IBM IMS High Performance Unload for z/OS 1.2

User's Guide

SC27-0936-12

Note:

Before using this information and the product it supports, read the information in “Notices” on page 517.

13th Edition (September 2024)

This edition applies to Version 1.2 of IBM IMS High Performance Unload for z/OS (program number 5655-E06) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC27-0936-11.

Contents

About this information

IBM IMS High Performance Unload for z/OS® (also referred to as IMS High Performance Unload or IMS HP Unload) provides high speed unloading of IMS databases and improves performance of IMS data retrieval application programs by using the Unload application programming interface (API).

These topics are designed to help system programmers, application programmers, system analysts, database administrators, and computer operators to do these tasks:

- Understand the functions of IMS High Performance Unload
- Run and use IMS High Performance Unload
- Use DD statements to control how you use IMS High Performance Unload
- Interpret IMS High Performance Unload reports
- Diagnose and recover from IMS High Performance Unload problems

Attention: These topics contain specific settings and target values for tuning databases. See “IMPORTANT NOTICE” on page 321 before you use these topics.

Always refer to the IMS Tools Product Documentation web page for complete product documentation resources:

https://www.ibm.com/support/pages/node/712955

The IMS Tools Product Documentation web page includes:

- Links to IBM Documentation for the user guides ("HTML")
- PDF versions of the user guides ("PDF")
- Program Directories for IMS Tools products
- Technical notes from IBM Software Support, referred to as "Tech notes"
- White papers that describe product business scenarios and solutions

Part 1. IMS High Performance Unload overview

IBM IMS High Performance Unload for z/OS (also referred to as IMS High Performance Unload or IMS HP Unload) provides high speed database unloading and capabilities that are not included in the basic utilities that are provided by IMS.

Topics:

Chapter 1. Introduction to IMS High Performance Unload

IMS High Performance Unload provides high speed unloading of IMS databases and improves the performance of IMS data retrieval application programs by using the unload application programming interface (API).

Topics:

What's new in IMS High Performance Unload

This topic summarizes the technical changes for this edition.

New and changed information is indicated by a vertical bar (|) to the left of a change. Editorial changes that have no technical significance are not noted.

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.

SC27-0936-12 (September 2024)

Description	Related APARs
STAGING control statement, which is for obtaining the pending database from the IMS catalog staging data set, is supported for the SYSIN DD statement of the FABHURG1 unload utility. The following topics are updated: • "FABHURG1 SYSIN input data set" on page 41 • "IMS HD Reorganization Unload JCL for running FABHURG1" on page 48	PH62672

SC27-0936-11 (November 2023)

Description	Related APARs
Fix for migration unload issue of a non-HALDB with a virtual logical child segment to a HALDB. For more information, see the following topics: • "Restrictions for IMS High Performance Unload" on page 22 • "Migration unload" on page 107 • "Fallback unload" on page 112 • Modified message "FABH0400I" on page 439	PH57574

SC27-0936-10 (October 2022)

Description	Related APARs
Modified messages FABH0646E and FABH0856E.	N/A

SC27-0936-09 (August 2020)

Description	Related APARs
Support for IMS OSAM database enhancement.	PH22529
Support encrypted VSAM (ESDS) DBDS even when Media Manager is used.	PH00816
Support for the Language Environment® preinitialization service (CEEPIPI).	PI91820
A new control statement, ZIIPMODE, which specifies whether HSSR engine offloads eligible VSAM ESDS I/O workloads to zIIP processors. For more information, see "ZIIPMODE control statement" on page 178.	PI89050
Enables to use database definitions in the IMS catalog instead of DBD libraries when IMS-managed ACBs is enabled. For more information, see "Considerations for IMS-managed ACBs environment" on page 28.	PI83671
A new control statement, CABBASE, which specifies the basic size of I/O buffers that the CAB buffer handler allocates. For more information, see "CABBASE control statement" on page 162.	PI77214
Support for IMS 15.1.	PI73493
Fix for DBRC database authorization issues of HSSR Engine.	PI60116
A new control statement, COMPAUTH, which specifies to call the segment compression exit in supervisor state. For more information, see "COMPAUTH control statement" on page 165.	PI51721
Other APAR related doc changes.	PI59811

SC27-0936-08 (March 2015)

Description	Related APARs
Support for IMS 14.1.	PI27636
Support unloading of databases when database versioning is enabled in the IMS system.	PI10157

Description	Related APARs
You can use the new control statement, SKIPAUTH, for the HSSR Engine. Use this control statement to bypass IMS DBRC database authorization for HALDB partitions. For more information, see "SKIPAUTH control statement" on page 175.	PM97558
Support IMS catalogs. Information is added to "Considerations for unloading an IMS catalog" on page 27.	PM65262
Other APAR related doc changes.	PM84374, PM88606, PI19021

SC27-0936-07

Description	Related APARs
Support for IMS 12.1	PM22119
*CP format (the communication industry partitioned format) has been supported. This format is useful if the database is a HALDB with partitioned secondary index (PSINDEX) databases and if *CS format cannot be used due to the presence of the PSINDEX. For details, see "*CP format" on page 39.	PK90234
Support for IMS 11.1.	PK74302
Other APAR related doc changes.	PK61726, PK77437, PK90234, PM12285, PM48532, PM55775, PM58705

SC27-0936-06

Description	Related APARs
Functional and performance improvements: • In the unload utilities (FABHURG1 and FABHFSU), the number of buffers that are automatically set for unload data set are increased. For details, see "FABHURG1 JCL requirements" on page 39 and "FABHFSU JCL requirements" on page 54. • If the STEPLIB is APF-authorized, HSSR Engine uses the Media Manager to read VSAM ESDS database data sets. The use of Media Manager saves the use of CPU time.	N/A
Improvements in the usability and readability of the product information are made in Chapter 7, "Application programming interface for using HSSR Engine," on page 79.	N/A
IMS High Performance Unload for z/OS 1.2 does not support IMS that are lower than 7.1. Therefore, the names of the IMS libraries that are referred to by the catalog procedure, provided by the product, are changed to those names of IMS 8.1 and later.	N/A
Other APAR related doc changes.	PK38688, PK47931, PK49836

SC27-0936-05

Description	Related APARs
Support for IMS 10.1: • To use the parallel RECONs, you must specify the IMSPLEX= and the DBRCGRP= parameters in the JCL EXEC statement. For more information, see “Preparing the basic JCL” on page 29. • The FABHURG1 and FABHFSU unload utilities support the large format data set for unload data sets.	PK33118
The Get Next call with two SSAs to retrieve the second-level-dependent segment has been supported in HP Unload API. For more information, see “DL/I calls supported by each API set” on page 85.	PK32595
The SKIPVLC control statement has been added to HSSR Engine. For more information, see “SKIPVLC control statement” on page 176.	PK28097
The CHECKREC control statement has been added to the FABHURG1 unload utility. For more information, “CHECKREC control statement” on page 41.	PK11534
Other APAR related doc changes.	PK17804, PK24577, PK24974

