes the functional area. If you need this information because the trace is in a raw format, the codes can be obtained by assembling the following macro statement **DFSKDT FUNC=EQUATES**.

**SEQ NUM**

Two-byte trace sequence number assigned by the IMS trace component.

## TIME STAMP

Bytes 3 through 6 of the system clock (STCK) at the time the trace entry was created.

Words 1 through 6 contain data specific to each trace entry, as described below: The letter A followed by parentheses () indicates "address of" in all dispatcher trace entries listed below.

**TRACE ID** = X'01'  
**DESC** = FRR driven attempting to schedule a RESUME SRB  
in IPOST common (DFSIPOTC)  
word 1 - A(Target ECB being IPOSTed). If high X'80' on,  
this indicates recursive FRR entry  
word 2 - SAPCNTRL field from target ECB's SAP  
word 3 - Abend code  
word 4 - A(target dispatcher work area)  
word 5 - IPOST common caller's return address  
word 6 - IPOST common caller's R13

**TRACE ID** = X'02'  
**DESC** = ECB dispatch - ITASK started (created)  
word 1 - A(ITASK ECB)  
word 2 - ECB contents  
word 3 - A(ITASK SAP)  
word 4 - EPFFLAGS field from ECB prefix  
word 5 - A(CULE) if present in ECB prefix  
word 6 - A(Routine to get control)

**TRACE ID** = X'03'  
**DESC** = ECB dispatch - ITASK terminated  
word 1 - A(ITASK ECB)  
word 2 - ECB contents  
word 3 - A(ITASK SAP)  
word 4 - EPFFLAGS field from ECB prefix  
word 5 - A(CULE) if present in ECB prefix  
word 6 - 0

**TRACE ID** = X'04'  
**DESC** = IWAIT called  
word 1 - A(ITASK ECB)  
word 2 - ECB contents prior to IWAIT  
word 3 - IWAIT return address  
word 4 - 0  
word 5 - 0  
word 6 - SAPCNTRL contents

**TRACE ID** = X'05'  
**DESC** = ECB dispatch - ITASK reinstated  
word 1 - A(ITASK ECB)  
word 2 - ECB contents  
word 3 - SAPCNTRL field from ITASK's SAP  
word 4 - EPFFLAGS field from ECB prefix  
word 5 - Reinstated address (return address)  
word 6 - 0

**TRACE ID** = X'06'  
**DESC** = IPOST called  
word 1 - A(POSTer's ECB) (A(TCB) if ITASK=NO)  
word 2 - IPOST return  
word 3 - A(ECB to be POSTed)  
word 4 - Contents of ECB before IPOST  
word 5 - POST code at entry to IPOST (may be complimented)  
word 6 - 0

**TRACE ID** = X'07'  
**DESC** = IXCTL called  
word 1 - A(Current ITASK ECB)  
word 2 - A(IXCTL target ECB)  
word 3 - IXCTL return address  
word 4 - A(CULE) from current ECB prefix

word 5 – 0  
word 6 – 0

**TRACE ID** = X'08'  
**DESC** = ISWITCH TO= invoked

word 1 – A(Current ECB)  
word 2 – ISWITCH return address  
word 3 – A(target dispatcher work area)  
word 4 – SAPCNTRL field from ECB's SAP  
word 5 – SAPXFLAG contents  
word 6 – 0

**TRACE ID** = X'09'  
**DESC** = UN-INITIALIZE ECB called

word 1 – A(Target ECB)  
word 2 – UNINIT return address  
word 3 – UNINIT return code  
word 4 – EPFFLAGS from ECB prefix  
word 5 – ECB contents  
word 6 – 0

**TRACE ID** = X'0A'  
**DESC** = Dependent region reattach

word 1 – A(Related PST)  
word 2 – A(Dependent region dispatcher work area)  
word 3 – SAPCNTRL field from PST's SAP  
word 4 – 0  
word 5 – 0  
word 6 – 0

**TRACE ID** = X'0B'  
**DESC** = Process IMS TCB signoff

word 1 – A(Related PST)  
word 2 – A(Released dispatcher work area)  
word 3 – Signoff return address  
word 4 – 0  
word 5 – 0  
word 6 – 0

**TRACE ID** = X'0D'  
**DESC** = INITECB called

word 1 – A(Current ECB)  
word 2 – INITECB return address  
word 3 – A(ECB being initialized)  
word 4 – Contents of ECB before being initialized  
word 5 – INITECB RC  
word 6 – If INITECB RC=0C, WD5 = A(SAP) from target ECB prefix.  
If INITECB RC=10, WD5 = A(dispatcher work area)  
from target ECB prefix.

**TRACE ID** = X'0E'  
**DESC** = Memory change done via PC/PT

word 1 – A(Current ECB) (X'80' on=PC; off=PT)  
word 2 – Old primary ASID | Secondary ASID  
word 3 – If Word 1 indicates PT: PKM ASID for PT  
If Word 1 indicates PC: PC # issued  
word 4 – A(Current dispatcher work area)  
word 5 – New Primary ASN-Second-Table-Entry Instance Number  
(PASTEIN), or 0 if none.  
word 6 – High half word = 0. Low half word = old PKM.

**TRACE ID** = X'0F'  
**DESC** = Dispatcher ABEND issued ("other diagnostics"  
dependent on ABEND issuer)

word 1 – A(Current ECB)  
word 2 – Other diagnostics  
word 3 – ABEND code | reason code  
word 4 – Other diagnostics (usually the dispatcher work area  
address of the abending TCB)

word 5 – Other diagnostics  
word 6 – Other diagnostics

**TRACE ID** = X'10'  
**DESC** = Cross memory ISWITCH TO=XM or TO=HOME

word 1 – A(Current ECB)  
word 2 – ISWITCH return address  
word 3 – Target code (00=HOME, 01=CTL, 02=DLI)  
word 4 – SAPCNTRL field from ECB's SAP  
word 5 – Home ASID of target | Primary ASID of target  
word 6 – SAPXFLAG contents

**TRACE ID** = X'11'  
**DESC** = Cross memory state change

word 1 – A(Current ECB)  
word 2 – Old primary ASID | Secondary ASID  
word 3 – New primary ASID | Secondary ASID  
word 4 – A(current dispatcher work area)  
word 5 – High half word = low 16 bits of new Primary ASN-  
Second-Table-Entry Instance Number (PASTEIN), or  
zero if none. Low half word = low 16 bits of new  
Secondary ASN-Second-Table-Entry Instance Number  
(SASTEIN), or zero if none.  
word 6 – High half word = New PSW Key Mask (PKM) . Low half  
word = Old PKM.

**TRACE ID** = X'12'  
**DESC** = DFSKPXT-POST code stored in ECB (ECB was not waiting)

word 1 – A(ECB) to be POSTed  
word 2 – POST code  
word 3 – Contents of ECB on prior to store  
word 4 – 0  
word 5 – 0  
word 6 – 0

**TRACE ID** = X'13'  
**DESC** = DFSKPXT-Special MVS branch-entry POST call

word 1 – A(Caller's TCB) (0 if SRB)  
word 2 – Caller's return address  
word 3 – A(ECB) to be POSTed  
word 4 – Caller's home ASID  
word 5 – 0  
word 6 – 0

**TRACE ID** = X'14'  
**DESC** = DFSCIR called to create an ITASK

word 1 – A(ECB) or -A(ECB list)  
word 2 – ITASK type code  
word 3 – DFSCIR return address  
word 4 – A(ITASK main program)  
word 5 – 0  
word 6 – 0

**TRACE ID** = X'15'  
**DESC** = DFSKPXT issued branch-entry MVS POST (local)

word 1 – A(ECB) to be POSTed  
word 2 – ECB POST code  
word 3 – ECB contents prior to the POST  
word 4 – 0  
word 5 – 0  
word 6 – 0

**TRACE ID** = X'16'  
**DESC** = POST exit POSTed ECB enqueue

word 1 – A(ECB) being POSTed  
word 2 – ECB POST code  
word 3 – Previous POST queue header contents  
word 4 – 0

word 5 - 0  
word 6 - 0

**TRACE ID** = X'17'  
**DESC** = POST exit RESUME target MVS TCB

word 1 - A(TCB) (SRB=0)  
word 2 - Home ASID | Primary ASID  
word 3 - Target TCB's ASID  
word 4 - 0  
word 5 - 0  
word 6 - 0

**TRACE ID** = X'18'  
**DESC** = IPOST common store POST code in ECB (ECB was not waiting)

word 1 - A(ECB) being IPOSTed  
word 2 - POST code  
word 3 - ECB contents prior to the IPOST  
word 4 - A(ECB's dispatcher work area)  
word 5 - IPOST common caller's return address  
word 6 - 0

**TRACE ID** = X'19'  
**DESC** = IPOST common POSTed ECB enqueue

word 1 - A(ECB) being enqueued  
word 2 - ECB POST code  
word 3 - Previous POSTed queue header contents  
word 4 - A(ECB's dispatcher work area)  
word 5 - IPOST common caller's return address  
word 6 - 0

**TRACE ID** = X'1A'  
**DESC** = IPOST common RESUME target IMS TCB

word 1 - A(current TCB) (0=SRB)  
word 2 - Home ASID or Primary ASID  
word 3 - Target TCB's home ASID  
word 4 - A(resumed TCB's dispatcher work area)  
word 5 - 0  
word 6 - 0

**TRACE ID** = X'1B'  
**DESC** = INITECB ECB store results

word 1 - A(ECB) being initialized  
word 2 - WAIT code being stored into ECB  
word 3 - ECB contents prior to INITECB store  
word 4 - 0  
word 5 - 0  
word 6 - 0

**TRACE ID** = X'1C'  
**DESC** = INITECB POSTed ECB enqueue

word 1 - A(ECB) being initialized  
word 2 - ECB POST code  
word 3 - Previous POSTed queue header contents  
word 4 - 0  
word 5 - 0  
word 6 - 0

**TRACE ID** = X'1D'  
**DESC** = SUSPEND back out RESUME issued

word 1 - POSTed queue header contents  
word 2 - Home ASID | Primary ASID  
word 3 - A(SRB) (0 = no SRB)  
word 4 - 0  
word 5 - 0  
word 6 - 0

**TRACE ID** = X'1E'  
**DESC** = SRB scheduled for alternate IPOST

word 1 – A(ECB) to be IPOSTed  
word 2 – Primary ASID | target ASID  
word 3 – A(IPOST SRB) (0 if MVS branch entry XM-POST)  
word 4 – A(current ASCB)  
word 5 – POST code  
word 6 – 0

**TRACE ID** = X'1F'  
**DESC** = IPOST called with TOSAP= option

word 1 – A(Poster's ECB) (A(TCB) if ITASK=NO)  
word 2 – IPOST return address  
word 3 – A(ECB to be POSTed)  
word 4 – 0  
word 5 – POST code at entry to IPOST (may be complimented)  
word 6 – 0

**TRACE ID** = X'20'  
**DESC** = Dependent region shutdown ISWITCH

word 1 – A(Related PST)  
word 2 – A(Special exit)  
word 3 – SAPCNTRL field from PST's SAP  
word 4 – A(Home dispatcher work area)  
word 5 – 0  
word 6 – 0

**TRACE ID** = X'21'  
**DESC** = Entry to Post-Exit Routine

word 1 – A(ECB) being POSTed  
word 2 – ECB Contents  
word 3 – EPFFLAGS from ECB prefix  
word 4 – 0  
word 5 – 0  
word 6 – 0

**TRACE ID** = X'22'  
**DESC** = ABTERM ISWITCH entered

word 1 – A(ECB) to be switched  
word 2 – ECB contents  
word 3 – SAPCNTRL contents  
word 4 – SAPCNTRL2 contents  
word 5 – Posted Q contents  
word 6 – SAPCMEM | SAPCFLGS

**TRACE ID** = X'23'  
**DESC** = ISERWAIT called

word 1 – A(ITASK ECB)  
word 2 – ECB contents prior to ISERWAIT  
word 3 – ISERWAIT return address  
word 4 – 0  
word 5 – 0  
word 6 – SAPCNTRL contents

**TRACE ID** = X'24'  
**DESC** = ISWITCH TO=, STACK=YES called

word 1 – A(Current ECB)  
word 2 – ISWITCH return address  
word 3 – A(Target dispatcher work area)  
word 4 – SAPCNTRL field from ITASK's SAP  
word 5 – SAPXFLAG contents  
word 6 – 0

**TRACE ID** = X'25'  
**DESC** = POST ABTERM ISWITCH

word 1 – A(ECB) to be switched

word 2 – ECB POST code  
word 3 – previous posted Q contents  
word 4 – A(Target dispatcher work area)  
word 5 – IPOTC/IPEXT caller's return  
word 6 – 0

**TRACE ID** = X'26'  
**DESC** = Branch entry SCP POST

word 1 – A(ECB) to be POSTed  
word 2 – ECB POST code  
word 3 – A(ASCB) of ECB's address space  
word 4 – A(Current TCB)  
word 5 – A(Current ASCB)  
word 6 – 0

**TRACE ID** = X'27'  
**DESC** = SUSPEND IMS TCB

word 1 – A(Related PST) (0 if not a dependent region/LSD)  
word 2 – Home ASID | Primary ASID  
word 3 – A(Suspended dispatcher work area)  
word 4 – A(TCB being suspended)  
word 5 – Low order word of STORE CLOCK (STCK)  
word 6 – High order word of STORE CLOCK (STCK)

**TRACE ID** = X'28'  
**DESC** = Dependent region open dispatcher–signon

word 1 – A(Related PST)  
word 2 – Home ASID  
word 3 – A(Current TCB)  
word 4 – 0  
word 5 – 0  
word 6 – 0

**TRACE ID** = X'29'  
**DESC** = ISWITCH TO=UNSTACK

word 1 – A(Current ECB)  
word 2 – ISWITCH return address  
word 3 – X'80000000'  
word 4 – SAPCNTRL field from ECB's SAP  
word 5 – SAPXFLAG contents  
word 6 – 0

**TRACE ID** = X'2A'  
**DESC** = IMS list IPOST called

word 1 – A(ECB) to be IPOSTed  
word 2 – List IPOST return address  
word 3 – A(POST list)  
word 4 – 0  
word 5 – 0  
word 6 – 0

**TRACE ID** = X'2B'  
**DESC** = SCP WAIT issued (SVC WAIT)

word 1 – A(WAIT ECB)  
word 2 – SCP WAIT return address  
word 3 – A(Current TCB)  
word 4 – ECB contents prior to WAIT  
word 5 – 0  
word 6 – 0

**TRACE ID** = X'2C'  
**DESC** = SCP WAIT complete (SVC WAIT)

word 1 – A(WAIT ECB)  
word 2 – ECB POST code  
word 3 – A(Current TCB)  
word 4 – 0  
word 5 – 0  
word 6 – 0

**TRACE ID** = X'2D'  
**DESC** = ISWITCH TO=RET called

word 1 - A(Current ECB)  
word 2 - ISWITCH return address  
word 3 - 0  
word 4 - SAPCNTRL field from ECB's SAP  
word 5 - SAPXFLAG contents  
word 6 - 0

**TRACE ID** = X'2E'  
**DESC** = Shutdown ISWITCH reinstate

word 1 - A(PST)  
word 2 - A(Return save area)  
word 3 - A(Shutdown ECB)  
word 4 - 0  
word 5 - 0  
word 6 - 0

**TRACE ID** = X'2F'  
**DESC** = Dependent region open dispatcher-TCB switch

word 1 - A(Related PST)  
word 2 - A(Previous TCB)  
word 3 - A(Current TCB)  
word 4 - 0  
word 5 - 0  
word 6 - 0

**TRACE ID** = X'30'  
**DESC** = IWAIT called with IXCTL=YES option

word 1 - A(Current ECB)  
word 2 - ECB Contents prior to IWAIT  
word 3 - IWAIT Return address  
word 4 - A(Target ECB)  
word 5 - Target ECB Contents  
word 6 - 0

### Example of a dispatcher trace

```
**DTR          DISPATCHER TRACE
*****
***TRACE PRINTED FROM OLDEST TO MOST CURRENT ENTRY**
*****
  FUNCTION      WORD 0      WORD 1      WORD 2      WORD 3      WORD 4      WORD 5      WORD 6      WORD 7
XM ISWITCH STK  10035E11  05B5A060  80BBE2E8  80000002  00800001  001B001B  00000000  9AB7A070  MPP
TO=XMDLI
MEM CHANGE     11035E12  05B5A060  001B001B  0084001B  00B16A40  00000000  00000000  9AB7A1B3  MPP
IPOST(ECB=)    06035E17  05B5A060  80B8F516  00B21140  80B48CD7  40C1E6C5  00000000  9AB7A23D  MPP
AWE
IPC ENQ        19015E18  00B21140  40C1E6C5  FF4B7340  00B48CC0  80BE4208  00000000  9AB7A2CB  LOG
AWE
IPC RESUME     1A015E19  006DEE88  001B0084  00000082  00B48CC0  00000000  00000000  9AB7A3FC  LOG
ISERWAIT       23035E1A  85B5A060  00000000  80B8F602  00000000  00000000  00000000  9AB7A5AC  MPP
IECB STORE     1B035E1B  05B5A060  80B16A57  00000000  00000000  00000000  00000000  9AB7A671  MPP
SUSPEND        27035E1C  05B5A060  001B0084  00B16A40  00000000  00000000  00000000  9AB7A6CE  MPP
XM ISWITCH STK  10035E1E  05B4B060  867851F0  80000001  00000001  00320032  00000000  9AB7A7F1  MPP
TO=XMCTL
MEM CHANGE     11035E1F  05B4B060  00320032  00820032  00B22E00  00000000  00000000  9AB7A92D  MPP
IPOST(ECB=)    06FE5E25  006D77F0  80B91FA6  00BA156C  80B48417  40E3D9C1  00000000  9AB7A93D  N/A
TRA
IPC ENQ        19025E26  00BA156C  40E3D9C1  FF4B7C00  00B48400  80BE4208  00000000  9AB7A9A1  CTL
TRA
IPC RESUME     1A025E27  006D77F0  00820082  00000082  00B48400  00000000  00000000  9AB7A9F2  CTL
RE-DISPATCH   05015E28  00B21140  40C1E6C5  40000000  00000000  801504A6  00000000  9AB7ABA1  LOG
IWAIT          04015E2C  00B21140  00C1E6C5  801504A6  00000000  00000000  00000000  9AB7AC31  LOG
AWE
ISWITCH UNSTK  29035E2E  05B4B060  86785246  80000000  00000041  00000000  00000000  9AB7AD61  MPP
IECB STORE     1B015E2F  00B21140  80B48CD7  00C1E6C5  00000000  00000000  00000000  9AB7AF15  LOG
SUSPEND        27015E30  00000000  00820082  00B48CC0  00000000  00000000  00000000  9AB7AF7C  LOG
RE-DISPATCH   05035E31  05B4B060  00025EE4  00000003  00000000  00B22E00  00000000  9AB7AF8F  MPP
MEM CHANGE     11035E32  05B4B060  00820032  00320032  00B22E00  00000000  00000000  9AB7B04E  MPP
ITASK START    02025E33  00BA156C  40E3D9C1  064BC040  00000000  066C6440  00B7E7E0  9AB7B171  CTL
TRA
```



IPOST (ECB=)	06FE5E34	00000000	8007EAB8	05B37060	80AF3917	801A1D2C	00000000	9AB7B1C7	N/A
VSM									
IPC ENQ	19035E35	05B37060	7FE5E2D4	FF50C700	00AF3900	80BE4208	00000000	9AB7B374	MPP
VSM									
IPC RESUME	1A035E36	00000000	00840084	00000052	00AF3900	00000000	00000000	9AB7B4EF	MPP
IPOST (SAP=)	1FFE5E37	006CFE88	80B7E94C	00167060	00000000	00000000	00000000	9AB7B569	N/A
IPC ENQ	19155E39	00167060	40E3D9C1	FF4B7840	00B487C0	80BE4394	00000000	9AB7B5BC	TRA
TRA									
IPC RESUME	1A155E3A	006CFE88	00820082	00000082	00B487C0	00000000	00000000	9AB7B692	TRA
ISERWAIT	23025E3D	00BA156C	00E3D9C1	80B7E956	00000000	00000000	00000000	9AB7B843	CTL
TRA									
IECB STORE	1B025E3E	00BA156C	80B48417	00E3D9C1	00000000	00000000	00000000	9AB7B88D	CTL
SUSPEND	27025E40	00000000	00820082	00B48400	00000000	00000000	00000000	9AB7B8D7	CTL
XM ISWITCH STK	10035E44	05B4B060	80BBE2E8	80000002	00000001	00320032	00000000	9AB7B90E	MPP
TO=XMDLI									
RE-DISPATCH	05155E45	00167060	40E3D9C1	40000000	00000000	8015EC84	00000000	9AB7B9FB	TRA
MEM CHANGE	11035E46	05B4B060	00320032	00840032	00B22E00	00000000	00000000	9AB7BA3B	MPP
RE-DISPATCH	05035E48	05B37060	7FE5E2D4	00000041	00000000	8007E9FA	00000000	9AB7BA87	MPP
KPOST LIST	2A155E4A	00167060	8015EC36	00167064	00000000	00000000	00000000	9AB7BACC	TRA
IPC ENQ	19025E4B	00BA156C	40E3D9C1	FF4B7C00	00B48400	80BE456E	00000000	9AB7BC79	CTL
TRA									
IPC RESUME	1A025E4D	006CEE88	00820082	00000082	00B48400	00000000	00000000	9AB7BE28	CTL
IPOST (ECB=)	06035E4F	05B4B060	80B90B8E	00B21140	80B48CD7	40C1E6C5	00000000	9AB7BE86	MPP
AWE									
IPC ENQ	19015E50	00B21140	40C1E6C5	FF4B7340	00B48CC0	80BE4208	00000000	9AB7BF72	LOG
AWE									
IPC RESUME	1A015E51	006DEE88	00320084	00000082	00B48CC0	00000000	00000000	9AB7C0CB	LOG
IWAIT	04155E52	00167060	00E3D9C1	8015EC84	00000000	00000000	00000000	9AB7C1E7	TRA
TRA									
IECB STORE	1B155E54	00167060	80B487D7	00E3D9C1	00000000	00000000	00000000	9AB7C324	TRA
SUSPEND	27155E55	00000000	00820082	00B487C0	00000000	00000000	00000000	9AB7C4B1	TRA
ISERWAIT	23035E56	85B4B060	00000000	80B8F602	00000000	00000000	00000000	9AB7C661	MPP
IECB STORE	1B035E57	05B4B060	80B22E17	00000000	00000000	00000000	00000000	9AB7C7AE	MPP
SUSPEND	27035E58	05B4B060	00320084	00B22E00	00000000	00000000	00000000	9AB7C917	MPP
RE-DISPATCH	05015E5B	00B21140	40C1E6C5	40000000	00000000	801504A6	00000000	9AB7CA0E	LOG
IWAIT	04015E5D	00B21140	00C1E6C5	801504A6	00000000	00000000	00000000	9AB7CBB5	LOG
AWE									

### Related reference

[“IMS type-1 trace function codes” on page 577](#)

You use trace function codes to help you diagnose performance problems or other problems with IMS.

## ITASK ECB posting

The post exit routine and the IMS posting routine add all ECBs to the posted queue. When an IMS TCB waits for work, IMS issues a z/OS SUSPEND. This task is reactivated by a RESUME invoked by the post exit posting routine or the IMS posting routine.

## System post codes

The subset of the system post codes are listed and described.

### System post codes

The following table lists only a subset of the possible post codes.

*Table 181. System post codes*

Code	Description
X'40', C'BTR'	PST posted by scheduler as a result of BMP termination (Subqueues 4, 5)
X'40', C'CHK'	PST posted by checkpoint (Subqueues 3, 4, 5, 6)
X'40', C'SMB'	PST posted by SMB enqueue when a message is received that can be processed by the PST (Subqueue 3 or 6)
X'40', C'CMD'	PST posted by command processor when <b>/START PGM, /START TRAN</b> , or a similar command is entered (Subqueues 3, 6)

Table 181. System post codes (continued)

Code	Description
X'40', C'ABD'	PST posted by DFSCPY00 as a result of an abend in a dependent region (Subqueues 3, 4, 5, 6)
X'40', C'PRG'	PST posted by scheduler to stop region when checkpoint purge (that is, all messages processed) is complete—this is used if MPP issued last message (Subqueue 3)
X'40', C'STP'	PST posted by DFSSTOPO when the region is waiting in scheduler and is to be stopped (Subqueues 3, 4, 5)
X'40', C'DLG'	PST posted by DFSRDLG0 when dynamic log is free (Subqueues 3, 4, 5, 6)
X'40', C'CF4'	PST posted by DFSASK00 as a result of an abend in a dependent region (Subqueues 3, 4, 5, 6)
X'40', C'DEQ'	Terminate control processor ECB posted by DFSRST00 at restart completion
X'40', C'TO'	PST posted after ISWITCH to IMS control region TCB
X'40', C'RET'	PST posted after ISWITCH return to dependent region TCB

## External subsystem trace

You enable the external subsystem trace by using the **/TRACE SET ON TABLE SUBS** command. When you specify **OPTION LOG**, IMS writes the trace externally as type X'67FA' records.

The External Subsystem (ESS) trace entries help you analyze problems for either:

- A connection problem between the IMS control region and the external subsystem (for example, Db2 for z/OS)
- Any problem between the IMS dependent region and the external subsystem

The following figure illustrates the external subsystem (ESS) trace record format. Each of the sixteen words is 4 bytes long. Words 0 and 1 hold the standard ESS trace record prefix. The Module ID and Sub function (WORD 1) determines what information appears in words 2 through 15.

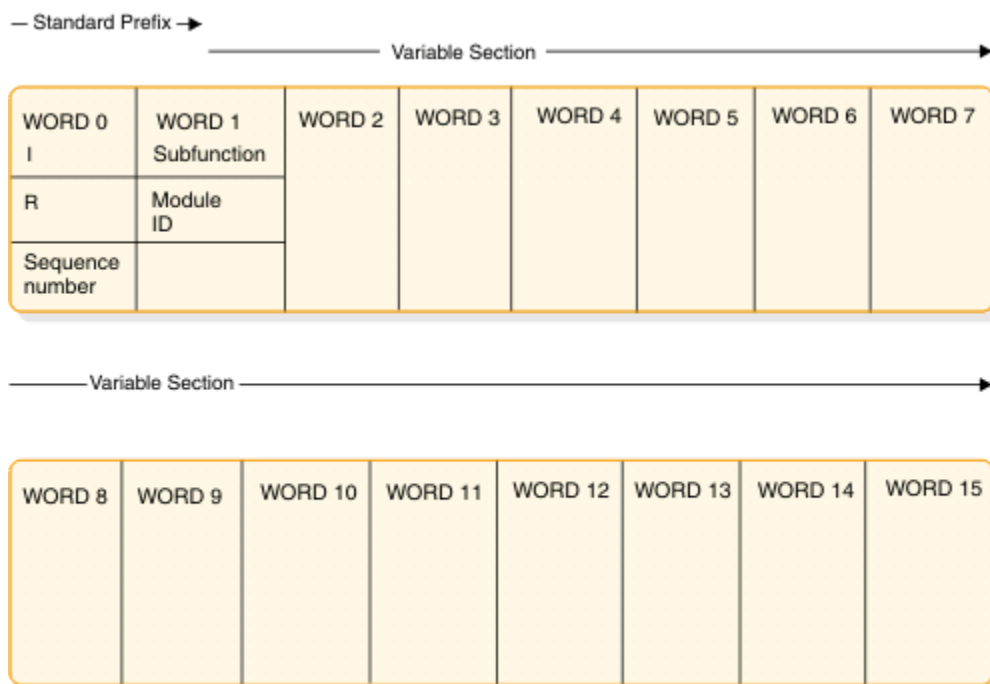


Figure 98. External subsystem (ESS) trace record format

where

**represents**

**I**

This 1-byte field contains the hexadecimal trace record ID. Two possible ID values are X'57' and X'58'. The X'57' record ID is created by a module that executes in the IMS control region (for example, the ESS mother task DFSIESIO). The X'58' record ID is created by a module that executes in an IMS dependent region (for example, DFSESCT0).

**R**

This 1-byte field is reserved.

**SEQ NUM**

This 2-byte field contains the hexadecimal trace record sequence number assigned by the IMS trace component.

**MOD ID**

This 2-byte field contains a hexadecimal value that identifies the module that created the trace record. Each ESS module has an associated module ID. Macro DFSESFC contains the complete list of IDs.

**SUB FUNC**

This 2-byte field contains a hexadecimal value that identifies the subfunction that created the trace record within the module. For example, if a module creates a trace record in each of five internal subroutines, each subroutine has a unique SUB FUNC ID.

The following table shows:

- The ID of the module that created the trace record
- The ID of the subfunction (within the module) that created the record
- The name of the module that created the record
- A description of the event being traced

Table 182. Module ID and subfunction table

Module ID	Subfunction	Module	Description
X'0015'	X'0015'	DFSESS40	ESS message service exit

Table 182. Module ID and subfunction table (continued)

Module ID	Subfunction	Module	Description
X'0016'	X'0014'	DFSESS30	ESS logging exit
X'0017'	X'0011' X'0012' X'0040' X'0041'	DFSESS10	IMS control region identify Dependent region identify Control region identify error Identify error subsystem stopped
X'0018'	X'0013'	DFSESS20	ESS termination exit (if X'57') Dependent region ESS term (if X'58')
X'0285'	X'0010'	DFSESD80	Dependent region ESS initialization
X'0288'	X'0001'	DFSESS00	Dependent region ESS sign on
X'0289'	X'0003'	DFSESD50	Dependent region ESS signoff
X'0290'	X'0005'	DFSESECT0	Dependent region ESS create thread
X'0291'	X'0002' X'0003' X'0004'	DFSESD50	Dependent region ESS term thread Dependent region ESS term thread region ESS signoff Dependent region ESS term identify
X'0292'	X'0004'	DFSESD50	Dependent region ESS term identify
X'0293'	X'0007'	DFSESAB0	Dependent region ESS ABORT
X'0294'	X'0008'	DFSESP10	Dependent region ESS commit prep
X'0295'	X'0009'	DFSESP20	Dependent region ESS commit cont
X'0307'	X'0016' X'0017' X'0018'	DFSFEPS0	ESS commit processor entered ESS commit processor exited ESS commit processor R-I-D request
X'0402'	X'0020' X'0021' X'0022' X'0023' X'0024' X'0025' X'0026' X'0027' X'0028' X'0029' X'0030' X'0031'	DFSESI30	IMS control region daughter identify IMS control region resolve-in-doubt IMS control region ESS CMD IMS control region ESS RRE IMS control region ESS ECHO IMS control region terminate identify IMS control region terminate subsystem IMS control region /STOP CMD IMS control region ESS term record IMS control region ESS shutdown IMS control region ESS termination IMS control region ESS AWE error
X'0403'	X'0019'	DFSESI50	Control region ESS initialization

Table 182. Module ID and subfunction table (continued)

Module ID	Subfunction	Module	Description
X'0404'	X'0042'	DFSESI60	Control region ESS R-I-D exit
X'0405'	X'0032'	DFSESI70	Control region ESS /CHANGE
X'0409'	X'0001'	DFSIESIO	Mother ITASK request
	X'0002'		Control region ESS attach
X'0506'	X'0006'	DFSESPR0	Dependent region ESS program request handler
	X'0019'		Dependent region ESS program request recursive call
	X'0020'		Dependent region ESS Subsystem
			Not Operational (SNOX)

#### Related reference

[“IMS type-1 trace function codes” on page 577](#)

You use trace function codes to help you diagnose performance problems or other problems with IMS.

## Layout of the X'57' variable section

The layout of the X'57' variable section is depicted.