SC27-0936-04

Description	Related APARs
JCL compatibility with IMS HD Reorganization Unload is supported. For more information, see “IMS HD Reorganization Unload JCL for running FABHURG1” on page 48.	PK11209
Function to retrieve HDAM or HIDAM root segments in a secondary index sequence has been supported. For more information, see “Considerations for using a secondary index” on page 27.	PK07882
Keywords, URG1BUFNO and FSUBUFNO, have been added to the default option table (FABHOPT). For more information, see “Replacing the HSSR option table (FABHOPT)” on page 262.	PK06057
Get Next call with a qualified SSA on root segments is supported in HP Unload API. For more information, see “DL/I calls supported by each API set” on page 85.	PK04911
IBM-provided exit routine FABHKEYX is available for the FABHURG1 unload utility. For more information, see “Migration unload: Exit routine FABHKEYX for distributing unload records” on page 109.	PK01994
Other APAR related doc changes.	PQ97692, PQ99842, PK07881, PK08900

SC27-0936-03

Description	Related APARs
Support for IMSDALIB DD statement. IMS Parallel Reorganization for z/OS 3.1 requires IMS High Performance Unload that has this APAR applied.	PQ93668
Function to set buffer numbers for VSAM KSDS automatically.	PQ85786

Description	Related APARs
Support for HALDB partitions that are defined as HALDB Online Reorganization (OLR) capable. See the considerations described in “Considerations for HALDB Online Reorganization capable partitions” on page 26.	PQ83387
Support for DFSHALDB DD statement, which is used to select a single partition to be processed in the API function (DLI or DBB region) of IMS High Performance Unload. For details, see “Consideration for HALDB single partition processing” on page 95.	PQ81675
Enables the product to run under IMS 9.1.	PQ80191
Other APAR related doc changes.	PQ69413, PQ70680, PQ70768, PQ72433, PQ76387, PQ77099, PQ79194

SC27-0936-01 and SC27-0936-02

Description	Related APARs
The following functions are added to the HSSR call: • The application programming interface for IMS High Performance Unload is extended to support GN and GNP calls with an unqualified SSA for a dependent segment type of an HD database. • The application programming interface for IMS High Performance Unload is extended to support EXEC DLI commands.	PQ67004, PQ67296, PQ69199
Complemented explanation of HALDB formats. The description is now as follows: A HALDB can be unloaded in *HD format by FABHURG1, or in UL format by FABHFSU. Whichever format the HALDB is in, both the IPR Reload utility of IMS Parallel Reorganization for z/OS 2.1 and IMS High Performance Load for OS/390® 1.1 regard it internally, and therefore refer to it, as having PHD format.	PQ66914
Support for the Return Code Edit exit. This APAR provides a new exit for the HSSR Engine of IMS High Performance Unload. This exit, called the Return Code Edit exit (FABHRCEX), can be used to modify the return codes issued by HSSR application programs.	PQ62843
Added a description of the conditions in which the header record flag introduced by APAR PQ22654 for HSSR 2.3 causes an incompatibility. If an unloaded data set has been created by specifying the DECN control statement for a database that contains a compressed segment, that data set is not compatible with an unloaded data set created by the IMS HD Reorganization Unload utility.	PQ59762
Explanations for three messages: • The explanation for message FABH0431W is expanded to cover all the conditions in which that message is issued. • Explanations of messages FABH0092W and FABH0358W, which had been missing, are added.	PQ59358
Enables IMS High Performance Unload to run under IMS8.1.	PQ57581

Description	Related APARs
IMS High Performance Unload is modified so that Segment Edit/Compression routines are loaded and deleted by the IMS IMODULE service. The APAR also adds message FABH0853E.	PQ57815
New control statement, PARTEXTR, for the HSSREXTR DD for the FABHEXTR exit routine of the FABHURG1 unload utility. The PARTEXTR control statement specifies how many database records are to be extracted from each HALDB partition.	PQ55674
IMS High Performance Unload is modified so that using an uninitialized OSAM database as input causes an error. IMS High Performance Unload is modified so that, before getting access to a database, it checks the status of each data set in that database. If it finds an uninitialized data set, it causes an abend U4013. For High Availability Large Database (HALDB), the first time IMS High Performance Unload gets access to a partition, it checks the status of the data sets in that partition. If it finds an uninitialized data set, it causes an abend U4013.	PQ49420
Added a description that the GOT control statement becomes valid if DBRC is inactive in the FABHFSU unload utility.	PQ48541

SC27-0936-00

This documentation covers IMS High Performance Unload, which is a follow-on product to the IMS System Utilities/Data Base Tools (DBT) Version 2 High Speed Sequential Retrieval (DBT V2 HSSR).

Description	Related APARs
The compatibility with DBT V2 HSSR and other prior products is summarized in the following topics: • Chapter 30, “Compatibility with DBT V2 HSSR,” on page 357 • Chapter 31, “Compatibility with DBT V1 HSSR,” on page 365 • Chapter 32, “Compatibility with PO HSSR,” on page 367 • Chapter 33, “Compatibility with FSU II,” on page 373	N/A
One of the major enhancements to IMS High Performance Unload is the addition of support for IMS 7.1, specially High Availability Large Database (HALDB), introduced in IMS 7.1. See Chapter 8, “Methods for processing High Availability Large Databases,” on page 97.	N/A

What does IMS High Performance Unload do?

IMS High Performance Unload includes two unload utilities, FABHURG1 and FABHFSU, that provide high speed unloading capability. It also includes an application programming interface (API) for DL/I application programs that use GN calls.

As processing volumes increase, more work needs to be done in a shorter time due to shrinking batch windows. IMS High Performance Unload saves you time and money by reducing the CPU and elapsed time that is required to unload IMS databases and to run IMS data retrieval application programs. Powerful functions such as the ability to continue processing after a pointer error, a user exit facility, and various unloaded record formats are provided, all with the goal of improving availability and throughput.

IMS High Performance Unload is designed for use with IMS Database Reorganization Expert for z/OS and IMS Online Reorganization Facility, as well as with other high performance IMS Tools products to provide the most efficient and powerful end-to-end solution for IMS database reorganization.

IMS High Performance Unload replaces the functionality of the IMS HD Reorganization Unload utility (DFSURGU0).

IMS High Performance Unload is serviced by a high-performance database retrieval engine called *High Speed Sequential Retrieval (HSSR) Engine*. IMS High Performance Unload also has an application programming interface (API) that is compatible with the HSSR call API provided by High Speed Sequential Retrieval of IMS System Utilities, Data Base Tools (PID: 5685-093).

Subtopics:

Two unload utilities

IMS High Performance Unload provides two unload utilities; FABHURG1 and FABHFSU. Both utilities offer the following functions:

Fast segment retrieval from databases
The segment retrieval engine of IMS High Performance Unload, HSSR Engine, provides the facility to retrieve segments much faster than DL/I. See "IMS High Performance Unload system structure" on page 11.

Unloading a database without decompressing compressed segments
Both unload utilities can unload a database that uses the Segment Edit/Compression Exit without decompressing the segments. This method can decrease the CPU time and elapsed time. See the following topics to activate the decompress option:

- For FABHURG1, see "DEC control statement" on page 42.
- For FABHFSU, see "DEC control statement" on page 59.

Unloading a selected partition or a selected sequence of partitions of a HALDB
Both unload utilities support unloading of a HALDB. For details, see Chapter 8, "Methods for processing High Availability Large Databases," on page 97.

Ability to monitor the need for database reorganization by examining statistical reports
The unload utilities can generate statistical reports that can be used for database tuning purposes. For details, see Chapter 26, "Obtaining statistics for database tuning," on page 307.

Ability to monitor the quality of HDAM or PHDAM randomizing by examining statistical reports
The unload utilities can generate statistical reports that can be used to measure the quality of HDAM or PHDAM randomizing. For details, see Chapter 26, "Obtaining statistics for database tuning," on page 307.

Ability to continue processing after sequence errors
HSSR Engine optionally performs sequence-key checks for twin chains. You can select the behavior for a sequence error from the following options:

- HSSR Engine gets to abend.
- HSSR Engine returns a GG status code and unloads neither the segment in error nor its dependents.
- HSSR Engine returns a GX status code and unloads the segment in error.

Reading a corrupted database
Options in IMS High Performance Unload increases the control you have when you unload a database that contains an incorrect pointer or incorrect HISAM records:

- Incorrect HD pointers can be bypassed.
- The processing of the remainder of the incorrect HISAM record can be skipped.
- The root segments of HIDAM or PHIDAM database can be accessed by use of the primary index, instead of traversing the root twin chain and the RAP chain.

For details, see Chapter 9, "Utility options for unloading corrupted databases," on page 115.

Application programing interface

The application programming interface (API) of IMS High Performance Unload enables an IMS DL/I batch application program, which reads a database sequentially, to use HSSR Engine without being recompiled or relink-edited. The API enables DL/I application programs to use HSSR Engine to read a database by using GN calls. Through this API, the application program can retrieve a large number of segments more efficiently than the native IMS DL/I and the elapsed time and CPU time can be reduced.

Database organizations supported

IMS High Performance Unload supports the following database organizations:

- HDAM (Hierarchical Direct Access Method)
- HIDAM (Hierarchical Indexed Direct Access Method)
- PHDAM (Partitioned HDAM)
- PHIDAM (Partitioned HIDAM)
- HISAM (Hierarchical Indexed Sequential Access Method)
- SHISAM (Simple HISAM)

You can unload a partition or a sequence of partitions of a PHDAM or PHIDAM database.

Other database organizations are not supported by IMS High Performance Unload.

Notes:

- Secondary index databases are supported, but are processed as independent databases. Processing of a partitioned secondary index is not supported.
- IMS/ESA® Partition Support Product for MVS/ESA (5697-A06) and IMS/ESA Partition Support Product for MVS/ESA Version 2 (5697-D85) are not supported.

Compatibility with prior products

IMS High Performance Unload is compatible with the following prior products:

DBT V2 HSSR
IMS System Utilities/Data Base Tools Version 2, High Speed Sequential Retrieval (5685-093)

DBT V1 HSSR
IMS System Utilities/Data Base Tools Version 1, High Speed Sequential Retrieval (5668-856)

PO HSSR
Program Offering High Speed Sequential Retrieval Version 2 (IFP 5787-LAC)

FSU II
IMS/VS Fast Scan Utility II Version 2 (FDP 5798-DFN)

For details about compatibility with these products, see the following topics:

- Chapter 30, “Compatibility with DBT V2 HSSR,” on page 357
- Chapter 31, “Compatibility with DBT V1 HSSR,” on page 365
- Chapter 32, “Compatibility with PO HSSR,” on page 367
- Chapter 33, “Compatibility with FSU II,” on page 373

IMS High Performance Unload features and benefits

IMS High Performance Unload provides high speed database unloading capabilities, as well as additional capabilities that are not included in the IMS base utilities.

Fast segment retrieval from databases

The segment retrieval engine of IMS High Performance Unload, HSSR Engine, provides the function to retrieve segments much faster than DL/I. The unload utilities FABHURG1 and FABHFSU benefit.

Also, batch DL/I application programs that use GN calls to read database can use the facility. Programs that read large portions of a database sequentially may get significant reductions in elapsed time and CPU use. You do not have to recompile or relink-edit the application program to use HSSR Engine.

User exit facility to allow additional processing for retrieved segments

Each of the unload utilities provides the user exit facility, which provides more control on unload process than the IMS HD Reorganization Unload utility.

Ability to continue processing after sequence errors

IMS High Performance Unload optionally does sequence-key checks for twin chains. You can select the behavior for a sequence error from the following methods:

- IMS High Performance Unload gets to abend (KEYCHECK ABEND)
- IMS High Performance Unload returns a GG status code and does not unload the segment containing the error, or its dependents (KEYCHECK GG and SKERROR *n*)
- IMS High Performance Unload returns a GX status code and unloads the segment in error (KEYCHECK GX)

If the DIAGG control statement is specified, IMS High Performance Unload provides diagnostic information of each sequence error.

Reading a corrupted database

IMS High Performance Unload options give you greater control when you are unloading a database that has an incorrect pointer:

- SKERROR option can be used to bypass pointer errors.
- BYINDEX option can be used to force the roots of HIDAM or PHIDAM database to be accessed from the primary index.

If the DIAGG control statement is specified, IMS High Performance Unload provides diagnostic information about each pointer error.

IMS High Performance Unload system structure

Any application program that runs the IMS High Performance Unload's runtime environment and is serviced by HSSR Engine, is called an *HSSR application program*. Unload utilities FABHURG1 and FABHFSU are HSSR application programs. You can also write your own HSSR application program.

The following figure provides an overview of the structure of IMS High Performance Unload and its data flow.