### MOD ID = X'0015' SUB FUNC = X'0015' DFSESS40 External SubSys MESSAGE service request record

```
MOD ID   = X'0015'
SUB FUNC = X'0015' DFSESS40 External SubSys MESSAGE service request
                    record

word 2 -- External SubSystem name
words 3 through 15 not used
```

### MOD ID = X'0016' SUB FUNC = X'0014' DFSESS30 External SubSys LOGGING service request record

```
MOD ID   = X'0016'
SUB FUNC = X'0014' DFSESS30 External SubSys LOGGING service request
                    record

word 2 -- External SubSystem name
words 3 through 15 not used
```

### MOD ID = X'0017' SUB FUNC = X'0011' DFSESS10 control region External SubSys IDENTIFY record

```
MOD ID   = X'0017'
SUB FUNC = X'0011' DFSESS10 control region External SubSys IDENTIFY record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 not used
        byte 2 GESEGF1 (DFSGESE macro global flag1)
        byte 3 GESEGF2 (DFSGESE macro global flag2)
word 4 -- byte 0 GESEGF3 (DFSGESE macro global flag3)
        byte 1 not used
        byte 2 SSIDFLG1 (DFSSSIE subsys status flag1)
        byte 3 SSIDFLG2 (DFSSSIE subsys status flag2)
word 5 -- bytes 0-1 not used
        bytes 2-3 AWQRC (DFSawe DFSESI30 identify return code)

words 6 through 15 not used
```

## SUB FUNC = X'0040' DFSESS10 External SubSys GLOBAL identify error record

```
SUB FUNC = X'0040' DFSESS10 External SubSys GLOBAL identify error record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 not used
      byte 2 GESEGF1 (DFSGESE macro global flag1)
      byte 3 GESEGF2 (DFSGESE macro global flag2)
word 4 -- byte 0 GESEGF3 (DFSGESE macro global flag3)
      byte 1 not used
      byte 2 SSIDFLG1 (DFSSSIE subsys status flag1)
      byte 3 SSIDFLG2 (DFSSSIE subsys status flag2)
words 5 through 15 not used
```

## SUB FUNC = X'0041' DFSESS10 External SubSys identify with External SubSystem

```
SUB FUNC = X'0041' DFSESS10 External SubSys identify with External SubSystem

stopped or stopping record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 not used
      byte 2 GESEGF1 (DFSGESE macro global flag1)
      byte 3 GESEGF2 (DFSGESE macro global flag2)
word 4 -- byte 0 GESEGF3 (DFSGESE macro global flag3)
      byte 1 not used
      byte 2 SSIDFLG1 (DFSSSIE subsys status flag1)
      byte 3 SSIDFLG2 (DFSSSIE subsys status flag2)
words 5 through 15 not used
```

## MOD ID = X'0018' SUB FUNC = X'0013' DFSESS20 External SubSys termination record

```
MOD ID = X'0018'
SUB FUNC = X'0013' DFSESS20 External SubSys termination record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 not used
      byte 2 GESEGF1 (DFSGESE macro global flag1)
      byte 3 GESEGF2 (DFSGESE macro global flag2)
word 4 -- byte 0 GESEGF3 (DFSGESE macro global flag3)
      byte 1 not used
      byte 2 SSIDFLG1 (DFSSSIE subsys status flag1)
      byte 3 SSIDFLG2 (DFSSSIE subsys status flag2)
words 5 through 15 not used
```

## MOD ID = X'0402' SUB FUNC = X'0020' DFSESI30 External SubSys IDENTIFY exit record

```
MOD ID = X'0402'
SUB FUNC = X'0020' DFSESI30 External SubSys IDENTIFY exit record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 not used
      byte 2 GESEGF1 (DFSGESE macro global flag1)
      byte 3 GESEGF2 (DFSGESE macro global flag2)
word 4 -- byte 0 GESEGF3 (DFSGESE macro global flag3)
      byte 1 ESSTERRC (External SubSys termination reason)
      byte 2 SSIDFLG1 (DFSSSIE subsys status flag1)
      byte 3 SSIDFLG2 (DFSSSIE subsys status flag2)
word 5 -- bytes 0-1 not used
      bytes 2-3 External SubSys exit routine return code
words 6 through 15 not used
```

## SUB FUNC = X'0021' DFSESI30 External SubSys RESOLVE IN DOUBT record

```
SUB FUNC = X'0021' DFSESI30 External SubSys RESOLVE IN DOUBT record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 not used
      byte 2 GESEGF1 (DFSGESE macro global flag1)
      byte 3 GESEGF2 (DFSGESE macro global flag2)
word 4 -- byte 0 GESEGF3 (DFSGESE macro global flag3)
```

```

        byte 1  ESSTERRC (External SubSys termination reason)
        byte 2  SSIDFLG1 (DFSSSIE subsys status flag1)
        byte 3  SSIDFLG2 (DFSSSIE subsys status flag2)
word 5 -- bytes 0-1 not used
        bytes 2-3 AWQRC (DFSAWE return code, see DFSESSEC)
words 6 through 7 not used
words 8 through 11 RRETOKEN (DFSRRE UOW recovery token)
word 12 -- bytes 0-1 RRECI (DFSRRE commit indicator)
        bytes 2-3 not used
words 13 through 15 not used

```

## SUB FUNC = X'0022' DFSESI30 External SubSys /SSR COMMAND exit record

**SUB FUNC = X'0022'** DFSESI30 External SubSys /SSR COMMAND exit record

```

word 2 -- External SubSystem name
word 3 -- bytes 0-1 not used
        byte 2  GESEGF1 (DFSGESE macro global flag1)
        byte 3  GESEGF2 (DFSGESE macro global flag2)
word 4 -- byte 0  GESEGF3 (DFSGESE macro global flag3)
        byte 1  ESSTERRC (External SubSys termination reason)
        byte 2  SSIDFLG1 (DFSSSIE subsys status flag1)
        byte 3  SSIDFLG2 (DFSSSIE subsys status flag2)
word 5 -- bytes 0-1 not used
        bytes 2-3 External SubSys exit routine return code
words 6 through 15 not used

```

## SUB FUNC = X'0023' DFSESI30 External SubSys specific RRE request record

**SUB FUNC = X'0023'** DFSESI30 External SubSys specific RRE request record

```

word 2 -- External SubSystem name
word 3 -- bytes 0-1 not used
        byte 2  GESEGF1 (DFSGESE macro global flag1)
        byte 3  GESEGF2 (DFSGESE macro global flag2)
word 4 -- byte 0  GESEGF3 (DFSGESE macro global flag3)
        byte 1  ESSTERRC (External SubSys termination reason)
        byte 2  SSIDFLG1 (DFSSSIE subsys status flag1)
        byte 3  SSIDFLG2 (DFSSSIE subsys status flag2)
words 5 through 7 not used
words 8 through 11 RRETOKEN (DFSRRE UOW recovery token)
word 12 -- bytes 0-1 RRECI (DFSRRE commit indicator)
        bytes 2-3 not used
words 13 through 15 not used

```

## SUB FUNC = X'0024' DFSESI30 External SubSys ECHO exit record

**SUB FUNC = X'0024'** DFSESI30 External SubSys ECHO exit record

```

word 2 -- External SubSystem name
word 3 -- bytes 0-1 not used
        byte 2  GESEGF1 (DFSGESE macro global flag1)
        byte 3  GESEGF2 (DFSGESE macro global flag2)
word 4 -- byte 0  GESEGF3 (DFSGESE macro global flag3)
        byte 1  ESSTERRC (External SubSys termination reason)
        byte 2  SSIDFLG1 (DFSSSIE subsys status flag1)
        byte 3  SSIDFLG2 (DFSSSIE subsys status flag2)
word 5 -- bytes 0-1 not used
        bytes 2-3 External SubSys exit routine return code
words 6 through 7 not used
words 8 through 11 RRETOKEN (DFSRRE UOW recovery token)
word 12 -- bytes 0-1 RRECI (DFSRRE commit indicator)
        bytes 2-3 not used
words 13 through 15 not used

```

## SUB FUNC = X'0025' DFSESI30 External SubSys TERMINATE IDENTIFY exit

**SUB FUNC = X'0025'** DFSESI30 External SubSys TERMINATE IDENTIFY exit record

```

word 2 -- External SubSystem name
word 3 -- bytes 0-1 not used
        byte 2  GESEGF1 (DFSGESE macro global flag1)

```

```

        byte 3  GESEGF2  (DFSGESE macro global flag2)
word  4 -- byte 0  GESEGF3  (DFSGESE macro global flag3)
        byte 1  ESSTERRC (External SubSys termination reason)
        byte 2  SSIDFLG1 (DFSSSIE subsys status flag1)
        byte 3  SSIDFLG2 (DFSSSIE subsys status flag2)
word  5 -- bytes 0-1 not used
        bytes 2-3 External SubSys exit routine return code
words 6 through 15  not used

```

### SUB FUNC = X'0026' DFSESI30 External SubSys TERMINATE SUBSYSTEM record

**SUB FUNC = X'0026'** DFSESI30 External SubSys TERMINATE SUBSYSTEM record

```

word  2 -- External SubSystem name
word  3 -- bytes 0-1 not used
        byte 2  GESEGF1  (DFSGESE macro global flag1)
        byte 3  GESEGF2  (DFSGESE macro global flag2)
word  4 -- byte 0  GESEGF3  (DFSGESE macro global flag3)
        byte 1  ESSTERRC (External SubSys termination reason)
        byte 2  SSIDFLG1 (DFSSSIE subsys status flag1)
        byte 3  SSIDFLG2 (DFSSSIE subsys status flag2)
words 5 through 15  not used

```

### SUB FUNC = X'0027' DFSESI30 External SubSys /STOP command record

**SUB FUNC = X'0027'** DFSESI30 External SubSys /STOP command record

```

word  2 -- External SubSystem name
word  3 -- bytes 0-1 not used
        byte 2  GESEGF1  (DFSGESE macro global flag1)
        byte 3  GESEGF2  (DFSGESE macro global flag2)
word  4 -- byte 0  GESEGF3  (DFSGESE macro global flag3)
        byte 1  ESSTERRC (External SubSys termination reason)
        byte 2  SSIDFLG1 (DFSSSIE subsys status flag1)
        byte 3  SSIDFLG2 (DFSSSIE subsys status flag2)
words 5 through 15  not used

```

### SUB FUNC = X'0028' DFSESI30 External SubSys IMS termination record

**SUB FUNC = X'0028'** DFSESI30 External SubSys IMS termination record

```

word  2 -- External SubSystem name
word  3 -- bytes 0-1 not used
        byte 2  GESEGF1  (DFSGESE macro global flag1)
        byte 3  GESEGF2  (DFSGESE macro global flag2)
word  4 -- byte 0  GESEGF3  (DFSGESE macro global flag3)
        byte 1  ESSTERRC (External SubSys termination reason)
        byte 2  SSIDFLG1 (DFSSSIE subsys status flag1)
        byte 3  SSIDFLG2 (DFSSSIE subsys status flag2)
words 5 through 15  not used

```

### SUB FUNC = X'0029' DFSESI30 External SubSys IMS shutdown record

**SUB FUNC = X'0029'** DFSESI30 External SubSys IMS shutdown record

```

word  2 -- External SubSystem name
word  3 -- bytes 0-1 not used
        byte 2  GESEGF1  (DFSGESE macro global flag1)
        byte 3  GESEGF2  (DFSGESE macro global flag2)
word  4 -- byte 0  GESEGF3  (DFSGESE macro global flag3)
        byte 1  ESSTERRC (External SubSys termination reason)
        byte 2  SSIDFLG1 (DFSSSIE subsys status flag1)
        byte 3  SSIDFLG2 (DFSSSIE subsys status flag2)
words 5 through 15  not used

```

### SUB FUNC = X'0030' DFSESI30 External SubSys TERMINATION exit record

**SUB FUNC = X'0030'** DFSESI30 External SubSys TERMINATION exit record

```

word  2 -- External SubSystem name
word  3 -- bytes 0-1 not used

```



```

        byte 2 GESEGF1 (DFSGESE macro global flag1)
        byte 3 GESEGF2 (DFSGESE macro global flag2)
word 4 -- byte 0 GESEGF3 (DFSGESE macro global flag3)
        byte 1 ESSTERRC (External SubSys termination reason)
        byte 2 SSIDFLG1 (DFSSSIE subsys status flag1)
        byte 3 SSIDFLG2 (DFSSSIE subsys status flag2)
word 5 -- bytes 0-1 not used
        bytes 2-3 External SubSys exit routine return code
words 6 through 15 not used

```

#### **SUB FUNC = X'0031' DFSESI30 AWE error record**

**SUB FUNC = X'0031'** DFSESI30 AWE error record

```

word 2 -- External SubSystem name
word 3 -- bytes 0-1 not used
        byte 2 GESEGF1 (DFSGESE macro global flag1)
        byte 3 GESEGF2 (DFSGESE macro global flag2)
word 4 -- byte 0 GESEGF3 (DFSGESE macro global flag3)
        byte 1 ESSTERRC (External SubSys termination reason)
        byte 2 SSIDFLG1 (DFSSSIE subsys status flag1)
        byte 3 SSIDFLG2 (DFSSSIE subsys status flag2)
word 5 -- bytes 0-1 not used
        bytes 2-3 AWQRC (DFSAWE return code)
words 6 through 15 not used

```

#### **MOD ID = X'0403' SUB FUNC = X'0019' DFSESI50 External SubSys INITIALIZATION exit record**

**MOD ID = X'0403'**  
**SUB FUNC = X'0019'** DFSESI50 External SubSys INITIALIZATION exit record

```

word 2 -- External SubSystem name
word 3 -- bytes 0-1 not used
        byte 2 GESEGF1 (DFSGESE macro global flag1)
        byte 3 GESEGF2 (DFSGESE macro global flag2)
word 4 -- byte 0 GESEGF3 (DFSGESE macro global flag3)
        byte 1 not used
        byte 2 SSIDFLG1 (DFSSSIE subsys status flag1)
        byte 3 SSIDFLG2 (DFSSSIE subsys status flag2)
word 5 -- bytes 0-1 not used
        bytes 2-3 External SubSys exit routine return code
words 6 through 15 not used

```

#### **MOD ID = X'0404' SUB FUNC = X'0042' DFSESI60 External SubSys RESOLVE IN DOUBT exit record**

**MOD ID = X'0404'**  
**SUB FUNC = X'0042'** DFSESI60 External SubSys RESOLVE IN DOUBT exit record

```

word 2 -- External SubSystem name
word 3 -- bytes 0-1 not used
        byte 2 GESEGF1 (DFSGESE macro global flag1)
        byte 3 GESEGF2 (DFSGESE macro global flag2)
word 4 -- byte 0 GESEGF3 (DFSGESE macro global flag3)
        byte 1 not used
        byte 2 SSIDFLG1 (DFSSSIE subsys status flag1)
        byte 3 SSIDFLG2 (DFSSSIE subsys status flag2)
word 5 -- bytes 0-1 not used
        bytes 2-3 External SubSys exit routine return code
words 6 through 7 not used
words 8 through 11 RRETOKEN (DFSRRE UOW recovery token)
word 12 -- bytes 0-1 RRECI (DFSRRE commit indicator)
        bytes 2-3 not used
words 13 through 15 not used

```

#### **MOD ID = X'0405' SUB FUNC = X'0032' DFSESI70 External SubSys /CHANGE command record**

**MOD ID = X'0405'**  
**SUB FUNC = X'0032'** DFSESI70 External SubSys /CHANGE command record

```

word 2 -- External SubSystem name
word 3 -- bytes 0-1 not used
        byte 2 GESEGF1 (DFSGESE macro global flag1)
        byte 3 GESEGF2 (DFSGESE macro global flag2)

```

```

word 4 -- byte 0 GESEGF3 (DFSGESE macro global flag3)
        byte 1 not used
        byte 2 SSIDFLG1 (DFSSSIE subsys status flag1)
        byte 3 SSIDFLG2 (DFSSSIE subsys status flag2)
words 5 through 15 not used

```

#### **MOD ID = X'0409' SUB FUNC = X'0001' DFSIESI0 mother ITASK request record**

```

MOD ID   = X'0409'
SUB FUNC = X'0001' DFSIESI0 mother ITASK request record

word 2 -- not used
word 3 -- bytes 0-1 function requested
        Function requested:
        X'0002' terminate the mother ITASK TCB
        X'0003' build / merge subsystem definitions
        X'0004' SSM JCL parameter
        X'0005' attach external subsystem ITASK TCB
        X'0007' /START command
        X'0008' sync request
        bytes 2-3 not used
word 4 -- not used
word 5 -- bytes 0-1 not used
        bytes 2-3 AWQRC (DFSAWE DFSIESI0 return code)
words 6 through 15 not used

```

#### **SUB FUNC = X'0002' DFSIESI0 External Subsys ATTACH record**

```

SUB FUNC = X'0002' DFSIESI0 External Subsys ATTACH record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 function requested
        Function requested:
        X'0005' attach external subsystem ITASK TCB
        X'0007' /START command
        byte 2 GESEGF1 (DFSGESE macro global flag1)
        byte 3 GESEGF2 (DFSGESE macro global flag2)
word 4 -- byte 0 GESEGF3 (DFSGESE macro global flag3)
        byte 1 not used
        byte 2 SSIDFLG1 (DFSSSIE subsys status flag1)
        byte 3 SSIDFLG2 (DFSSSIE subsys status flag2)
word 5 -- bytes 0-1 not used
        bytes 2-3 AWQRC (DFSAWE attach process return code)
words 6 through 15 not used

```

## **Layout of the X'58' variable section**

The layout of the X'58' variable section is depicted.

#### **MOD ID = X'0015' SUB FUNC = X'0015' DFSESS40 External SubSys MESSAGE service request record**

```

MOD ID   = X'0015'
SUB FUNC = X'0015' DFSESS40 External SubSys MESSAGE service request
record

word 2 -- External SubSystem name
words 3 through 15 not used

```

#### **MOD ID = X'0016' SUB FUNC = X'0014' DFSESS30 External SubSys LOGGING service request record**

```

MOD ID   = X'0016'
SUB FUNC = X'0014' DFSESS30 External SubSys LOGGING service request
record

word 2 -- External SubSystem name
words 3 through 15 not used

```

## MOD ID = X'0017' SUB FUNC = X'0011' DFSESS10 control region External SubSys IDENTIFY record

```
MOD ID   = X'0017'
SUB FUNC = X'0011' DFSESS10 control region External SubSys IDENTIFY record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 PSTID (IMS dependent region ID)
        byte 2 GESEGF1 (DFSGESE macro global flag1)
        byte 3 GESEGF2 (DFSGESE macro global flag2)
word 4 -- byte 0 GESEGF3 (DFSGESE macro global flag3)
        byte 1 not used
        byte 2 SSIDFLG1 (DFSSSIE subsys status flag1)
        byte 3 SSIDFLG2 (DFSSSIE subsys status flag2)
word 5 -- bytes 0-1 not used
        bytes 2-3 AWQRC (DFSASWE DFSESI30 identify return code)
words 6 through 7 not used
words 8 through 11 LCRETOKN (DFSALCRE UOW recovery token)
words 12 through 15 not used
```

## SUB FUNC = X'0012' DFSESS10 dependent region External SubSys IDENTIFY record

```
SUB FUNC = X'0012' DFSESS10 dependent region External SubSys IDENTIFY
record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 PSTID (IMS dependent region ID)
        byte 2 EZSGFL (DFSEZS connection status byte1)
        byte 3 EZSLFL (DFSEZS connection status byte2)
word 4 -- byte 0 EZSEFL1 (DFSEZS thread startup status)
        byte 1 EZSEFL2 (DFSEZS thread commit status)
        byte 2 EZSEFL3 (DFSEZS thread termination status)
        byte 3 EZSEFL4 (DFSEZS termination flag)
word 5 -- bytes 0-1 not used
        bytes 2-3 AWQRC (DFSASWE DFSESI30 identify return code)
words 6 through 7 not used
words 8 through 11 LCRETOKN (DFSALCRE UOW recovery token)
words 12 through 15 not used
```

## SUB FUNC = X'0040' DFSESS10 IMS detected External SubSys IDENTIFY error record

```
SUB FUNC = X'0040' DFSESS10 IMS detected External SubSys IDENTIFY error
record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 PSTID (IMS dependent region ID)
        byte 2 GESEGF1 (DFSGESE macro global flag1)
        byte 3 GESEGF2 (DFSGESE macro global flag2)
word 4 -- byte 0 GESEGF3 (DFSGESE macro global flag3)
        byte 1 not used
        byte 2 SSIDFLG1 (SSIDX subsys status flag1)
        byte 3 SSIDFLG2 (SSIDX subsys status flag2)
words 5 through 7 not used
words 8 through 11 LCRETOKN (DFSALCRE UOW recovery token)
words 12 through 15 not used
```

## SUB FUNC = X'0041' DFSESS10 IMS detected External SubSys IDENTIFY with External SubSystem stopped or stopping record

```
SUB FUNC = X'0041' DFSESS10 IMS detected External SubSys IDENTIFY with
External SubSystem stopped or stopping record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 PSTID (IMS dependent region ID)
        byte 2 GESEGF1 (DFSGESE macro global flag1)
        byte 3 GESEGF2 (DFSGESE macro global flag2)
word 4 -- byte 0 GESEGF3 (DFSGESE macro global flag3)
        byte 1 not used
        byte 2 SSIDFLG1 (SSIDX subsys status flag1)
        byte 3 SSIDFLG2 (SSIDX subsys status flag2)
words 5 through 7 not used
words 8 through 11 LCRETOKN (DFSALCRE UOW recovery token)
words 12 through 15 not used
```

## MOD ID = X'0018' SUB FUNC = X'0013' DFSESS20 External SubSys termination record

MOD ID = X'0018'  
SUB FUNC = X'0013' DFSESS20 External SubSys termination record

word	2	--	External SubSystem	name
word	3	--	bytes 0-1	not used
		byte	2	GESEGF1 (DFSGESE macro global flag1)
		byte	3	GESEGF2 (DFSGESE macro global flag2)
word	4	--	byte 0	GESEGF3 (DFSGESE macro global flag3)
		byte	1	not used
		byte	2	SSIDFLG1 (DFSSSIE subsys status flag1)
		byte	3	SSIDFLG2 (DFSSSIE subsys status flag2)
words	5	through	15	not used

## MOD ID = X'0285' SUB FUNC = X'0010' DFSESD80 dep region External SubSys INITIALIZATION exit

MOD ID = X'0285'  
SUB FUNC = X'0010' DFSESD80 dep region External SubSys INITIALIZATION exit record

word	2	--	External SubSystem	name
word	3	--	bytes 0-1	PSTID (IMS dependent region ID)
		byte	2	EZSGFL (DFSEZS connection status byte1)
		byte	3	EZSLFL (DFSEZS connection status byte2)
word	4	--	byte 0	EZSEFL1 (DFSEZS thread startup status)
		byte	1	EZSEFL2 (DFSEZS thread commit status)
		byte	2	EZSEFL3 (DFSEZS thread termination status)
		byte	3	EZSEFL4 (DFSEZS termination flag)
word	5	--	bytes 0-1	not used
		bytes	2-3	External SubSys exit routine return code
words	6	through	7	not used
words	8	through	11	LCRETOKN (DFSLCRE UOW recovery token)
words	12	through	15	not used

## MOD ID = X'0288' SUB FUNC = X'0001' DFSESS00 External SubSys SIGNON exit record

MOD ID = X'0288'  
SUB FUNC = X'0001' DFSESS00 External SubSys SIGNON exit record

word	2	--	External SubSystem	name
word	3	--	bytes 0-1	PSTID (IMS dependent region ID)
		byte	2	EZSGFL (DFSEZS connection status byte1)
		byte	3	EZSLFL (DFSEZS connection status byte2)
word	4	--	byte 0	EZSEFL1 (DFSEZS thread startup status)
		byte	1	EZSEFL2 (DFSEZS thread commit status)
		byte	2	EZSEFL3 (DFSEZS thread termination status)
		byte	3	EZSEFL4 (DFSEZS termination flag)
word	5	--	bytes 0-1	not used
		bytes	2-3	External SubSys exit routine return code
word	6	--	byte 0	RLA parameter for Db2 subsystem
		byte	1-3	not used
Word	7			not used
words	8	through	11	LCRETOKN (DFSLCRE UOW recovery token)
words	12	through	15	not used

## MOD ID = X'0290' SUB FUNC = X'0005' DFSESD50 External SubSys CREATE THREAD exit record

MOD ID = X'0289'  
SUB FUNC = X'0003' DFSESD50 External SubSys SIGNOFF exit record

word	2	--	External SubSystem	name
word	3	--	bytes 0-1	PSTID (IMS dependent region ID)
		byte	2	EZSGFL (DFSEZS connection status byte1)
		byte	3	EZSLFL (DFSEZS connection status byte2)
word	4	--	byte 0	EZSEFL1 (DFSEZS thread startup status)
		byte	1	EZSEFL2 (DFSEZS thread commit status)
		byte	2	EZSEFL3 (DFSEZS thread termination status)
		byte	3	EZSEFL4 (DFSEZS termination flag)
word	5	--	bytes 0-1	not used
		bytes	2-3	External SubSys exit routine return code
words	6	through	7	not used
words	8	through	11	LCRETOKN (DFSLCRE UOW recovery token)
words	12	through	15	not used

**MOD ID = X'0290' SUB FUNC = X'0005' DFSE SCT0 External SubSys CREATE THREAD exit record**

```

MOD ID   = X'0290'
SUB FUNC = X'0005' DFSE SCT0 External SubSys CREATE THREAD exit record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 PSTID (IMS dependent region ID)
        byte 2 EZSGFL (DFSEZS connection status byte1)
        byte 3 EZSLFL (DFSEZS connection status byte2)
word 4 -- byte 0 EZSEFL1 (DFSEZS thread startup status)
        byte 1 EZSEFL2 (DFSEZS thread commit status)
        byte 2 EZSEFL3 (DFSEZS thread termination status)
        byte 3 EZSEFL4 (DFSEZS termination flag)
word 5 -- bytes 0-1 not used
        bytes 2-3 External SubSys exit routine return code
words 6 through 7 not used
words 8 through 11 LCRETOKN (DFSLCRE UOW recovery token)
words 12 through 15 not used

```

**MOD ID = X'0291' SUB FUNC = X'0002' DFSE SD50 External SubSys TERMINATE THREAD exit record**

```

MOD ID   = X'0291'
SUB FUNC = X'0002' DFSE SD50 External SubSys TERMINATE THREAD exit record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 PSTID (IMS dependent region ID)
        byte 2 EZSGFL (DFSEZS connection status byte1)
        byte 3 EZSLFL (DFSEZS connection status byte2)
word 4 -- byte 0 EZSEFL1 (DFSEZS thread startup status)
        byte 1 EZSEFL2 (DFSEZS thread commit status)
        byte 2 EZSEFL3 (DFSEZS thread termination status)
        byte 3 EZSEFL4 (DFSEZS termination flag)
word 5 -- bytes 0-1 not used
        bytes 2-3 External SubSys exit routine return code
words 6 through 7 not used
words 8 through 11 LCRETOKN (DFSLCRE UOW recovery token)
words 12 through 15 not used

```

**MOD ID = X'0292' SUB FUNC = X'0004' DFSE SD50 External SubSys TERMINATE IDENTIFY exit**

```

MOD ID   = X'0292'
SUB FUNC = X'0004' DFSE SD50 External SubSys TERMINATE IDENTIFY exit
                      record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 PSTID (IMS dependent region ID)
        byte 2 EZSGFL (DFSEZS connection status byte1)
        byte 3 EZSLFL (DFSEZS connection status byte2)
word 4 -- byte 0 EZSEFL1 (DFSEZS thread startup status)
        byte 1 EZSEFL2 (DFSEZS thread commit status)
        byte 2 EZSEFL3 (DFSEZS thread termination status)
        byte 3 EZSEFL4 (DFSEZS termination flag)
word 5 -- bytes 0-1 not used
        bytes 2-3 External SubSys exit routine return code
words 6 through 7 not used
words 8 through 11 LCRETOKN (DFSLCRE UOW recovery token)
words 12 through 15 not used

```

**MOD ID = X'0293' SUB FUNC = X'0007' DFSE SAB0 External SubSys ABORT exit record**

```

MOD ID   = X'0293'
SUB FUNC = X'0007' DFSE SAB0 External SubSys ABORT exit record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 PSTID (IMS dependent region ID)
        byte 2 EZSGFL (DFSEZS connection status byte1)
        byte 3 EZSLFL (DFSEZS connection status byte2)
word 4 -- byte 0 EZSEFL1 (DFSEZS thread startup status)
        byte 1 EZSEFL2 (DFSEZS thread commit status)
        byte 2 EZSEFL3 (DFSEZS thread termination status)
        byte 3 EZSEFL4 (DFSEZS termination flag)
word 5 -- bytes 0-1 not used
        bytes 2-3 External SubSys exit routine return code
words 6 through 7 not used

```

words 8 through 11	LCRETOKN (DFSLCRE UOW recovery token)
words 12 through 15	not used

### MOD ID = X'0294' SUB FUNC = X'0008' DFSESP10 External SubSys COMMIT PREPARE exit record

```

MOD ID   = X'0294'
SUB FUNC = X'0008' DFSESP10 External SubSys COMMIT PREPARE exit record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 PSTID (IMS dependent region ID)
      byte 2 EZSGFL (DFSEZS connection status byte1)
      byte 3 EZSLFL (DFSEZS connection status byte2)
word 4 -- byte 0 EZSEFL1 (DFSEZS thread startup status)
      byte 1 EZSEFL2 (DFSEZS thread commit status)
      byte 2 EZSEFL3 (DFSEZS thread termination status)
      byte 3 EZSEFL4 (DFSEZS termination flag)
word 5 -- bytes 0-1 not used
      bytes 2-3 External SubSys exit routine return code
words 6 through 7 not used
words 8 through 11 LCRETOKN (DFSLCRE UOW recovery token)
words 12 through 15 not used

```

### MOD ID = X'0295' SUB FUNC = X'0009' DFSESP20 External SubSys COMMIT CONTINUE exit record

```

MOD ID   = X'0295'
SUB FUNC = X'0009' DFSESP20 External SubSys COMMIT CONTINUE exit record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 PSTID (IMS dependent region ID)
      byte 2 EZSGFL (DFSEZS connection status byte1)
      byte 3 EZSLFL (DFSEZS connection status byte2)
word 4 -- byte 0 EZSEFL1 (DFSEZS thread startup status)
      byte 1 EZSEFL2 (DFSEZS thread commit status)
      byte 2 EZSEFL3 (DFSEZS thread termination status)
      byte 3 EZSEFL4 (DFSEZS termination flag)
word 5 -- bytes 0-1 not used
      bytes 2-3 External SubSys exit routine return code
words 6 through 7 not used
words 8 through 11 LCRETOKN (DFSLCRE UOW recovery token)
words 12 through 15 not used

```

### MOD ID = X'0297' SUB FUNC = X'000A' DFSESP30 External SubSys COMMIT VERIFY exit record

```

MOD ID   = X'0297'
SUB FUNC = X'000A' DFSESP30 External SubSys COMMIT VERIFY exit record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 PSTID (IMS dependent region ID)
      byte 2 EZSGFL (DFSEZS connection status byte1)
      byte 3 EZSLFL (DFSEZS connection status byte2)
word 4 -- byte 0 EZSEFL1 (DFSEZS thread startup status)
      byte 1 EZSEFL2 (DFSEZS thread commit status)
      byte 2 EZSEFL3 (DFSEZS thread termination status)
      byte 3 EZSEFL4 (DFSEZS termination flag)
word 5 -- bytes 0-1 not used
      bytes 2-3 External SubSys exit routine return code
words 6 through 7 not used
words 8 through 11 LCRETOKN (DFSLCRE UOW recovery token)
words 12 through 15 not used

```

### MOD ID = X'0307' SUB FUNC = X'0016' DFSFESP0 External SubSys commit processor entry record

```

MOD ID   = X'0307'
SUB FUNC = X'0016' DFSFESP0 External SubSys commit processor entry record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 PSTID (IMS dependent region ID)
      byte 2 EZSGFL (DFSEZS connection status byte1)
      byte 3 EZSLFL (DFSEZS connection status byte2)
word 4 -- byte 0 EZSEFL1 (DFSEZS thread startup status)
      byte 1 EZSEFL2 (DFSEZS thread commit status)
      byte 2 EZSEFL3 (DFSEZS thread termination status)

```

```

word 5 -- byte 3 EZSEFL4 (DFSEZS termination flag)
        byte 0 PSTFUNCT (IDLI function code)
        byte 1 PSTSYNFC (sync function code)
        byte 2 SSTTFGT1 (DFSSSOB termination flag)
        byte 3 not used
word 6 -- bytes 0-1 SSTTCOMP (DFSSSOB user completion bytes 2,3)
        byte 2 LCREFL1 (DFSLCRE status indicators)
        byte 3 LCREFL2 (DFSLCRE region connection status)
word 7 -- byte 0 LCREFL3 (DFSLCRE thread status)
        byte 1 LCREFL4 (DFSLCRE internal resource manager status)
        byte 2 LCREESSST (DFSLCRE ESS resource manager status byte1)
        byte 3 LCREESSF (DFSLCRE ESS resource manager status byte2)
words 8 through 11 RRETOKEN (DFSRRE UOW recovery token)
word 12 -- bytes 0-1 RRECI (DFSRRE commit indicator)
        bytes 2-3 not used
words 13 through 15 not used

```

## SUB FUNC = X'0017' DFSFESPO External SubSys commit processor exit record

SUB FUNC = X'0017' DFSFESPO External SubSys commit processor exit record

```

word 2 -- External SubSystem name
word 3 -- bytes 0-1 PSTID (IMS dependent region ID)
        byte 2 EZSGFL (DFSEZS connection status byte1)
        byte 3 EZSLFL (DFSEZS connection status byte2)
word 4 -- byte 0 EZSEFL1 (DFSEZS thread startup status)
        byte 1 EZSEFL2 (DFSEZS thread commit status)
        byte 2 EZSEFL3 (DFSEZS thread termination status)
        byte 3 EZSEFL4 (DFSEZS termination flag)
word 5 -- byte 0 PSTFUNCT (IDLI function code)
        byte 1 PSTSYNFC (sync function code)
        byte 2 SSTTFGT1 (DFSSSOB termination flag)
        byte 3 not used
word 6 -- bytes 0-1 SSTTCOMP (DFSSSOB user completion bytes 2,3)
        byte 2 LCREFL1 (DFSLCRE status indicators)
        byte 3 LCREFL2 (DFSLCRE region connection status)
word 7 -- byte 0 LCREFL3 (DFSLCRE thread status)
        byte 1 LCREFL4 (DFSLCRE internal resource manager status)
        byte 2 LCREESSST (DFSLCRE ESS resource manager status byte1)
        byte 3 LCREESSF (DFSLCRE ESS resource manager status byte2)
words 8 through 11 RRETOKEN (DFSRRE UOW recovery token)
word 12 -- bytes 0-1 RRECI (DFSRRE commit indicator)
        bytes 2-3 not used
words 13 through 15 not used

```

## SUB FUNC = X'0018' DFSFESPO External SubSys commit processor Resolve

SUB FUNC = X'0018' DFSFESPO External SubSys commit processor Resolve  
In Doubt requested record

```

word 2 -- External SubSystem name
word 3 -- bytes 0-1 PSTID (IMS dependent region ID)
        byte 2 EZSGFL (DFSEZS connection status byte1)
        byte 3 EZSLFL (DFSEZS connection status byte2)
word 4 -- byte 0 EZSEFL1 (DFSEZS thread startup status)
        byte 1 EZSEFL2 (DFSEZS thread commit status)
        byte 2 EZSEFL3 (DFSEZS thread termination status)
        byte 3 EZSEFL4 (DFSEZS termination flag)
word 5 -- byte 0 PSTFUNCT (IDLI function code)
        byte 1 PSTSYNFC (sync function code)
        byte 2 SSTTFGT1 (DFSSSOB termination flag)
        byte 3 not used
word 6 -- bytes 0-1 SSTTCOMP (DFSSSOB user completion bytes 2,3)
        byte 2 LCREFL1 (DFSLCRE status indicators)
        byte 3 LCREFL2 (DFSLCRE region connection status)
word 7 -- byte 0 LCREFL3 (DFSLCRE thread status)
        byte 1 LCREFL4 (DFSLCRE internal resource manager status)
        byte 2 LCREESSST (DFSLCRE ESS resource manager status byte1)
        byte 3 LCREESSF (DFSLCRE ESS resource manager status byte2)
words 8 through 11 RRETOKEN (DFSRRE UOW recovery token)
word 12 -- bytes 0-1 RRECI (DFSRRE commit indicator)
        bytes 2-3 not used
words 13 through 15 not used

```

## MOD ID = X'0506' SUB FUNC = X'0006' DFSESPR0 External SubSys PROGRAM REQUEST HANDLER

```
MOD ID   = X'0506'
SUB FUNC = X'0006' DFSESPR0 External SubSys PROGRAM REQUEST HANDLER
                      record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 PSTID (IMS dependent region ID)
      byte 2 EZSGFL (DFSEZS connection status byte1)
      byte 3 EZSLFL (DFSEZS connection status byte2)
word 4 -- byte 0 EZSEFL1 (DFSEZS thread startup status)
      byte 1 EZSEFL2 (DFSEZS thread commit status)
      byte 2 EZSEFL3 (DFSEZS thread termination status)
      byte 3 EZSEFL4 (DFSEZS termination flag)
word 5 -- bytes 0-1 not used
      bytes 2-3 External SubSys exit routine return code
words 6 through 7 not used
words 8 through 11 LCRETOKN (DFSLCRE UOW recovery token)
words 12 through 15 not used
```

## SUB FUNC = X'0019' DFSESPR0 External SubSys PROGRAM REQUEST recursive

```
SUB FUNC = X'0019' DFSESPR0 External SubSys PROGRAM REQUEST recursive
                      call record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 PSTID (IMS dependent region ID)
      byte 2 EZSGFL (DFSEZS connection status byte1)
      byte 3 EZSLFL (DFSEZS connection status byte2)
word 4 -- byte 0 EZSEFL1 (DFSEZS thread startup status)
      byte 1 EZSEFL2 (DFSEZS thread commit status)
      byte 2 EZSEFL3 (DFSEZS thread termination status)
      byte 3 EZSEFL4 (DFSEZS termination flag)
word 5 -- bytes 0-1 not used
      bytes 2-3 External SubSys exit routine return code
words 6 through 7 not used
words 8 through 11 LCRETOKN (DFSLCRE UOW recovery token)
words 12 through 15 not used
```

## SUB FUNC = X'0020' DFSESPR0 External SubSys NOT OPERATIONAL (SNOX) exit

```
SUB FUNC = X'0020' DFSESPR0 External SubSys NOT OPERATIONAL (SNOX) exit
                      record

word 2 -- External SubSystem name
word 3 -- bytes 0-1 PSTID (IMS dependent region ID)
      byte 2 EZSGFL (DFSEZS connection status byte1)
      byte 3 EZSLFL (DFSEZS connection status byte2)
word 4 -- byte 0 EZSEFL1 (DFSEZS thread startup status)
      byte 1 EZSEFL2 (DFSEZS thread commit status)
      byte 2 EZSEFL3 (DFSEZS thread termination status)
      byte 3 EZSEFL4 (DFSEZS termination flag)
word 5 -- bytes 0-1 not used
      bytes 2-3 External SubSys exit routine return code
words 6 through 7 not used
words 8 through 11 LCRETOKN (DFSLCRE UOW recovery token)
words 12 through 15 not used
```

“Type-1 trace table interface” on page 574 shows an example of an external subsystem trace with both X'57' and X'58' record IDs. The ESS trace is called the subsystem (SST) trace in a dump.