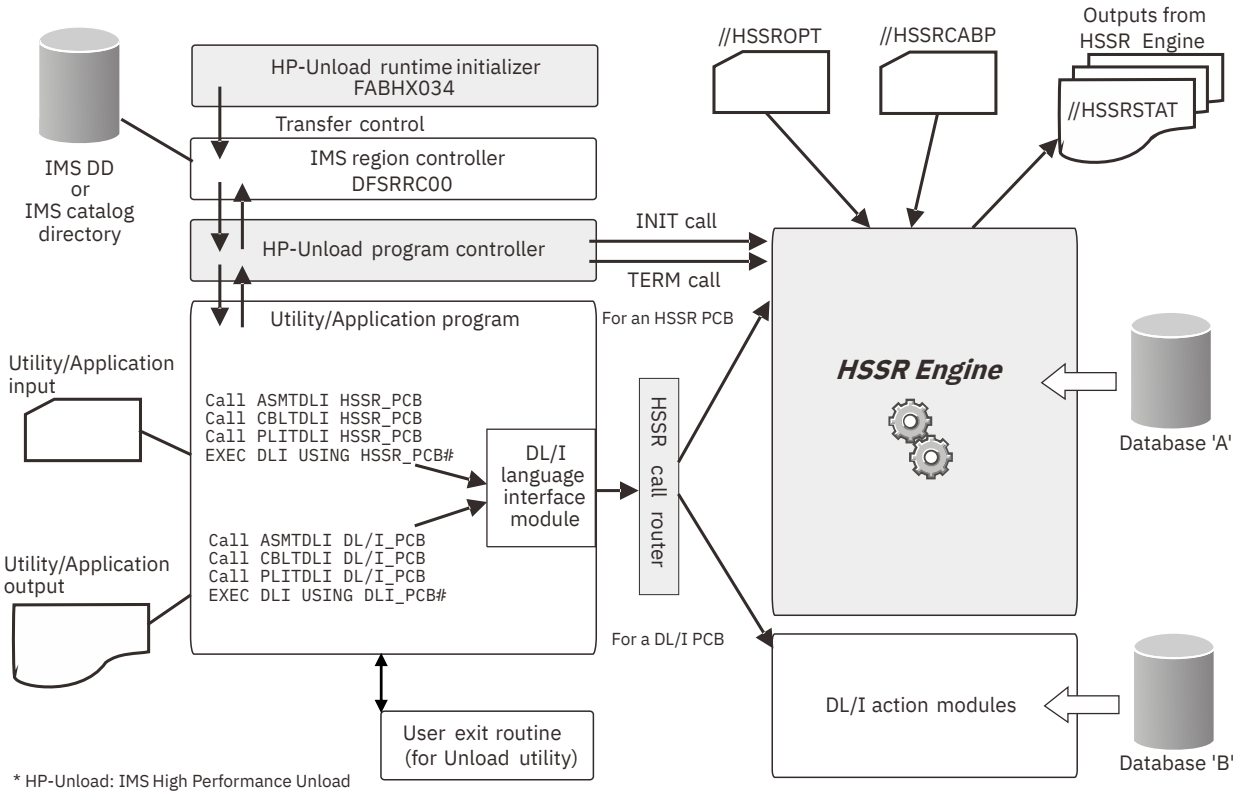

Figure 1. System structure and data flow

IMS High Performance Unload runs in an IMS batch environment. IMS High Performance Unload runtime environment initializer (FABHX034) is invoked first. The initializer gives control to the IMS region controller DFSRRC00, which passes control to the IMS program controller. IMS High Performance Unload program controller is then invoked by the IMS program controller as an IMS batch application program.

The IMS High Performance Unload program controller calls HSSR Engine. HSSR Engine then reads the HSSROPT data set to initialize the call analyzer, call handler, and trace and diagnosis facilities, and reads the HSSRCABP data set to initialize the buffer handler. Call analyzer initializes PCBs that have been specified to be serviced by HSSR Engine. Such PCBs are called *HSSR PCBs* (for details about HSSR PCB, see Chapter 7, “Application programming interface for using HSSR Engine,” on page 79). Call analyzer also initializes control blocks relating to the HSSR PCBs. Buffer handler allocates a buffer pool for each data set group for each HSSR PCB.

The IMS High Performance Unload program controller loads the application program. Each DL/I call or EXEC DLI command to an HSSR PCB is processed by HSSR Engine. This call or command is called an *HSSR call*. If the PCB is not an HSSR PCB, HSSR call router passes control to the DL/I program request handler.

After the application program runs, it returns control to the IMS High Performance Unload program controller. HSSR Engine terminates the processing and writes statistical reports to the HSSRSTAT data set. Then the program controller returns control to IMS.

For details about each component of IMS High Performance Unload, see the following topics:

- Chapter 5, “FABHURG1 unload utility,” on page 35
- Chapter 6, “FABHFSU unload utility,” on page 51
- Chapter 7, “Application programming interface for using HSSR Engine,” on page 79

To run these unload utilities or to use HSSR engine with your application program, you must prepare basic JCL. For information, see Chapter 4, “Basic job control language,” on page 29.

Roadmap to IMS High Performance Unload information

The *IMS High Performance Unload User's Guide* provides complete information for using IMS High Performance Unload.

This information is designed for system programmers, application programmers, system analysts, database administrators, and computer operators who have a working knowledge of IMS and need to learn how to set up and use IMS High Performance Unload. Before reading this information, you should understand basic IMS concepts, the IMS environment, and your installation's IMS system.

If you are not experienced with IMS System Utilities/Data Base Tools, High Speed Sequential Retrieval (DBT HSSR) or Program Offering High Speed Sequential Retrieval (PO HSSR), read Chapter 1, "Introduction to IMS High Performance Unload," on page 3 for a technical overview of this product. This topic covers the following general information:

- Introduction to the functions of IMS High Performance Unload
- Typical benefits you can get by using IMS High Performance Unload
- System structure of IMS High Performance Unload

See Chapter 3, "Introduction to the unload utilities," on page 21 for information about the two unload utilities (FABHURG1 and FABHFSU). This information will help you determine which utility to use.

Then proceed to Chapter 4, "Basic job control language," on page 29 to learn how to write JCL for IMS High Performance Unload. This topic covers the following information:

- JCL statements for IMS High Performance Unload
- IBM-supplied cataloged procedures

IMS High Performance Unload provides an application programming interface. If you want to write your own application program and use HSSR Engine, see Chapter 7, "Application programming interface for using HSSR Engine," on page 79.

In addition, the following topics help you to complete your tasks with IMS High Performance Unload:

- To process a HALDB, see Chapter 8, "Methods for processing High Availability Large Databases," on page 97.
- To unload a database that has a pointer error, see Chapter 9, "Utility options for unloading corrupted databases," on page 115.
- To change the behavior of IMS High Performance Unload by coding control statements, see Chapter 11, "Options for HSSR Engine," on page 155.
- To interpret the reports generated by HSSR Engine, see Chapter 12, "Reports and output from HSSR Engine," on page 181.
- To tune or customize IMS High Performance Unload jobs, see the following topics:
  - Chapter 13, "Overview of the buffer handlers," on page 207
  - Chapter 18, "System programming interfaces," on page 243
  - Chapter 19, "Site default options," on page 261
- To tune your databases by using the Database Tuning Statistics function, see Chapter 26, "Obtaining statistics for database tuning," on page 307.
- If you are migrating from DBT HSSR or PO HSSR, see the following topics:
  - Chapter 30, "Compatibility with DBT V2 HSSR," on page 357
  - Chapter 31, "Compatibility with DBT V1 HSSR," on page 365
  - Chapter 32, "Compatibility with PO HSSR," on page 367
  - Chapter 33, "Compatibility with FSU II," on page 373

IMS High Performance Unload terminology

IMS High Performance Unload includes several unique terms that you should understand before you begin to use IMS High Performance Unload.

Short names used in this information

In these topics, all supported versions of IMS are referred to as IMS, except where distinctions among them need to be made.

In these topics, the following short names for product names are used:

Short name	Product name
Db2®	Db2 UDB for z/OS (currently supported versions)
IMS	IMS Database Manager (currently supported versions)
IMS Database Reorganization Expert	IBM IMS Database Reorganization Expert for z/OS 4.1 (5655-S35) **Note:** IMS Database Reorganization Expert is the successor product of IMS Parallel Reorganization.
IMS High Performance Load	IBM IMS High Performance Load for z/OS 2.1 (5655-M26)
IMS High Performance Unload	IBM IMS High Performance Unload for z/OS 1.2 (this product)

For withdrawn products, the following short names are used:

Short name	Product name
DBT HSSR	The generic name for the following products: • IMS System Utilities/Data Base Tools Version 2, High Speed Sequential Retrieval (5685-093) • IMS System Utilities/Data Base Tools Version 1, High Speed Sequential Retrieval (5668-856)
IPR	IBM IMS Parallel Reorganization for z/OS 3.2 (5655-M28)
PO HSSR	Program Offering High Speed Sequential Retrieval Version 2 (IFP 5787-LAC)
FSU II	IMS/VS Fast Scan Utility II Version 2 (FDP 5798-DFN)

Service updates and support information

Service updates and support information for this product, including software fix packs, PTFs, frequently asked questions (FAQs), technical notes, troubleshooting information, and downloads, are available from the web.