## Example of an external subsystem trace (SST)

```
*****
***TRACE PRINTED FROM OLDEST TO MOST CURRENT ENTRY**
*****
FUNCTION      WORD 0      WORD 1      WORD 2      WORD 3      WORD 4      WORD 5      WORD 6      WORD 7
ESI5 CTL INIT  5700198F  04030019  F1F0F0F1  00000000  00000000  00000000  00000000  00000000
ESI3 IDENT    570019B8  04020020  F1F0F0F1  00000800  00000000  00000000  00000000  00000000
ESS4 MESSAGE   570019BD  00150015  F1F0F0F1  00000000  00000000  00000000  00000000  00000000
ESI3 R-I-D     570019C6  04020021  F1F0F0F1  00002C00  00000000  00000000  00000000  00000000
ESS3 LOGGING   570019CF  00160014  F1F0F0F1  00000000  00000000  00000000  00000000  00000000
ESCT CRT THRD  58003165  02900005  F1F0F0F1  0001CC0C  81000000  00000000  00000000  00000000
FESP SYNC STA  580035D0  03070016  F1F0F0F1  0001CC0C  8C100000  42048000  03F00000  00000000
```



```

ESI3 RRE REQ      570035EC 04020023 F1F0F0F1 00008C00 00000000 00000000 00000000 00000000
ESI3 XS ECHO      570035F1 04020024 F1F0F0F1 00008C00 00000000 00000000 00000000 00000000
ESI3 R-I-D        570035F6 04020021 F1F0F0F1 00008C00 00000000 00000000 00000000 00000000
ESS3 LOGGING      57003608 00160014 F1F0F0F1 00000000 00000000 00000000 00000000 00000000
ESCT CRT THRD     58003A8F 02900005 F1F0F0F1 0001CC0C 81000000 00000000 00000000 00000000
FESP SYNC STA     58003AA1 03070016 F1F0F0F1 0001CC0C 8C100000 01080000 00000000 00000000
ESP1 COM PREP     58003AC8 02940008 F1F0F0F1 0001CC0C 8C500000 00000000 00000000 00000000
FESP SYNC END     58003ACB 03070017 F1F0F0F1 0001CC0C 8CD00000 01080000 00000080 02940000
FESP SYNC STA     58003B1A 03070016 F1F0F0F1 0001CC0C 8CD00000 010C0000 00002080 00000000
ESP2 COM CONT     58003B3D 02950009 F1F0F0F1 0001CC0C 8CD40000 00000000 00000000 00000000
FESP SYNC END     58003B44 03070017 F1F0F0F1 0001CC0C 9CCC0000 010C0000 000020C0 02950000
FESP SYNC STA     58003BA3 03070016 F1F0F0F1 0001CC0C 9CCC0000 42080000 00000000 00000000
FESP SYNC END     58003BA4 03070017 F1F0F0F1 0001CC0C 9CCC0000 42080000 00000080 02950000
FESP SYNC STA     58003BDF 03070016 F1F0F0F1 0001CC0C 9CCC0000 420C0000 00002080 00000000
ESD5 TRM THRD     58003BE7 02910002 F1F0F0F1 0001CC0C 9CCC0000 00000000 00000000 00000000
FESP SYNC END     58003BF1 03070017 F1F0F0F1 0001CC0C 95000C00 420C0000 00002080 00000000

```

```

GLOBAL ESET PREFIX
BLOCK AT 00BED480

```

```

PGES 00BED4A4 PLES 00000000 SCDAD 00BEA2B0 PCPE 00000000 ESGL
PICT 00000001 POCT 00000001 00000000

```

```

*** GLOBAL ESET BLOCK ***

```

```

00BED4A4 00000000 0059E9C0 00BED480 F1F0F0F1 40404040 E2E8E2F1 C4E2D5D4 C9D5F1F0
00BED4C4 40404040 40404040 D9F14040 0FC4E2D7 00B4DB40 001547C0 00A0C4C0 80B4DB57
00BED4E4 0FC4E2D7 00B4DB40 00153868 80A0C550 80B4DB57 108021DE 00000022 0059F9C8
00BED504 00000000 00000000 00000000 00000000 00005628 005B85A0 FF412B0C 00000000
00BED524 8C000000 009DC078 0059F998

```

## Resource Recovery Services trace

The Resource Recovery Services trace (RRST) provides information about relevant z/OS Resource Recovery Services (RRS) events in the IMS dependent region. Use the trace under direction of IBM Software Support when problems are suspected in the RRS area.

### About this task

You enable the Resource Recovery Services trace by using the **/TRACE SET ON TABLE RRST** command. When you specify **OPTION LOG**, IMS writes the trace externally as type X'67FA' records.

### Related concepts

[“Diagnostics for use with synchronous APPC and OTMA with shared queues” on page 364](#)

Synchronous APPC and OTMA message processing in the shared queues environment introduces additional diagnostic considerations for the message flow.

### Related reference

[“IMS type-1 trace function codes” on page 577](#)

You use trace function codes to help you diagnose performance problems or other problems with IMS.

## Format of Resource Recovery Services trace records

z/OS Resource Recovery Services trace (RRST) records contain standard fields, and all RRS calls are associated with subfunction codes.

### About this task

Word 0	-- byte 1	One-byte trace ID field. This byte indicates the type of the trace entry.
	byte 2	One-byte trace sub function code.
	byte 3-4	Two-byte trace sequence number assigned by the IMS trace component.
Word 1	-- byte 1	One-byte numeric Resource Recovery Service call code (see the RRS call table below).
	byte 2	One-byte LCREGFLG.
	byte 3-4	Two-byte RRS return code.
Word 2	-- byte 1-2	Two-byte PST number - PSTPSTNR.
	byte 3-4	Not used.
Word 3	-- byte 1-4	Four-byte LCRERRSF.
Word 4	-- Not used.	

Word 5 -- Not used.  
Word 6 -- Not used.  
Word 7 -- byte 1-4 Bytes 3 through 6 of the system clock (STCK) at  
the time the trace entry was created.

The following table shows the RRS calls that are associated with the subfunction codes:

*Table 183. Resource Recovery Services calls associated with the subfunction codes*

<b>Subfunction code</b>	<b>Function</b>
X'00'	ATRBAC
X'01'	ATRCMT
X'02'	ATRDINT
X'03'	ATREINT
X'04'	ATREINT5
X'05'	ATREND
X'06'	ATRIERS
X'07'	ATRIERS
X'08'	ATRIRLN
X'09'	ATRIRNI
X'0A'	ATRIRRI
X'0B'	ATRISLN
X'0C'	ATRPDUE
X'0D'	ATRREIC
X'0E'	ATRRURD
X'0F'	ATRRWID
X'10'	ATRSROI
X'11'	ATRSIT
X'12'	ATRSPID
X'13'	ATRSUSI2
X'14'	CRGDRM
X'15'	CRGGRM
X'16'	CRGSEIF
X'17'	CRXSEIF
X'18'	CTXBEGC
X'19'	CTXEINT1
X'1A'	CTXDINT
X'1B'	CTXENDC
X'1C'	CTXSWCH
X'1D'	CTXSCID
X'1E'	CTXSDTA
X'1F'	IEANTCR
X'20'	IEANTCT

The following table shows the RRS function routines that are associated with the DFSRRSI function routine codes (FRCs).

*Table 184. Resource Recovery Services function routines associated with DFSRRSI function routine codes*

<b>Function routine codes</b>	<b>Function routine</b>
X'01'	Register
X'02'	Restart
X'03'	End_Restart
X'04'	Unregister
X'05'	Switch_Context
X'06'	Determine_Syncpt_Coord
X'07'	Initiate_Syncpt
X'08'	End_Context
X'09'	Retain_Interest
X'0A'	Post_Deferred_UR
X'0B'	Disassociate_Context
X'0C'	Coordinate_Backout
X'0D'	Perform_Syncpt
X'0E'	Identify_Context
X'0F'	Post_Deferred_Backout
X'10'	Unhook_for_Phase2
X'11'	RRS_Validation
X'12'	Delete_UR_Interest
X'13'	Retrieve_XID
X'14'	Determine_Batch_Coord
X'15'	Create_Context
X'16'	Set_Side_Information
X'17'	Create_Cascaded_UR
X'18'	Express_UR_Interest
X'19'	Commit_UR
X'1A'	Backout_UR
X'1B'	Associate_Context
X'1C'	Application_Abend
X'A6'	Enter Commit (DFSRGFS0)
X'A7'	Exit Commit (DFSRGFS0)
X'A8'	RRS Error Occurred (DFSRGFS0)
X'A9'	RRS Abend Occurred (DFSRGFS0)
X'AA'	Token Trace

The following diagrams show the format of the trace records. Each trace record has a trace function code of X'A5' and is X'20' bytes long.

Subfunction Code	= X'00'
Description	= Resource Recovery Services - ATRBACK
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG - byte 3-4 - RRS return code
Word 2	- byte 1-2 - PSTPSTNR - byte 3-4 - Not used
Word 3	- LCRERRSF
Word 4	- Not used
Words 5-6	- LCURIDNT
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'01'
Description	= Resource Recovery Services - ATRCMIT
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG - byte 3-4 - RRS return code
Word 2	- byte 1-2 - PSTPSTNR - byte 3-4 - Not used
Word 3	- LCRERRSF
Word 4	- Not used
Words 5-6	- LCURIDNT
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'02'
Description	= Resource Recovery Services - ATRDINT
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG - byte 3-4 - RRS return code
Word 2	- byte 1-2 - PSTPSTNR - byte 3-4 - Not used
Word 3	- LCRERRSF
Word 4	- Not used
Words 5-6	- LCURIDNT
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'03'
Description	= Resource Recovery Services - ATREINT
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG - byte 3-4 - RRS return code
Word 2	- byte 1-2 - PSTPSTNR - byte 3-4 - Not used
Word 3	- LCRERRSF
Word 4	- Not used
Words 5-6	- LCURIDNT
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'04'
Description	= Resource Recovery Services - ATREINT5
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG - byte 3-4 - RRS return code
Word 2	- byte 1-2 - PSTPSTNR - byte 3-4 - Not used
Word 3	- LCRERRSF
Word 4	- Not used
Words 5-6	- LCURIDNT
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'05'
Description	= Resource Recovery Services - ATREND
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG - byte 3-4 - RRS return code
Word 2	- byte 1-2 - PSTPSTNR

Word 3	- byte 3-4 - Not used
Word 4	- Not used
Word 5	- Not used
Word 6	- Not used
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'06'
Description	= Resource Recovery Services - ATRIBRS
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG
Word 2	- byte 3-4 - RRS return code - byte 1-2 - PSTPSTNR
Word 3	- byte 3-4 - Not used
Word 4	- LCRERRSF
Word 5	- Not used
Word 6	- Not used
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'07'
Description	= Resource Recovery Services - ATRIERs
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG
Word 2	- byte 3-4 - RRS return code - byte 1-2 - PSTPSTNR
Word 3	- byte 3-4 - Not used
Word 4	- LCRERRSF
Word 5	- Not used
Word 6	- Not used
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'08'
Description	= Resource Recovery Services - ATRIRLN
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG
Word 2	- byte 3-4 - RRS return code - byte 1-2 - PSTPSTNR
Word 3	- byte 3-4 - Not used
Word 4	- LCRERRSF
Word 5	- Not used
Word 6	- Not used
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'09'
Description	= Resource Recovery Services - ATRIRNI
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG
Word 2	- byte 3-4 - RRS return code - byte 1-2 - PSTPSTNR
Word 3	- byte 3-4 - Not used
Word 4	- LCRERRSF
Word 5	- Not used
Word 6	- Not used
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'0A'
Description	= Resource Recovery Services - ATRIRRI
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG
Word 2	- byte 3-4 - RRS return code - byte 1-2 - PSTPSTNR
Word 3	- byte 3-4 - Not used
Word 4	- LCRERRSF
Word 5	- Not used
Word 6	- Not used
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'0B'
Description	= Resource Recovery Services - ATRISLN
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG
Word 2	- byte 3-4 - RRS return code - byte 1-2 - PSTPSTNR
Word 3	- byte 3-4 - Not used
Word 4	- LCRERRSF
Word 5	- Not used
Word 6	- Not used
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'0C'
Description	= Resource Recovery Services - ATRPDUE
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG
Word 2	- byte 3-4 - RRS return code - byte 1-2 - PSTPSTNR
Word 3	- byte 3-4 - Not used
Word 4	- LCRERRSF
Word 5	- byte 1-2 - ATRPDUEEXITNUMBER
Word 6	- byte 3-4 - ATRPDUECOMPLETION
Word 7	- Not used
	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'0D'
Description	= Resource Recovery Services - ATRREIC
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG
Word 2	- byte 3-4 - RRS return code - byte 1-2 - PSTPSTNR
Word 3	- byte 3-4 - Not used
Word 4	- LCRERRSF
Words 5-6	- Not used
Word 7	- LCURCNTX
	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'0E'
Description	= Resource Recovery Services - ATRRURD
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG
Word 2	- byte 3-4 - RRS return code - byte 1-2 - PSTPSTNR
Word 3	- byte 3-4 - Not used
Word 4	- LCRERRSF
Words 5-6	- Not used
Word 7	- IMS_PCTASK_URI_TOKEN
	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'0F'
Description	= Resource Recovery Services - ATRRWID
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG
Word 2	- byte 3-4 - RRS return code - byte 1-2 - PSTPSTNR
Word 3	- byte 3-4 - Not used
Word 4	- LCRERRSF
Words 5-6	- Not used
Word 7	- URI_Token
	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'10'
Description	= Resource Recovery Services - ATRSR0I
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG
Word 2	- byte 3-4 - RRS return code - byte 1-2 - PSTPSTNR
	- byte 3-4 - Not used

Word 3	- LCRERRSF
Word 4	- Not used
Word 5	- Not used
Word 6	- URID
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'11'
Description	= Resource Recovery Services - ATRSIT
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG - byte 3-4 - RRS return code
Word 2	- byte 1-2 - PSTPSTNR - byte 3-4 - Not used
Word 3	- LCRERRSF
Word 4	- Not used
Word 5	- Not used
Word 6	- Not used
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'12'
Description	= Resource Recovery Services - ATRSPID
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 0 - LCREGFLG - byte 3-4 - RRS return code
Word 2	- byte 1-2 - PSTPSTNR - byte 3-4 - Not used
Word 3	- LCRERRSF
Word 4	- Not used
Words 5-6	- IMS_PCTASK_URI_TOKEN
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'13'
Description	= Resource Recovery Services - ATRSUSI2
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG - byte 3-4 - RRS return code
Word 2	- byte 1-2 - PSTPSTNR - byte 3-4 - Not used
Word 3	- LCRERRSF
Word 4	- Not used
Words 5-6	- IMS_PCTASK_RUI_TOKEN
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'14'
Description	= RRMS Registration Services - CRGDRM
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG - byte 3-4 - RRS return code
Word 2	- byte 1-2 - PSTPSTNR - byte 3-4 - Not used
Word 3	- LCRERRSF
Word 4	- Not used
Word 5	- Not used
Word 6	- Not used
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'15'
Description	= RRMS Registration Services - CRGGRM
Word 1	- byte 1 - Numeric DFSRRSI function routine code - byte 2 - LCREGFLG - byte 3-4 - RRS return code
Word 2	- byte 1-2 - PSTPSTNR - byte 3-4 - Not used
Word 3	- LCRERRSF
Word 4	- Not used
Word 5	- Not used
Word 6	- Not used
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'16'
Description	= RRMS Registration Services - CRGSEIF

Word 1	- byte 1 - Numeric DFSRRSI function routine code
	- byte 2 - LCREGFLG
Word 2	- byte 3-4 - RRS return code
	- byte 1-2 - PSTPSTNR
Word 3	- byte 3-4 - Not used
Word 4	- LCRERRSF
Word 5	- Not used
Word 6	- Not used
Word 7	- Not used
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'17'
Description	= RRMS Registration Services - CRXSEIF
Word 1	- byte 1 - Numeric DFSRRSI function routine code
	- byte 2 - LCREGFLG
Word 2	- byte 3-4 - RRS return code
	- byte 1-2 - PSTPSTNR
Word 3	- byte 3-4 - Not used
Word 4	- LCRERRSF
Word 5	- Not used
Word 6	- Not used
Word 7	- Not used
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'18'
Description	= RRMS Registration Services - CTXBEGC
Word 1	- byte 1 - Numeric DFSRRSI function routine code
	- byte 2 - LCREGFLG
Word 2	- byte 3-4 - RRS return code
	- byte 1-2 - PSTPSTNR
Word 3	- byte 3-4 - Not used
Word 4	- LCRERRSF
Word 5	- Not used
Words 5-6	- LCURCNTX
Word 7	- Not used
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'19'
Description	= RRMS Context Services - CTXEINT1
Word 1	- byte 1 - Numeric DFSRRSI function routine code
	- byte 2 - LCREGFLG
Word 2	- byte 3-4 - RRS return code
	- byte 1-2 - PSTPSTNR
Word 3	- byte 3-4 - Not used
Word 4	- LCRERRSF
Word 5	- Address(LCRE)
Word 6	- Not used
Word 7	- Not used
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'1A'
Description	= RRMS Context Services - CTXDINT
Word 1	- byte 1 - Numeric DFSRRSI function routine code
	- byte 2 - LCREGFLG
Word 2	- byte 3-4 - RRS return code
	- byte 1-2 - PSTPSTNR
Word 3	- byte 3-4 - Not used
Word 4	- LCRERRSF
Word 5	- Address(IMS_PC_CI-Token)
Word 6	- Not used
Word 7	- Not used
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'1B'
Description	= RRMS Context Services - CTXENDC
Word 1	- byte 1 - Numeric DFSRRSI function routine code
	- byte 2 - LCREGFLG
Word 2	- byte 3-4 - RRS return code
	- byte 1-2 - PSTPSTNR
Word 3	- byte 3-4 - Not used
Word 4	- LCRERRSF
Word 4	- Not used



Words 5-6	- LCURCNTX
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'1C'
Description	= RRMS Context Services - CTXSWCH
Word 1	- byte 1 - Numeric DFSRRSI function routine code
	- byte 2 - LCREGFLG
	- byte 3-4 - RRS return code
Word 2	- byte 1-2 - PSTPSTNR
	- byte 3-4 - Not used
Word 3	- LCRERRSF
Word 4	- Not used
Words 5-6	- LCURCNTX
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'1D'
Description	= RRMS Context Services - CTXSCID
Word 1	- byte 1 - Numeric DFSRRSI function routine code
	- byte 2 - LCREGFLG
	- byte 3-4 - RRS return code
Word 2	- byte 1-2 - PSTPSTNR
	- byte 3-4 - Not used
Word 3	- LCRERRSF
Word 4	- Address(LCRE)
Word 5	- Not used
Word 6	- Not used
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'1E'
Description	= ODBA Set Context Data - CTXSDTA
Word 1	- byte 1 - Numeric DFSRRSI function routine code
	- byte 2 - LCREGFLG
	- byte 3-4 - RRS return code
Word 2	- byte 1-2 - PSTPSTNR
	- byte 3-4 - Not used
Word 3	- LCRERRSF
Word 4	- Not used
Word 5	- Not used
Word 6	- Not used
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'1F'
Description	= MVS Name/Token Services - IEANTCR
Word 1	- byte 1 - Numeric DFSRRSI function routine code
	- byte 2 - LCREGFLG
	- byte 3-4 - RRS return code
Word 2	- byte 1-2 - PSTPSTNR
	- byte 3-4 - Not used
Word 3	- LCRERRSF
Word 4	- Not used
Word 5	- Not used
Word 6	- Not used
Word 7	- Low 4 byte time stamp (STCK)

Subfunction Code	= X'20'
Description	= MVS Name/Token Services - IEANTRT
Word 1	- byte 1 - Numeric DFSRRSI function routine code
	- byte 2 - LCREGFLG
	- byte 3-4 - RRS return code
Word 2	- byte 1-2 - PSTPSTNR
	- byte 3-4 - Not used
Word 3	- LCRERRSF
Word 4	- Not used
Word 5	- Not used
Word 6	- Not used
Word 7	- Low 4 byte time stamp (STCK)

## RRST trace examples

The RRST trace in an OTMA and an APPC environment are depicted. The traces were gathered with tracing volume set to HIGH.

### About this task

#### RRST trace in an OTMA environment

```
CONTROL  CNTL STOPAFT=EOF
*****
* INPUT LOG DATA SET NAME(S): *
* DARIO.IMS1.OLDSP0.OTMU01.DECKS2
*****
*
* SELECTION FOR INTERNAL TRACE LOG RECORD(S) *
*
OPTION PRINT 0=5,V=67FA,L=2,E=DFSERA60,C=E
1 FUNCTION WORD 0 WORD 1 WORD 2 WORD 3 WORD 4 WORD 5 WORD 6 WORD 7
PAGE 0001
0* RR1 TRACE TABLE - DATE 2004173 TIME 224754462929 OFFSET 028D SKIP 0000 TOTAL SKIP 00000000 RECORD NUMBER 0000028F
-DFSSLUM0 Exit 7B027F86 01C90000 00000000 00000000 00000000 00000000 00000000 BB676D0D B53E27A0 SYNCHRONOUS OUTPUT LU
MANAGER
DFSRLM10 Exit 7B0281E8 06400000 C9D4E2F2 40404040 00000000 00000000 BB676D7A 2B82CF89 RECEIVE LU MANAGER
RECEIVER
DFSSLUM0 Exit 7B0282FD 01C90000 00000000 00000000 00000000 00000000 BB676DBE 97EF7323 SYNCHRONOUS OUTPUT LU
MANAGER
DFSRLM10 Exit 7B02856E 06400000 C9D4E2F3 40404040 00000000 00000000 BB676E36 DD321202 RECEIVE LU MANAGER
RECEIVER
DFSRGFS0 (RRS) A0A68972 480005A6 0CB252F8 0CB25060 00010000 00010000 BB676EB9 17C56A2A ENTER COMMIT
CTXEINT1 (RRS) A5188973 15000000 00010000 00000000 00000000 00000000 BB676EB9 B917C5DD CREATE_CONTEXT
[NO LCRE FLAGS
ATREINT5 (RRS) A5048974 18000000 00010000 04000000 00000000 BB676EB9 7E71E000 B917C977 EXPRESS_UR_INTRST
[NO LCRE FLAGS
DFSRGFS0 (RRS) A0A78975 480005A7 0CB252F8 0CB25060 00000001 00000000 BB676EB9 17ED934A EXIT COMMIT
DFSRGFS0 (RRS) A0AA8976 480005AA 0001E4D9 7E71E000 7E71E000 01000000 17ED934A TKN TRACE (A0A7)
DFSYTIB0 (RRS) 5A00897D 28010084 0CB25060 7E71E000 7E71E000 01000000 0100001E B917EEAD INPT MSG ENQUEUED(UR
TOKEN WORD3-6)
CTXDINT (RRS) A5198C04 05000000 00010000 00000000 0AFBA048 19070000 00000000 BEB54D12 SWITCH_CONTEXT
[NO LCRE FLAGS
ATREINT (RRS) A5038C05 05000000 00010000 40000000 00000000 BB676EBE 7E71E374 BEB54EDC SWITCH_CONTEXT
[NO LCRE FLAGS
ATTRURD (RRS) A50D8C88 06400000 00010000 60000000 00000000 BB676EBE B54E530B BF5AF676 DETRMN_SYNC_COORD
[OUTPUT SENT
DFSRGFS0 (RRS) A0A68CF7 480003A6 0CB252F8 0AFF5060 00010000 00010001 BB676EBF 5BACC466 ENTER COMMIT
ATRDINT (RRS) A5028CF8 12000000 00010000 20000000 00000000 00000000 00000000 BF5BAD3B DELETE_UR_INTRST
[NO LCRE FLAGS
DFSRGFS0 (RRS) A0A78CF9 480003A7 0CB252F8 0AFF5060 00000001 00000000 BB676EBF 5BB19926 EXIT COMMIT
DFSRGFS0 (RRS) A0AA8CFA 480003AA 0001C3E7 BB676EB9 17C86FAA 01000000 022A4060 5BB19926 TKN TRACE (A0A7)
DFSRGFS0 (RRS) A0A68D0A 480001A6 0CB252F8 0AFF5060 00010000 00010002 BB676EBF 5BB30B26 ENTER COMMIT
CTXSCID (RRS) A51C8D0B 19000000 00010000 00000000 00000000 BB676EB9 17C86FAA BF5BB31D COMMIT_UR
[NO LCRE FLAGS
ATRCMIT (RRS) A5018D0C 19000000 00010000 00080000 00000000 00000000 00000000 BF5BB422 COMMIT_UR
[NO LCRE FLAGS
DFSRGFS0 (RRS) A0A78D0D 480001A7 0CB252F8 0AFF5060 00000001 00000000 BB676EBF 5BFA1C26 EXIT COMMIT
DFSRGFS0 (RRS) A0AA8D0E 480001AA 0001C3E7 BB676EB9 17C86FAA 01000000 022A4060 5BFA1C26 TKN TRACE (A0A7)
ATRDINT (RRS) A5028D5E 08400000 00010000 62002001 00000000 BB676EBE 7E71E374 BF5C4396 END_CONTEXT
[OUTPUT SENT
CTXENDC (RRS) A51A8D5F 08400000 00010000 62002001 00000000 BB676EBE B54E9F6B BF5C4602 END_CONTEXT
[OUTPUT SENT
DFSRGFS0 (RRS) A0A68F9F 480005A6 0CB252F8 0CB25060 00010000 00010003 BB676ED7 2204448D ENTER COMMIT
CTXEINT1 (RRS) A5188FA0 15000000 00010000 00000000 00000000 00000000 00000000 D72204AF CREATE_CONTEXT
[NO LCRE FLAGS
ATREINT5 (RRS) A5048FA1 18000000 00010000 04000000 00000000 BB676ED7 7E71E000 D72206A2 EXPRESS_UR_INTRST
[NO LCRE FLAGS
DFSRGFS0 (RRS) A0A78FA2 480005A7 0CB252F8 0CB25060 00000001 00000000 BB676ED7 220DB20D EXIT COMMIT
DFSRGFS0 (RRS) A0AA8FA3 480005AA 0001E4D9 7E71E000 7E71E000 01000002 0100001E 220DB20D TKN TRACE (A0A7)
DFSYTIB0 (RRS) 5A008FAA 28010084 0CB25060 7E71E000 7E71E000 01000002 0100001E D7223231 INPT MSG ENQUEUED(UR
TOKEN WORD3-6)
DFSASW0 (RRS) 7B0890AB 47000000 0B27C060 40C1D6E2 0CB252F8 00000000 BB676ED8 D8D3326F APPC/OTMA SMQ AWE
server
DFSAPPC0 Exit 7B0290AC 02400000 00000000 0CB25060 00000000 00000000 BB676ED8 D8D3630F DFSAPPC MSG SWITCH
PROCESSOR
DFSSLUM0 Exit 7B0290AD 01800000 00000002 00000000 00000000 00000000 BB676ED8 D8DD6EAF SYNCHRONOUS OUTPUT LU
MANAGER
DFSYTIB0 (RRS) 5A0090B5 2802A084 0CB25060 7E71E000 7E71E000 01000002 0100001E D8D8E13F BE RESPONSE RECVD(UR
TOKEN WORD3-6)
DFSYLUS0 (RRS) 5A0091F1 2A060002 0CB25060 7E71E000 7E71E000 01000002 0100001E D8F4361F OTMA SERVICE:UNK FUNC
DFSRGFS0 (RRS) A0A691FA 480001A6 0CB252F8 0CB25060 00020000 00010000 BB676ED8 F4376B81 ENTER COMMIT
CTXSCID (RRS) A51C91FB 19000000 00020000 00000000 00000000 BB676ED7 22060A4D D8F437D0 COMMIT_UR
[NO LCRE FLAGS
ATRCMIT (RRS) A50191FC 19000000 00020000 00080000 00000000 00000000 00000000 D8F439BA COMMIT_UR
[NO LCRE FLAGS
DFSRGFS0 (RRS) A0A791FD 480001A7 0CB252F8 0CB25060 00000001 00000000 BB676ED8 FC093A68 EXIT COMMIT
DFSRGFS0 (RRS) A0AA91FE 480001AA 0002E4D9 7E71E000 7E71E000 01000002 0100001E FC093A68 TKN TRACE (A0A7)
DFSRGFS0 (RRS) A0A69401 480005A6 0CB252F8 0CB25060 00010000 00010004 BB676EFE B7210502 ENTER COMMIT
CTXEINT1 (RRS) A5189402 15000000 00010000 00000000 00000000 00000000 00000000 FEB72191 CREATE_CONTEXT
[NO LCRE FLAGS
ATREINT5 (RRS) A5049403 18000000 00010000 04000000 00000000 BB676EFE 7E71E000 FEB7242F EXPRESS_UR_INTRST
```

```

|NO LCRE FLAGS
DFSRGFSO (RRS) A0A79404 480005A7 0CB252F8 0CB25060 00000001 00000000 BB676EFE B72C3502 EXIT COMMIT
DFSRGFSO (RRS) A0AA9405 480005AA 0001E4D9 7E71E000 7E71E000 01000004 0100001E B72C3502 TKN TRACE (A0A7)
DFSYTIBO (RRS) 5A00940C 28010084 0CB25060 7E71E000 7E71E000 01000004 0100001E FEB7314C INPT MSG ENQUEUED(UR
TOKEN WORD3-6)
DFAOSW0 (RRS) 7B0895E1 47000000 0B27C060 40C1D6E2 0CB252F8 00000000 BB676F0A 3B00B4A3 APPC/OTMA SMQ AWE
server
DFSAPPC0 Exit 7B0295E2 02400000 00000000 0CB25060 00000000 00000000 BB676F0A 3B00E323 DFSAPPC MSG SWITCH
PROCESSOR
DFSSLUM0 Exit 7B0295E3 01800000 00000003 00000000 00000000 00000000 BB676F0A 3B0ABC23 SYNCHRONOUS OUTPUT LU
MANAGER
DFSYTIBO (RRS) 5A0095EB 2802A084 0CB25060 7E71E000 7E71E000 01000004 0100001E 0A3B0E44 BE RESPONSE RECVD(UR
TOKEN WORD3-6)
DFSYLUS0 (RRS) 5A0096BC 2A060002 0CB25060 7E71E000 7E71E000 01000004 0100001E 0A7AD931 OTMA SERVICE:UNK FUNC
DFSRGFSO (RRS) A0A696C5 480001A6 0CB252F8 0CB25060 00020000 00010000 BB676F0A 7ADED120 ENTER COMMIT
1 FUNCTION WORD 0 WORD 1 WORD 2 WORD 3 WORD 4 WORD 5 WORD 6
7 PAGE 0002
-CTXSCID (RRS) A51C96C6 19000000 00020000 00000000 00000000 BB676EFE B7236902 0A7ADF43 COMMIT_UR
|NO LCRE FLAGS
ATRCMIT (RRS) A50196C7 19000000 00020000 00080000 00000000 00000000 00000000 0A7AE4B4 COMMIT_UR
|NO LCRE FLAGS
DFSRGFSO (RRS) A0A79739 480001A7 0CB252F8 0CB25060 00000001 00000000 BB676F0A D26D5C0E EXIT COMMIT
DFSRGFSO (RRS) A0AA973A 480001AA 0002E4D9 7E71E000 7E71E000 01000004 0100001E D26D5C0E TKN TRACE (A0A7)
DFSRGFSO (RRS) A0A69881 480005A6 0CB252F8 0CB25060 00010000 00010005 BB676F22 71395349 ENTER COMMIT
CTXEINT1 (RRS) A5189882 15000000 00010000 00000000 00000000 00000000 00000000 227139DC CREATE_CONTEXT
|NO LCRE FLAGS
ATREINT5 (RRS) A5049883 18000000 00010000 04000000 00000000 BB676F22 7E71E000 22713C07 EXPRESS_UR_INTRST
|NO LCRE FLAGS
DFSRGFSO (RRS) A0A79884 480005A7 0CB252F8 0CB25060 00000001 00000000 BB676F22 71439009 EXIT COMMIT
DFSRGFSO (RRS) A0AA9885 480005AA 0001E4D9 7E71E000 7E71E000 01000006 0100001E 71439009 TKN TRACE (A0A7)
DFSYTIBO (RRS) 5A00988C 28010080 0CB25060 7E71E000 7E71E000 01000006 0100001E 22714D7A INPT MSG ENQUEUED(UR
TOKEN WORD3-6)
DFAOSW0 (RRS) 7B08999C 47000000 0B27C060 40C1D6E2 0CB252F8 00000000 BB676F22 F5CA6326 APPC/OTMA SMQ AWE
server
DFSAPPC0 Exit 7B02999D 02400000 00000000 0CB25060 00000000 00000000 BB676F22 F5CA8E46 DFSAPPC MSG SWITCH
PROCESSOR
DFSSLUM0 Exit 7B02999E 01800000 00000004 00000000 00000000 00000000 BB676F22 F5D4A386 SYNCHRONOUS OUTPUT LU
MANAGER
DFSYTIBO (RRS) 5A0099A6 2802C080 0CB25060 7E71E000 7E71E000 01000006 0100001E 22F5D812 BE RESPONSE RECVD(UR
TOKEN WORD3-6)
DFSYLUS0 (RRS) 5A0099B9 2A060002 0CB25060 7E71E000 7E71E000 01000006 0100001E 22F5DEAF OTMA SERVICE:UNK FUNC
DFSRGFSO (RRS) A0A699C5 480001A6 0CB252F8 0CB25060 00030000 00010000 BB676F22 F5F4C026 ENTER COMMIT
CTXSCID (RRS) A51C99C6 19000000 00030000 00000000 00000000 BB676F22 713B6A69 22F5F53D COMMIT_UR
|NO LCRE FLAGS
ATRCMIT (RRS) A50199C7 19000000 00030000 00080000 00000000 00000000 00000000 22F5F72B COMMIT_UR
|NO LCRE FLAGS
DFSRGFSO (RRS) A0A799CA 480001A7 0CB252F8 0CB25060 00000001 00000000 BB676F22 FBFA8107 EXIT COMMIT
DFSRGFSO (RRS) A0AA99CB 480001AA 0003E4D9 7E71E000 7E71E000 01000006 0100001E FBFA8107 TKN TRACE (A0A7)
DFSRGFSO (RRS) A0A69C13 480005A6 0CB252F8 0CB25060 00010000 00010006 BB676F52 B618084F ENTER COMMIT
CTXEINT1 (RRS) A5189C14 15000000 00010000 00000000 00000000 00000000 00000000 52B6187C CREATE_CONTEXT
|NO LCRE FLAGS
ATREINT5 (RRS) A5049C15 18000000 00010000 04000000 00000000 BB676F52 7E71E000 52B61858 EXPRESS_UR_INTRST
|NO LCRE FLAGS
DFSRGFSO (RRS) A0A79C16 480005A7 0CB252F8 0CB25060 00000001 00000000 BB676F52 B6230D2F EXIT COMMIT
DFSRGFSO (RRS) A0AA9C17 480005AA 0001E4D9 7E71E000 7E71E000 01000008 0100001E B6230D2F TKN TRACE (A0A7)
DFSYTIBO (RRS) 5A009C1E 28010084 0CB25060 7E71E000 7E71E000 01000008 0100001E 52B62C94 INPT MSG ENQUEUED(UR
TOKEN WORD3-6)
DFAOSW0 (RRS) 7B089E2A 47000000 0B27C060 40C1D6E2 0CB252F8 00000000 BB676F5C AB73B56C APPC/OTMA SMQ AWE
server
DFSAPPC0 Exit 7B029E2B 02400000 00000000 0CB25060 00000000 00000000 BB676F5C AB73E68C DFSAPPC MSG SWITCH
PROCESSOR
DFSSLUM0 Exit 7B029E2C 01800000 00000005 00000000 00000000 00000000 BB676F5C AB7BB62C SYNCHRONOUS OUTPUT LU
MANAGER
DFSYTIBO (RRS) 5A009E34 2802A084 0CB25060 7E71E000 7E71E000 01000008 0100001E 5CAB7F1D BE RESPONSE RECVD(UR
TOKEN WORD3-6)
DFSYLUS0 (RRS) 5A009F94 2A060002 0CB25060 7E71E000 7E71E000 01000008 0100001E 5CD77072 OTMA SERVICE:UNK FUNC
DFSRGFSO (RRS) A0A69F9D 480001A6 0CB252F8 0CB25060 00020000 00010000 BB676F5C D771B08A ENTER COMMIT
CTXSCID (RRS) A51C9F9E 19000000 00020000 00000000 00000000 BB676F52 B61A728F 5CD7720F COMMIT_UR
|NO LCRE FLAGS
ATRCMIT (RRS) A5019F9F 19000000 00020000 00080000 00000000 00000000 00000000 5CD773D6 COMMIT_UR
|NO LCRE FLAGS
DFSRGFSO (RRS) A0A79FA0 480001A7 0CB252F8 0CB25060 00000001 00000000 BB676F5D 26DB750C EXIT COMMIT
DFSRGFSO (RRS) A0AA9FA1 480001AA 0002E4D9 7E71E000 7E71E000 01000008 0100001E 26DB750C TKN TRACE (A0A7)
DFS707I END OF FILE ON INPUT
DFS708I OPTION COMPLETE
DFS703I END OF JOB

```

## RRST trace in an APPC environment

```

CONTROL CNTL STOPAFT=EOF
*****
* INPUT LOG DATA SET NAME(S): *
* DARIO.IMS1.OLDSP0.ASQA01.DECKS2
*****
*
* SELECTION FOR INTERNAL TRACE LOG RECORD(S) *
*
OPTION PRINT 0=5,V=67FA,L=2,E=DFSERA60,C=E
1 FUNCTION WORD 0 WORD 1 WORD 2 WORD 3 WORD 4 WORD 5 WORD 6 WORD
7 PAGE 0001
0* RR1 TRACE TABLE - DATE 2004173 TIME 232543999620 OFFSET 028D SKIP 0000 TOTAL SKIP 00000000 RECORD NUMBER 0000025E

```

## 622 IMS: Diagnosis

```

DFSSLUM0 Exit 7B028C5B 01800000 00000005 00000000 00000000 00000000 00000000 BB6777D2 E4567063 SYNCHRONOUS OUTPUT LU
MANAGER
DFS6LUS0 (RRS) 7B098C8D 0A000001 0CB25700 7E71E000 7E71E000 01000006 0100001E 00000000 LU62 SERVICES
INTERFACE 1
DFSRGFS0 (RRS) A0A68C96 480001A6 0CB25998 0CB25700 00030000 00010000 BB6777D2 E4A89B69 ENTER COMMIT
CTXSCID (RRS) A51C8C97 19000000 00030000 00000000 00000000 BB6777D2 31746F2A D2E4A90C COMMIT_UR
[NO LCRE FLAGS
ATRCMIT (RRS) A5018C98 19000000 00030000 00080000 00000000 00000000 00000000 D2E4BD92 COMMIT_UR
[NO LCRE FLAGS
DFSRGFS0 (RRS) A0A78C99 480001A7 0CB25998 0CB25700 00000001 00000000 BB6777D2 EF0C5785 EXIT COMMIT
DFS6LUS0 (RRS) A0A8C9A 480001AA 0003E4D9 7E71E000 7E71E000 01000006 0100001E EF0C5785 TKN TRACE (A0A7)
DFS6LUS0 (RRS) A0A68D66 480005A6 0CB25998 0CB25700 00010000 00010004 BB6777D7 0AB06B68 ENTER COMMIT
CTXEINT1 (RRS) A5188D67 15000000 00010000 00000000 00000000 00000000 00000000 D70AB0E6 CREATE_CONTEXT
[NO LCRE FLAGS
ATREINT5 (RRS) A5048D68 18000000 00010000 04000000 00000000 BB6777D7 7E71E000 D70AB49F EXPRESS_UR_INTRST
[NO LCRE FLAGS
DFS6LUS0 (RRS) A0A78D69 480005A7 0CB25998 0CB25700 00000001 00000000 BB6777D7 0ABC5A28 EXIT COMMIT
DFS6LUS0 (RRS) A0AA8D6A 480005AA 0001E4D9 7E71E000 7E71E000 01000008 0100001E 0ABC5A28 TKN TRACE (A0A7)
DFSRLM10 (RRS) 7B008D71 06000000 0CB25700 7E71E000 7E71E000 01000008 0100001E 00000000 RECEIVE LU MANAGER
RECEIVER
DFS6LUS0 (RRS) 7B088EE9 47000000 0BB80060 40C1D6E2 0CB25998 00000000 BB6777D7 A738EA6F APPC/OTMA SMQ AWE
server
DFSAPPC0 Exit 7B028EEA 02400000 00000000 0CB25700 00000000 00000000 BB6777D7 A739192F DFSAPPC MSG SWITCH
PROCESSOR
DFS6LUS0 Exit 7B028EEB 01800000 00000006 00000000 00000000 00000000 BB6777D7 A744892F SYNCHRONOUS OUTPUT LU
MANAGER
DFS6LUS0 (RRS) 7B098F1D 0A000001 0CB25700 7E71E000 7E71E000 01000008 0100001E 00000000 LU62 SERVICES
INTERFACE 1
DFS6LUS0 (RRS) A0A68F26 480001A6 0CB25998 0CB25700 00020000 00010000 BB6777D7 A7D2356F ENTER COMMIT
CTXSCID (RRS) A51C8F27 19000000 00020000 00000000 00000000 BB6777D7 0AB3A848 D7A7D2B2 COMMIT_UR
[NO LCRE FLAGS
ATRCMIT (RRS) A5018F28 19000000 00020000 00080000 00000000 00000000 00000000 D7A7D4D7 COMMIT_UR
[NO LCRE FLAGS
DFS6LUS0 (RRS) A0A78F29 480001A7 0CB25998 0CB25700 00000001 00000000 BB6777D7 B160BD29 EXIT COMMIT
DFS6LUS0 (RRS) A0AA8F2A 480001AA 0002E4D9 7E71E000 7E71E000 01000008 0100001E B160BD29 TKN TRACE (A0A7)
CTXDINT (RRS) A5198FD5 0E000000 00000000 00000000 0C1833D8 19070000 00000000 DD59F8EF IDENTIFY_CONTEXT
[NO LCRE FLAGS
DFS6LUS0 (RRS) A0A6901F 480004A6 0CB25998 0CB25700 00010000 00010005 BB6777DD 5B634549 ENTER COMMIT
ATREINT5 (RRS) A5049020 18000000 00010000 04000000 00000000 BB6777DD 7E71E374 DD5B63AD EXPRESS_UR_INTRST
[NO LCRE FLAGS
DFS6LUS0 (RRS) A0A79021 480004A7 0CB25998 0CB25700 00000001 00000000 BB6777DD 5B699BA9 EXIT COMMIT
DFS6LUS0 (RRS) A0AA9022 480004AA 0001E4D9 7E71E374 7E71E374 01000002 0100001E 5B699BA9 TKN TRACE (A0A7)
DFSRLM10 (RRS) 7B009029 06000000 0CB25700 7E71E374 7E71E374 01000002 0100001E 00000000 RECEIVE LU MANAGER
RECEIVER
DFS6LUS0 (RRS) 7B0891A1 47000000 0BB80060 40C1D6E2 0CB25998 00000000 BB6777DD E519D12A APPC/OTMA SMQ AWE
server
DFSAPPC0 Exit 7B0291A2 02400000 00000000 0CB25700 00000000 00000000 BB6777DD E519FD2A DFSAPPC MSG SWITCH
PROCESSOR
DFS6LUS0 Exit 7B0291A3 01800000 00000007 00000000 00000000 00000000 BB6777DD E5241B2A SYNCHRONOUS OUTPUT LU
MANAGER
DFS6LUS0 (RRS) 7B0991E8 0A000001 0CB25700 7E71E374 7E71E374 01000002 0100001E 00000000 LU62 SERVICES
INTERFACE 1
DFS6LUS0 (RRS) A0A691F1 480001A6 0CB25998 0CB25700 00050000 00010000 BB6777DD E935F58C ENTER COMMIT
CTXSCID (RRS) A51C91F2 19000000 00050000 00000000 00000000 BB6777DD 591A1C0A DDE93669 COMMIT_UR
[NO LCRE FLAGS
ATRCMIT (RRS) A50191F3 19000000 00050000 00080000 00000000 00000000 00000000 DDE93885 COMMIT_UR
[NO LCRE FLAGS
DFS6LUS0 (RRS) A0A79272 480001A7 0CB25998 0CB25700 00000001 00000000 BB6777DD FC906981 EXIT COMMIT
DFS6LUS0 (RRS) A0AA9273 480001AA 0005E4D9 7E71E374 7E71E374 01000002 0100001E FC906981 TKN TRACE (A0A7)
CTXSWCH (RRS) A51B9299 08000000 00000000 00002000 00000000 BB6777DD 591A1C0A DE02A604 END_CONTEXT
[NO LCRE FLAGS
DFS707I END OF FILE ON INPUT
DFS708I OPTION COMPLETE
DFS703I END OF JOB

```

## RRST entries for OTMA modules

The formats of OTMA trace entries for z/OS Resource Recovery Services related events are shown in the following diagrams.

### TRACE ID = X'5A00'

```

TRACE ID = X'5A00'
Word 1          - byte 0 - 2A, module number for DFSYLUS0
                  - byte 1 - 01, OTMA GU was invoked
                  - byte 2 - DLAFLAG1
                  - byte 3 - DLAFLAG4
Word 2          - Back-end YTIB CLB address
Words 3-6       - RRS parent UR token
Word 7          - Time stamp (short)

```

### TRACE ID = X'5A00'

```

TRACE ID = X'5A00'
Word 1          - byte 0 - 2A, module number for DFSYLUS0

```

	- byte 1 - 02, Fastpaht GU was invoked
	- byte 2 - DLAFLAG1
	- byte 3 - DLAFLAG4
Word 2	- Front-end IMS YTIB CNT address
Words 3-6	- RRS parent UR token
Word 7	- Time stamp (short)

#### **TRACE ID = X'5A00'**

TRACE ID = X'5A00'	
Word 1	- byte 0 - 2A, module number for DFSYLUS0
	- byte 1 - 03, Back-end IMS issued a DFS2224 message
	- byte 2 - DLAFLAG1
	- byte 3 - DLAFLAG4
Word 2	- Front-end IMS YTIB CLB address (CNT address for fastpath transaction)
Words 3-6	- RRS parent UR token
Word 7	- Time stamp (short)

#### **TRACE ID = X'5A00'**

TRACE ID = X'5A00'	
Word 1	- byte 0 - 2D, module number for DFSYSLM0
	- byte 1 - 01, back-end XCF send succeed
	- byte 2 - AOS_FLAGS
	- byte 3 - 0
Word 2	- Back-end YTIB address
Words 3-6	- RRS parent UR token
Word 7	- Time stamp (short)

#### **TRACE ID = X'5A00'**

TRACE ID = X'5A00'	
Word 1	- byte 0 - 2D, module number for DFSYSLM0
	- byte 1 - 02, back-end XCF send failed
	- byte 2 - AOS_FLAGS
	- byte 3 - 0
Word 2	- Back-end YTIB address
Words 3-6	- RRS parent UR token
Word 7	- Time stamp (short)

#### **TRACE ID = X'5A00'**

TRACE ID = X'5A00'	
Word 1	- byte 0 - 25, module number for DFSYPSI0
	- byte 1 - 01, OTMA Protected Trans was submitted
	- byte 2 - 0
	- byte 3 - 0
Word 2	- Front-end YTIB CLB address
Words 3-6	- Context token
Word 7	- Time stamp (short)

#### **TRACE ID = X'5A00'**

TRACE ID = X'5A00'	
Word 1	- byte 0 - 28, module number for DFSYTIB0
	- byte 1 - 01, OTMA input message is about to enqueue
	- byte 2 - 0
	- byte 3 - YTIB_MSG_TYPE
Word 2	- Front-end YTIB CLB address
Words 3-6	- RRS parent UR token
Word 7	- Time stamp (short)

#### **TRACE ID = X'5A00'**

TRACE ID = X'5A00'	
Word 1	- byte 0 - 28, module number for DFSYTIB0
	- byte 1 - 02, response from back-end is received
	- byte 2 - YTIB_MSG_STATUS_3
	- byte 3 - YTIB_MSG_TYPE
Word 2	- Front-end YTIB CLB address
Words 3-6	- RRS parent UR token
Word 7	- Time stamp (short)

## z/OS Resource Recovery Services entries logged by OTMA modules

RRS entries logged by OTMA modules are depicted.

### About this task

#### TRACE ID = X'5A00'

```
TRACE ID = X'5A00'
Word 1          - byte 0 - 2A, module number for DFSYLUS0
                  - byte 1 - 06, XCF_good_Send or - 05, XCF_bad_Send
Word 2          - A(ECB)
Words 3-6      - LUP_UR_TOKEN (AWRRURTK)
Word 7          - Time stamp (short)
```

## Trace entries logged by z/OS Resource Recovery Services related modules

Trace entry records for DFSRGSF0 and TOKEN Tracing are depicted.

#### TRACE ID = X'A0A6' DFSRGSF0 Entry record

```
TRACE ID = X'A0A6' DFSRGSF0 Entry record
Word 1          - byte 0 - 72, module number for DFSRGSF0
                  - bytes 1-2 - Function, SEE AWRRFUNC
                  - byte 3 - A6 Module entry
Word 2          - A(TIB)
Word 3          - A(ECB) - from AWRRECB
Word 4          - bytes 0-2 - PST number
Word 5          - Number of queued AWEs
Word 7          - Time stamp (STCK)
```

#### TRACE ID = X'A0A7' DFSRGSF0 Exit record

```
TRACE ID = X'A0A7' DFSRGSF0 Exit record
Word 1          - byte 0 - 72, module number for DFSRGSF0
                  - bytes 1-2 - Function, SEE AWRRFUNC
                  - byte 3 - A7 Module exit
Word 2          - A(TIB)
Word 3          - A(ECB) - from AWRRECB
Word 4          - bytes 0-1 - Highest number of AWEs
                  - bytes 2-3 - Number greater of AWEs than TCBs
Word 5          - Trace return code (AWRRETCD)
Word 7          - Time stamp (STCK)
```

```
TRACE ID = X'A0A8' DFSRGSF0 Error Occurred
```

```
TRACE ID = X'A0A9' DFSRGSF0 ABEND Occurred
```

#### TRACE ID = X'A0AA' TOKEN Tracing record

```
TRACE ID = X'A0AA' TOKEN Tracing record
Word 1          - byte 0 - 72, module number for DFSRGSF0
                  - bytes 1-2 - Function, SEE AWRRFUNC
                  - byte 3 - AA - Token trace record
Word 2          - bytes 0-1 - PST Number
                  - bytes 2-3 - Token id (CX=Context,IN=Interest,
                               UR=Unit of recovery)
Words 4-7      - TOKEN
```

## Scheduler trace

When you use the **/TRACE SET ON TABLE SCHD** command, IMS enables the scheduler trace. When you specify **OPTION LOG**, IMS sends these entries to the log as type X'67FA' records.

### Scheduler trace record formats

The following code samples show the formats of the scheduler trace records for function codes X'41' through .

#### Scheduler trace record format for function code X'41'

```
TRACE ID   = X'41'
word 0 - byte 1 - X'41' Scheduling starts, traced by DFSSBMP0
          byte 2 - PST number
          bytes 3-4 - Trace sequence number
word 1 - SCHD must be addressable by caller
word 2 - Reserved
word 3 - SAPCNTRL
words 4-5 - Reserved
word 6 - Module identifier
word 7 - Store clock value
```

#### Scheduler trace record format for function code X'42'

```
TRACE ID   = X'42'
word 0 - byte 1 - X'42' Block mover, traced by DFSSBMP0,
          DFSSBMP0, DFSSMSC0
          byte 2 - PST number
          bytes 3-4 - Trace sequence number
word 1 - byte 1 - PDIRCODE
          byte 2 - PDIROPTC
          byte 3 - PSTSCHDF
          byte 4 - PSTCODE1
word 2 - PSTPSB
word 3 - PSTSMB
words 4-5 - Reserved
word 6 - Module identifier
word 7 - Store clock value
```

#### Scheduler trace record format for function code X'43'

```
TRACE ID   = X'43'
word 0 - byte 1 - X'43' Scheduling ends
          byte 2 - PST number
          bytes 3-4 - Trace sequence number
word 1 - PSTABTRM
word 2 - PSTPSB
word 3 - SAPCNTRL
words 4-5 - Reserved
word 6 - Module identifier
word 7 - Store clock value
```

#### Scheduler trace record format for function code X'44'

```
TRACE ID   = X'44'
word 0 - byte 1 - X'44' IRC started
          byte 2 - PST number
          bytes 3-4 - Trace sequence number
word 1 - SSIMCOMP
word 2 - Reserved
word 3 - SAPCNTRL
words 4-5 - Reserved
word 6 - Module identifier
word 7 - Store clock value
```



### Scheduler trace record format for function code X'45'

```
TRACE ID   = X'45'
word 0 - byte 1 - X'45' TMS00 started
          byte 2 - PST number
          bytes 3-4 - Trace sequence number
word 1 - A(PST)
word 2 - Sync point function code (COMMIT/P1/P2/BACKOUT)
word 3 - Caller of TMS00
word 4 - TPI (first four bytes)
word 5 - TPI (last four bytes)
word 6 - Module identifier
word 7 - Store clock value
```

### Scheduler trace record format for function code X'46'

```
TRACE ID   = X'46'
word 0 - byte 1 - X'46' TMS00 finished
          byte 2 - PST number
          bytes 3-4 - Trace sequence number
word 1 - A(PST)
word 2 - Sync point function code (COMMIT/P1/P2/BACKOUT)
word 3 - Return code
word 4 - TPI (first four bytes)
word 5 - TPI (last four bytes)
word 6 - Module identifier
word 7 - Store clock value
```

### Scheduler trace record format for function code X'47'

```
TRACE ID   = X'47'
word 0 - byte 1 - X'47' APPC extract call made
          byte 2 - PST number
          bytes 3-4 - Trace sequence number
word 1 - Function code (FPRETRY/PUSER)
word 2 - Abend code (PSTABTRM)
word 3 - Return code from DFSTMR00
word 4 - Return code from APPC extract call
word 5 - Reserved
word 6 - Module identifier
word 7 - Store clock value
```

### Scheduler trace record format for function code X'48'

```
TRACE ID   = X'48'
word 0 - byte 1 - X'48' Scheduling failed
          byte 2 - PST number
          bytes 3-4 - Trace sequence number
word 1 - byte 1 - PDIRCODE
          byte 2 - PDIROPTC
          byte 3 - PSTSCHDF
          byte 4 - PSTCODE1
word 2 - PSTPSB
word 3 - PSTSMB
words 4-5 - Reserved
word 6 - Module identifier
word 7 - Store clock value
```

### Scheduler trace record format for function code X'49'

```
TRACE ID   = X'49'
word 0 - byte 1 - X'49' Schedule Serial Program
          byte 2 - PST number
          bytes 3-4 - Trace sequence number
word 1 - Address of the PSB
word 2 - RM update return code
word 3 - RM update reason code
word 4 - Address RM parameter list
word 5 - Address RM List Header
word 6 - Module ID 'SPM0'
word 7 - Store clock value
```

### Scheduler trace record format for function code X'4A'

TRACE ID = X'4A'

- word 0 - byte 1 - X'4A' Release Serial Program
- byte 2 - PST number
- bytes 3-4 - Trace sequence number
- word 1 - Address of the PSB
- word 2 - RM delete return code
- word 3 - RM delete reason code
- word 4 - Address RM parameter list
- word 5 - Address RM List Header
- word 6 - Module ID 'SPM0'
- word 7 - Store clock value

### Scheduler trace record format for function code X'4B'

TRACE ID = X'4B'

- word 0 - byte 1 - X'4B' Release Notify
- byte 2 - PST number
- bytes 3-4 - Trace sequence number
- word 1 - Address of the PSB
- word 2 - Address of Tran name
- word 3 - SCI delete return code
- word 4 - SCI delete reason code
- word 5 - Address Notify message area
- word 6 - Module ID 'SPM0'
- word 7 - Store clock value

### Scheduler trace record format for function code X'4C'

TRACE ID = X'4C'

- word 0 - byte 1 - X'4C' Catalog PSB Dynamic Attach
- byte 2 - PST number
- bytes 3-4 - Trace sequence number
- word 1 - bytes 1-2 - PDIR code
- byte 3 - Schedule failure code
- byte 4 - Schedule special processing code
- word 2 - Address of the PSB
- word 3 - Address of the catalog PDIR
- words 4-5 - Reserved
- word 6 - Module ID
- word 7 - Store clock value

### Example of a scheduler trace

```
*****
***TRACE PRINTED FROM OLDEST TO MOST CURRENT ENTRY**
*****
FUNCTION      WORD 0      WORD 1      WORD 2      WORD 3
              WORD 4      WORD 5      WORD 6      WORD 7
BLOCK MOVER   4207E98A   44060000   16F90598   00800041
              00000000   00000000   00000000   F89569D5
SCHED END     4307E994   00000000   16F90598   00800001
              00000000   00000000   00000000   F8956BD3
SCHED START   4156F4D0   E2C3C8C4   16920060   00800001
              00000000   00000000   00000000   F89973E5
BLOCK MOVER   4256F4DE   44060000   170305E8   00800041
              00000000   00000000   00000000   F89979DA
SCHED END     4356F4E8   00000000   170305E8   00800001
              00000000   00000000   00000000   F8997B43
IRC START     44560737   00000000   00000000   00800001
              00000000   00000000   16CAF7A0   F8A95716
IRC START     4407077F   00000000   00000000   00800001
              00000000   00000000   16CAF7A0   F8A9CA44
SCHED START   4107078C   E2C3C8C4   15AB5060   00800001
              00000000   00000000   00000000   F8A9D45F
BLOCK MOVER   4207079A   44060000   16F90598   00800041
              00000000   00000000   00000000   F8A9DF19
SCHED END     430707A4   00000000   16F90598   00800001
              00000000   00000000   00000000   F8A9E0C4
SCHED START   417007B5   E2C3C8C4   15A48060   00800001
              00000000   00000000   00000000   F8AA4B87
```

BLOCK MOVER	42700804	44060000	16F91740	00800041
	00000000	00000000	00000000	F8AB0631
SCHED END	4370080E	00000000	16F91740	00800001
	00000000	00000000	00000000	F8AB07C2
IRC START	447008CE	00000000	00000000	00800001
	00000000	00000000	16CAF7A0	F8ABC593
SCHED START	417008DB	E2C3C8C4	15A48060	00800001
	00000000	00000000	00000000	F8ABDC00
BLOCK MOVER	427008E9	44060000	16F91740	00800041
	00000000	00000000	00000000	F8ABD209
SCHED END	437008F3	00000000	16F91740	00800001

### Related reference

[“IMS type-1 trace function codes” on page 577](#)

You use trace function codes to help you diagnose performance problems or other problems with IMS.

## Storage manager trace

The storage manager trace writes a record each time it is called to allocate a pool, get a buffer, or release a buffer. The storage manager traces requests from the following pools: AOIP, CESS, CIOP, CMDP, DYNP, EMHB, FPWP, HIOP, LUMC, LUMP, and SPAP.

You can enable the storage manager trace during IMS initialization with the STRG= option in the DFSVSMxx PROCLIB member, or online using the **/TRACE** command. The **/TRACE SET ON TABLE STRG** command activates the trace and sends the output to an internal trace table. When you specify OPTION LOG on the **/TRACE** command, IMS sends the output to the system log or external trace data set.

You can format the internal trace table using the Offline Dump Formatter under IPCS with either the **VERBX** command or the IMS Dump Formatter panels. To format the trace records, any storage manager control blocks, and pool storage, you can specify ALL as the POOL ID, as shown in the following example.

```
FMTIMS ...(POOL,NAME,ALL),...or you can specify FMTIMS (TRACE, NAME, SM).
```

To locate the storage manager trace in a formatted dump, look for eye catcher **\*\*SMTR**.

To locate the trace tables in an unformatted dump, look for the trace identifier SM in the trace table header record.

The following tables show the format of each storage manager trace record.

Table 185. TRACE ID = X'5F03' (get trace record)

WORD 0	WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7
Control Information	Pool Name	Variable Pool Size	Variable Pool Address Fixed Pool Upper Limit	0	Caller's Return Address	Return Code	0

Table 186. TRACE ID = X'5F04' (get trace record)

WORD 0	WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7
Control Information	Pool Name	Buffer Request Size	Buffer Address	Address of Caller's ECB	Caller's Return Address	Return Code	Current Pool Size

Table 187. TRACE ID = X'5F05' (release trace record)

WORD 0	WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7
Control Information	Pool Name	0	Buffer Address	Address of Caller's ECB	Caller's Return Address	Return Code	Current Pool Size

## Related concepts

[“Formatting IMS dumps offline” on page 512](#)

Two methods are available for formatting IMS dumps offline: interactive formatting, performed through a series of panels which provide formatting choices, and formatting by using JCL.

## Related reference

[“IMS type-1 trace function codes” on page 577](#)

You use trace function codes to help you diagnose performance problems or other problems with IMS.

## Latch trace

When you use the **/TRACE SET ON TABLE LATC** command, IMS traces events related to its internal serialization services (latch manager, use manager, and system locate control function) to an internal table.

The following table shows the general format of a latch trace entry.

*Table 188. Format of a latch trace entry*

WORD 0			WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7
I	S	SEQ NUM	ENTRY TYPE						

where:

### I

One-byte trace ID field. This byte indicates the type of the trace entry. It is always X'70' for latch trace entries.

### S

One-byte trace subtype field. This field is used for latch manager trace entries to denote the latch function being traced. It is not currently used for Use Manager trace entries.

### SEQ NUM

Two-byte trace sequence number assigned by the IMS trace component.

### ENTRY TYPE

For Use Manager trace entries only: 4-byte printable character string, indicating the type of entry.

Words 2 through 6 contain data specific to each trace entry.

## Related reference

[“IMS type-1 trace function codes” on page 577](#)

You use trace function codes to help you diagnose performance problems or other problems with IMS.

## Latch manager trace entries

The sub functions GET, GETU, REL, and RCOV are listed.

### **Sub Function: X'01' Get latch (GET)**

```
Sub Function: X'01' Get latch (GET)
Description:  Get a latch
  word  1  -- Caller's SAP address
  word  2  -- Latch name
  word  3  -- Caller's return address
  word  4  -- Resource header address
  word  5  -- 1st halfword = latch level;
             2nd halfword = flags from latch manager parmlist
  word  6/7 -- 8-byte STCK value
```

### **Sub Function: X'02' - Upgrade latch (GETU)**

```
Sub Function: X'02' - Upgrade latch (GETU)
Description: Upgrade a latch from shared to exclusive
word 1 -- Caller's SAP address
word 2 -- Latch name
word 3 -- Caller's return address
word 4 -- Resource header address
word 5 -- 1st halfword = latch level;
          2nd halfword = flags from latch manager parmlist
word 6/7 -- 8-byte STCK value
```

### **Sub Function: X'03' - Release latch (REL)**

```
Sub Function: X'03' - Release latch (REL)
Description: Release a latch
word 1 -- Caller's SAP address
word 2 -- Latch name
word 3 -- Caller's return address
word 4 -- Resource header address
word 5 -- 1st halfword = latch level;
          2nd halfword = flags from latch manager parmlist
word 6/7 -- 8-byte STCK value
```

### **Sub Function: X'04' - Recover latch (RCOV)**

```
Sub Function: X'04' - Recover latch (RCOV)
Description: Recover a latch
word 1 -- SAP, TCB, or ASCB address
word 2 -- Latch name
word 3 -- Caller's return address
word 4 -- 0
word 5 -- 1st halfword = latch level;
          2nd halfword = flags from latch manager parmlist
word 6/7 -- 8-byte STCK value
```

## **Use manager trace entries**

Eleven entry types are listed and described.

### **Entry Type: USE**

```
Entry Type: USE
Description: Inuse request trace entry
word 1 -- 'USE'
word 2 -- Block type
word 3 -- Call ID
word 4 -- Work ID
word 5 -- Block address
word 6 -- SAP address
word 7 -- Caller's return address
```

### **Entry Type: LOK**

```
Entry Type: LOK
Description: Lock request trace entry

word 1 -- 'LOK'
word 2 -- Block type
word 3 -- Call ID
word 4 -- Work ID
word 5 -- Block address
word 6 -- SAP address
word 7 -- Caller's return address
```

### **Entry Type: CON**

```
Entry Type: CON
Description: Connect request trace entry
```

```

word 1 -- 'CON'
word 2 -- Block type
word 3 -- Call ID
word 4 -- Work ID
word 5 -- Block address
word 6 -- SAP address
word 7 -- Caller's return address

```

### Entry Type: MRG

Entry Type: MRG  
Description: Merge request trace entry

```

word 1 -- 'MRG'
word 2 -- Block type
word 3 -- Call ID
word 4 -- Work ID
word 5 -- Block address
word 6 -- SAP address
word 7 -- Caller's return address

```

### Entry Type: INQ

Entry Type: INQ  
Description: Inquiry request trace entry

```

word 1 -- 'INQ'
word 2 -- Block type
word 3 -- Call ID
word 4 -- Work ID
word 5 -- Block address
word 6 -- SAP address
word 7 -- Caller's return address

```

### Entry Type: NUSE

Entry Type: NUSE  
Description: Nouse request trace entry

```

word 1 -- 'NUSE'
word 2 -- Block type
word 3 -- Call ID
word 4 -- Work ID
word 5 -- Block address
word 6 -- SAP address
word 7 -- Caller's return address

```

### Entry Type: NLOK

Entry Type: NLOK  
Description: Unlock request trace entry

```

word 1 -- 'NLOK'
word 2 -- Block type
word 3 -- Call ID
word 4 -- Work ID
word 5 -- Block address
word 6 -- SAP address
word 7 -- Caller's return address

```

### Entry Type: NCON

Entry Type: NCON  
Description: Disconnect request trace entry

```

word 1 -- 'NCON'
word 2 -- Block type
word 3 -- Call ID
word 4 -- Work ID
word 5 -- Block address

```

```
word 6 -- SAP address
word 7 -- Caller's return address
```

### Entry Type: RCOV (SAP level)

Entry Type: RCOV (SAP level)  
Description: Use recovery performed at the SAP (ITASK) level trace entry

```
word 1 -- 'RCOV'
word 2 -- 'SAP'
word 3 -- Block Type
word 4 -- SAP address
word 5 -- Ø
word 6 -- Ø
word 7 -- Caller's return address
```

### Entry Type: RCOV (TCB level)

Entry Type: RCOV (TCB level)  
Description: Use recovery performed at the TCB level trace entry

```
word 1 -- 'RCOV'
word 2 -- 'TCB'
word 3 -- Block Type
word 4 -- Ø
word 5 -- TCB address
word 6 -- Ø
word 7 -- Caller's return address
```

### Entry Type: RCOV (address space level)

Entry Type: RCOV (address space level)  
Description: Use recovery performed at the address space level trace entry

```
word 1 -- 'RCOV'
word 2 -- 'MEM'
word 3 -- Block Type
word 4 -- Ø
word 5 -- ASCB address
word 6 -- Ø
word 7 -- Caller's return address
```

## System locate control function entries

Entry types for SLC0, SLC1, and a latch trace example are depicted.

### Entry Type: SLC0

Entry Type: SLC0  
Description: Locate a block and issue a use manager inuse call against it

```
word 1 -- 'SLC0'
word 2 -- Block Type
word 3 -- Work ID
word 4 -- Call ID
word 5 -- ''
word 6 -- SAP address
word 7 -- Caller's return address
```

### Entry Type: SLC1

Entry Type: SLC1  
Description: Locate a block and issue a use manager nouse call against it

```
word 1 -- 'SLC1'
word 2 -- Block Type
word 3 -- Work ID
word 4 -- Call ID
word 5 -- ''
word 6 -- SAP address
word 7 -- Caller's return address
```

## Latch trace example

```
**LTR                                LATCH TRACE
*****
***TRACE PRINTED FROM OLDEST TO MOST CURRENT ENTRY**
*****
  FUNCTION      WORD 0      WORD 1      WORD 2      WORD 3      WORD 4      WORD 5      WORD 6      WORD 7
COMMON LATCH    70006A98    GET      QMGR      SHR      00005F28    00290000    065975F0    8004BABE
COMMON LATCH    70006A99    REL      QMGR      ANY      00005F28    00290000    065975F0    800EAA62
COMMON LATCH    70006A9A    GET      QMGR      SHR      00005F28    00290000    065975F0    8004BABE
COMMON LATCH    70006A9B    REL      QMGR      ANY      00005F28    00290000    065975F0    800EAA62
COMMON LATCH    70006A9C    GET      DCSL      SHR      05B581B0    00030000    065975F0    8004F2C4
COMMON LATCH    70006A9E    GET      LOGL      EXCL     05B58F70    002F0000    065975F0    85B0EED4
COMMON LATCH    70006A9F    REL      LOGL      EXCL     05B58F70    002F0000    065975F0    85B0E53C
COMMON LATCH    70006AA1    GET      QMGR      SHR      00005F28    00290000    065975F0    8004BABE
COMMON LATCH    70006AA2    REL      QMGR      ANY      00005F28    00290000    065975F0    800EAA62
COMMON LATCH    70006AA3    REL      DCSL      SHR      05B581B0    00030000    065975F0    80046012
COMMON LATCH    70006AA4    NUSE     ALLW      ....     05F66060    00000000    065975F0    06D2CCC2
COMMON LATCH    70006AA6    GET      LOGL      EXCL     05B58F70    002F0000    065975F0    85B0EED4
COMMON LATCH    70006AA7    REL      LOGL      EXCL     05B58F70    002F0000    065975F0    85B0E53C
COMMON LATCH    70006AAD    GET      LOGL      EXCL     05B58F70    002F0000    065975F0    85B0EED4
COMMON LATCH    70006AB2    REL      LOGL      EXCL     05B58F70    002F0000    065975F0    85B0E53C
COMMON LATCH    70006AB4    GET      TCTB      EXCL     05B71858    00130000    065975F0    85B5CB3A
COMMON LATCH    70006AB5    REL      TCTB      EXCL     05B71858    00130000    065975F0    85B5CD78
COMMON LATCH    70006AB6    GET      SMGT      EXCL     05C47288    002B0000    065975F0    85B0BAEA
COMMON LATCH    70006AB7    REL      SMGT      EXCL     05C47288    002B0000    065975F0    85B0BBB6
COMMON LATCH    70006AB8    GET      PDRB      EXCL     05BA9E90    00150000    065975F0    85B5AB26
COMMON LATCH    70006AB9    GET      PSBP      SHR      05B587A0    00160000    065975F0    85B5ABE6
COMMON LATCH    70006ABA    REL      PDRB      EXCL     05BA9E90    00150000    065975F0    85B5AED4
COMMON LATCH    70006ABB    REL      PSBP      ANY      05B587A0    00160000    065975F0    85B5AF90
COMMON LATCH    70006ABC    GET      SUBQ      SHR      05B71418    00200000    065975F0    85B4291E
COMMON LATCH    70006ABD    REL      SUBQ      SHR      05B71418    00200000    065975F0    85B42A60
COMMON LATCH    70006ABE    GET      SUBQ      SHR      05B71430    00200000    065975F0    85B4291E
COMMON LATCH    70006ABF    REL      SUBQ      SHR      05B71430    00200000    065975F0    85B42A60
COMMON LATCH    70006AC7    GET      QMGR      SHR      00005F28    00290000    06597790    8004BABE
COMMON LATCH    70006AC8    REL      QMGR      ANY      00005F28    00290000    06597790    800EAA62
COMMON LATCH    70006ACA    SLC0     LNBQ      .. -     C4D3C1F3    40404040    06597790    05B7BD2A
COMMON LATCH    70006ACB    GET      VLQB      SHR      00BD2230    00260000    06597790    800511A4
COMMON LATCH    70016ACC    USE      CNT      DLA3      05FB4060    07926568    06597790    05B312AE
COMMON LATCH    70006ACD    REL      VLQB      ANY      00BD2230    00260000    06597790    800511A4
COMMON LATCH    70006ACE    REL      SCHD      ANY      05B58660    00120000    06597790    85B60CB4
```

## Queue manager trace

The queue manager trace provides information about relevant queue manager functional and exceptional events. Use the trace under the direction of IBM support personnel when problems are suspected in the queue manager area.

### Turning on the queue manager trace

You can turn on the queue manager trace in two ways:

- During IMS online initialization with the QMGR parameter in the DFSVSMxx IMS.PROCLIB member.
- During online operation, with the **/TRACE** command.

You can specify trace output destination and tracing volume on both the QMGR parameter and the **/TRACE** command.

If you send output to the common trace table, you can format the table using the Offline Dump Formatter under IPCS, using either the **VERBX** command or the IMS Dump Formatter panels. If you send the output to an external data set, you can use the File Select and Formatting Print utility (DFSERA10) with exit routine DFSERA60 to format the trace entries.

To locate the queue manager trace in a formatted dump, look for eye catcher **\*\*QMGR**. To locate the trace table in an unformatted dump, look for the trace identifier **QM** in the trace table header record.

### Related reference

[“IMS type-1 trace function codes” on page 577](#)



You use trace function codes to help you diagnose performance problems or other problems with IMS.

## Format of trace records

The following diagrams show the format of the trace records. Each trace record has a trace function code of X'4E' and is X'20' bytes long.