To find service updates and support information, see the following website:

IBM Support: IMS High Performance Unload for z/OS

Product documentation and updates

IMS Tools information is available at multiple places on the web. You can receive updates to IMS Tools information automatically by registering with the IBM My Notifications service.

Information on the web

Always refer to the IMS Tools Product Documentation web page for complete product documentation resources:

https://www.ibm.com/support/pages/node/712955

The IMS Tools Product Documentation web page includes:

- Links to IBM Documentation for the user guides ("HTML")
- PDF versions of the user guides ("PDF")
- Program Directories for IMS Tools products
- Technical notes from IBM Software Support, referred to as "Tech notes"
- White papers that describe product business scenarios and solutions

IBM Redbooks® publications that cover IMS Tools are available from the following web page:

http://www.redbooks.ibm.com

The IBM Information Management System website shows how IT organizations can maximize their investment in IMS databases while staying ahead of today's top data management challenges:

https://www.ibm.com/software/data/ims

Receiving documentation updates automatically

To automatically receive emails that notify you when new technote documents are released, when existing product documentation is updated, and when new product documentation is available, you can register with the IBM My Notifications service. You can customize the service so that you receive information about only those IBM products that you specify.

To register with the My Notifications service:

1. Go to https://www.ibm.com/support/mynotifications
2. Enter your IBM ID and password, or create one by clicking **register now**.
3. When the My Notifications page is displayed, click **Subscribe** to select those products that you want to receive information updates about. The IMS Tools option is located under **Software** > **Information Management**.
4. Click **Continue** to specify the types of updates that you want to receive.
5. Click **Submit** to save your profile.

How to send your comments

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 Tools information, see How to provide feedback in IBM Documentation.

When you provide feedback, 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.

Accessibility features

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 major accessibility feature in IMS High Performance Unload is keyboard-only operation for ISPF editors. It uses the standard TSO/ISPF interface.

Keyboard navigation

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

For information about navigating the IMS ISPF panels using TSO/E or ISPF, refer to *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.

IBM and accessibility

See the IBM Human Ability and Accessibility Center at www.ibm.com/able for more information about the commitment that IBM has to accessibility.

Chapter 2. Hardware and software prerequisites

Before you install IMS High Performance Unload, make sure that your environment meets the following minimum hardware and software requirements.

Hardware requirements

IMS High Performance Unload operates on any hardware configuration that supports the required versions of IMS.

Software requirements

IMS High Performance Unload requires Database Manager of one of the currently supported versions of IMS or IMS Database Value Unit Edition.

The operating system requirements of IMS High Performance Unload are the same as those required by the corresponding IMS release.

To use the Data Conversion exit of IMS, IMS/ESA Year 2000 Exit Tool Version 1 (5697-E04) is required (this product supports IMS versions up to 7.1).

Note: No user-written Data Conversion exit routine (DFSDBUX1) is supported.

To use the Db2 DL/I Batch support, one of the currently supported versions of Db2 UDB for z/OS is required.

Restrictions

IMS High Performance Unload is subject to the following restrictions with respect to processing environment:

- It cannot be run under IMS Utility Control Facility (UCF).
- It does not support the checkpoint and restart capability.

Restrictions that are related with each utility program are described in “Restrictions for IMS High Performance Unload” on page 22.

Part 2. Unloading IMS databases

You can use the FABHURG1 utility or the FABHFSU utility to unload IMS databases, or you can enable your application program to use the High Speed Sequential Retrieval Engine (HSSR Engine) of IMS High Performance Unload.

Topics:

Chapter 3. Introduction to the unload utilities

IMS High Performance Unload provides two unload utilities, FABHURG1 and FABHFSU, both of which run as HSSR application programs. The utilities take advantage of the services provided by HSSR Engine.

The unloaded data set produced by these unload utilities can be used as input to the IMS HD Reorganization Reload utility and to other reload utilities that are compatible with it.

Topics:

Selecting an unload utility for your use

IMS High Performance Unload provides two unload utilities, FABHURG1 and FABHFSU, both of which run as HSSR application programs. Before selecting which utility to use, you must understand the functional differences between each utility.

Procedure

Use the following information to select the unload utility that meets your unloading needs:

The FABHURG1 utility is relatively simple. The FABHFSU utility provides more functions than FABHURG1, and is compatible with FSU II.

Both unload utilities are usually run in the ULU region. In that ULU region, all database segment types are unloaded automatically. These utilities can also be run in DLI region, and FABHURG1 can be run in the DBB region.

Consider using the FABHURG1 unload utility if you want to:

Unload a database into one of the six standard formats.
One of the six standard formats is acceptable as input to IMS HD Reorganization Reload utility or a compatible utility. For input processing by application programs, the other four standard formats are more practical. The six standard formats include the communication industry standard format (*CS format).

Note: The *CS format cannot be selected in FABHFSU jobs.

Do a migration unload to HALDB or a fallback unload from HALDB.
FABHURG1 can unload a database to migrate an HIDAM or HDAM database to a PHIDAM database or a PHDAM database. FABHURG1 can also unload a database to do fallback from a PHIDAM or PHDAM database to an HIDAM or HDAM database. The unloaded data set created is compatible with the unloaded data set created by IMS HD Reorganization Unload utility.

Note: You can also use FABHFSU in PSF mode to do a migration unload. By using FABHFSU in PSF mode, you can process the partitions in parallel, which results in faster processing. For more information, see "Parallel migration unload" on page 110. Fallback unload from HALDB is not supported by FABHFSU.

Create a database extract to perform database tuning experiments.
Using the exit routine FABHEXTR provided by IBM, you can create a small database extract from a large database, for use in database tuning experiments.

Use a user record-formatting routine to format output records or to edit segment data (for system programmers).
FABHURG1 accepts a user exit that enables system programmers to write a user record-formatting routine for formatting output records and an optional exit routine for editing segment data.

Note: User exit that enables system programmers to write a user record-formatting routine for formatting output records or optional exit routines for editing segment data are not supported by FABHFSU.

Consider using the FABHFSU unload utility if you want to:

Create up to three data sets in one single execution.
Different record formats can be specified for each output data set. If FABHFSU runs in the DLI region, different PCBs that refer to the same DBD can be specified for each output data set.

Note: FABHURG1 does not support generating multiple outputs with different formats.

Use a user exit that is compatible with user exit of IPR Unload utility or DBT HSSR (for application programmers).
The user exit is compatible with the user exit of FABHFSU in DBT HSSR and with the user exit of IPR Unload. A different user exit routine can be specified for each output.

If you want to edit segment data or to discard some segments on the basis of criteria you chose, you should use FABHFSU and write an exit routine for it.

Notes:

- User exits are also provided for FABHURG1, but they are intended to be used mainly by system programmers who have good knowledge of IMS database and HSSR Engine.
- FABHURG1 does not support user exits that are compatible with user exits of IPR Unload utility.

Unload a specified range or portion of an HISAM, HIDAM, or HDAM database.
You can unload the portion of the database to be read. For HISAM or HIDAM, the limits of the range are specified as keys. For HDAM, the limits of the range are defined as relative block numbers of the CIs or blocks in the Root Addressable Area.

Unload a multivolume database by scanning separately or concurrently (PSF mode).
FABHFSU has two different modes: the standard mode and the Parallel Scan Facility (PSF) mode. In the PSF mode, FABHFSU performs individual scan phases, which can run separately or concurrently to unload a multivolume database.