### Subfunction codes

#### SC

##### FUNCTION

#### X'00'

GET PREFIX

#### X'01'

CANCEL INPUT

#### X'02'

GET UNIQUE

#### X'03'

GET NEXT

#### X'04'

DEQUEUE

#### X'05'

SAVE

#### X'06'

REJECT

#### X'07'

DELETE

#### X'08'

CANCEL OUTPUT (LOG)

#### X'09'

CANCEL OUTPUT (NOLOG)

#### X'0C'

ENQUEUE (FIFO)

#### X'0D'

ENQUEUE (LIFO)

#### X'0E'

REENQUEUE (FIFO)

#### X'0F'

REENQUEUE (LIFO)

#### X'10'

REPOSITION

#### X'11'

AOI COMMAND INPUT

#### X'12'

AOI MESSAGE TO MASTER

#### X'13'

AOI CANCEL UEHB

#### X'14'

AOI TERMINATION

#### X'17'

UNUSED OP CODE

**X'18'**  
UNUSED OP CODE

**X'19'**  
UNUSED OP CODE

**X'1A'**  
INSERT PREFIX

**X'1C'**  
CONDITIONAL ENQUEUE (FIFO)

**X'1D'**  
CONDITIONAL ENQUEUE (LIFO)

**X'1E'**  
TRANSFER

**X'1F'**  
NOTE/POINT

### Low level trace record format

**FUNCTION: See above listing**  
Subfunction Code: See above listing

word	0	-- Control information
word	1	-- A(ECB)
word	2	-- A(QTPPCB)
word	3	-- byte 1 - Current call type byte 2 - Prior call type byte 3 - (unused) byte 4 - (unused)
word	4	-- Caller's ID (WORD 1)
word	5	-- Caller's ID (WORD 2)
word	6	-- Unused (zero)
word	7	-- Time stamp

### Medium level trace record format - X'21'

"Medium level trace record format - X'21'" on page 636 depicts the trace (medium level) record format of the following function with subfunction code X'21':

**FUNCTION: EXIT FROM QUEUE MANAGER**  
Subfunction Code: X'21'

word	0	-- Control information
word	1	-- PCB Contents (WORD 1)
word	2	-- A(QTPPCB)
word	3	-- Return code
word	4	-- PCB contents (WORD 4)
word	5	-- PCB contents (WORD 5)
word	6	-- PCB contents (WORD 6)
word	7	-- Time stamp

### Medium level trace record format - X'20'

"Medium level trace record format - X'20'" on page 636 depicts the trace (medium level) record format of the following function with subfunction code X'20':

**FUNCTION: ENTRY TO QUEUE MANAGER**  
Subfunction Code: X'20'

word	0	-- Control information
word	1	-- PCB Contents (WORD 1)
word	2	-- A(QTPPCB)
word	3	-- PCB contents (WORD 3)
word	4	-- PCB contents (WORD 4)
word	5	-- PCB contents (WORD 5)
word	6	-- PCB contents (WORD 6)
word	7	-- Time stamp

### Medium level trace record format - X'22'

This figure depicts the trace (medium level) record format of the following function with subfunction code X'22':

**FUNCTION: Special- Not Applicable**

Subfunction Code: X'22'

```
word 0 -- Control information
word 1 -- Varies by use
word 2 -- Varies by use
word 3 -- Varies by use
word 4 -- Varies by use
word 5 -- Varies by use
word 6 -- Varies by use
word 7 -- Time stamp
```

### Low level trace record format - X'08', X'15', X'1B'

“Low level trace record format - X'08', X'15', X'1B'” on page 637 depicts the trace (low level) record format of the following functions with these subfunction codes (SC):

#### SC

##### FUNCTION

**X'08'**

**X'15'**

MESSAGE REROUTE

**X'1B'**

INSERT MOVE SPANNABLE

**FUNCTION:** See above list

Subfunction Code: See above list

```
word 0 -- Control information
word 1 -- A(ECB)
word 2 -- A(QTPPCB)
word 3 -- byte 1 - Current call type
           byte 2 - Prior call type
           byte 3 - (unused)
           byte 4 - (unused)
word 4 -- Caller's ID (WORD 1)
word 5 -- Caller's ID (WORD 2)
word 6 -- byte 1 - Length of user segment
           byte 2 - Length of user segment
           byte 3 - (unused)
           byte 4 - (unused)
word 7 -- Time stamp
```

### Low level trace record format - X'0A'

“Low level trace record format - X'0A'” on page 637 depicts the trace (low level) record format of the following function with subfunction code X'0A':

**FUNCTION:** INSERT LOCATE

Subfunction Code: X'0A'

```
word 0 -- Control information
word 1 -- A(ECB)
word 2 -- A(QTPPCB)
word 3 -- byte 1 - Current call type
           byte 2 - Prior call type
           byte 3 - (unused)
           byte 4 - (unused)
word 4 -- Caller's ID (WORD 4)
word 5 -- Caller's ID (WORD 2)
word 6 -- Length of requested message area
word 7 -- Time stamp
```

## Low level trace record format - X'16'

[“Low level trace record format - X'16'” on page 638](#) depicts the trace (low level) record format of the following function with subfunction code X'16':

```
FUNCTION: RELEASE
Subfunction Code: X'16'

word 0 -- Control information
word 1 -- A(ECB)
word 2 -- A(QTPPCB)
word 3 -- byte 1 - Current call type
           byte 2 - Prior call type
           byte 3 - (unused)
           byte 4 - (unused)
word 4 -- Caller's ID (WORD 1)
word 5 -- Caller's ID (WORD 2)
word 6 -- Contents of DECAREA
word 7 -- Time stamp
```

## Shared queues interface trace

The shared queues interface trace provides information about errors associated with the interface between IMS and CQS.

Examples of errors that are traced are:

- CQS Request errors
- CQS Inform errors
- Service errors
- Storage errors

Use this trace under the direction of IBM Software Support when problems are suspected in the interface between IMS and CQS.

You can turn on the shared queues interface trace in two ways:

- During IMS online initialization, with the SQTT parameter in the DFSVSMxx IMS.PROCLIB member.
- During online operation, with the **/TRACE** command.

Each trace entry is X'20' bytes long.

You can specify trace output destination and tracing volume on both the SQTT parameter and the **/TRACE** command.

The **/TRACE SET ON TABLE SQTT** command activates the trace and sends the output to an internal trace table that consists of 126 entries. If you specify OPTION LOG on the **/TRACE** command, IMS sends the output to the system log or an external trace data set in groups of 126. Other parameters control the volume of output.

You can format trace table entries with the Offline Dump Formatter under IPCS, using either the VERBX parameter or the IMS Dump Formatter panels. You can use the File Select and Formatting Print utility (DFSERA10) with exit routine DFSERA60 to format the trace entries written to an external data set.

To locate the shared queues interface trace in a dump, look for eye catcher \*\*SQTT.

To display the status of the trace, use the **/DISPLAY TRACE** command.

### Related reference

[“IMS type-1 trace function codes” on page 577](#)

You use trace function codes to help you diagnose performance problems or other problems with IMS.

## Fast Path trace

When you use the **/TRACE SET ON TABLE FPTT** command, IMS enables the Fast Path trace. The Fast Path trace resides in the internal IMS trace tables, with the **OPTION LOG** parameter causing the trace to also be written to the IMS logs.

If the **OPTION LOG** parameter is not specified (or the **OPTION NOLOG** parameter is specified), the trace resides only in the IMS internal trace tables and is formatted through the IMS Dump Formatter. If the **OPTION LOG** parameter is specified, the trace will also reside on the logs and can be formatted with DFSERA60 for log type X'67FA' or through the IMS Dump Formatter.

### Trace formats

Fast Path reserves X'9C' and X'9D' trace entries. X'9C' is reserved for tracing notifies and X'9D' is reserved for all other Fast Path traces.

### X'9C' trace format

The following table shows the format of the X'9C' trace entry. Following the table are the trace IDs and descriptions of content of trace entry.

Table 189. Format of the Fast Path X'9C' trace entry

Word 0	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7
aabbcccc	dddddddd	dddddddd	dddddddd	dddddddd	dddddddd	dddddddd	dddddddd

#### Trace ID

##### Description of Content of Trace Entry

#### aa

The FP Notify trace code, X'9C'

#### bb

The trace subcode, used so that each IMSFP trace entry has a unique code or subcode, avoiding duplication of the same trace function.

#### X'01'

DBFNOTM0 Entry

#### X'02'

NCB contents at entry to DBFNOTM0

#### X'03'

DBFNOTM0 NOTEXC (DFS LLM->IRLM)

#### X'04'

DBFNOTM0 IWAIT

#### X'05'

DBFNOTM0 after IWAIT

#### X'06'

DBFICLI0 Entry

#### X'07'

NCB contents at entry to DBFICLI0

#### X'08'

DBFICLI0 Response decrement EPSTNCTR

#### X'09'

DBFICLI0 IPOST

- X'0A'**  
DBFCSTS2 EPST Timeout Candidate
- X'0B'**  
DBFCSTS2 EPST Timeout IPOST
- cccc**  
The Trace Sequence Number
- dddddddd**  
Data, specific for each trace entry.

## X'9D' trace format

The following table shows the format of the X'9D' trace entry. Following the table are the trace IDs and descriptions of content of trace entry.

Table 190. Format of the Fast Path X'9D' trace entry							
Word 0	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7
aabbcccc	dddddddd	dddddddd	dddddddd	dddddddd	dddddddd	dddddddd	dddddddd
<div> <div>Trace ID</div> <div>Description of Content of Trace Entry</div> </div> <div> <div>aa</div> <div>The FP General trace code, X'9D'</div> </div> <div> <div>bb</div> <div>The FP latch trace subcodes</div> </div> <div> <div>X'01'</div> <div>DBFELOCK DMAC</div> </div> <div> <div>X'02'</div> <div>DBFELOCK DMCB</div> </div> <div> <div>X'03'</div> <div>DBFELOCK DSM</div> </div> <div> <div>X'04'</div> <div>DBFELOCK FLD</div> </div> <div> <div>X'05'</div> <div>DBFELOCK FNCB</div> </div> <div> <div>X'06'</div> <div>DBFELOCK MSDB</div> </div> <div> <div>X'07'</div> <div>DBFELOCK TRAT</div> </div> <div> <div>X'08'</div> <div>DBFELOCK VSO</div> </div> <div> <div>X'09'</div> <div>DBFELOCK VSTR</div> </div> <div> <div>X'0A'</div> <div>DBFELOCK XCRB</div> </div> <div> <div>X'10'</div> <div>Resource Latch</div> </div> <div> <div>X'11'</div> <div>DBFSYNL Latch</div> </div> <div> <div>X'12'</div> <div>DBFBUFL Latch</div> </div>							

**X'13'**

DBFEMHBL Latch

**X'14'**

DBFLATCH Latch

**X'15'**

DBFALOCK Latch

**X'16'**

DBFHLOCK Latch

**X'17'**

DBFPLOCK Latch

**cccc**

The Trace Sequence Number

**dddddddd**

Data, specific for each trace entry

**Related reference**

“IMS type-1 trace function codes” on page 577

You use trace function codes to help you diagnose performance problems or other problems with IMS.

## Fast Path trace entries

Fast Path trace entries are listed, including Trace ID, Module name, and Trace point.

**Trace entries for Fast Path**

The following table describes the Fast Path trace entries.

*Table 191. Fast Path trace entries*

Trace ID	Module	Trace point	Comments
ALOC	DBFALOC0	Entry	FP trace data set allocation
ALOX	DBFALOC0	Exit	FP trace data set allocation
RTYE	DBFALOC0	EMHB Present	FP trace data set allocation
RTYS	DBFALOC0	No EMHB Present	FP trace data set allocation
ALOP	DBFALOC0	Trace data set already allocated	FP trace data set allocation
BBIN	DBFBBIN0	Entry, Exit (Shift)	MSDB Binary Search
OFSE	DBFBBIN0	Binary Search	MSDB Binary Search
BIN1	DBFBBIN0	Binary Search Entry, Exit (Shift)	MSDB Binary Search
BCHG	DBFBCHG0	Entry	MSDB FLD Call Change
RCHG	DBFBCHG0	Exit	MSDB FLD Call Change
BOFL	DBFBCHG0	Overflow	MSDB FLD Call Change
BCL0	DBFBCL10	Entry	MSDB Call Handler
RCL0	DBFBCL10	Exit	MSDB Call Handler
IRC2	DBFBCL10	Copy Call Data	MSDB Call Handler
BDLT	DBFBDLT0	Entry	MSDB Delete Call

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
CDLT	DBFBDLT0	Delete OK	MSDB Delete Call
RDLT	DBFBDLT0	Exit	MSDB Delete Call
BENQ	DBFBENQ0	Entry	MSDB Resource Locking
NQ16	DBFBENQ0	Function 16 = Enqueue	MSDB Resource Locking
ENQ1	DBFBENQ0	Resource Locked, call Lock Manger	MSDB Resource Locking
ENQ2	DBFBENQ0	Resource Locked	MSDB Resource Locking
RENQ	DBFBENQ0	Exit	MSDB Resource Locking
BDEQ	DBFBENQ0	Dequeue	MSDB Resource Locking
BFLD	DBFBFLD0	Entry	MSDB FLD Call Processor
RFLD	DBFBFLD0	Exit	MSDB FLD Call Processor
BGET	DBFBGET0	Entry	MSDB Get Processor
RGET	DBFBGET0	Exit	MSDB Get Processor
BINC	DBFBINC0	Entry, Exit (Shift)	MSDB Decimal Field Verify
BNXT	DBFBNXT0	Entry	MSDB Get Next
RNXT	DBFBNXT0	Exit	MSDB Get Next
BRPL	DBFBRPL0	Entry	MSDB Replace
RRPL	DBFBRPL0	Exit	MSDB Replace
BSEQ	DBFBSEQ0	Entry, Exit (Shift)	MSDB Sequential Search
SEQ1	DBFBSEQ0	ECNT Search Entry,Exit (Shift)	MSDB Sequential Search
SEQI	DBFBSEQ0	ECNT Scan	MSDB Sequential Search
SEQ2	DBFBSEQ0	Segment Search Entry, Exit (Shift)	MSDB Sequential Search
SEQ3	DBFBSEQ0	Search Forward Entry, Exit (Shift)	MSDB Sequential Search
BSRT	DBFBSRT0	Entry	MSDB Insert Processor
CSRT	DBFBSRT0	Count Free Segments	MSDB Insert Processor
DSRT	DBFBSRT0	Insert Complete	MSDB Insert Processor
RSRT	DBFBSRT0	Exit	MSDB Insert Processor
BUPB	DBFBUPB0	Entry	MSDB Update Buffer Space Handler
RUPB	DBFBUPB0	Exit	MSDB Update Buffer Space Handler
BVAL	DBFBVAL0	Entry,Exit (Shift)	MSDB Decimal Segment Validate
BVfy	DBFBVfy0	Entry	MSDB Field Verify Processor
RVfy	DBFBVfy0	Exit	MSDB Field Verify Processor



Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
BXTR	DBFBXTR0	Entry,Exit (Shift)	MSDB Hex Field Translator
CBHL	DBFCBHL0	Entry	DEDB Hard Luck Buffer Handler (Buffer Steal)
YBHL	DBFCBHL0	Exit to caller or to wait for buffer	DEDB Hard Luck Buffer Handler (Buffer Steal)
ZBHL	DBFCBHL0	OBA required	DEDB Hard Luck Buffer Handler (Buffer Steal)
BDU0	DBFDBDU0	Entry	MSDB Log Update Processor
CHGA	DBFDBDU0	Change - Before	MSDB Log Update Processor
CHGB	DBFDBDU0	Change - After	MSDB Log Update Processor
DECA	DBFDBDU0	Decimal Operation - Before	MSDB Log Update Processor
DECB	DBFDBDU0	Decimal Operation - After	MSDB Log Update Processor
DLTA	DBFDBDU0	Delete - Before	MSDB Log Update Processor
DLBT	DBFDBDU0	Delete - After	MSDB Log Update Processor
SRTA	DBFDBDU0	Insert - Before	MSDB Log Update Processor
SRTB	DBFDBDU0	Insert - After	MSDB Log Update Processor
RDU0	DBFDBDU0	Exit	MSDB Log Update Processor
DCAP	DBFDCAP0	Entry	DEDB Change Data Capture
CAPD	DBFDCAP0	Build CAPD Block	DEDB Change Data Capture
DATA	DBFDCAP0	Build CAPD_DATA Blocks	DEDB Change Data Capture
READ	DBFDCAP0	Read DEDB CI	DEDB Change Data Capture
DCAX	DBFDCAP0	Should not occur, invalid call type	DEDB Change Data Capture
SLG2	DBFDLG20	Good Sync	FP Resync Commit/Abort Log Processor
SLOG	DBFDLG20	Bad Sync	FP Resync Commit/Abort Log Processor
TLG2	DBFDLG20	Exit	FP Resync Commit/Abort Log Processor
DRSC	DBFDRSC0	Entry	FP Resync Controller
DSRP	DBFDSRP0	Entry	DEDB SDEP Resync Processor
DSRN	DBFDSRP0	Exit	DEDB SDEP Resync Processor
HCHG	DBFHCHG0	Entry	EMH Alt PCB CHNG Call Processor
NCHG	DBFHCHG0	Exit	EMH Alt PCB CHNG Call Processor
HCL0	DBFHCL00	Entry	EMH and FP Utility Call Analyzer
NCL0	DBFHCL00	Exit	EMH and FP Utility Call Analyzer
HGN0	DBFHGN00	Entry	EMH Get Next Call Processor
NGN0	DBFHGN00	Exit	EMH Get Next Call Processor

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
HGU1	DBFHGU10	Entry	EMH Get Unique + Sync Control Processor
NGU1	DBFHGU10	Exit	EMH Get Unique + Sync Control Processor
EOTR	DBFHGU10	End of Thread	EMH Get Unique + Sync Control Processor
RTRY	DBFHGU10	Retried Transaction	EMH Get Unique + Sync Control Processor
BOTR	DBFHGU10	Start of Thread	EMH Get Unique + Sync Control Processor
HRLB	DBFHRLB0	Entry	EMH ROLB Processor
NRLB	DBFHRLB0	Exit	EMH ROLB Processor
HSRT	DBFHSRT0	Entry	EMH TP PCB ISRT Processor
NSRT	DBFHSRT0	Exit	EMH TP PCB ISRT Processor
FPR3	DBFIRC10	DL/I Call Start	FP Inter-Region Communication
IRC1	DBFIRC10	DL/I Call	FP Inter-Region Communication
IR09	DBFIRC10	Post Call, DEDB FLD, or MSDB	FP Inter-Region Communication
OPMV	DBFIRC10	Post Call, Move Data to Dependent	FP Inter-Region Communication
IRCZ	DBFIRC10	Post Call, Pseudo Abend Set	FP Inter-Region Communication
MBED	DBFMBED0	Entry	DEDB Get CI
MBE2	DBFMBED0	HSSP Async Read Ahead Wait	DEDB Get CI
MBEH	DBFMBED0	HSSP, found CI in Private Buffer	DEDB Get CI
GPRS	DBFMBED0	Exit without XCRB	DEDB Get CI
NBED	DBFMBED0	Exit	DEDB Get CI
BFL9	DBFMBFL9	Entry	Build FLDC for ISRT
BFLX	DBFMBFL9	Exit	Build FLDC for ISRT
MBMM	DBFMBMM9	Entry	Build SSAs, set Minimum   Maximum
MB02	DBFMBMM9	GT Operator no Minimum	Build SSAs, set Minimum   Maximum
MB03	DBFMBMM9	GT Operator Minimum	Build SSAs, set Minimum   Maximum
MB04	DBFMBMM9	GE Operator no Minimum	Build SSAs, set Minimum   Maximum
MB05	DBFMBMM9	GE Operator Minimum	Build SSAs, set Minimum   Maximum
MB06	DBFMBMM9	LT Operator no Maximum	Build SSAs, set Minimum   Maximum

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
MB07	DBFMBMM9	LT Operator Maximum	Build SSAs, set Minimum   Maximum
MB08	DBFMBMM9	LE Operator no Maximum	Build SSAs, set Minimum   Maximum
MB09	DBFMBMM9	Invalid Boolean Operator	Build SSAs, set Minimum   Maximum
MB10	DBFMBMM9	EQ Operator set Maximum	Build SSAs, set Minimum   Maximum
MB11	DBFMBMM9	EQ Operator Maximum already set	Build SSAs, set Minimum   Maximum
MB12	DBFMBMM9	Set Minimum	Build SSAs, set Minimum   Maximum
MB13	DBFMBMM9	Minimum already set	Build SSAs, set Minimum   Maximum
MB14	DBFMBMM9	NE Operator	Build SSAs, set Minimum   Maximum
MB15	DBFMBMM9	No Key Fields	Build SSAs, set Minimum   Maximum
MB16	DBFMBMM9	Error in Maximum or Minimum	Build SSAs, set Minimum   Maximum
MB17	DBFMBMM9	Set Maximum into SSA	Build SSAs, set Minimum   Maximum
MB18	DBFMBMM9	Set Maximum into SSA	Build SSAs, set Minimum   Maximum
MB19	DBFMBMM9	Set Maximum into SSA	Build SSAs, set Minimum   Maximum
MB20	DBFMBMM9	Maximum zero	Build SSAs, set Minimum   Maximum
MB21	DBFMBMM9	Set Minimum into SSA	Build SSAs, set Minimum   Maximum
MB22	DBFMBMM9	Set Minimum into SSA	Build SSAs, set Minimum   Maximum
MB23	DBFMBMM9	Set Minimum into SSA	Build SSAs, set Minimum   Maximum
MB25	DBFMBMM9	Set Minimum into SSA	Build SSAs, set Minimum   Maximum
MB26	DBFMBMM9	Minimum zero	Build SSAs, set Minimum   Maximum
MBMM	DBFMBMM9	Exit	Build SSAs, set Minimum   Maximum
CVAL	DBFMCCV9	Entry, Exit (Shift)	Check Command Code Validity
SSP1	DBFMCCV9	Subset Pointer	Check Command Code Validity
SSR1	DBFMCCV9	Command Code R	Check Command Code Validity
SSP2	DBFMCCV9	Other Subset Command	Check Command Code Validity
SSP3	DBFMCCV9	Check for Conflicts	Check Command Code Validity

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
SSPX	DBFMCCV9	Check Subset Pointer Conflict	Check Command Code Validity
SSP4	DBFMCCV9	Set Pointers	Check Command Code Validity
SSP5	DBFMCCV9	Not Command Code C	Check Command Code Validity
SSP6	DBFMCCV9	Command Code C	Check Command Code Validity
SSP7	DBFMCCV9	Command Code F or R	Check Command Code Validity
SSP8	DBFMCCV9	F, R, or L not at ISRT Level	Check Command Code Validity
LOPP	DBFMCCV9	Loop checking position	Check Command Code Validity
ELOP	DBFMCCV9	End of loop checking position	Check Command Code Validity
SSP9	DBFMCCV9	F, R, or L at ISRT level	Check Command Code Validity
SSPA	DBFMCCV9	Command Code U	Check Command Code Validity
LOPU	DBFMCCV9	Loop checking position back to root	Check Command Code Validity
SSPB	DBFMCCV9	Command Code V	Check Command Code Validity
AMST	DBFMCCV9	Status Code AM set	Check Command Code Validity
AJST	DBFMCCV9	Status Code AJ set	Check Command Code Validity
MCHG	DBFMCHG0	Entry	DEDB FLD Call Processor
XCHG	DBFMCHG0	Exit	DEDB FLD Call Processor
MOFL	DBFMCHG0	Overflow	DEDB FLD Call Processor
MCL0	DBFMCLX0	Entry	DEDB Call Analyzer
SSAX	DBFMCLX0	Count SSAs	DEDB Call Analyzer
SEG4	DBFMCLX0	Good RC, Trace Segment	DEDB Call Analyzer
PARP	DBFMCLX0	Trace Parent	DEDB Call Analyzer
NCL0	DBFMCLX0	Exit	DEDB Call Analyzer
CRP9	DBFMCRP9	Entry	Check for Subset Pointer
PPRE	DBFMCRP9	Trace Parent Prefix	Check for Subset Pointer
CRPX	DBFMCRP9	Exit	Check for Subset Pointer
MCSS	DBFMCSS9	Entry	Compare Current Segment Field to SSA
CALL	DBFMCSS9	Entry, Trace Call Argument	Compare Current Segment Field to SSA
MC01	DBFMCSS9	Key SSA + Key Value	Compare Current Segment Field to SSA

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
MC02	DBFMCSS9	Compare Key	Compare Current Segment Field to SSA
MC03	DBFMCSS9	Compare Key	Compare Current Segment Field to SSA
MC04	DBFMCSS9	Compare Key	Compare Current Segment Field to SSA
MC05	DBFMCSS9	Not Satisfied RC=8	Compare Current Segment Field to SSA
MC06	DBFMCSS9	Compare Key	Compare Current Segment Field to SSA
MC07	DBFMCSS9	Compare Key	Compare Current Segment Field to SSA
MC08	DBFMCSS9	Not Satisfied RC=12	Not Satisfied RC=12
MC09	DBFMCSS9	Compare Key	Compare Current Segment Field to SSA
MC10	DBFMCSS9	Compare Key	Compare Current Segment Field to SSA
MC11	DBFMCSS9	Compare	Compare Current Segment Field to SSA
MC12	DBFMCSS9	Compare	Compare Current Segment Field to SSA
MC13	DBFMCSS9	Compare	Compare Current Segment Field to SSA
MC14	DBFMCSS9	Compare	Compare Current Segment Field to SSA
MC1A	DBFMCSS9	No Match	Compare Current Segment Field to SSA
MC1B	DBFMCSS9	Compare	Compare Current Segment Field to SSA
CSSF	DBFMCSS9	Compare, no Boolean	Compare Current Segment Field to SSA
MCS2	DBFMCSS9	Relational Operator Satisfied	Compare Current Segment Field to SSA
MCS1	DBFMCSS9	Relational Operator Not Satisfied	Compare Current Segment Field to SSA
CSS9	DBFMCSS9	Exit	Compare Current Segment Field to SSA
CSL9	DBFMCSL9	Entry,Exit (Shift)	Compare Current Segment Field to SSA
CALL	DBFMCSL9	Entry, Trace Call Argument	Compare Current Segment Field to SSA
LOPC	DBFMCSL9	Compare Loop	Compare Current Segment Field to SSA
NEXT	DBFMCSL9	Read Next Buffer	Compare Current Segment Field to SSA
MCTL	DBFMCTL0	Entry, Exit (Shift)	Check this Level
COML	DBFMCTL0	Command Code L	Check this Level
FRST	DBFMCTL0	Command Code F, R, or unqualified	Check this Level
GETN	DBFMCTL0	Get Next	Check this Level
ISRT	DBFMCTL0	ISRT Here	Check this Level
MCT3	DBFMCTL0	Trace Process Return Code	Check this Level
MDEQ	DBFMDEQ0	Entry	DEDB DEQ Command Processor
XDEQ	DBFMDEQ0	Exit	DEDB DEQ Command Processor

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
MDLT	DBFMDLT0	Entry	DEDB Direct Delete
EPCB	DBFMDLT0	Check for other PCBs	DEDB Direct Delete
PRBA	DBFMDLT0	Update other PCB PRBA	DEDB Direct Delete
CRBA	DBFMDLT0	Clear other PCB CRBA	DEDB Direct Delete
KILL	DBFMDLT0	Reset Parentage other PCB	DEDB Direct Delete
NRBA	DBFMDLT0	Update other PCB NRBA	DEDB Direct Delete
XRBA	DBFMDLT0	Update other PCB XRBA	DEDB Direct Delete
GRBA	DBFMDLT0	Update other PCB GRBA	DEDB Direct Delete
DPTX	DBFMDPT9	Entry	Delete PCL and Subset Pointers in Parent Prefix
DPTX	DBFMDPT9	Exit	Delete PCL and Subset Pointers in Parent Prefix
MDRA	DBFMDRA9	Entry,Exit (Shift)	Determine possibility of randomizing
MD01	DBFMDRA9	Read First Root	Determine possibility of randomizing
MD02	DBFMDRA9	Use Current Position	Determine possibility of randomizing
MD03	DBFMDRA9	Cannot use CP, randomize	Determine possibility of randomizing
MD04	DBFMDRA9	Current key LT SSA min, randomize	Determine possibility of randomizing
MD05	DBFMDRA9	Must Move this Level	Determine possibility of randomizing
MD06	DBFMDRA9	Current Key GT SSA min	Determine possibility of randomizing
MD07	DBFMDRA9	Current Key GT SSA min	Determine possibility of randomizing
MD08	DBFMDRA9	Current Key GT SSA max	Determine possibility of randomizing
MD09	DBFMDRA9	Must Move this Level	Determine possibility of randomizing
MD10	DBFMDRA9	Level Acceptable	Determine possibility of randomizing
MD11	DBFMDRA9	Current Key LT SSA max	Determine possibility of randomizing
MD12	DBFMDRA9	Current Key GT SSA max	Determine possibility of randomizing
MD13	DBFMDRA9	Current Key LT SSA max	Determine possibility of randomizing

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
MD14	DBFMDRA9	Must Move this Level	Determine possibility of randomizing
MD15	DBFMDRA9	Level Acceptable	Determine possibility of randomizing
MA13	DBFMDRA9	SSA Min = SSA Max, randomizer	Determine possibility of randomizing
MD19	DBFMDRA9	Set SSA Min, Max	Determine possibility of randomizing
MD20	DBFMDRA9	No Low Boundary	Determine possibility of randomizing
MD21	DBFMDRA9	Current Key LT SSA Max	Determine possibility of randomizing
MD22	DBFMDRA9	Must Move this Level	Determine possibility of randomizing
MD23	DBFMDRA9	Do Nothing	Determine possibility of randomizing
MD24	DBFMDRA9	Current Key GT SSA Max	Determine possibility of randomizing
MD26	DBFMDRA9	Do Sequential Read	Determine possibility of randomizing
MD27	DBFMDRA9	Do Sequential Read	Determine possibility of randomizing
MD28	DBFMDRA9	Do Nothing	Determine possibility of randomizing
MD29	DBFMDRA9	Current Key LT SSA Max	Determine possibility of randomizing
MD31	DBFMDRA9	Current Key GT FDLC Low Key	Determine possibility of randomizing
MD32	DBFMDRA9	Current Key GT FDLC Low Key	Determine possibility of randomizing
MD33	DBFMDRA9	Do Nothing	Determine possibility of randomizing
MD34	DBFMDRA9	Do Sequential Read	Determine possibility of randomizing
MD35	DBFMDRA9	Do Nothing	Determine possibility of randomizing
MD36	DBFMDRA9	Do Sequential Read	Determine possibility of randomizing
MD37	DBFMDRA9	Do Nothing	Determine possibility of randomizing
MD38	DBFMDRA9	Go to Next Set Process	Determine possibility of randomizing
MD41	DBFMDRA9	Search for Lowest Min GT Current	Determine possibility of randomizing
MD42	DBFMDRA9	Search for Next Higher Set Minimum	Determine possibility of randomizing
MD43	DBFMDRA9	Address next Set	Determine possibility of randomizing
MD44	DBFMDRA9	Set Min Found, Randomize	Determine possibility of randomizing
MD45	DBFMDRA9	Goto NOUSE	Determine possibility of randomizing
MD46	DBFMDRA9	Read First Root	Determine possibility of randomizing

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
MD47	DBFMDRA9	Do Nothing - current position good	Determine possibility of randomizing
MD48	DBFMDRA9	Clear current position	Determine possibility of randomizing
MD49	DBFMDRA9	Call Randomizer	Determine possibility of randomizing
MD50	DBFMDRA9	Continue Sequential Read	Determine possibility of randomizing
MDRB	DBFMDBR0	Entry	Delete, get root backwards
MDRT	DBFMDBRX0	Entry, Exit DDEP,SDEP (Shift)	DEDB Insert
DSG9	DBFMDSG9	Entry	DEDB Delete Direct Dependent
RECU	DBFMDSG9	Recursive Call Stack Information	DEDB Delete Direct Dependent
DSG1	DBFMDSG9	Trace Segment to be processed	DEDB Delete Direct Dependent
LOP1	DBFMDSG9	Twin Chain Loop	DEDB Delete Direct Dependent
SIBL	DBFMDSG9	First Child	DEDB Delete Direct Dependent
LO1X	DBFMDSG9	Loop over Parent Prefix Complete	DEDB Delete Direct Dependent
FRE1	DBFMDSG9	Call DBFMFSE0 to free space	DEDB Delete Direct Dependent
LOP2	DBFMDSG9	Twin Chain Loop	DEDB Delete Direct Dependent
FRE2	DBFMDSG9	Call DBFMFSE0 to free space	DEDB Delete Direct Dependent
DSGX	DBFMDSG9	Exit	DEDB Delete Direct Dependent
DS14	DBFMDSG9	Return after Recursive Call	DEDB Delete Direct Dependent
MFLO	DBFMFL00	Entry	DEDB FLD Call Processor
XMFL	DBFMFL00	Exit	DEDB FLD Call Processor
MFLD	DBFMFL10	Entry	DEDB FSA Processor
XFLD	DBFMFL10	Exit	DEDB FSA Processor
MFSE	DBFMFSE0	Entry	DEDB Space Manager
MFSS	DBFMFSE0	Scrap Handling	DEDB Space Manager
NFSE	DBFMFSE0	Exit	DEDB Space Manager
OFSE	DBFMFSE0	Read AP or Root CI to find space	DEDB Space Manager
PFSE	DBFMFSE0	Read 1st DOVF CI	DEDB Space Manager
GPDS	DBFMFSE0	Got Conditional Lock IOVF SM CI	DEDB Space Manager



Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
GPDN	DBFMFSE0	Bad Conditional Lock IOVF SM CI	DEDB Space Manager
MGAP	DBFMGAP0	Entry	DEDB Get Anchor Point
ERAN	DBFMGAP0	Entry to Randomizer	DEDB Get Anchor Point
XRAN	DBFMGAP0	Exit from Randomizer	DEDB Get Anchor Point
MGA1	DBFMGAP0	PROCOPT=P UOW BDY crossed	DEDB Get Anchor Point
MGA2	DBFMGAP0	PROCOPT=P UOW Set GC status	DEDB Get Anchor Point
MGA3	DBFMGAP0	PROCOPT=H Save Position	DEDB Get Anchor Point
NGAP	DBFMGAP0	Exit	DEDB Get Anchor Point
MGFD	DBFMGFD0	Entry	DEDB Initialize Level Table
NGFD	DBFMGFD0	Exit	DEDB Initialize Level Table
MGL9	DBFMGLA9	Entry, Exit (Shift)	DEDB Get Last Occurrence of Segment Under Parent
CLA9	DBFMGLA9	Enter, Exit(shift) EP DBFMCLA9	DEDB Check if Another Occurrence of Segment under Parent satisfies SSA
MGNR	DBFMGNR0	Entry	DEDB Get Next Root
NGNR	DBFMGNR0	Exit	DEDB Get Next Root
EXAP	DBFMGNR0	Get Next RAP with a root start loop	DEDB Get Next Root
EXA1	DBFMGNR0	Get Next RAP with a root block#	DEDB Get Next Root
EOC1	DBFMGNR0	Out of Area Range	DEDB Get Next Root
EXA2	DBFMGNR0	Read CI	DEDB Get Next Root
XXAP	DBFMGNR0	Return the RAP	DEDB Get Next Root
ECAL	DBFMGNR0	Calc UOW#, DMAC from BLK#	DEDB Get Next Root
ECA1	DBFMGNR0	PROCOPT=P set GC status	DEDB Get Next Root
ECA2	DBFMGNR0	PROCOPT=P set GC status	DEDB Get Next Root
XCAL	DBFMGNR0	Rerun UOW#, DMAC	DEDB Get Next Root
MGN0	DBFMGNX0	Entry, Exit(Shift)	DEDB Get Next Root
GPDE	DBFMGPD0	Entry	DEDB Retrieve Sequential Dependent Segment
GPD1	DBFMGPD0	Scan Segment Chain	DEDB Retrieve Sequential Dependent Segment

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
GPD2	DBFMGPD0	SDEP Pointer -> uncommitted seg	DEDB Retrieve Sequential Dependent Segment
GPD3	DBFMGPD0	SDEP Pointer -> normal seg	DEDB Retrieve Sequential Dependent Segment
GPD4	DBFMGPD0	Must Read CI	DEDB Retrieve Sequential Dependent Segment
GPDS	DBFMGPD0	IRLM Notify to Partner	DEDB Retrieve Sequential Dependent Segment
GDPN	DBFMGPD0	SDEP CI found to be locked	DEDB Retrieve Sequential Dependent Segment
GDP7	DBFMGPD0	Re-read CI	DEDB Retrieve Sequential Dependent Segment
GDPC	DBFMGPD0	Compare Segment to SSA	DEDB Retrieve Sequential Dependent Segment
GPD5	DBFMGPD0	SSA does not match this segment	DEDB Retrieve Sequential Dependent Segment
GPD6	DBFMGPD0	Copy segment	DEDB Retrieve Sequential Dependent Segment
GDPX	DBFMGPD0	Exit	DEDB Retrieve Sequential Dependent Segment
MGPF	DBFMGPFO	Entry	Get Page of Common Storage
NGPF	DBFMGPFO	Exit	Get Page of Common Storage
MGRF	DBFMGRFO	Entry	Get Root Forward Search
MGR1	DBFMGRFO	Run Chain in RAP CI	Get Root Forward Search
MGR2	DBFMGRFO	Scan RAP CI	Get Root Forward Search
MGR3	DBFMGRFO	Run Chain next CI	Get Root Forward Search
MGR4	DBFMGRFO	Scan next CI	Get Root Forward Search
MGR5	DBFMGRFO	Root does not exist (status GE)	Get Root Forward Search
MGR6	DBFMGRFO	Root does not exist, other roots found	Get Root Forward Search
MGR8	DBFMGRFO	Root found during a scan	Get Root Forward Search
MGR9	DBFMGRFO	Root found by Run Chain	Get Root Forward Search
NGRF	DBFMGRFO	Exit	Get Root Forward Search
MGRG	DBFMGRFO	Anchor Point Scan routine	Get Root Forward Search
MGRL	DBFMGRFO	CI Scan routine	Get Root Forward Search
MGRM	DBFMGRFO	Nextitem routine	Get Root Forward Search

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
MGRC	DBFMGRF0	Call DBFMPG00 check PROCOPT	Get Root Forward Search
MGRD	DBFMGRF0	Return from DBFMPG00	Get Root Forward Search
MGU0	DBFMGUX0	Entry, Exit(Shift)	Get Unique, Unqualified
MGXC	DBFMGXCO	Entry, Entry SEGLOCK	Get Control of Resource
NGXC	DBFMGXCO	Exit, Entry SEGLOCK, Exit CI EXCL	Get Control of Resource
ENQR	DBFMGXCO	Lock Resource for this caller	Get Control of Resource
ENQO	DBFMGXCO	Lock Resource on behalf of other	Get Control of Resource
SHXC	DBFMGXCO	Just CI SHR Lock, Entry/Exit	Get Control of Resource
EXXC	DBFMGXCO	Just CI Lock Exclusive, Entry	Get Control of Resource
VLOC	DBFMGXCO	DBFVLOCK Entry	Get Control of Resource
NLOC	DBFMGXCO	DBFVLOCK Exit	Get Control of Resource
MINC	DBFMINC0	Entry	DEDB Included Decimal FLD Call
XINC	DBFMINC0	Exit	DEDB Included Decimal FLD Call
IRC9	DBFMIRC9	Entry, Exit(Shift)	DEDB Retrieve Previous Parent, Set MLTE Fields
MIRT	DBFMIRT0	Entry, Exit(Shift)	DEDB Insert
MIR1	DBFMIRT0	Trace CI RBA	DEDB Insert
MIR2	DBFMIRT0	Previous Root Twin not in Same CI	DEDB Insert
MIR3	DBFMIRT0	Trace after reading CI	DEDB Insert
MIR4	DBFMIRT0	Previous Segment not in Same CI	DEDB Insert
MIR5	DBFMIRT0	Previous Segment not Parent	DEDB Insert
MIRB	DBFMIRT0	Set PCF Pointer in Parent	DEDB Insert
MIR6	DBFMIRT0	Previous Segment in Same CI	DEDB Insert
MIR7	DBFMIRT0	Update RAP -> new segment	DEDB Insert
MIR8	DBFMIRT0	Previous is not a RAP	DEDB Insert

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
MIR9	DBFMIRT0	Previous Segment is Twin	DEDB Insert
MIRA	DBFMIRT0	Set Log Data	DEDB Insert
ISLL	DBFMISL9	Entry, Exit(Shift)	DEDB Process Insert Last Level
MLCS	DBFMLCL0	Entry	DEDB Logical Close Area
MLCE	DBFMLCL0	Exit	DEDB Logical Close Area
MLEV	DBFMLEV0	Entry	DEDB Adjust MLTE Sequence Numbers
NLEV	DBFMLEV0	Exit	DEDB Adjust MLTE Sequence Numbers
MLOG	DBFMLOG0	Entry	DEDB SDEP CI Logging
NLOG	DBFMLOG0	Exit	DEDB SDEP CI Logging
MLOS	DBFMLOP0	Entry	DEDB Logical Open Area
MLOE	DBFMLOP0	Exit	DEDB Logical Open Area
MMIT	DBFMMIT0	Entry	DEDB Media Manager Connect/Disconnect
MMIE	DBFMMIT0	Exit	DEDB Media Manager Connect/Disconnect
MOCI	DBFMOCIO	Entry	DEDB DMAC Update
NOCI	DBFMOCIO	Exit	DEDB DMAC Update
PCC9	DBFMPCC9	Entry, Exit(Shift)	DEDB Process 'C' Command Code
MCLS	DBFMPCL0	Entry	DEDB Physical Area Close
MCLE	DBFMPCL0	Exit	DEDB Physical Area Close
PED9	DBFMPED9	Entry,Exit (Shift)	DEDB Process Position Fields in Parallel EPCBs - DELETE
PEI9	DBFMPEI9	Entry, Exit (Shift)	DEDB Process Position Fields in Parallel EPCBs - INSERT
LOP1	DBFMPEI9	Loop through EPCBs	DEDB Process Position Fields in Parallel EPCBs - INSERT
LOP2	DBFMPEI9	Loop through MLTEs	DEDB Process Position Fields in Parallel EPCBs - INSERT
FRGU	DBFMPEI9	Update before EPCB GU Position	DEDB Process Position Fields in Parallel EPCBs - INSERT
FGRN	DBFMPEI9	Update before EPCB GN Position	DEDB Process Position Fields in Parallel EPCBs - INSERT
AFGU	DBFMPEI9	Update after EPCB GU Position	DEDB Process Position Fields in Parallel EPCBs - INSERT
DPTE	DBFMPEI9	Entry	DEDB Relocate PCL or SSPT in Parent
DPTX	DBFMPEI9	Exit	DEDB Relocate PCL or SSPT in Parent
MPGO	DBFMPGO0	Entry, Exit(Shift)	DEDB Process PROCOPT GOX, GON
PIO9	DBFMPIO9	Entry, Exit(Shift)	DEDB Process I/O Area for REPLACE

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
MOPS	DBFMPOPO	Entry	DEDB Physical Area Open
MOPE	DBFMPOPO	Exit	DEDB Physical Area Open
MPOS	DBFMPOS0	Entry	DEDB POS Call
NPOS	DBFMPOS0	Exit	DEDB POS Call
MGUP	DBFMPOS0	Entry to Find Root Segment	DEDB POS Call
MGNP	DBFMPOS0	Entry to find next SDEP Segment	DEDB POS Call
NGN0	DBFMPOS0	Exit from find next SDEP Segment	DEDB POS Call
MPO2	DBFMPOS0	Notify Partners to Harden SDEP Cis	DEDB POS Call
VMAI	DBFMPOS0	Exit from Notify Partners	DEDB POS Call
MPSG	DBFMPSG9	Entry, Exit(Shift)	DEDB Process Subset Pointer Commands S W Z M
SSPL	DBFMPSG9	Loop down through MLTEs	DEDB Process Subset Pointer Commands S W Z M
LOPC	DBFMPSG9	Loop over SSPTRs	DEDB Process Subset Pointer Commands S W Z M
SCOM	DBFMPSG9	Command Code S	DEDB Process Subset Pointer Commands S W Z M
WCOM	DBFMPSG9	Command Code W	DEDB Process Subset Pointer Commands S W Z M
ZCOM	DBFMPSG9	Command Code Z	DEDB Process Subset Pointer Commands S W Z M
MPUG	DBFMPUG0	Entry, Exit(Shift)	DEDB Process Unqualified GN
FCHL	DBFMPUG0	First Child	DEDB Process Unqualified GN
MUP1	DBFMPUG0	Move Up A Level	DEDB Process Unqualified GN
VIO1	DBFMPUG0	Violation 1	DEDB Process Unqualified GN
VIO2	DBFMPUG0	Violation 2	DEDB Process Unqualified GN
GSBL	DBFMPUG0	Get Sibling	DEDB Process Unqualified GN
MPU2	DBFMPUG0	Sibling Located	DEDB Process Unqualified GN
MRCU	DBFMRCU0	Entry, Exit(Shift)	DEDB Read Current Dependent Segment
MRPU	DBFMRPU0	Entry, Exit(Shift)	DEDB Reset Position after Unqualified GN
LMLT	DBFMRPU0	Loop through MLTEs	DEDB Reset Position after Unqualified GN
MRPL	DBFMRPX0	Entry, Exit(Shift)	DEDB REPLACE
AMST	DBFMRPX0	Return AM status code	DEDB REPLACE

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
SPR9	DBFMRPX0	Update PRBA in MLTE of children	DEDB REPLACE
SPR1	DBFMRPX0	Loop through siblings	DEDB REPLACE
SPR2	DBFMRPX0	End loop through siblings	DEDB REPLACE
PED9	DBFMRPX0	End child PRBA updates	DEDB REPLACE
MRQC	DBFMRQC0	Entry, Exit(Shift)	DEDB Retrieve by Qualified Call
MRQU	DBFMRQC0	Check current position	DEDB Retrieve by Qualified Call
MRUU	DBFMRQC0	MLTE not qualified	DEDB Retrieve by Qualified Call
MRNQ	DBFMRQC0	Get Next Loop	DEDB Retrieve by Qualified Call
MRNU	DBFMRQC0	Not qualified SSA	DEDB Retrieve by Qualified Call
MRUF	DBFMRQC0	At Least Root Satisfies	DEDB Retrieve by Qualified Call
REQ1	DBFMRQC0	Diverge U or GN *F	DEDB Retrieve by Qualified Call
REQL	DBFMRQC0	MLTE downward loop qualification	DEDB Retrieve by Qualified Call
RC04	DBFMRQC0	Found, moved off current position	DEDB Retrieve by Qualified Call
RC08	DBFMRQC0	Not found at that level	DEDB Retrieve by Qualified Call
LROT	DBFMRQC0	Loop up to root	DEDB Retrieve by Qualified Call
TOGH	DBFMRQC0	GU, all levels satisfied current pos	DEDB Retrieve by Qualified Call
DIVE	DBFMRQC0	MLTE Loop to clear DIVERGE flag	DEDB Retrieve by Qualified Call
NOPA	DBFMRQC0	Not a PATH call	DEDB Retrieve by Qualified Call
PMVE	DBFMRQC0	Data to be moved	DEDB Retrieve by Qualified Call
PHIL	DBFMRQC0	Path call Highest Level to move	DEDB Retrieve by Qualified Call
PCOM	DBFMRQC0	Path call complete	DEDB Retrieve by Qualified Call
PLOP	DBFMRQC0	Loop for P command up in MLTEs	DEDB Retrieve by Qualified Call
MRUC	DBFMRUC0	Entry, Exit(Shift)	DEDB Reset U Command at Current/Lower Level
SFIT	DBFMFSI9	Entry, Exit(Shift)	DEDB Search Field Name
SFLP	DBFMFSI9	Loop over Fields in Segment	DEDB Search Field Name

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
SFTP	DBFMFSI9	Verify Relational Operator	DEDB Search Field Name
CALL	DBFMFSI9	Trace SSA and Fields	DEDB Search Field Name
SFO9	DBFMSFO9	Entry, Exit(Shift)	DEDB Set First Position of Segment Type
PPRE	DBFMSFO9	Trace Parent Prefix	DEDB Set First Position of Segment Type
SFOT	DBFMSFO9	No Floating Pointer in Call	DEDB Set First Position of Segment Type
SIMP	DBFMSIM9	Entry, Exit(Shift)	DEDB Set Implied for Upper Levels of Call
MSPC	DBFMSPC0	Entry	DEDB IOVF Free Space Calculator
NSPC	DBFMSPC0	Exit	DEDB IOVF Free Space Calculator
MSRB	DBFMSTRB0	Entry	DEDB Schedule DBFMIOS0 SRB routine
NSRB	DBFMSTRB0	Exit	DEDB Schedule DBFMIOS0 SRB routine
MSRT	DBFMSRT0	Entry	DEDB Insert SDEP Segment to LSRT
NSRT	DBFMSRT0	Exit	DEDB Insert SDEP Segment to LSRT
MSR1	DBFMSRT0	MSRTCIAL entry	DEDB Preallocate SDEP Cis
MSR2	DBFMSRT0	After Recheck #1 still need PA	DEDB Preallocate SDEP Cis
MSR3	DBFMSRT0	After Recheck #2 still need PA	DEDB Preallocate SDEP Cis
MSR4	DBFMSRT0	DMAC Read Error	DEDB Preallocate SDEP Cis
MSR5	DBFMSRT0	DMAC Read Successful	DEDB Preallocate SDEP Cis
MSR6	DBFMSRT0	SDEP Part full after DMAC Read	DEDB Preallocate SDEP Cis
MSR7	DBFMSRT0	Allocate an RBAT	DEDB Preallocate SDEP Cis
MSR8	DBFMSRT0	DMACLBTS was zero, set here	DEDB Preallocate SDEP Cis
MSR9	DBFMSRT0	Trace ACCUM_LENG	DEDB Preallocate SDEP Cis
MSRA	DBFMSRT0	Trace RBAT, #Cis to allocate	DEDB Preallocate SDEP Cis
MSRB	DBFMSRT0	Trace RBAT, min # Cis	DEDB Preallocate SDEP Cis
MSRD	DBFMSRT0	SDEP Part now full	DEDB Preallocate SDEP Cis
MSRE	DBFMSRT0	SDEP Part now full	DEDB Preallocate SDEP Cis
MSRF	DBFMSRT0	Lock failure on SDEP PACI	DEDB Preallocate SDEP Cis
MSRG	DBFMSRT0	Add SDEP CI XCRBs to RBAT	DEDB Preallocate SDEP Cis

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
MSRH	DBFMSRT0	Trace ACCUM_LEN	DEDB Preallocate SDEP Cis
MLOG	DBFMSRT0	Log 5953 Record - entry	DEDB Preallocate SDEP Cis
NLOG	DBFMSRT0	Log 5953 Record - exit	DEDB Preallocate SDEP Cis
SSA9	DBFMSSA9	Entry, Exit(Shift)	DEDB Search SSA for Data
MMOV	DBFMSSA9	Must Move	DEDB Search SSA for Data
BACK	DBFMSSA9	Must Move Back	DEDB Search SSA for Data
STAY	DBFMSSA9	Stay at this Level	DEDB Search SSA for Data
FOND	DBFMSSA9	Found	DEDB Search SSA for Data
COML	DBFMSSA9	Command Code L	DEDB Search SSA for Data
TWLF	DBFMSSA9	Not found - higher key found	DEDB Search SSA for Data
EIGH	DBFMSSA9	Not found, no higher key found	DEDB Search SSA for Data
SAGI	DBFMSSC9	Entry, Exit(Shift)	DEDB SSA Handler for GET and INSERT
ACST	DBFMCSS9	Count SSAs	DEDB SSA Handler for GET and INSERT
NAMF	DBFMCSS9	Segment Name Found	DEDB SSA Handler for GET and INSERT
DESC	DBFMCSS9	Level Descending	DEDB SSA Handler for GET and INSERT
NDES	DBFMCSS9	Level Not Descending	DEDB SSA Handler for GET and INSERT
NOTN	DBFMCSS9	Segment Name Not Found	DEDB SSA Handler for GET and INSERT
NOSG	DBFMCSS9	No Segment Name Found	DEDB SSA Handler for GET and INSERT
ENAC	DBFMCSS9	Hierarchy Error in Segment Name	DEDB SSA Handler for GET and INSERT
SSD9	DBFMSSD9	Entry, Exit(Shift)	DEDB SSA Handler for DELETE
SAGE	DBFMSSG9	Entry,Exit (Shift)	DEDB SSA Handler for GET
SAIN	DBFMSSI9	Entry,Exit (Shift)	DEDB SSA Handler for INSERT
SSP9	DBFMSSP9	Entry,Exit (Shift)	DEDB SSA Handler for POS
SSR9	DBFMSSR9	Entry,Exit (Shift)	DEDB SSA Handler for REPLACE
MSTP	DBFMSTP0	Entry	DEDB I/O substitute routine when ADS is closing/closed
NSTP	DBFMSTP0	Exit	DEDB I/O substitute routine when ADS is closing/closed
SVC9	DBFMSVC9	Entry, Exit(Shift)	DEDB set V Command as U command in MLTE
MUHE	DBFMUHE0	Entry	DEDB Update Log Entry in DMHR



Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
NUHE	DBFMUHE0	Exit	DEDB Update Log Entry in DMHR
MUH1	DBFMUHE1	Entry, Exit(Shift)	DEDB Front End/Back End Elimination
MUPB	DBFMUPB0	Entry	DEDB View=MSDB Update Buffer handler
XUPB	DBFMUPB0	Exit	DEDB View=MSDB Update Buffer handler
MVFY	DBFMVFY0	Entry	DEDB FLD Call Verify Processor
XVFY	DBFMVFY0	Exit	DEDB FLD Call Verify Processor
VSNA	DBFMVSN9	Entry, Exit(Shift)	DEDB Call Handler Verify Segment Name
FOND	DBFMVSN9	Segment Name Found	DEDB Call Handler Verify Segment Name
NOTM	DBFNOTM0	Entry	Intersystem NOTIFY processor
EOTM	DBFNOTM0	Exit	Intersystem NOTIFY processor
PDNA	DBFPDNA0	Entry	DEDB SETR Positioning
PGA5	DBFPDNA0	Exit	DEDB SETR Positioning
PENQ	DBFPENQ0	Entry	DEDB UOW Resource Enqueue
NENQ	DBFPENQ0	Exit	DEDB UOW Resource Enqueue
PFDS	DBFPFDS0	Entry	DEDB Unallocate, Unchain, and release ADSC
PGAB	DBFPGAB0	Entry	DEDB Get Private Buffer or Buffers
NGAB	DBFPGAB0	Exit	DEDB Get Private Buffer or Buffers
PGAP	DBFPGAP0	Entry	DEDB HSSP Positioning
PGAE	DBFPGAP0	Exit	DEDB HSSP Positioning
PGA1	DBFPGAP0	New AREA	DEDB HSSP Positioning
PGA2	DBFPGAP0	Previous AREA still active	DEDB HSSP Positioning
PGA3	DBFPGAP0	Same AREA	DEDB HSSP Positioning
PGDS	DBFPGDS0	Entry	DEDB Allocate and Chain ADSC
PHST	DBFPHST0	Entry	DEDB HSSP/Utility Process Termination
PICS	DBFPICS0	Entry	DEDB HSSP Image Copy Process Setup
NICS	DBFPICS0	Exit	DEDB HSSP Image Copy Process Setup
PIOS	DBFPIOS0	Entry	DEDB HSSP Image Copy I/O Routine
IOSH	DBFPIOS0	Imac_IC_cursor GT req uow 1st CI	DEDB HSSP Image Copy I/O Routine
IOSL	DBFPIOS0	Imac_IC_cursor LT req uow 1st CI	DEDB H DEDB HSSP Image Copy I/O Routine
IOSE	DBFPIOS0	Imac_IC_cursor EQ req uow 1st CI	DEDB HSSP Image Copy I/O Routine
NIOS	DBFPIOS0	Exit	DEDB HSSP Image Copy I/O Routine

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
PRAB	DBFPRAB0	Entry	DEDB Release Current UOW Resources
NRAB	DBFPRAB0	Exit	DEDB Release Current UOW Resources
PSET	DBFPSET0	Entry, Exit(Shift)	DEDB HSSP Process Setup
PULI	DBFPULIO	Entry	DEDB UOW Lock Mode Initiation
NULI	DBFPULIO	Exit	DEDB UOW Lock Mode Initiation
PUL1	DBFPULIO	Wait for CI locks to be released	DEDB UOW Lock Mode Initiation
PUXC	DBFPUXC0	Entry	DEDB UOW Resource Handler
NUXC	DBFPUXC0	Exit	DEDB UOW Resource Handler
LUXC	DBFPUXC0	Lock Subroutine Entry	DEDB UOW Resource Handler
PUXR	DBFPUXR0	Entry	DEDB Release UXRBS
NUXR	DBFPUXR0	Exit	DEDB Release UXRBS
BSBP	DBFSBP10	Entry	MSDB Syncpoint Phase I
RSBP	DBFSBP10	Exit	MSDB Syncpoint Phase I
BACC	DBFSBP10	Use MSDB data for operation	MSDB Syncpoint Phase I
RACC	DBFSBP10	Use Record data for operation	MSDB Syncpoint Phase I
CACC	DBFSBP10	Trace Segment to be processed	MSDB Syncpoint Phase I
DACC	DBFSBP10	After move to EPST work area	MSDB Syncpoint Phase I
EACC	DBFSBP10	Move MSDB Data	MSDB Syncpoint Phase I
FACC	DBFSBP10	Move MSDB Data	MSDB Syncpoint Phase I
BMSG	DBFSBP10	Setup Arithmetic Overflow message	MSDB Syncpoint Phase I
SDEQ	DBFSDEQ0	Entry	FP Resource Dequeue
TDEQ	DBFSDEQ0	Exit	FP Resource Dequeue
UOWX	DBFSDEQ0	Change UOW Lock Ownership	FP Resource Dequeue
SFLD	DBFSFLD0	Entry	DEDB Syncpoint Phase I FLD Call
RSFL	DBFSFLD0	Exit	DEDB Syncpoint Phase I FLD Call
SGAB	DBFSGAB0	Entry	DEDB Get Buffer from Shared Pool
ZGAB	DBFSGAB0	Exit	DEDB Get Buffer from Shared Pool
SHQD	DBFSHDQ0	Entry	DEDB HSSP Resource Dequeue Phase II
SHDX	DBFSHDQ0	Exit	DEDB HSSP Resource Dequeue Phase II

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
HPRE	DBFSHDQ0	Trace DMAC	DEDB HSSP Resource Dequeue Phase II
HPR1	DBFSHDQ0	Application got GC status	DEDB HSSP Resource Dequeue Phase II
HPR3	DBFSHDQ0	Application did not get GC status	DEDB HSSP Resource Dequeue Phase II
HPRA	DBFSHDQ0	Sync Abort Flow	DEDB HSSP Resource Dequeue Phase II
SIC1	DBFSIC10	Entry DBFSIC10	DEDB HSSP Image Copy Phase I and Phase II
NIC1	DBFSIC10	Exit DBFSIC10	DEDB HSSP Image Copy Phase I and Phase II
SIC2	DBFSIC10	Entry DBFSIC20	DEDB HSSP Image Copy Phase I and Phase II
NIC2	DBFSIC10	Exit DBFSIC10	DEDB HSSP Image Copy Phase I and Phase II
SICC	DBFSIC10	Good Sync, enqueue DMHRSET	DEDB HSSP Image Copy Phase I and Phase II
SICE	DBFSIC10	Last UOW, enqueue TERM AWE	DEDB HSSP Image Copy Phase I and Phase II
SLGE	DBFSLGE0	Entry, Exit(Shift)	DEDB Sync Log Exit
SLGE2S	DBFSLGE2	Entry	DEDB Sync Log Exit for Segment Level Locking
SLGE20E	DBFSLGE2	Exit	DEDB Sync Log Exit for Segment Level Locking
SLG2	DBFSLG20	Entry, Exit(Shift)	DEDB Sync/Abort Log Processor
SLOG	DBFSLOG0	Entry,Bad Sync(Shift)	FP Log Processor
TLOG	DBFSLOG0	Exit	FP Log Processor
MP10	DBFSMP10	Entry	DEDB SDEP Syncpoint Phase I
SYPB	DBFSMP10	New current SDEP Buffer	DEDB SDEP Syncpoint Phase I
MP11	DBFSMP10	Trace #PA Cis, #required Cis	DEDB SDEP Syncpoint Phase I
MP12	DBFSMP10	Trace needed CI Space	DEDB SDEP Syncpoint Phase I
MP13	DBFSMP10	Trace new CI space	DEDB SDEP Syncpoint Phase I
NP10	DBFSMP10	Exit	DEDB SDEP Syncpoint Phase I
SPIX	DBFSPIX0	Entry	DEDB Process unused XCRB/DMHR at Sync
XPIX	DBFSPIX0	Exit	DEDB Process unused XCRB/DMHR at Sync
BSY0	DBFSYN00	Entry	Pure FP Sync Point Controller
SYN1	DBFSYN10	Entry	FP Syncpoint Phase I Controller
TYN1	DBFSYN10	Exit	FP Syncpoint Phase I Controller
SYN2	DBFSYN20	Entry, Entry 2nd Call	FP Syncpoint Phase II Controller
TYN2	DBFSYN20	Exit	FP Syncpoint Phase II Controller
SYP2	DBFSYP20	Entry	FP Syncpoint Phase II

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
TYP2	DBFSYP20	Exit	FP Syncpoint Phase II
SYPB	DBFSYP20	New current SDEP Buffer	FP Syncpoint Phase II
AFCE	DBFTAFC9	Entry	Analyze FPTCNTRL control cards
AFCX	DBFTAFC9	Exit	Analyze FPTCNTRL control cards
ATCE	DBFTATC9	Entry	Analyze Trace Calls
TON0	DBFTATC9	Trace ON	Analyze Trace Calls
TOFF	DBFTATC9	Trace OFF	Analyze Trace Calls
TSEL	DBFTATC9	TOM Table built	Analyze Trace Calls
TDAT	DBFTATC9	Deactivate Trace	Analyze Trace Calls
TACT	DBFTATC9	Activate Trace	Analyze Trace Calls
ATCX	DBFTATC9	Exit without error	Analyze Trace Calls
ATCY	DBFTATC9	Error Exit	Analyze Trace Calls
BLTE	DBFTBLT9	Entry	Build TOM Table Structure
BLTX	DBFTBLT9	Exit	Build TOM Table Structure
BMIE	DBFTBMI9	Entry	Build Trace Message In I/O Area
BMIX	DBFTBMI9	Exit	Build Trace Message In I/O Area
COTE	DBFTCOT9	Entry	Construct Trace Option Table
COT1	DBFTCOT9	Trace before IMODULE LOAD	Construct Trace Option Table
COT2	DBFTCOT9	After call to DBFTBLT9	Construct Trace Option Table
ILAR	DBFTCOT9	After GETMAIN	Construct Trace Option Table
COT3	DBFTCOT9	After GETMAIN	Construct Trace Option Table
COT4	DBFTCOT9	Delete TOM	Construct Trace Option Table
COTX	DBFTCOT9	Exit without error	Construct Trace Option Table
COTY	DBFTCOT9	Error Exit	Construct Trace Option Table
ABN2	DBFTDEB9	Trace without SDWA	Provide Debugging Information for Abending Module
ABN1	DBFTDEB9	Trace with SDWA	Provide Debugging Information for Abending Module
FTOE	DBFTDEB9	Entry	Free Previous TOM
FTOX	DBFTDEB9	Exit	Free Previous TOM
TRAF	DBFTIR1S	Trace OFF	DBFIRC10 Connection to FP Trace
TRAN	DBFTIR1S	Trace ON	DBFIRC10 Connection to FP Trace
SIEE	DBFTSIE9	Entry	Setup Initial Environment for FP Trace

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
SIEX	DBFTSIE9	Exit	Setup Initial Environment for FP Trace
STS9	DBFTSTS9	Entry	Set Trace Suppress Flag
STSX	DBFTSTS9	Exit	Set Trace Suppress Flag
VIAE	DBFTVIA9	Entry	Verify I/O Area of Trace Call
TCAL	DBFTVIA9	Trace I/O Area	Verify I/O Area of Trace Call
TC01	DBFTVIA9	Call is TON or TOFF	Verify I/O Area of Trace Call
TC02	DBFTVIA9	Call is TSEL	Verify I/O Area of Trace Call
TC03	DBFTVIA9	Trace 1st ID in I/O Area	Verify I/O Area of Trace Call
VIAX	DBFTVIA9	Exit without error	Verify I/O Area of Trace Call
VIAY	DBFTVIA9	Error Exit	Verify I/O Area of Trace Call
24BE	DBFT24B0	Entry	Trace Get/Put Routines in 24-bit mode
24BX	DBFT24B0	Exit	Trace Get/Put Routines in 24-bit mode
UHAC	DBFUHAC7	Entry	DEDB HSRE Access segment in non reorganized CI
UHAX	DBFUHAC7	Exit	DEDB HSRE Access segment in non reorganized CI
UHAR	DBFUHAR0	Entry	DEDB HS Utility Async Read-Ahead
HAR2	DBFUHAR0	Wait for inflight async read ahead	DEDB HS Utility Async Read-Ahead
HAR1	DBFUHAR0	Setup SRB for async read ahead	DEDB HS Utility Async Read-Ahead
UHAX	DBFUHAR0	Exit	DEDB HS Utility Async Read-Ahead
PCH0	DBFUHAR0	Entry to routine UHARPCHO	DEDB HS Utility Async Read-Ahead
PCH1	DBFUHAR0	Found non-HSSP XCRB in UOW	DEDB HS Utility Async Read-Ahead
PCH2	DBFUHAR0	XCRB has DMHR	DEDB HS Utility Async Read-Ahead
PCH3	DBFUHAR0	Data copied to HSSP buffer	DEDB HS Utility Async Read-Ahead
PCH4	DBFUHAR0	Release DMHR and XCRB	DEDB HS Utility Async Read-Ahead
PCHX	DBFUHAR0	Exit from routine UHARPCHO	DEDB HS Utility Async Read-Ahead
UHDA	DBFUHDA0	Entry	DEDB HSRE Process Alloc/Dealloc of IOVF Cis
UHDX	DBFUHDA0	Exit	DEDB HSRE Process Alloc/Dealloc of IOVF Cis
UHDS	DBFUHDS0	Entry	DMAC Sync
UHDX	DBFUHDS0	Exit	DMAC Sync

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
UHGS	DBFUHGS7	Entry	DEDB HSRE Get Space to Copy Segment
UHGX	DBFUHGS7	Exit	DEDB HSRE Get Space to Copy Segment
UHIO	DBFUHIO0	Entry	DEDB HSRE Read a specified CI
UHIX	DBFUHIX0	Exit	DEDB HSRE Read a specified CI
UHPR	DBFUHPR7	Entry	DEDB HSRE Process Root Chain of RAP CI
UHPX	DBFUHPR7	Exit	DEDB HSRE Process Root Chain of RAP CI
UHRD	DBFUHRD7	Entry	DEDB HSRE Read Database Record
UHRX	DBFUHRD7	Exit	DEDB HSRE Read Database Record
UHRE	DBFUHRE0	Entry	DEDB HSRE UOW Reorg Mainline
UHRX	DBFUHRE0	Exit	DEDB HSRE UOW Reorg Mainline
UHSR	DBFUHSR0	Entry	DEDB HSRE Mainline
UHSX	DBFUHSR0	Exit	DEDB HSRE Mainline
UHSS	DBFUHSS0	Entry	DEDB High Speed Utility Services
UHS1	DBFUHSS0	Wait for inflight async read ahead	DEDB High Speed Utility Services
NHSS	DBFUHSS0	Exit	DEDB High Speed Utility Services
UHSI	DBFUHSS0	HSSP Image Copy Termination	DEDB High Speed Utility Services
UMAF	DBFUMAF0	Entry	DEDB Utility Page Fix Services
VMAF	DBFUMAF0	Exit	DEDB Utility Page Fix Services
UMAI	DBFUMAI0	Entry	DEDB Utility I/O Services
ENQR	DBFUMAI0	Test for SDEP CI Lock	DEDB Utility I/O Services
VMAI	DBFUMAI0	Exit	DEDB Utility I/O Services
UMAL	DBFUMAL0	Entry	DEDB Utility Logging
VMAL	DBFUMAL0	Exit	DEDB Utility Logging
UMAN	DBFUMAN0	Entry	DEDB Utility Services
VMAN	DBFUMAN0	Exit	DEDB Utility Services
UMAV	DBFUMAV0	Entry	DEDB Utility Set ADS Available
VMAC	DBFUMAV0	Exit	DEDB Utility Set ADS Available
UMDS	DBFUMDS0	Entry	DEDB Utility DMAC Sync (CONNECT/DISCONNECT)
VMDS	DBFUMDS0	Exit	DEDB Utility DMAC Sync (CONNECT/DISCONNECT)
UMFT	DBFUMFT0	Entry	DEDB ADS Format
VMFT	DBFUMFT0	Exit	DEDB ADS Format
UMMT	DBFUMMT0	Entry	DEDB Utility MTO Message Services

Table 191. Fast Path trace entries (continued)

Trace ID	Module	Trace point	Comments
UNMT	DBFUMMT0	Exit	DEDB Utility MTO Message Services
UMNO	DBFUMNO0	Entry	DEDB Utility Notify Partners to Open ADS
VMNO	DBFUMNO0	Exit	DEDB Utility Notify Partners to Open ADS
VSCL	DBFVSCL0	Entry	DEDB VSO Area Close
VSCE	DBFVSCL0	Exit	DEDB VSO Area Close
VSOP	DBFVSOP0	Entry	DEDB VSO Area Open
VSOE	DBFVSOP0	Exit	DEDB VSO Area Open
XPIX	DBFXPIX0	Entry	Free a chain of XCRBs/UXRBS
NPIX	DBFXPIX0	Exit	Free a chain of XCRBs/UXRBS

## Type-2 trace tables

You can activate IMS type-2 trace tables at IMS startup or while IMS is active. Unlike type-1 trace tables, type-2 trace tables can be updated dynamically using the **UPDATE TRACE** command while IMS is up, and they can be queried to gather diagnostic information.

The settings for type-2 trace tables are specified in the DFSDFxxx PROCLIB member by using the TRCLEV statement. The size of each type-2 trace table can be specified by the user, whereas the size of type-1 trace tables is a fixed value determined by IMS. Type-2 trace tables can be used in the same environments as type-1 trace tables.

For more information about the differences between type-1 and type-2 trace tables, see [IMS Trace Facility \(System Administration\)](#).

Type-2 trace records are written to one or more of the trace tables that are shown in the following table:

Table 192. Type-2 trace tables for type-2 trace records	
Table type	Table description
DDL	Traces DDL events within IMS.
ERR	Traces error events within IMS
TRCE	Traces IMS trace services events
USRX	Traces IMS User Exit services

### Related concepts

[IMS Trace Facility \(System Administration\)](#)

### Related tasks

[Setting up type-2 trace tables \(System Definition\)](#)

### Related reference

[QUERY TRACE command \(Commands\)](#)

[UPDATE TRACE command \(Commands\)](#)

## Type-2 trace records

You can analyze type-2 trace records to gather diagnostic information.

Type-2 trace records are mapped by a macro that follows the naming convention of DFSTxxx macro, where xxx represents the type-2 trace table name as shown in the following table:

Table 193. Type-2 trace record mapping macros	
Trace macro name	Description
DFSTDDL	DDL trace records

#### Related tasks

[Setting up type-2 trace tables \(System Definition\)](#)

## IMS shutdown trace table

When IMS begins shutdown processing, IMS populates a trace table for the shutdown activity. The shutdown trace table is included in a dump of the IMS control region, and can help you determine which module is preventing IMS shutdown from completing.

The shutdown trace table is written to memory and contains 64 one-byte entries.

You can find the IMS shutdown trace table in a formatted dump from the pointer to DGA\_SDTT in the Diagnostic Anchor Block (DGA), which is pointed to by SCDDGA in the System Contents Directory (SCD) control block. Each entry is one byte and contains the ID of the module entry or exit point. The table structure is defined in the DFSSDTT macro. Entries are written to the shutdown trace table in the same sequence in which they arrived. When the end of the table is reached, new entries are wrapped to the start of the table and overwrite the oldest existing entries.

The following table shows the trace value for each module entry or exit point. Unused shutdown trace table entries contain zeroes.

Table 194. Trace-to-code-module-name cross reference for the IMS shutdown trace table

Module name	Entry	Exit: RC=0
DBFTERM0	X'40'	X'80'
DBFTERM1(EP)	X'41'	X'81'
DFSCPCP0	X'42'	X'82'
DFSCST00	X'43'	X'83'
DFSDLOC0	X'44'	X'84'
DFSICIO0	X'45'	X'85'
DFSICLX0	X'46'	X'86'
DFSICL20	X'47'	X'87'
DFSIPCP0	X'48'	X'88'
DFSRCPO0	X'49'	X'89'
DFSRCRT0	X'4A'	X'8A'
DFSSDL40	X'4B'	X'8B'
DFSTERM0	X'4C'	X'8C'
DFSTRM00	X'4D'	X'8D'
UNLOCK00	X'4E'	X'8E'
ESSSHUT0	X'4F'	X'8F'
DFSIPCP1	X'50'	X'90'
DFSIPCP2	X'51'	X'91'



Table 194. Trace-to-code-module-name cross reference for the IMS shutdown trace table (continued)

Module name	Entry	Exit: RC=0
DFSIPCP3	X'52'	X'92'
DFSICM30	X'53'	X'93'
DFSQC010	X'54'	X'94'
OTMAPHA1	X'55'	X'95'
OTMAPHA2	X'56'	X'96'
OTMAPHA3	X'57'	X'97'
DFS6ECT0	X'58'	X'98'



## Notices

---

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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 grant 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 character set (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 publication 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 wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM 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 document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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 IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate 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. Sample Programs. © Copyright IBM Corp. \_enter the year or years\_.

## Programming interface information

---

This information is intended to help programmers, operators, and system support personnel diagnose IMS problems. This information also documents Diagnosis, Modification or Tuning Information provided by IMS.

Diagnosis, Modification or Tuning information is provided to help you diagnose, modify, or tune IMS. Do not use this Diagnosis, Modification or Tuning information as a programming interface.