Note: In both the standard mode and the PSF mode, FABHFSU runs as an HSSR application program.

Use FSU II JCL.
You can use your FSU II JCL to run FABHFSU by making small modifications to your FSU II JCL.

Note: FABHURG1 is not compatible with FSU II.

What to do next

See also "Restrictions for IMS High Performance Unload" on page 22 and "Considerations for using the unload utilities" on page 25 to learn the restrictions and considerations that apply to each utility.

When you have determined the unload utility to use, see the following topics for the instructions to use the utility:

- For FABHURG1 utility, see Chapter 5, "FABHURG1 unload utility," on page 35.
- For FABHFSU utility in standard mode, see Chapter 6, "FABHFSU unload utility," on page 51.
- For FABHFSU utility in PSF mode, see Chapter 10, "Parallel Scan Facility of FABHFSU," on page 121.

Restrictions for IMS High Performance Unload

Certain restrictions apply when using the unload utilities or the API of IMS High Performance Unload.

Subtopics:

- “Restrictions specific to FABHFSU” on page 25

Restrictions common to all HSSR applications

The following restrictions apply to all HSSR application programs including the FABHURG1 utility and the FABHFSU utility:

- Logical DBD is not supported.
- A sensitive virtual logical child (LCHILD) is not supported.
- Field sensitivity for an HSSR PCB is not supported.
- For the restrictions of DBRC authorization for database access, see “Support for database level sharing” on page 92.
- For the restrictions of secondary index processing, see “Considerations for using a secondary index” on page 27.
- If IMS database versioning is enabled and any database version other than the current version is specified, HSSR Engine passes the DL/I calls to the DL/I call handler of IMS. Certain HSSROPT control statements are ignored. For more information, see the explanation of message “FABH0638W” on page 467. For the FABHURG1 utility and the FABHFSU utility, any database version other than the current version is not supported.
- The following restrictions apply to HSSR calls:
  - For details of call types supported in each API set, see “DL/I calls supported by each API set” on page 85.
    - If APISET 1 or 2 is specified, or if the target is a non-HD database, the following restrictions are applied:
      - An unqualified or qualified SSA is not fully supported.
      - Three or more SSAs are supported restrictively.
      - Command codes except for NULL('*-') command code are not supported.
      - Multiple qualification statements are not supported.
    - If APISET 3 is specified for non-HD database and the unsupported call is issued, HSSR Engine ends abnormally without passing the call to IMS DL/I.
    - A get-next call issued after GU calls that returned a GE status might not return the same segment as DL/I.
  - For details about the EXEC DLI command types supported in each API set, see “EXEC DLI commands supported by each API set” on page 86. For the EXEC DLI commands, the restrictions corresponding those for the DL/I calls apply.
- The following restrictions apply to Application Interface Block (AIB) interface:
  - HSSR calls using AIB interface are not supported.

    An HSSR call that uses the interface is passed to IMS's DL/I call handler and is processed as a DL/I call or an EXEC DLI command for the corresponding DL/I PCB.
  - An HSSR application program that uses AIB system service calls might have problems. The use of AIB system service calls is not recommended.
- In APISET 1 and 2, by default, for a logical child with a logical parent's concatenated key (LPCK) that is specified as VIRTUAL on the SEGM statement of the DBD, HSSR Engine returns blanks in the I/O area that would usually hold the LPCK (as if field sensitivity were in effect). For compatibility with FSU II, FABHFSU returns binary zeros to the I/O area instead of blanks. If you need to have LPCKs built in the I/O area, you must specify the BLDLPCK control statement in the HSSROPT data sets.
- The data capture exit routine is supported, with the following restrictions:
  - A data capture exit routine cannot issue an HSSR call.

  - If an application program issues an HSSR call for the PCB that follows, the call is transferred to the DL/I language interface and the message FABH0671W is issued with RC=USREXIT. The reason that caused the message is as follows:
    - PCB is generated with PROCOPT=R
    - The DBD referred to by the PCB is generated with the data capture exit routine.

    In this case, therefore, the HSSR application program cannot take advantage of the HSSR Engine.
- No user-written Data Conversion exit routine (DFSDBUX1) is supported.
- Db2 DL/I Batch interface is supported, with restrictions.

  See "Considerations for Db2 DL/I Batch interface" on page 88.
- DL/I checkpoint and restart calls are supported, with restrictions.

  See "Considerations for checkpoint and restart" on page 89.
- An HSSR application program can run in a database-sharing environment. However, if the application program reads a database that is concurrently being updated by IMS, HSSR Engine does not guarantee read integrity.

  See "Consideration for database sharing" on page 90.
- An application program that gets information from DL/I control blocks might have problems in the IMS High Performance Unload environment.

Restrictions common to FABHURG1 and FABHFSU

The following restrictions are common to FABHURG1 and FABHFSU:

- The DBD that specifies the input database must be a physical DBD.
- If IMS database versioning is enabled, only the current version of the database can be used. Any other versions of the database are not supported.
- FABHURG1 and FABHFSU do not support sensitive, virtually paired segments.

  In the ULU region, the virtual logical child segment is ignored. In the DLI and the DBB regions, the processing depends on the SKIPVLC control statement. For details of the SKIPVLC control statement, see "SKIPVLC control statement" on page 176.

  The restriction applies to the unloading of databases for reorganization because the data set input to IMS HD Reorganization Reload utility does not contain virtually paired segments.

  Note: Virtually paired segments are unloaded when migration unload (to HALDB) is performed.
- If either of the following segment occurrences is skipped or lost during the unload, FABHURG1 and FABHFSU cannot be used in a reorganization to unload a corrupted database that has logical relationships:
  - A logical parent segment that has one or more logical children.
  - A logical child segment that is physically paired.
- Make sure that no segments required by the IMS HD Reorganization Reload utility are missing during reorganization for either of the following reasons:
  - Having no segment types defined as data-sensitive by a SENSEG statement (this condition does not happen when you use a ULU region in unloading.)
  - Having a user exit routine set a return code indicating that some segments should be skipped.
- No support is available for multivolume output data sets whose extents reside on volumes of more than one device type.
- FABHURG1 and FABHFSU do not support IMS Tools Knowledge Base. It is supported by the IPR Unload utility. For the details, see the *IMS Database Reorganization Expert User's Guide*.
- For migration unload, the following restrictions apply:
  - Migration unload of the secondary index is not supported.

- Migration unload of the HISAM database is not supported.
- If PTR=H or PTR=HB is defined as the parent segment of virtual logical child, migration unload of the database is not supported.

Restrictions specific to FABHURG1

The following restrictions apply only to FABHURG1:

- If the BLDLPCK control statement is not specified in the HSSROPT data set, the following restriction applies:
  - When FABHURG1 retrieves a logical child segment for which the logical parent's concatenated key (LPCK) has been specified as VIRTUAL on the SEGM statement of the DBD, blanks are returned to the part of the I/O area where the LPCK is to be placed.

  Therefore, you must specify a BLDLPCK control statement when unloading such a database if you specified the database in the DBIL= statement when running the IMS Database Prereorganization utility. For performance reason, you should not specify a BLDLPCK statement if you specified the database in the DBR= statement when running the Prereorganization utility.
- Fallback unload of the partitioned secondary index (PSINDEX) is not supported.

Restrictions specific to FABHFSU

The following restrictions apply only to FABHFSU:

- FABHFSU cannot be run in the DBB region; FABHDBB procedure cannot be used for FABHFSU.
- HSSRPCB and HSSRDBD control statements specified in the HSSROPT data set are always ignored. In FABHFSU, HSSR PCBs are specified by the DBD control statement and the PSB control statement in the CARDIN data set (see “FABHFSU CARDIN input data set” on page 55).
- By default, for a logical child segment for which a logical parent's concatenated key (LPCK) is specified as VIRTUAL on the SEGM statement of the DBD, binary zeros are returned to the I/O area when that segment is retrieved. This is because of the compatibility with FSU II. If you need to have the LPCK built —for example, when you have specified the DBIL control statement in the Database Prereorganization utility—you must specify the BLDLPCK control statement in the HSSROPT data set when you run the FABHFSU job.
- The PROCOPT statement fields on the PCB and SENSEG statements can have any value acceptable to IMS, although only the PROCOPTs that are listed in “Processing option (PROCOPT) requirements” on page 80 have meaning to FABHFSU.

  Note: In a database-sharing environment, DBRC uses the PROCOPT values to check whether database access can be granted (see “Handling data set extensions” on page 91).
- Field sensitivity can be specified during PSBGEN, but is ignored by FABHFSU.
- Fallback unload is not supported.

Considerations for using the unload utilities

Certain considerations apply when using the unload utilities of IMS High Performance Unload.

Considerations for a logical parent's concatenated key

By default, for a logical child segment with a logical parent's concatenated key (LPCK) that is specified as VIRTUAL on the SEGM statement of the DBD, FABHURG1 returns blanks in the I/O area that usually contains LPCK, and FABHFSU returns binary zeros in the area.

To have HSSR Engine build the LPCK and return it in the I/O area, you must specify BLDLPCK control statement in the HSSROPT data set. For details, see “BLDLPCK control statement” on page 160.

When you unload an uncorrupted database that has a logical child whose LPCK is defined as virtual, and if BLDLPCK statement is not specified, you must run the IMS Database Prereorganization utility with the

control statement DBR= to get a successful reload and prefix resolution. The control statement DBIL= gives incorrect results in this case.

You must specify BLDLPCK statement when you unload a corrupted database that has a logical pointer error in a logical child whose LPCK is defined as virtual. Since you suspect that logical pointers are incorrect, you must also run the Database Prereorganization utility, using the DBIL= control statement. Otherwise, you will get an incorrect reload that would be detected during prefix resolution.

Considerations for an unloaded data set used for reorganization

Unloaded data sets that are created under certain conditions cannot be reloaded with the IMS HD Reorganization Reload utility (DFSURGL0).

A data set created by FABHURG1 in *HD format or by FABHFSU in UL format can be used as input for the IMS HD Reorganization Reload utility (DFSURGL0). If you create an unloaded data set by specifying a DECN control statement for a database that contains a compressed segment, that data set is not compatible with the unloaded data set created by the IMS HD Reorganization Unload utility. For details, see the following topics:

- For FABHURG1, see “DEC control statement” on page 42.
- For FABHFSU, see “DEC control statement” on page 59.

You cannot reload such an unloaded data set by using the IMS HD Reorganization Reload utility (DFSURGL0), but you can reload it by using the IMS High Performance Load (Load utility or PSSR utility) or the IPR Reload utility.

There are the following differences, which do not affect DFSURGL0:

- IMS adds padding bytes of binary zeros to make the length of the segment even; FABHURG1 and FABHFSU do not.
- IMS overrides the BLKSIZE specified in the JCL with the computed values. FABHURG1 and FABHFSU is overridden by the BLKSIZE specified.

Considerations for database sharing

In general, like any other unload utility during reorganization, FABHURG1 and FABHFSU must have exclusive control of the database while they run. In some cases, however, you might want to read the database while it is being updated by IMS.

The update might be done either by IMS within the same program or by another IMS subsystem through concurrent execution. For such cases, see “Database sharing support” on page 90. The considerations in the topic also apply to FABHURG1 and FABHFSU.

Considerations for HALDB Online Reorganization capable partitions

IMS High Performance Unload supports HALDBs. IMS High Performance Unload supports also the PHDAM or the PHIDAM partitions that are defined as HALDB Online Reorganization (OLR) capable.

However, if one or more partitions are in the following HALDB OLR status, IMS High Performance Unload cannot process the HALDB:

- HALDB OLR is currently running.
- HALDB OLR for the partition is suspended and one of the following options is specified for FABHURG1 or FABHFSU:
  - DECN
  - User exit routine
  - *CS format for PHDAM
  - PARTEXTR

When none of these options is specified, IMS High Performance Unload can process the partition. However, the following options for the HSSR Engine are ignored:

- BYINDEX
- CO
- DBSTATS
- KEYCHECK
- SKERROR

Because the DBSTATS control statement is ignored, the following reports are not printed:

- DB Statistics report
- Randomizing Statistics report
- DB Record Length Distribution report

The CAB Statistics report for the partition of which HALDB OLR is suspended and partitions that follow this partition are not printed.

In this case, the performance decreases because the HSSR Engine passes the DL/I calls to the IMS's DL/I call handler from the partition. Consider running IMS High Performance Unload after the completion of the HALDB OLR.

Related concepts

Methods for processing High Availability Large Databases
You can use the FABHURG1 unload utility, the FABHFSU unload utility, or your HSSR application program to process High Availability Large Databases.

Considerations for using a secondary index

A secondary index can be used to retrieve HIDAM or HDAM root segments. The target segment of the secondary index must be a root segment.

In the ULU region, you can specify the name of the secondary index by using the following control statement:

- For FABHFSU, "DBD control statement" on page 58.
- For others, "BYINDEX control statement" on page 162.

In the DLI or the DBB region, the index name on these control statements is ignored. HSSR Engine uses the secondary index, which is coded by the PROCSEQ= parameter in the specified PCB.

When the secondary index is used, the value of the search field of the index segment is set to the key feedback area of the HSSR PCB, instead of the root key. User exits can specify a next root segment by the value of the search field.

Restriction

The following secondary indexes are not supported:

- Secondary index whose target is not a root segment
- Secondary index with symbolic pointing
- Secondary index that contains index pointer segments with non-unique keys
- Secondary index for databases other than HIDAM or HDAM

Considerations for unloading an IMS catalog

You can use IMS High Performance Unload to unload data from an IMS catalog.

IMS catalogs are HALDB databases. The unload method differs whether DBRC is used to manage the IMS catalog. If DBRC is used to manage the IMS catalog, you can unload data from the IMS catalog in the same manner as you do to unload a HALDB database.

To unload data from an IMS catalog that is not managed by DBRC, use either of the following methods:

- Add the DFSDF=*xxx* parameter as shown in the following examples. You can use either format.

  Example 1:

```
//UNLOAD    EXEC PGM=FABHX034,PARM=(DFSRRC00/ULU,FABHURG1,DFSCD000,
//               ,,,,,,,,,N,N,,N,,,'',,,,,,,'DFSDF=CAT')
```

  Example 2:

```
//UNLOAD    EXEC FABHULU,
//               MBR=FABHURG1,DBD=DFSCD000,DBRC=N,
//               PARM1='DFSDF=CAT'
```

  Here, CAT is the suffix of the DFSDF*xxx* member that contains the UNREGCATLG parameter.