Diagnosis, Modification or Tuning Information is identified where it occurs, either by an introductory statement to a section or topic, or by the following marking: Diagnosis, Modification or Tuning Information.



## 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 <http://www.ibm.com/legal/copytrade.shtml>.

Adobe, the Adobe logo, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux® is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.





# 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.

To learn more, see [IBM Privacy Statement](#).



## Bibliography

---

This bibliography lists all of the publications in the IMS 15.6 library.

<b>Title</b>	<b>Acronym</b>
<i>IMS Version 15.5 Application Programming</i>	APG
<i>IMS Version 15.5 Application Programming APIs</i>	APR
<i>IMS Version 15.5 Commands, Volume 1: IMS Commands A-M</i>	CR1
<i>IMS Version 15.5 Commands, Volume 2: IMS Commands N-V</i>	CR2
<i>IMS Version 15.5 Commands, Volume 3: IMS Component and z/OS Commands</i>	CR3
<i>IMS Version 15.5 Communications and Connections</i>	CCG
<i>IMS Version 15.5 Database Administration</i>	DAG
<i>IMS Version 15.5 Database Utilities</i>	DUR
<i>IMS Version 15.5 Diagnosis</i>	DGR
<i>IMS Version 15.5 Exit Routines</i>	ERR
<i>IMS Version 15.5 Installation</i>	INS
<i>IMS Version 15.5 Licensed Program Specifications</i>	LPS
<i>IMS Version 15.5 Messages and Codes, Volume 1: DFSMessages</i>	MC1
<i>IMS Version 15.5 Messages and Codes, Volume 2: Non-DFS Messages</i>	MC2
<i>IMS Version 15.5 Messages and Codes, Volume 3: IMSAbend Codes</i>	MC3
<i>IMS Version 15.5 Messages and Codes, Volume 4: IMSComponent Codes</i>	MC4
<i>IMS Version 15.5 Operations and Automation</i>	OAG
<i>IMS Version 15.5 Release Planning</i>	RPG
<i>IMS Version 15.5 System Administration</i>	SAG
<i>IMS Version 15.5 System Definition</i>	SDG
<i>IMS Version 15.5 System Programming APIs</i>	SPR
<i>IMS Version 15.5 System Utilities</i>	SUR



---

# Index

## Special Characters

/DIAGNOSE command SNAP function  
    console alternative [573](#)  
/TRACE command  
    starting DC trace [279](#)  
    stopping DC trace [280](#)

## Numerics

01/03 log record  
    trace [316](#)  
3270 error recovery  
    sense-status message [324](#)  
5A0B trace entry [383](#)  
6701 log records  
    OTMA  
        shared queues log records [509](#)  
        synchronous callout log record format [506](#)  
        TIB3 record [509](#)  
    synchronous callout log record format [506](#), [509](#)  
6701-MRQB records (command for obtaining) [304](#)  
6701-MRQE diagnostic records  
    control blocks and mapping macros [303](#)  
    description [302](#)  
    sample JCL for printing [303](#)  
67D0 log record  
    about [395](#)  
    DSECT name [484](#)  
    IMS Spool API [395](#)  
    issuing module [484](#)  
    mapping macro [484](#)

## A

abend dumps  
    causes [160](#)  
abend processing for Spool API support [395](#)  
abends  
    IMS abend search and notification [31](#)  
    research [31](#)  
    search and notification, enabling [31](#)  
    searching online documents and technical support  
        databases [31](#)  
ABENDU1026  
    Fast Path problem analysis  
        description [401](#)  
        procedure [401](#)  
ABENDUxxxx keyword procedure [36](#)  
ABENDxxx keyword procedure [35](#)  
abnormal save area set [49](#)  
accessibility  
    features [xiii](#)  
    keyboard shortcuts [xiii](#)  
active save set  
    finding during DC analysis [323](#)  
ADSC definition/mapping macro [72](#)

AIBREASN code  
    message determination [304](#)  
AIBREASN codes (Queue Control Facility/Message  
    Requeuer)  
    description [304](#)  
ALDS definition/mapping macro [73](#)  
AMPB definition/mapping macro [73](#)  
analysis  
    dumps [563](#)  
analyzing problems  
    using log records [469](#)  
APARs  
    preparing [68](#)  
    procedure [68](#)  
APPC problem, diagnosing [19](#)  
areas  
    global [410](#)  
    load list [410](#)  
AS [391](#)  
audit log [24](#)  
audit log records  
    Repository Server  
        managing [25](#)

## B

BALG definition/mapping macro [73](#)  
base primitive environment (BPE)  
    external trace  
        formatting panels [559](#)  
Base Primitive Environment (BPE)  
    DBRC [267](#)  
    trace entry [12](#)  
    trace records [267](#)  
batch environment  
    call image capture trace [159](#)  
BFSP definition/mapping macro [73](#)  
BFUS definition/mapping macro [73](#)  
BGNRETRY trace entry [247](#)  
BHDR definition/mapping macro [73](#)  
BLOCKHDR definition/mapping macro [73](#)  
BMP  
    dependent regions  
        useful dumps for [554](#)  
BMP dependent region, useful dumps for [554](#)  
BPE  
    DBRC  
        trace records [267](#)  
    external trace  
        formatting panels [559](#)  
    IMS Connect  
        displaying status of external trace [433](#)  
        formatting external trace data [434](#)  
    trace entry, formatting [12](#)  
    tracing  
        IMS Connect [429](#)  
BPE-based DBRC

- BPE-based DBRC (*continued*)
  - example trace records [268](#)
- BPE-based DBRC service aids [267](#)
- BSPH definition/mapping macro [73](#)
- BUFC definition/mapping macro [73](#)
- BUFENTRY definition/mapping macro [73](#)
- buffer handler
  - function codes [201](#)
  - module trace IDs [204](#)
  - pool (VSAM) [88](#)
  - return codes [204](#)
- buffer handler trace entries
  - DL/I [204](#)
  - space management [204](#)
- buffers
  - receive-any [322](#)
  - space [322](#)
- BUFMSTRA (message processing) trace
  - description [447](#)
- BUFSMVID trace description [464](#)

## C

- CADSECT definition/mapping macro [73](#)
- call image capture trace
  - batch environment [159](#)
  - online environment [160](#)
  - retrieving data from log data set [160](#)
- CALLER= parameter
  - FMTIMS statement example [517](#)
- calls used with Spool API support
  - CHNG [391](#)
- CBT (control block table) pool [412](#)
- CBT (control block table) pools
  - descriptions [81](#)
  - listed [81](#)
- CBT definition/mapping macro [73](#)
- CCB definition/mapping macro [73](#)
- channel-to-channel access method trace stack [458](#)
- CHE FREEZE [59](#)
- checkpoint messages
  - CQS [140](#)
- CHNG call
  - Spool API [391](#)
- CI (control interval)
  - DEDB problem
    - CI 0 [405](#)
    - CI 1 [405](#)
    - common data [405](#)
    - first DOVF CI [405](#)
    - first IOVF CI [405](#)
    - other DOVF CIs [405](#)
    - other IOVF CIs [405](#)
    - other SDEP CI [405](#)
    - RAP CI [405](#)
    - scraps [405](#)
    - type identification [405](#)
- CIB definition/mapping macro [73](#)
- CIBSTRAC trace
  - content
    - entry [326](#)
    - example [326](#)
    - locating [326](#)
- CIBTRACE trace

- CIBTRACE trace (*continued*)
  - content
    - entry [326](#)
    - example [326](#)
    - locating [326](#)
- CICS
  - diagnosis
    - ISC TCP/IP links [21](#)
  - ISC TCP/IP
    - diagnosis [21](#)
- CIRCA definition/mapping macro [73](#)
- CLB definition/mapping macro [73](#)
- CLLE definition/mapping macro [73](#)
- CNT definition/mapping macro [73](#)
- codes
  - error code examples [391](#)
  - status of CHNG and SETO calls [391](#)
- collecting data
  - APPC-related problem [19](#)
  - control or DL/I region loop [10](#)
  - control region hang [9](#)
  - control region wait [9](#)
  - CQS-related problem [137](#)
  - database-related problem [27](#)
  - DBCTL-related problem [15](#)
  - DBRC-related problem [15](#)
  - DC-related problem [16, 17](#)
  - ESAF Interface-related problem [27](#)
  - IMS dependent region loop [11](#)
  - IMS dependent region wait [11](#)
  - OTMA-related problem [17](#)
  - Recovery Resource Service-related problem [28](#)
- commands
  - /DISPLAY TRACE EXIT [313](#)
  - /TRA SET ON TABLE FAST [416](#)
  - TRA SET OFF TABLE FAST [416](#)
  - VERBX [530](#)
- Common Service Layer (CSL)
  - service aids [151](#)
  - trace [582](#)
- Common Service Layer trace (CSLT)
  - format [582](#)
- Common Storage Area (CSA)
  - global [410](#)
  - load list [410](#)
- common trace interface
  - trace tables [577](#)
- communication analyzer (DFSICIO0)
  - description [277](#)
  - device-dependent module
    - entry point [277](#)
    - trace ID [277](#)
  - save area [277](#)
  - trace output [279](#)
  - trace record example [278](#)
  - trace record format [278](#)
- communication task trace
  - description [445](#)
- COMPARE option [229](#)
- COMPARE statement [157](#)
- component identification keyword procedure [34](#)
- contents
  - DBRC [239](#)
- control (CTL) address space



- control (CTL) address space *(continued)*
  - online formatted dump [567](#)
- control address space [567](#)
- control block
  - external SNAP call [157](#)
  - interrelationship diagram [88](#)
  - linkage for static DB/DC environment [71](#)
  - locating in an IMS
    - Fast Path [408](#)
  - locating using load list [410](#)
  - logged at time of error [303](#)
  - mapping macros [303](#)
  - relationships created for MAIN pool [88](#)
  - relationships for DFSCBT00 pools [88](#)
  - relationships for DFSPOOL pools [88](#)
  - relationships for preallocated storage blocks [88](#)
  - sequential buffering diagram [88](#)
- control block table (CBT) pool [412](#)
- control block table (CBT) pools
  - descriptions [81](#)
  - listed [81](#)
- control blocks
  - acronym [72](#)
  - definitions [72–80](#)
  - description [72](#)
  - macros that generate [72](#)
  - mapping macros [72](#)
- control blocks, relocation during special abend processing [395](#)
- control interval
  - DEDB problem
    - CI 0 [405](#)
    - CI 1 [405](#)
  - common data [405](#)
  - first DOVF CI [405](#)
  - first IOVF CI [405](#)
  - other DOVF CIs [405](#)
  - other IOVF CIs [405](#)
  - other SDEP CI [405](#)
  - RAP CI [405](#)
  - scraps [405](#)
  - type identification [405](#)
- control region loop, diagnosing [10](#)
- control region wait or hang, diagnosing [9](#)
- control region, FMTIMS statement example [517](#)
- CPM definition/mapping macro [73](#)
- CPT definition/mapping macro [73](#)
- CQS
  - checkpoint messages [140](#)
  - log records [149](#)
  - structure
    - rebuild [141](#)
- CQS (Common Queue Server)
  - additional manual intervention [138](#)
  - CQS structure recovery data set [140](#)
  - CQS z/OS log stream [140](#)
  - diagnosis [137](#)
  - log records [146](#)
  - printing [148](#)
  - problem diagnostics [23](#)
  - structure dump contents [138](#)
- CRB definition/mapping macro [73](#)
- creating search arguments [34](#)
- CRTU

- CRTU *(continued)*
  - formatted log record [293](#)
- CSAB definition/mapping macro [73](#)
- CSL
  - problem diagnostics [23](#)
  - trace records [151](#)
- CSL (Common Service Layer)
  - trace [582](#)
- CSLT (Common Service Layer trace)
  - format [582](#)
- CSV definition/mapping macro [73](#)
- CTB definition/mapping macro [73](#)
- CTL (control) address space
  - online formatted dump [567](#)
- CTM definition/mapping macro [74](#)
- CTT definition/mapping macro [74](#)
- CULE definition/mapping macro [74](#)
- CVB definition/mapping macro [74](#)
- CXB definition/mapping macro [74](#)

## D

- data
  - general problems
    - collecting [1](#)
- data collection
  - DB2 ESS interface problems [27](#)
- data communication (DC)
  - call analyzer (DFSDLA30) [317](#)
  - FMTIMS statement example [518](#)
  - service aid
    - DC trace [279](#)
    - description [277](#)
    - finding the active save set [323](#)
    - IBM 3270 error recovery analysis [324](#)
    - IMS transaction trace [317](#)
    - IMS-VTAM interface [323](#)
    - message format service module traces [326](#)
    - OTMA dumps [391](#)
    - OTMA log records [390](#)
    - OTMA module-to-code cross reference table [388](#)
    - OTMA trace [365](#)
    - OTMA verb-to-code cross reference table [389](#)
    - receive-any buffer analysis [321](#)
    - terminal communication task trace [277](#)
- Data Communication (DC)
  - trace output example [296](#)
- data management block (DMB) diagram [88](#)
- data sets
  - non-SMS-managed
    - recovering after Sx37 abends [234](#)
  - SMS-managed
    - extending [236](#)
    - recovering after Sx37 abends [236](#)
- database
  - control block diagram [88](#)
  - diagnosis [27](#)
  - diagnostic techniques [155](#)
  - log analysis [221](#)
  - log record (X'50') DSECT [221](#)
  - Recovery Resource Service [28](#)
  - searching techniques [66](#)
- Database Control (DBCTL)
  - DRA dumps [397](#)

Database Control (DBCTL) (*continued*)

- dump title format [543](#)
- problem, diagnosing [15](#)
- recovery tokens [398](#)
- service aids [397](#)

Database Recovery Control

- control block diagram [88](#)
- external trace
  - example [263](#)
  - record format [263](#)
- FMTIMS statement example [518](#)
- RECON data set
  - contents [239](#)
  - diagnostic aid [239](#)
- service aids [239](#)

database recovery control (DBRC)

- trace header record [245](#)

Database Recovery Control (DBRC)

- group services entries [254](#)
- internal trace example [255](#), [273](#)

database resource adapter (DRA)

- dump title format [543](#)

Database Resource Adapter (DRA)

- Analyzing DRA Problems [398](#)
- dumps [397](#)
- recovery tokens [398](#)
- service aids [397](#)

databases

- dump
  - formatting [563](#)

DB2 ESS interface problems

- diagnosing [27](#)

Db2 for z/OS stored procedures

- diagnosing hung threads and UORs [465](#)
- diagnosis [465](#)
- hung threads and UORs, diagnosing [465](#)

DBCTL (Database Control)

- DRA dumps [397](#)
- dump title format [543](#)
- problem, diagnosing [15](#)
- recovery tokens [398](#)
- service aids [397](#)

DBPCB definition/mapping macro [74](#)

DBRC

- diagnosis
  - internal trace [243](#)
  - security override [15](#)

DBRC (database recovery control)

- trace header record [245](#)

DBRC (Database Recovery Control)

- control block diagram [88](#)
- external trace
  - example [263](#)
  - record format [263](#)
- FMTIMS statement example [518](#)
- group services entries [254](#)
- internal trace example [255](#), [273](#)
- RECON data set
  - contents [239](#)
  - diagnostic aid [239](#)
- service aids [239](#)

DBRC formatted output

- example [264](#)

DBRC problem, diagnosing [15](#)

DBRC service aids

- BPE-based [267](#)

DBRC trace (DSPTRACE)

- BGNRETRY entry [247](#)
- DSPCABNO entry [247](#)
- DSPCTRTO entry [247](#)
- DSPSTACK entry [245](#)
- DSPSTFRE entry [245](#)
- DSPSTGET entry [245](#)
- DSPURI00 entry [249](#)
- GETFEED entry [249](#)
- locating [244](#)
- processing flow [245](#)
- using [244](#)

DC (data communication)

- call analyzer (DFSDLA30) [317](#)
- FMTIMS statement example [518](#)
- service aid
  - DC trace [279](#)
  - description [277](#)
  - finding the active save set [323](#)
  - IBM 3270 error recovery analysis [324](#)
  - IMS transaction trace [317](#)
  - IMS-VTAM interface [323](#)
  - message format service module traces [326](#)
  - OTMA dumps [391](#)
  - OTMA log records [390](#)
  - OTMA module-to-code cross reference table [388](#)
  - OTMA trace [365](#)
  - OTMA verb-to-code cross reference table [389](#)
  - receive-any buffer analysis [321](#)
  - terminal communication task trace [277](#)

DC (Data Communication)

- trace output example [296](#)

DC problem, diagnosing [16](#), [17](#)

DC trace

- diagnosing line problem [291](#)
- diagnosing terminal problem [291](#)
- starting [279](#)
- stopping [280](#)
- trace record
  - identifiers [282](#)
  - printing [281](#)
  - table of record types and contents [284](#)

DCB definition/mapping macro [74](#)

DCB-EXT definition/mapping macro [74](#)

DDIR definition/mapping macro [74](#)

DDL trace

- format [665](#)

deadlock involving non-IRLM resources [62](#)

deadlock involving only IRLM resources [62](#)

debugging and diagnostic aids for IMS Spool API

- debugging tips [395](#)
- internal trace table [395](#)

DEDB (data entry database)

CI problem

- CI 0 [405](#)
- CI 1 [405](#)
- common data [405](#)
- diagnosis aids [405](#)
- first DOVF CI [405](#)
- first IOVF CI [405](#)
- other DOVF CIs [405](#)
- other IOVF CIs [405](#)

DEDB (data entry database) (*continued*)

CI problem (*continued*)

other SDEP CI [405](#)

RAP CI [405](#)

scraps [405](#)

type identification [405](#)

DELETE module

DL/I trace, using [207](#)

dependency keyword table [65](#)

dependency keywords [67](#)

dependent region address space (DP)

FMTIMS statement example [519](#)

dependent regions

BMP

useful dumps for [554](#)

IFP (IMS Fast Path)

useful dumps for [554](#)

IMS Fast Path (IFP)

useful dumps for [554](#)

developing search arguments [33](#)

DEVICE BUSY category

sense-status message

3270 recovery analysis [324](#)

DEVICE END category

sense-status message

3270 recovery analysis [324](#)

device-dependent module

communication analyzer

entry point [277](#)

save area [277](#)

trace ID [277](#)

trace output [279](#)

trace record example [278](#)

trace record format [278](#)

device-dependent module (DDM)

functions [445](#)

DFS070

diagnostic message example [306](#)

error conditions [306](#)

example diagnostic message [314](#)

DFS081

reason codes [313](#)

DFS1269E OTMA failure message [390](#)

DFS1959E

reason codes [351](#)

DFS1965E APPC/MVS

reason codes, call failures [364](#)

DFS2712I message

using in Fast Path problem analysis [401](#)

DFS3672I message [328](#)

DFS62FD0

reason codes [362](#)

DFS62FD1

reason codes [363](#)

DFS6CKP0

reason codes [357](#)

DFS6ECT0

reason codes [358](#)

DFS6IDC0

reason codes [358](#)

DFS6LUS0

reason codes [359](#)

DFS6LUS1are

reason codes [360](#)

DFS6LUS2

reason codes [360](#)

DFS6QFX0

reason codes [361](#)

DFS6RST0

reason codes [362](#)

DFSALM00

reason codes [352](#)

DFSAOSW0

reason codes [352](#)

DFSAPPC0

reason codes [353](#)

DFSATB00

reason codes [354](#)

DFSAVEC definition/mapping macro [74](#)

DFSCBT00 pools [88](#)

DFSCMC00 module, MSC analyzer [459](#)

DFSCMC10 module

abnormal-end appendage [461](#)

channel-end appendage [461](#)

shutdown appendage [462](#)

DFSCMC40 module

attention DIE routine [459](#)

I/O request DIE routine

[460](#)

DFSCMC50 module

shutdown processing routine [459](#)

DFSCMD00

reason codes [354](#)

DFSCMLC0

reason codes [354](#)

DFSCMS00

reason codes [355](#)

DFSCNXA0 module

error messages [336](#)

location codes for error messages [328](#)

tracing errors [328](#)

DFSCRTU0

return codes [293](#)

DFSDDLTO (DL/I test program) [157](#)

DFSDLA30

routing errors [316](#)

DFSDLA30 (DC call analyzer)

tracing using IMS transaction trace [317](#)

DFSDLTRO call image capture trace description [221](#)

DFSDOPTTE definition/mapping macro [74](#)

DFSDPBFH definition/mapping macro [74](#)

DFSERA10 (File Select and Formatting Print utility)

formatted output example [511](#)

printing DC trace records [281](#)

unformatted output example [511](#)

DFSERA20 [572](#)

DFSHAV70

reason codes [364](#)

DFSHCIO0

reason codes [355](#)

DFSICIO0 (communication analyzer)

description [277](#)

device-dependent module

entry point [277](#)

trace ID [277](#)

save area [277](#)

trace output [279](#)

trace record example [278](#)

- DFSICIO0 (communication analyzer) (*continued*)
  - trace record format [278](#)
- DFSLUM00
  - reason codes [363](#)
- DFSMSCEO
  - routing trace [312](#)
  - trace records [314](#)
  - user exit consolidation [306](#)
- DFSPPX0 parameter block diagram [88](#)
- DFSQMRQ0 processor module description [297](#)
- DFSRLM00
  - reason codes [355](#)
- DFSRLM10
  - reason codes [356](#)
- DFSSBHD0 utility
  - using with SB IMAGE CAPTURE option [229](#)
- DFSSBWO definition/mapping macro [74](#)
- DFSSLUM0
  - reason codes [357](#)
- DFSVTP00
  - overlay for posting of VTCBs [294](#)
- DFSXLUM0
  - reason codes [364](#)
- DFSZD510 control block dump
  - description [230](#)
  - formatted example [230](#)
  - unformatted example [231](#)
- diagnosing
  - a control or DL/I region loop [10](#)
  - a control region wait or hang [9](#)
  - a CQS-related problem [137](#)
  - a database related problem [27](#)
  - a DBCTL-related problem [15](#)
  - a DBRC-related problem [15](#)
  - a DC-related problem [16, 17](#)
  - a Recovery Resource Service related problem [28](#)
  - an APPC-related problem [19](#)
  - an ESAF interface related problem [27](#)
  - an IMS dependent region wait or loop [11](#)
  - Fast Path [401](#)
  - OTMA-related problem [17](#)
- diagnosis
  - CQS log records [149](#)
  - data collection
    - IMS Connect [20](#)
    - ISC TCP/IP links [21](#)
  - DB2 ESS interface problems [27](#)
  - DBRC internal trace [243](#)
  - DBRC trace entries [254](#)
  - Fast Path external trace [415](#)
  - IMS Connect [20](#)
  - ISC
    - TCP/IP links [21](#)
  - Repository Server [24](#)
- diagnostic aids
  - APPC/IMS [339](#)
  - OTMA [365](#)
- diagnostic message
  - DFS070 [306](#)
- diagnostics
  - abend search and notification [31](#)
  - formatting a dump for analysis [566](#)
  - gathering data [1](#)
- DIF/MID linkage diagram [88](#)

- dispatcher trace
  - example [587](#)
  - format [587](#)
- DL/I
  - analyzing problems [163](#)
  - buffer handler trace entries [204](#)
  - call image capture trace description [221](#)
  - control block
    - description [72](#)
  - data record format [127](#)
  - FMTIMS statement example [519](#)
  - online formatted dumps
    - data areas dumped [570](#)
  - test program
    - debugging in batch environment [160](#)
    - description [157](#)
  - trace
    - output sample [204](#)
  - trace, DL/I
    - buffer handler function codes [201](#)
    - buffer handler module trace IDs [204](#)
    - buffer handler return codes [204](#)
    - DELETE module, using DL/I trace [207](#)
    - description [165](#)
    - JRNAD codes [199](#)
    - PSTLRPRM codes [183–185](#)
    - record format [167](#)
  - trace, other database-related
    - JCB (job control block) [155](#)
    - locating [164](#)
    - PI (program isolation) [220](#)
    - retrieve [208](#)
- DL/I region loop, diagnosing [10](#)
- DL/I test program (DFSDDLTO) [157](#)
- DL/I trace [204](#)
- DMAC definition/mapping macro [74](#)
- DMB definition/mapping macro [74](#)
- DMBSEC definition/mapping macro [74](#)
- DMCB definition/mapping macro [74](#)
- DMHR definition/mapping macro [74](#)
- DOC keyword procedure [39](#)
- documentation management
  - description [2](#)
  - dump preservation [4](#)
  - IMS master console log preservation [3](#)
  - IMS OLDS/SLDS preservation [4](#)
  - JES JOBLLOG preservation [3](#)
  - SYS1.LOGREC preservation [3](#)
  - z/OS system console (syslog) preservation [2](#)
- DOF/MOD linkage diagram [88](#)
- DOVF CI
  - diagnosing CI problem in DEDB
    - first CIs [405](#)
    - other CIs [405](#)
- DP (dependent region address space)
  - FMTIMS statement example [519](#)
- DRA
  - SDUMP output [397](#)
- DRA (database resource adapter)
  - dump title format [543](#)
- DRA (Database Resource Adapter)
  - Analyzing DRA Problems [398](#)
  - dumps [397](#)

DRA (Database Resource Adapter) (*continued*)

recovery tokens [398](#)

service aids [397](#)

DSEB definition/mapping macro [74](#)

DSECT

for database log record (X'50') [221](#)

DSG definition/mapping macro [74](#)

DSPCABN0 trace entry [247](#)

DSPCRTR0 trace entry [247](#)

DSPRLX10

trace entry layout [254](#)

DSPRLX1M

trace entry layout [254](#)

DSPRSYNC

trace entry layout [253](#)

DSPSTACK trace entry [245](#)

DSPSTFRE trace entry [245](#)

DSPSTGET trace entry [245](#)

DSPTRACE (DBRC trace)

BGNRETRY entry [247](#)

DSPCABN0 entry [247](#)

DSPCRTR0 entry [247](#)

DSPSTACK entry [245](#)

DSPSTFRE entry [245](#)

DSPSTGET entry [245](#)

DSPURI00 entry [249](#)

GETFEED entry [249](#)

locating [244](#)

processing flow [245](#)

using [244](#)

DSPURI00

trace entry example [255](#), [273](#)

DSPURI00 module

calling [249](#)

entry trace entry [249](#)

exit routine trace entry [249](#)

GETFEED trace entry [249](#)

DSPURI00 trace entry [249](#)

DSPURI80

trace entry layout [253](#)

DSPWRK1 definition/mapping macro [74](#)

dump

formatted online

CTL address space [567](#)

dump formatter

IMS Connect

accessing [427](#)

dump formatter panel

IPCS [321](#)

dumping

IMS Dump Formatter [566](#)

DUMPQ [59](#)

dumps

activation

CQS [140](#)

BMP regions [554](#)

buffer handler request sequence analysis [162](#)

detailed analysis [162](#)

DL/I call sequence analysis [162](#)

FMT0 options [551](#)

FMT0 options, batch [555](#)

formatted online

description [566](#)

DL/I address space [570](#)

dumps (*continued*)

formatting [428](#)

formattingdatabase [563](#)

formattingFast Path [563](#)

formattinggeneric values, overwriting [563](#)

formattingSystem [563](#)

formattingTransaction Manager [563](#)

formattingtutorial [563](#)

IFP regions [554](#)

initialization [428](#)

introduction [161](#)

IRLM address space [441](#)

MPP regions [554](#)

save area analysis [162](#)

SNAP

output [398](#)

**E**

ECB definition/mapping macro [75](#)

ECNT definition/mapping macro [75](#)

edited command format [549](#)

EDSG definition/mapping macro [75](#)

EIB definition/mapping macro [75](#)

EMHB definition/mapping macro [75](#)

EPCB definition/mapping macro [75](#)

EPF definition/mapping macro [75](#)

EPST definition/mapping macro [75](#)

EQEL definition/mapping macro [75](#)

error codes

0002 [391](#)

0004 [391](#)

0006 [392](#)

0008 [393](#)

000A [393](#)

000C [393](#)

000E [394](#)

error location codes

dynamic logon errors [332](#)

error messages issued by DFSCNXA0 [328](#)

ISC processing [330](#)

logging-on device characteristics [335](#)

MSC errors [331](#)

related to existing ISC session errors [333](#)

user-logon exit processing [333](#)

error recovery

3270 device

sense-status message [324](#)

ESAF Interface

diagnosis [27](#)

ESCD definition/mapping macro [75](#)

ESRB definition/mapping macro [75](#)

ESRT definition/mapping macro [75](#)

ESS (external subsystem)

trace record

format [596](#)

module ID and subfunction table [596](#)

variable section layout [599](#)

external SNAP call, control blocks dumped [157](#)

external subsystem

DB2 [27](#)

trace [27](#)

external subsystem (ESS)

trace output example [574](#)

- external subsystem (ESS) (*continued*)
  - trace record
    - format [596](#)
    - module ID and subfunction table [596](#)
    - variable section layout [599](#)
- external trace
  - Fast Path [417](#)
- external trace, DBRC
  - RECON I/O error processing example [263](#)
  - record format [263](#)
  - router processing example [263](#)

## F

- FAQE definition/mapping macro [75](#)
- Fast Path
  - ABENDU1026 analysis [401](#)
  - control block
    - locating in an IMS [408](#)
  - detailed control block diagram [88](#)
  - diagnosing [401](#)
  - dumps
    - formatting [563](#)
  - FMTIMS statement example [519](#)
  - general control block diagram [88](#)
  - service aid
    - CI contention analysis [405](#)
    - DEDB CI problem assistance aids [405](#)
    - description [401](#)
    - locating control blocks and tables in an IMS [408](#)
  - trace [639](#)
  - trace entry [641](#)
  - trace, external [417](#)
  - transaction retry
    - description [404](#)
    - processing flow [404](#)
    - system programmer response [404](#)
  - work area
    - locating in an IMS [408](#)
- FDB definition/mapping macro [75](#)
- FDT definition/mapping macro [75](#)
- FEDB definition/mapping macro [75](#)
- FEIB definition/mapping macro [75](#)
- File Select and Formatting Print utility (DFSERA10)
  - formatted output example [511](#)
  - printing DC trace records [281](#)
  - unformatted output example [511](#)
- FMTIMS statement
  - choosing parameters for ODF [514](#), [516](#)
  - control region example [517](#)
  - DBRC example [518](#)
  - DC example [518](#)
  - DL/I example [519](#)
  - DP example [519](#)
  - Fast Path example [519](#)
  - formatted areas [521](#), [532](#)
  - LOG example [519](#)
  - options [520](#), [531](#)
  - sample statements [517](#)
  - syntax restrictions [530](#)
  - table [516](#)
  - VSAM example [520](#)
- format

- format (*continued*)
  - log records prefix [504](#)
- formatted dump
  - contents [541](#)
  - offline [541](#)
- formatting panels
  - BPE External Trace [559](#)
  - components [559](#)
- FRB definition/mapping macro [75](#)
- function codes
  - trace tables [577](#)
  - type-1 [577](#)

## G

- GB definition/mapping macro [75](#)
- GBCB definition/mapping macro [75](#)
- general problems
  - collecting data [1](#)
- Generalized Sequential Access Method (GSAM)
  - control block dump [230](#)
  - detailed control block diagram [88](#)
  - formatted (DFSZD510)
    - description [230](#)
  - formatted dump (DFSZD510)
    - example [230](#)
  - general control block diagram [88](#)
  - out-of-space abend [234](#)
  - out-of-space abends [234](#), [236](#)
  - unformatted dump (DFSZD510), example [231](#)
- GETFEED trace entry [249](#)
- GLT definition/mapping macro [75](#)
- GPT definition/mapping macro [75](#)
- GQCB definition/mapping macro [75](#)
- group services entries
  - DBRC [254](#)
- GSAM
  - non-striped data sets
    - extending [236](#)
- GSAM (Generalized Sequential Access Method)
  - control block dump [230](#)
  - detailed control block diagram [88](#)
  - formatted (DFSZD510)
    - description [230](#)
  - formatted dump (DFSZD510)
    - example [230](#)
  - general control block diagram [88](#)
  - out-of-space abend [234](#)
  - out-of-space abends [234](#), [236](#)
  - unformatted dump (DFSZD510), example [231](#)
- GTF (Generalized Trace Facility) trace
  - DBRC-related
    - example [263](#)
    - formatting and printing [263](#)
    - record format [263](#)
    - using [263](#)

## H

- hang, diagnosing a control region [9](#)
- hardware I/O category, 3270 error recovery analysis [324](#)
- HDAM database
  - OSAM ESDS block format [131](#)



- HDAM database (*continued*)
  - segment format [129](#)
  - VSAM ESDS block format [131](#)
- HIDAM database
  - OSAM and VSAM ESDS block format [131](#)
  - segment format [129](#)
  - VSAM ESDS block format [131](#)
- HIDAM index database
  - VSAM LRECL format [132](#)
- HISAM database
  - block format [128](#)
  - LRECL format [128](#)
  - segment format [128](#)
- HSAM database
  - block format [127](#)
  - delete byte format [127](#)
  - flag byte format [127](#)
  - segment format [127](#)
- HSSD definition/mapping macro [76](#)
- HSSO definition/mapping macro [76](#)
- HSSP definition/mapping macro [76](#)
- HSSR definition/mapping macro [76](#)
- HWSRCDR data set [434](#)

## I

- IBFPRF definition/mapping macro [76](#)
- IBPOOL definition/mapping macro [76](#)
- IDC0 trace table entries
  - internal trace formats that map entries [336](#)
- IDSC definition/mapping macro [76](#)
- IEADMCxx
  - dump activation [140](#)
- IEEQE definition/mapping macro [76](#)
- IFP (IMS Fast Path)
  - dependent regions
    - useful dumps for [554](#)
- IFP (IMS Fast Path) dependent region, useful dumps for [554](#)
- IMODULE facility
  - CBT pool [412](#)
  - IMS control block/work area, locating
    - using load list [410](#)
- IMS
  - Offline Dump Formatter [427](#)
- IMS abend search and notification
  - searching online documents and technical support databases [31](#)
- IMS component identification numbers [34](#)
- IMS Connect
  - BPE trace [429](#)
  - diagnosis
    - data collection [20](#)
    - ISC problems [21](#)
  - error codes
    - Transaction Manager [438](#)
  - problems
    - data collection [20](#), [21](#)
    - recorder log record mapping [434](#), [435](#), [437](#)
    - recorder trace [438](#)
    - recorder trace facility [429](#)
    - service aids [427](#)
    - traces [429](#), [438](#)
    - tracing
      - configuring BPE for external trace [431](#)

- IMS Connect (*continued*)
  - tracing (*continued*)
    - displaying status of external trace [433](#)
    - formatting external trace data [434](#)
    - starting a BPE external trace [432](#)
    - stopping a BPE external trace [433](#)
    - tracing to the HWSRCDR data set [434](#)
- IMS control blocks [71](#)
- IMS dependent region wait or loop, diagnosing [11](#)
- IMS Dump Formatter
  - description [550](#)
  - formatting a dump [566](#)
  - using [556](#)
- IMS Fast Path (IFP)
  - dependent regions
    - useful dumps for [554](#)
- IMS Fast Path (IFP) dependent region, useful dumps for [554](#)
- IMS Spool API
  - CHNG and SETO calls [391](#)
  - debugging tips [395](#)
  - feedback from parsing errors [391](#)
  - interfacing directly to [391](#)
  - log records produced by [395](#)
  - special abend processing [395](#)
- IMS sysplex dump considerations
  - sysplex IEADMCxx dump activation [7](#)
  - sysplex IEADMCxx example [7](#)
- IMS transaction trace
  - description [317](#)
  - example [318](#)
- IMS Transaction trace
  - content [317](#)
  - starting [317](#)
- IMS.ACBLIB
  - members layout [71](#)
  - partitioned data set [71](#)
- INCORROUT keyword procedure [41](#)
- intent conflict [55](#)
- Interactive Dump Formatter
  - dump options
    - available [551](#)
    - in effect [551](#)
- interactive problem control system (IPCS)
  - IMS Dump Formatter [550](#)
  - IRLM configuration [441](#)
  - using with ODF [514](#)
- internal resource lock manager (IRLM)
  - keyword procedure [38](#)
  - latch unavailable [63](#)
  - lock request example [88](#)
  - overall control block diagram [88](#)
  - procedure for WAIT state [62](#)
  - service aid
    - dumps [441](#)
    - SYS1.LOGREC [441](#)
  - service aids
    - description [441](#)
    - storage manager pool diagram [88](#)
- internal trace table
  - description [395](#)
- internal traces
  - DBRC [243](#)
- Intersystem Communication (ISC) link
  - starting DC trace [279](#)

## Intersystem Communication (ISC) link *(continued)*

- stopping DC trace [280](#)
- intersystem control block diagram [88](#)
- INTERVENTION REQUIRED category
  - 3270 error recovery analysis [324](#)
- IOVF CI
  - diagnosing CI problem in DEDB
    - first CIs [405](#)
    - other CIs [405](#)
- IPCS
  - dump formatter panel [321](#)
  - invoking the Offline Dump Formatter [530](#)
  - symbols [563](#)
- IPCS (interactive problem control system)
  - IMS Dump Formatter [550](#)
  - IRLM configuration [441](#)
  - using with ODF [514](#)
- IRLM (internal resource lock manager)
  - keyword procedure [38](#)
  - latch unavailable [63](#)
  - lock request example [88](#)
  - overall control block diagram [88](#)
  - procedure for WAIT state [62](#)
  - service aid
    - dumps [441](#)
    - SYS1.LOGREC [441](#)
  - service aids
    - description [441](#)
  - storage manager pool diagram [88](#)
- ISC
  - diagnosis
    - TCP/IP links [21](#)
  - TCP/IP
    - diagnosis [21](#)
- ISC (Intersystem Communication) link
  - starting DC trace [279](#)
  - stopping DC trace [280](#)
- ISL definition/mapping macro [76](#)
- ISPL definition/mapping macro [76](#)
- ITASK ECB posting [595](#)

## J

- JCB (job control block) trace
  - content [155](#)
  - function codes [156](#)
  - output example [155](#)
- JCB definition/mapping macro [76](#)
- JCL (job control language)
  - printing 6701-MRQE records [303](#)
  - printing QCF SCRAPLOG records [302](#)
- job control block (JCB) trace
  - content [155](#)
  - function codes [156](#)
  - output example [155](#)
- job control language (JCL)
  - printing 6701-MRQE records [303](#)
  - printing QCF SCRAPLOG records [302](#)
- JRNAD codes [199](#)

## K

- keyboard shortcuts [xiii](#)

## keyword

- component identification procedure [34](#)
- definition [33](#)
- selecting [34](#)
- type-of-failure [35](#)
- using dependency keywords [67](#)
- keyword dictionary [63](#)
- keyword procedures
  - ABENDUxxxx [36](#)
  - ABENDxxx [35](#)
  - DOC [39](#)
  - INCORROUT [41](#)
  - IRLM [38](#)
  - MSG [41](#)
  - PERFM [40](#)
  - SAP analysis [47](#)
  - WAIT/LOOP [44](#)
- keywords
  - types used with CHNG and SETO calls [391](#)

## L

- latch trace
  - example [633](#)
  - format [630](#)
  - latch manager trace entries [630](#)
  - system locate control function entries [633](#)
  - use manager trace entries [631](#)
- LCB definition/mapping macro [76](#)
- LCD definition/mapping macro [76](#)
- LCDSECT definition/mapping macro [76](#)
- LCRE definition/mapping macro [76](#)
- legal notices
  - notices [669](#)
  - trademarks [669](#), [673](#)
- LEV definition/mapping macro [76](#)
- limits for locking resources [220](#)
- line problem, diagnosing using DC trace [291](#)
- Link performance problems
  - diagnosing [453](#)
- link problem
  - MSS1 and MSS2 record description [454](#)
  - MSS3 and MSS4 record description [457](#)
- Link statistics
  - MSC (Multiple Systems Coupling)
    - using to diagnose link performance problems [453](#)
- LIPARMS definition/mapping macro [76](#)
- LLB definition/mapping macro [76](#)
- lock request not granted [63](#)
- locking resources, limiting with LOCKMAX [220](#)
- LOCKMAX parameter
  - determining [220](#)
  - exceeding [220](#)
- log
  - FMTIMS statement example [519](#)
- log analysis tools
  - log records formatting [565](#)
- log analysis, database-related [221](#)
- log data set [160](#)
- log record trace
  - 01/03 [316](#)
- log records
  - 6701
    - OTMA record formats [505](#)



log records (*continued*)

- all records used to analyze IMS problems [469](#)
- CQS [149](#)
- data area format [505](#)
- description [146](#)
- dumps [365](#)
- for service errors [395](#)
- formatting [565](#)
- Log Merge utility [512](#)
- log sequence field format [505](#)
- OTMA
  - X'6701' log record formats [505](#)
- prefix [504](#)
- printing [148](#)
- produced by IMS Spool API [395](#)
- Repository Server [24](#)
- Repository Server, printing [26](#)
- SNAPs [365](#)
- subrecord area format [505](#)
- table [146](#)
- type 67D0 [395](#), [484](#)
- type X'29' [492](#)
- type X'50' [221](#)
- type X'67' [503](#), [511](#)
- type X'67D0' [484](#)
- types [146](#)
- viewing format [146](#), [469](#)
- X'6701'
  - OTMA record formats [505](#)
- log sequence field format [505](#)
- log subrecord and data area formats [505](#)
- log, FMTIMS statement example [514](#)
- logical LINK
  - starting DC trace [279](#)
  - stopping DC trace [280](#)
- loop, diagnosing a control or DL/I region [10](#)
- loop, diagnosing an IMS dependent region [11](#)
- LRECL format [128](#)
- LTB definition/mapping macro [76](#)
- LU manager
  - trace example [347](#)
  - trace format [340](#)
  - trace, start [340](#)
- LU manager trace [339](#)
- LXB definition/mapping macro [76](#)
- LXB trace
  - DFSCMC00 module, MSC analyzer [459](#)
  - DFSCMC10 module
    - abnormal-end appendage [461](#)
    - channel-end appendage [461](#)
    - shutdown appendage [462](#)
  - DFSCMC40 module
    - attention DIE routine [459](#)
    - I/O request DIE routine [460](#)
  - DFSCMC50 module
    - shutdown processing routine [459](#)
  - example [462](#)
  - using [458](#)

**M**

macros for mapping control blocks [72–80](#)  
main storage-to-main-storage

main storage-to-main-storage (*continued*)

- access method trace description [447](#)
- save set trace description [447](#)
- Manual Intervention for Dump Creation
  - deciding when to dump [5](#)
  - description [4](#)
  - IEADMCxx, MVS SYS1.PARMLIB
    - EADMCxx DUMP activation [6](#)
    - EADMCxx example for IMS [6](#)
    - IMS dump techniques [5](#)
  - mapping macros for control blocks [72–80](#)
- Message Format Service (MFS)
  - diagnosing problems [324](#)
  - module trace
    - CIBSTRAC [326](#)
    - CIBTRACE [326](#)
- message processing
  - APPC [364](#)
  - OTMA [364](#)
- message processing (BUFMSTRA) trace
  - description [447](#)
- Message queue recovery
  - QCF (Queue Control Facility) MRQ (Message Requeuer) [297](#)
- Message Requeuer (MRQ) Queue Control Facility (QCF)
  - DFSQMRQ0 processor module [297](#)
- Message Requeuer (MRQ)/Queue Control Facility (QCF)
  - AIBREASN codes
    - description [304](#)
  - JCL
    - printing 6701-MRQE records [303](#)
    - printing QCF SCRAPLOG records [302](#)
  - key fields and offsets of diagnostic records [301](#)
  - key fields in message [301](#), [302](#)
  - messages successfully requested [305](#)
  - MRQE diagnostic records [302](#)
  - obtaining additional diagnostics [304](#)
  - sample of successful message requeue [305](#)
  - sample SCRAPLOG record and description [300](#)
- message routing problems
  - diagnosing [306](#)
- MFS (Message Format Service)
  - diagnosing problems [324](#)
  - module trace
    - CIBSTRAC [326](#)
    - CIBTRACE [326](#)
- MID/DIF linkage diagram [88](#)
- MOD/DOF linkage diagram [88](#)
- module directory, locating [71](#)
- MRMB definition/mapping macro [77](#)
- MRQ (Message Requeuer)/QCF (Queue Control Facility)
  - AIBREASN codes
    - description [304](#)
  - JCL
    - printing 6701-MRQE records [303](#)
    - printing QCF SCRAPLOG records [302](#)
  - key fields and offsets of diagnostic records [301](#)
  - key fields in message [301](#), [302](#)
  - messages successfully requested [305](#)
  - MRQE diagnostic records [302](#)
  - obtaining additional diagnostics [304](#)
  - sample of successful message requeue [305](#)

- MRQ (Message Requeuer)/QCF (Queue Control Facility) (*continued*)
    - sample SCRAPLOG record and description [300](#)
  - MRQE diagnostic records
    - control blocks and mapping macros [303](#)
    - description [302](#)
    - sample JCL for printing [303](#)
  - MSC (Multiple Systems Coupling)
    - abnormal-end appendage [461](#)
    - analyzer trace entry [459](#)
    - attention DIE routine [459](#)
    - BUFMSTRA (message processing) trace, description [447](#)
    - BUFSMVID trace [464](#)
    - channel-end appendage [461](#)
    - channel-to-channel access method trace stack [458](#)
    - communication task trace
      - description [445](#)
      - diagram [446](#)
    - detailed control block diagram [88](#)
    - device-dependent module [445](#)
    - general control block diagram [88](#)
    - I/O request DIE routine [460](#)
    - link statistics
      - diagnosing link performance problems [453](#)
    - main storage-to-main storage access method trace [447](#)
    - main storage-to-main storage save set trace [447](#)
    - MSS1 and MSS2 record description [454](#)
    - MSS3 and MSS4 record description [457](#)
    - service aids [445](#)
    - shutdown appendage [462](#)
    - shutdown processing routine [459](#)
  - MSC link analysis [29](#), [448](#)
  - MSG keyword procedure [41](#)
  - MSNB definition/mapping macro [77](#)
  - MSS1 record
    - description [454](#)
    - significant field [454](#)
  - MSS2 record
    - description [454](#)
    - significant field [454](#)
  - MSS3 record
    - description [457](#)
  - MSS4 record
    - description [457](#)
  - Multiple Systems Coupling (MSC)
    - abnormal-end appendage [461](#)
    - analyzer trace entry [459](#)
    - attention DIE routine [459](#)
    - BUFMSTRA (message processing) trace, description [447](#)
    - BUFSMVID trace [464](#)
    - channel-end appendage [461](#)
    - channel-to-channel access method trace stack [458](#)
    - communication task trace
      - description [445](#)
      - diagram [446](#)
    - detailed control block diagram [88](#)
    - device-dependent module [445](#)
    - general control block diagram [88](#)
    - I/O request DIE routine [460](#)
    - link statistics
      - diagnosing link performance problems [453](#)
    - main storage-to-main storage access method trace [447](#)
    - main storage-to-main storage save set trace [447](#)
    - MSS1 and MSS2 record description [454](#)
    - MSS3 and MSS4 record description [457](#)
  - Multiple Systems Coupling (MSC) (*continued*)
    - service aids [445](#)
    - shutdown appendage [462](#)
    - shutdown processing routine [459](#)
- ## N
- no work to do (wait/loop) [53](#)
  - node trace
    - DC-related problem diagnosis [16](#)
    - turning on [16](#)
    - using to diagnose routing problems [316](#)
    - using with INCORROUT procedure [41](#)
    - using with WAIT/LOOP procedure [45](#)
- ## O
- obtaining 6701-MRQB records [304](#)
  - ODBA (Open Database Access)
    - best practices
      - Db2 for z/OS stored procedures, diagnosing hung threads [465](#)
      - Db2 for z/OS stored procedures
        - diagnosing hung threads [465](#)
  - ODF (Offline Dump Formatter)
    - data set input [513](#)
    - description [512](#)
    - executing [514](#)
    - introduction [513](#)
    - SDUMP input [513](#)
    - using with IPCS [514](#)
  - ODF (offline formatter)
    - FMTIMS parameter
      - table [514](#), [516](#)
    - recommendations for using [514](#)
  - Offline Dump Formatter
    - invoke [530](#)
  - Offline Dump Formatter (ODF)
    - data set input [513](#)
    - description [512](#)
    - executing [514](#)
    - ICPS symbols [563](#)
    - introduction [513](#)
    - SDUMP input [513](#)
    - using with IPCS [514](#)
  - offline formatter (ODF)
    - FMTIMS parameter
      - table [514](#), [516](#)
    - recommendations for using [514](#)
  - online environment
    - call image capture trace [160](#)
  - Online Recovery Manager
    - trace, starting [213](#)
  - Online Recovery Manager (ORTT)
    - example [219](#)
    - format [213](#)
    - starting [213](#)
  - Open Database Access (ODBA)
    - best practices
      - Db2 for z/OS stored procedures, diagnosing hung threads [465](#)
      - Db2 for z/OS stored procedures
        - diagnosing hung threads [465](#)

## Open Transaction Manager Access (OTMA)

- DFS1269E message [390](#)
- dumps [391](#)
- log record format
  - shared queues [509](#)
  - TIB3 [509](#)
- log record format, synchronous callout [506](#)
- log records [390](#)
- module-to-cross reference table [388](#)
- shared queues
  - log record format, TIB3 [509](#)
- trace
  - description [365](#)
  - format of trace records [365](#)
  - parallel RESUME TPIPE request processing [369](#)
  - RESUME TPIPE requests, parallel processing [369](#)
  - sync callout [378](#)
  - sync switch [383](#)
  - synchronous callout [378](#)
  - synchronous program switch [383](#)
  - X'5A20' [369](#)
- verb-to-code cross reference table [389](#)
- X'6701' log record formats [505](#)

## ORTT

- trace [212](#)

## ORTT (Online Recovery Manager)

- example [219](#)
- format [213](#)
- starting [213](#)

## OSAM (Overflow Sequential Access Method)

- buffer pool diagram [88](#)
- DECB with IOB in use [88](#)

## OSAM and VSAM ESDS block format [131](#)

## OTMA

- trace entry, user exits [375](#)
- tracing [365](#)

## OTMA (Open Transaction Manager Access)

- DFS1269E message [390](#)
  - dumps [391](#)
  - log record format
    - shared queues [509](#)
    - TIB3 [509](#)
  - log records [390](#)
  - module-to-cross reference table [388](#)
  - shared queues
    - log record format, TIB3 [509](#)
  - synchronous callout log record format [506](#)
  - trace
    - description [365](#)
    - format of trace records [365](#)
    - parallel RESUME TPIPE request processing [369](#)
    - RESUME TPIPE requests, parallel processing [369](#)
    - sync callout [378](#)
    - sync switch [383](#)
    - synchronous callout [378](#)
    - synchronous program switch [383](#)
    - X'5A20' [369](#)
  - verb-to-code cross reference table [389](#)
  - X'6701' log record formats [505](#)
- ## OTMA problem, diagnosing [17](#)
- out-of-spaceabend, GSAM [234](#)
  - out-of-spaceabends (GSAM)
    - recovering [234](#), [236](#)

## Overflow Sequential Access Method (OSAM)

## Overflow Sequential Access Method (OSAM) (continued)

- buffer pool diagram [88](#)
- DECB with IOB in use [88](#)

## P

- PAC definition/mapping macro [77](#)
- PAPL definition/mapping macro [77](#)
- parallel RECON access
  - trace entries [253](#)
- PARMLIST definition/mapping macro [77](#)
- PAT definition/mapping macro [77](#)
- PATE definition/mapping macro [77](#)
- PCA definition/mapping macro [77](#)
- PCB definition/mapping macro [80](#)
- PCIB definition/mapping macro [77](#)
- PCPARMS definition/mapping macro [77](#)
- PCT definition/mapping macro [77](#)
- PDAE definition/mapping macro [77](#)
- PDEX definition/mapping macro [77](#)
- PDIR definition/mapping macro [77](#)
- PDL definition/mapping macro [77](#)
- PEC definition/mapping macro [77](#)
- PERFM keyword procedure [40](#)
- PHDAM database
  - segment format [129](#)
  - variable-length segment format [135](#)
- PHIDAM database
  - segment format [129](#)
  - variable-length segment format [135](#)
- PI (program isolation)
  - problem analysis [219](#)
  - trace facility [220](#)
- PNT definition/mapping macro [77](#)
- POOLHDR definition/mapping macro [77](#)
- post code list [595](#)
- posting of ITASK ECBs [595](#)
- PPRE definition/mapping macro [77](#)
- PQE definition/mapping macro [77](#)
- preparing APARs [68](#)
- processes
  - space issues [322](#)
- processor module for QCF [297](#)
- program isolation (PI)
  - problem analysis [219](#)
  - trace facility [220](#)
- program parameters
  - LOCKMAX [220](#)
- PSB definition/mapping macro [77](#)
- PSDB definition/mapping macro [77](#)
- pseudoabend, cause [160](#)
- PST active [52](#)
- PST analysis [51](#)
- PST definition/mapping macro [77](#)
- PSTLRPRM codes [183–185](#)
- PTBWA definition/mapping macro [77](#)
- PTE definition/mapping macro [77](#)
- PTK definition/mapping macro [77](#)
- PTX definition/mapping macro [77](#)
- PURGE [59](#)
- PXPARMS definition/mapping macro [77](#)

## Q

QCB definition/mapping macro [78](#)  
QCF  
    key fields and offsets [301](#)  
    overview [298](#)  
QCF (Queue Control Facility) MRQ (Message Requeuer)  
    DFSQMRQ0 processor module [297](#)  
    message queue recovery [297](#)  
QCF (Queue Control Facility)/MRQ (Message Requeuer)  
    JCL  
        printing 6701-MRQE records [303](#)  
        printing MRQ SCRAPLOG records [302](#)  
    key fields and offsets of diagnostic records [301](#)  
    key fields in message [301](#), [302](#)  
    messages successfully requested [305](#)  
    MRQE diagnostic records [302](#)  
    obtaining additional diagnostics [304](#)  
    sample of successful message requeue [305](#)  
    sample SCRAPLOG record and description [300](#)  
QCF SCRAPLOG records, sample JCL for printing [302](#)  
QCS  
    dump activation [140](#)  
QEL definition/mapping macro [78](#)  
QMBA definition/mapping macro [78](#)  
qualifier codes  
    ETO parsing errors [335](#)  
    screen-attribute errors [335](#)  
    VTCB creation errors [335](#)  
Queue Control Facility (QCF) Message Requeuer (MRQ)  
    DFSQMRQ0 processor module [297](#)  
Queue Control Facility (QCF)/Message Requeuer (MRQ)  
    JCL  
        printing 6701-MRQE records [303](#)  
        printing MRQ SCRAPLOG records [302](#)  
    key fields and offsets of diagnostic records [301](#)  
    key fields in message [301](#), [302](#)  
    messages successfully requested [305](#)  
    MRQE diagnostic records [302](#)  
    obtaining additional diagnostics [304](#)  
    sample of successful message requeue [305](#)  
    sample SCRAPLOG record and description [300](#)  
Queue Control Facility/Message Requeuer  
    SCRAPLOG records [300](#)  
queue manager trace  
    description [634](#)

## R

RAP CI  
    diagnosing CI problem in DEDB  
        CI format [405](#)  
RCPARMS definition/mapping macro [78](#)  
RCTE definition/mapping macro [78](#)  
RDLWA definition/mapping macro [78](#)  
receive-any buffer analysis [321](#)  
RECON  
    trace entries [253](#)  
RECON data set  
    listing records [239](#)  
RECON data sets  
    security override [15](#)  
record formats [71](#)  
recorder log record mapping [434](#), [435](#), [437](#)

REG0 trace [277](#)  
REPLACE module  
    DL/I trace, using [207](#)  
Repository Server  
    audit log [24](#)  
    audit log records [25](#)  
    diagnosis [24](#)  
    log records, printing [26](#)  
    managing [25](#)  
    z/OS logger errors [25](#)  
request parameter list (RPL) [323](#)  
Resource Recovery Services trace (RRST) [611](#)  
retrieve trace  
    ID table [209–211](#)  
    output sample [207](#)  
    using [208](#)  
retrieving call image capture data [160](#)  
return codes  
    DFSCRTU0 [293](#)  
RHB definition/mapping macro [78](#)  
RHT definition/mapping macro [78](#)  
RLB definition/mapping macro [78](#)  
RLCBT definition/mapping macro [78](#)  
RLMCB definition/mapping macro [78](#)  
RLPL definition/mapping macro [78](#)  
RLQD definition/mapping macro [78](#)  
RM  
    example trace record [154](#)  
routing  
    DFSDLA30, codes [316](#)  
Routing errors  
    diagnosing [315](#)  
RPL (request parameter list) [323](#)  
RPLI definition/mapping macro [78](#)  
RPST definition/mapping macro [78](#)  
RRE definition/mapping macro [78](#)  
RRS  
    trace entry [611](#)

## S

S  
    IRLM address space  
        formatting and printing [441](#)  
SAP analysis procedure [47](#)  
SAP definition/mapping macro [78](#)  
save area set  
    Fast Path problem analysis  
        example [401](#)  
        finding during DC analysis [323](#)  
save area set, abnormal [49](#)  
SB (sequential buffering)  
    COMPARE option, use in SB [229](#)  
    control block diagram [88](#)  
    DFSSBHD0 utility  
        using with SB IMAGE CAPTURE option [229](#)  
    DL/I trace table entry [227](#)  
    SB IMAGE CAPTURE option  
        using with DFSSBHD0 utility [229](#)  
    SBESNAP option, activating [228](#)  
    SBSNAP option  
        activating [228](#)  
        limiting output [228](#)  
    service aid tool [227](#)

- SBESNAP option, activating [228](#)
- SBHE definition/mapping macro [78](#)
- SBPARMS definition/mapping macro [78](#)
- SBPSS definition/mapping macro [78](#)
- SBPST definition/mapping macro [78](#)
- SBSCD definition/mapping macro [78](#)
- SBSNAP option
  - activating [228](#)
  - limiting output [228](#)
- SBUE definition/mapping macro [78](#)
- SBUF definition/mapping macro [79](#)
- SCA1 definition/mapping macro [79](#)
- SCAR definition/mapping macro [79](#)
- SCD definition/mapping macro [79](#)
- SCD diagram, online [88](#)
- scheduler trace
  - example [628](#)
  - format [626](#)
- SCRAPLOG for QCF
  - JCL for printing records [302](#)
  - sample record [300](#)
- SDB definition/mapping macro [79](#)
- SDB keyword dictionary [63](#)
- SDCB definition/mapping macro [79](#)
- SDEP CI
  - diagnosing CI problem in DEDB
    - format [405](#)
- SDSG definition/mapping macro [79](#)
- SDUMP
  - DRA, output [397](#)
  - IRLM address space
    - description [441](#)
- SDWA definition/mapping macro [79](#)
- search arguments
  - creating [34](#)
  - developing [33](#)
  - release levels [67](#)
- searching problem reporting databases [33](#)
- secondary index database
  - block format [133](#)
  - segment data format [133](#)
  - VSAM LRECL format [133](#)
- security override
  - DBRC [15](#)
  - RECON data sets [15](#)
- segment prefix mapping [129](#)
- selecting keywords [34](#)
- sense-status message [324](#)
- sequential buffering (SB)
  - COMPARE option, use in SB [229](#)
  - control block diagram [88](#)
  - DFSSBHD0 utility
    - using with SB IMAGE CAPTURE option [229](#)
  - DL/I trace table entry [227](#)
  - SB IMAGE CAPTURE option
    - using with DFSSBHD0 utility [229](#)
  - SBESNAP option, activating [228](#)
  - SBSNAP option
    - activating [228](#)
    - limiting output [228](#)
  - service aid tool [227](#)
- service aid
  - DBRC [239](#)
  - DC [277](#)
- service aid (*continued*)
  - Fast Path [401](#)
- service aids
  - CSL [151](#)
  - database [155](#)
  - IRLM [441](#)
  - MSC [445](#)
  - system [469](#)
- service error log records
  - causes [395](#)
  - type 67D0 [395](#)
- SETO call
  - Spool API [391](#)
- SGT definition/mapping macro [79](#)
- shared queues
  - APPC and OTMA message processing [364](#)
- shared queues interface trace
  - description [638](#)
- SHISAM database
  - block format [128](#)
  - LRECL format [128](#)
  - segment format [128](#)
- SHSAM database
  - block format [127](#)
  - delete byte format [127](#)
  - flag byte format [127](#)
  - segment format [127](#)
- shutdown analysis [59](#)
- shutdown processing [58](#)
- SIDB definition/mapping macro [79](#)
- SIDX definition/mapping macro [79](#)
- SMB definition/mapping macro [79](#)
- SMS-managed, non-striped data sets
  - converting to striped [236](#)
- SNAP
  - call facility (DFSERA20)
    - description [572](#)
    - output [572](#)
  - COMPARE statement, SNAP call [157](#)
  - control block output [157](#)
  - dump output [398](#)
  - exceptional condition [158](#)
  - SBESNAP option [228](#)
  - SBSNAP option [228](#)
  - specific call
    - description [158](#)
    - SB COMPARE option [158](#)
    - SBESNAP option [158](#)
    - SBSNAP option [158](#)
- Software Support Facility (SSF)
  - searching [66](#)
- space management module trace IDs [204](#)
- specialabend processing
  - IMS Spool API support [395](#)
- SPQB definition/mapping macro [79](#)
- SQPST definition/mapping macro [79](#)
- SRAN definition/mapping macro [79](#)
- SSF (Software Support Facility)
  - searching [66](#)
- SSIB definition/mapping macro [79](#)
- SSOB definition/mapping macro [80](#)
- SST (subsystem trace)
  - trace record
    - format [596](#)

- SST (subsystem trace) (*continued*)
  - trace record (*continued*)
    - module ID and subfunction table [596](#)
    - variable section layout [599](#)
- SSVP definition/mapping macro [80](#)
- static DB/DC environment [71](#)
- status codes associated with keywords
  - AR [391](#)
- storage management
  - control block relationships created for MAIN pool [88](#)
  - control block relationships for DFSCBT00 pools [88](#)
  - control block relationships for DFSPool pools [88](#)
  - control block relationships for preallocated storage blocks [88](#)
- storage manager trace [629](#)
- structures
  - CQS
    - rebuild [141](#)
- subsystem trace (SST)
  - trace record
    - format [596](#)
    - module ID and subfunction table [596](#)
    - variable section layout [599](#)
- Sx37abend
  - GSAM data sets [234](#)
- synchronous callout
  - 6701 log record formats [506](#)
  - log record format [506](#)
  - OTMA log record format [506](#)
  - X'6701' log record formats [506](#)
- syntax diagram
  - how to read [xi](#)
- SYS (system) service aids
  - printing log records and trace table entries
    - File Select and Formatting Print utility (DFSERA10) [511](#)
- SYS (systems)
  - service aid
    - common trace table interface [574](#)
    - dispatcher trace [587](#)
    - dumps, formatting online [566](#)
    - external subsystem trace [596](#)
    - ITASK ECB posting [595](#)
    - log record format (type X'29') [492](#)
    - log record format (type X'67') [503](#)
    - log record table [469](#)
    - scheduler trace [626](#)
    - type-2 trace table interface [665](#)
  - service aids
    - ODF (Offline Dump Formatter) [512](#)
    - Snap call facility [572](#)
- SYS1.DUMPXX data set
  - IRLM address space
    - description [441](#)
- SYS1.LOGREC record
  - IRLM diagnosis [441](#)
- SYS1.XX data set
  - IRLM address space
    - formatting and printing [441](#)
- SYSMDUMP statement
  - dump preservation [4](#)
- System
  - dumps
    - formatting [563](#)

- system analysis [469](#)
- system dump
  - contents [520](#)
- system post code list [595](#)
- system wait [58](#)
- SYSUDUMP statement
  - dump preservation [4](#)

## T

- TAB definition/mapping macro [80](#)
- TCT definition/mapping macro [80](#)
- technical support
  - searching databases [31](#)
- terminal communication task trace
  - entry point [277](#)
  - save area [277](#)
  - trace ID [277](#)
  - trace output [279](#)
  - trace record example [278](#)
  - trace record format [278](#)
- terminal problem
  - diagnosing using DC trace [291](#)
- trace
  - activation [416](#)
  - DBRC, output [245](#)
  - deactivation [416](#)
  - DFSMSCEO [312](#)
  - DFSMSCEO, starting [313](#)
  - DFSMSCEO, status [313](#)
  - DFSMSCEO, stopping [313](#)
  - DL/I [166](#)
  - Fast Path [415](#)
  - LU manager [347](#)
  - LU manager, format [340](#)
  - LU manager, start [340](#)
  - Online Recovery Manager, starting [213](#)
  - ORTT [212](#)
  - OTMA [365](#)
  - OTMA, user exits [375](#)
  - overview [416](#)
  - records, format [611](#)
  - RRS [625](#)
  - RRST [611](#), [620](#)
  - turning off [416](#)
  - X'0C', example [167](#)
  - X'D5', example [190](#)
- trace entries
  - BPE [12](#)
  - buffer handler
    - DL/I call [204](#)
  - CQS [142](#)
  - DSPRLX10 [254](#)
  - DSPRLX1M [254](#)
  - DSPRSYNC [253](#)
  - DSPURI00 [255](#), [273](#)
  - DSPURI80 [253](#)
  - parallel RECON access [253](#)
  - X'31', X'32', X'34', X'B1', and X'B2' [167](#)
  - X'63' [169](#)
  - X'64' [169](#)
  - X'65' [170](#)
  - X'6A' [170](#)
  - X'CB' [182](#)



trace entries (*continued*)

X'D0' [187](#)

X'D1' [188](#)

trace entry

Fast Path [641](#)

X'60' [168](#)

X'61' [168](#)

X'62' [169](#)

X'69' [170](#)

X'6B' [171](#)

X'6C' [172](#)

X'6F' [172](#)

X'80', X'81', X'82' [173](#)

X'AA' [173](#)

X'AB' [174](#)

X'AC' [175](#)

X'AD' [176](#)

X'C4' [177](#), [181](#)

X'C6' [177](#)

X'C7'

IRLM [178](#)

X'C7', example [178](#)

X'C8' [179](#)

X'C9' [179](#)

X'CA' [180](#)

X'CA' trace entry, subtype X'08' [181](#)

X'CF' [185](#)

X'DB' through X'FA [200](#)

trace header record

DBRC [245](#)

trace IDs

X'A0A6' DFSRGSF0 Entry record [625](#)

X'A0A7' DFSRGSF0 Exit record [625](#)

X'A0AA' TOKEN Tracing record [625](#)

Trace IDs

X'5A00' [623](#)

trace output

unformatted example [264](#)

trace record

RM [154](#)

trace records

BPE-based DBRC

examples [268](#)

CSL [151](#)

DC (Data Communication), example [296](#)

format [577](#), [665](#)

formats [635](#)

LU 6.2 module-to-code reference [348](#)

type-1 [577](#)

type-2 [665](#)

verb-to-code reference [350](#)

trace table

IMS shutdown [666](#)

trace tables

DASD log [577](#)

Dispatcher [577](#)

DL/I and lock [577](#)

External [577](#)

Fast Path [577](#)

function codes [577](#)

Latch [577](#)

locating in dump [575](#)

Queue Manager [577](#)

Resource Recovery Services [577](#)

trace tables (*continued*)

Scheduler [577](#)

Storage Manager [577](#)

Subsystem [577](#)

type-1 [575](#), [577](#)

traces

BPE

IMS Connect tracing [429](#)

CIBSTRAC [326](#)

CIBTRACE [326](#)

common trace table interface [574](#)

DBRC [244](#)

DBRC external [263](#)

DC [279](#)

dispatcher [587](#)

DL/I [165](#)

DL/I call image capture [159](#)

external subsystem [596](#)

Fast Path [639](#)

ICONER record [438](#)

ICONSN record [438](#)

IMS Connect

configuring BPE for external trace [431](#)

displaying status of external trace [433](#)

formatting external trace data [434](#)

overview [429](#)

starting a BPE external trace [432](#)

stopping a BPE external trace [433](#)

tracing to the HWSRCDR data set [434](#)

IMS transaction [317](#)

job control block [155](#)

log record [316](#)

LXB [459](#)

MSC communication task [445](#)

Online Recovery Manager (ORTT) [213](#)

ORTT (Online Recovery Manager) [213](#)

OTMA

parallel RESUME TPIPE request processing [369](#)

X'5A20' [369](#)

program isolation [220](#)

queue manager [634](#)

retrieve [208](#)

scheduler [626](#)

shared queues interface [638](#)

type-1 [574](#)

type-2 trace table interface [665](#)

trademarks [669](#), [673](#)

Transaction Manager

dumps

formatting [563](#)

Transaction Manager control block diagram [88](#)

transaction retry, Fast Path

description [404](#)

processing flow [404](#)

system programmer response [404](#)

tutorial

IMS Dump Formatter [563](#)

type X'68' [395](#)

type-2 trace table interface [665](#)

type-of-failure keyword [35](#)

## U

UEHB definition/mapping macro [80](#)

- UPAD codes [199](#)
- USR(FAD)
  - unformatted trace output example [264](#)
- utility
  - Log Merge [512](#)
- UXDT definition/mapping macro [80](#)
- UXRB control block [80](#)
- UXRB definition/mapping macro [80](#)

## V

- variable-length segment
  - HDAM format [135](#)
  - HIDAM format [135](#)
  - HISAM format [135](#)
  - PHDAM format [135](#)
  - PHIDAM format [135](#)
- Virtual Storage Access Method (VSAM)
  - FMTIMS statement example [520](#)
  - LRECL format [132](#)
  - PSINDEX [133](#)
  - secondary index [133](#)
- Virtual Telecommunications Access Method (VTAM)
  - request parameter list (RPL) [323](#)
  - RPL (request parameter list) [323](#)
  - terminal problem
    - starting DC trace [279](#)
    - stopping DC trace [280](#)
- VSAM (Virtual Storage Access Method)
  - FMTIMS statement example [520](#)
  - LRECL format [132](#)
  - PSINDEX [133](#)
  - secondary index [133](#)
- VSI definition/mapping macro [80](#)
- VTAM (Virtual Telecommunications Access Method)
  - request parameter list (RPL) [323](#)
  - RPL (request parameter list) [323](#)
  - terminal problem
    - starting DC trace [279](#)
    - stopping DC trace [280](#)
- VTCB
  - load module diagram [88](#)
  - posting of overlays in DFSVTP00 [294](#)

## W

- wait for input [56](#)
- wait, diagnosing a control region [9](#)
- wait, diagnosing an IMS dependent region [11](#)
- WAIT/LOOP procedure [44](#)
- WHB definition/mapping macro [80](#)
- work area
  - CBT pool [412](#)
  - Fast Path
    - locating in an IMS [408](#)
    - locating using load list [410](#)

## X

- X'29'
  - log record layout
    - X'2311' [499](#)
    - X'2900' [492](#)

- X'29' (*continued*)
  - log record layout (*continued*)
    - X'2910' [492](#)
    - X'2911' [493](#)
    - X'2920' [494](#)
    - X'2930' [495](#)
    - X'2940' [499](#)
    - X'2950' [500](#)
    - X'2970' [501](#)
    - X'2971' [502](#)
    - X'2990' [503](#)
  - X'58' variable section [604](#)
  - X'5A0B' trace entry [383](#)
  - X'5A20' trace entry [369](#)
  - X'61'
    - trace entry [168](#)
  - X'62'
    - trace entry [169](#)
  - X'63'
    - trace entry [169](#)
  - X'64'
    - trace entry [169](#)
  - X'65'
    - trace entry [170](#)
  - X'6701' log records
    - log sequence field [505](#)
    - map [293](#)
    - OTMA
      - shared queues log records [509](#)
      - synchronous callout log record format [506](#)
      - TIB3 record [509](#)
      - OTMA record formats [505](#)
      - subrecord and data area formats [505](#)
      - synchronous callout log record format [506](#)
  - X'68' log record [395](#)
  - X'69'
    - trace entry [170](#)
  - X'6B'
    - trace entry [171](#)
  - X'6C'
    - trace entry [172](#)
  - X'6F'
    - trace entry [172](#)
  - X'AA'
    - trace entry [173](#)
  - X'AB'
    - trace entry [174](#)
  - X'C4'
    - trace entry [177](#), [181](#)
  - X'C6'
    - trace entry [177](#)
  - X'C8'
    - trace entry [179](#)
  - X'C9'
    - trace entry [179](#)
  - X'CA'
    - trace entry [180](#)
  - X'CA' trace entry, subtype X'08'
    - trace entry [181](#)
  - X'D9' trace entry [191](#)
  - XCRB definition/mapping macro [80](#)
  - XMCA definition/mapping macro [80](#)
  - XMCI definition/mapping macro [80](#)
  - XRF environment



XRF environment (*continued*)  
starting DC trace [279](#)  
stopping DC trace [280](#)

## Z

z/OS component trace  
example [442](#)  
z/OS logger errors  
Repository Server [25](#)  
z/OS storage map diagram [88](#)  
ZIB definition/mapping macro [80](#)







Product Number: 5635-A06  
5655-DS5  
5655-TM4