  Then, specify the PROCLIB DD statement to point the DFSDF*xxx* member of the IMS.PROCLIB data set.

- Place the IMS Catalog Definition exit routine (DFS3CDX0) in the STEPLIB DD concatenation.

With either method, you must allocate the catalog partition definition data set. To do so, add the DFSHDBSC DD statement or the DFSMDA member with the TYPE=CATDBDEF statement.

Considerations for IMS-managed ACBs environment

When the IMS management of ACBs is enabled, IMS reads database descriptors (DBDs) from the IMS catalog instead of from the DBD library that is specified in the IMS DD statement.

Refer to the following JCL examples when you prepare IMS HP Unload JCL to run an unload job in an IMS-managed ACBs environment. For more information about coding IMS HP Unload JCL, see Chapter 4, “Basic job control language,” on page 29.

Example 1: Specifying the DFSDF= parameter and DBRC=Y

```
//UNLOAD   EXEC FABHULU,
//              MBR=FABHURG1,DBD=USERDBD,DBRC=Y,
//              PARM1='DFSDF=CAT'
//G.PROCLIB DD DSN=IMS.PROCLIB,DISP=SHR
//DBSAMP01  DD DSN=DBSAMP01.DATA1,DISP=OLD
//SYSUT2    DD DSN=DBSAMP01.UNLOAD,DISP=(NEW,PASS),
//             SPACE=(CYL,(20,10))
```

DFSDF=CAT parameter is specified on the EXEC statement. This parameter specifies the DFSDFCAT member that enables IMS-managed ACBs. The name of the IMS.PROCLIB data set in which the DFSDFCAT member exists is specified on the PROCLIB DD statement.

You must allocate the catalog partition definition data set when unloading a database with DBRC=NO. To do so, add the DFSHDBSC DD statement or the DFSMDA member with the TYPE=CATDBDEF statement.

Example 2: Using the IMS Catalog Definition exit routine

```
//UNLOAD   EXEC FABHULU,
//              MBR=FABHURG1,DBD=USERDBD,DBRC=Y
//G.STEPLIB DD
//          DD
//          DD DSN=user.DFS3CDX0,DISP=SHR
//DBSAMP01  DD DSN=DBSAMP01.DATA1,DISP=OLD
//SYSUT2    DD DSN=DBSAMP01.UNLOAD,DISP=(NEW,PASS),
//             SPACE=(CYL,(20,10))
```

The DFS3CDX0 exit routine must be bound into IMS.SDFSRESL or a concatenated library.

You must allocate the catalog partition definition data set when unloading a database with DBRC=NO. To do so, add the DFSHDBSC DD statement or the DFSMDA member with the TYPE=CATDBDEF statement.

Chapter 4. Basic job control language

An IMS High Performance Unload job can run in a DL/I, DBB, or ULU region by using DL/I JCL. The DL/I JCL requires minor changes for each region type.

Topics:

Preparing the basic JCL

You must select a region type and modify the DL/I JCL to prepare your JCL for an IMS High Performance Unload job.

Procedure

1. Select a region type for the IMS High Performance Unload job.

   An IMS High Performance Unload job can run in a DL/I, DBB, or ULU region:

   - The DL/I region is used to run an HSSR application program by using PSB and DBD libraries. The DBB region is used to run programs by using an ACB library.
   - The ULU region is used primarily in running the unload utilities FABHURG1 and FABHFSU. To run an application program in the ULU region, you must specify the physical DBD, but a PSB is not needed. This specification ensures that all segments are data-sensitive.
   - In a ULU region, IMS assumes that the IMS HD Reorganization Unload utility is running. Consequently, in a database sharing environment, DBRC grants the database access to an HSSR application program or utility under the same conditions as for the IMS HD Reorganization Unload utility. When an application program is called internally, the PCB list is passed to the application program in accordance with Assembler or COBOL conventions. A PCB for an HSSR application program that is run in a ULU region is defined as an HSSR PCB.

   Note: You can use the HSSROPT DBDL1 control statement to override the HSSR PCB and force DL/I calls to be made to the database.

   If your application program must be protected from the addition of new segment types to the database, do not select the ULU region.
2. Modify the DL/I JCL for IMS High Performance Unload.

   To run an IMS High Performance Unload job, you must modify the normal DL/I JCL procedure.

   A cataloged procedure is provided for each region type in the HPS.SHPSSAMP sample library. Use the appropriate cataloged procedure, or prepare a similar procedure of your own.

Table 1. Cataloged procedures for each region type

Cataloged procedure	Description
FABHDLI for DLI region	FABHDLI is used to run batch application programs by using DL/I and HSSR with PSBs and DBDs. It is similar to the DL/I procedure DLIBATCH.
FABHDBB for DBB region	FABHDBB is used in conjunction with ACBs to run batch application programs. It is similar to the DL/I procedure DBBBATCH.
FABHULU for ULU region	FABHULU is used in conjunction with the ULU region types to run utilities and generalized application programs. You must provide the name of a physical DBD with DBD=*keyword*.

Modify the JCL so that the basic JCL requirements for running IMS High Performance Unload utilities are met. For descriptions for the EXEC and DD statements, see “Basic JCL requirements” on page 30.

3. If any of the following conditions apply, follow the instructions in these topics:
   - “Considerations for a logical parent's concatenated key” on page 25
   - “Considerations for an unloaded data set used for reorganization” on page 26
   - “Considerations for database sharing” on page 26
   - “Considerations for HALDB Online Reorganization capable partitions” on page 26
   - “Considerations for using a secondary index” on page 27
   - “Considerations for unloading an IMS catalog” on page 27
   - “Considerations for IMS-managed ACBs environment” on page 28

Basic JCL requirements

The basic JCL requirements for IMS High Performance Unload are for the IMS High Performance Unload runtime environment initializer (FABHX034). Many programs of IMS High Performance Unload and user-written HSSR application programs require that the basic JCL requirements are met.

Subtopics:

- “EXEC and DD statements” on page 30
- “Additional JCL requirements for each utility” on page 33

EXEC and DD statements

The following table summarizes the DD statements for the runtime environment initializer (FABHX034).

Table 2. Runtime environment initializer (FABHX034) DD statements

DDNAME	Use	Format	Need
STEPLIB	Input	-	Required
HSSROPT	Input	LRECL=80	Optional
HSSRCABP	Input	LRECL=80	Optional
DFSVSAMP	Input	LRECL=80	Optional
ddname	Input	-	Optional
HSSRLDEF	Input	LRECL=80	Optional
RECONx	Input	-	Optional
IMSDALIB	Input	-	Optional
IEFRDER	Output	-	Required
HSSRSTAT	Output	LRECL=133	Optional
HSSRLOUT	Output	-	Optional
HSSRTRAC	Output	LRECL=133	Optional
HSSRBUTR	Output	-	Optional
HSSRSNAP	Output	LRECL=133	Required
IMS	Input	-	Optional
IMSACB	Input	IMSVS.ACBLIB	Optional
PROCLIB	Input	IMSVS.PROCLIB	Required

Table 2. Runtime environment initializer (FABHX034) DD statements (continued)

DDNAME	Use	Format	Need
IMSMON	Output	DUMMY	Required
DFSRESLB	Input	IMSVS.SDFSRESL	Required

The following list explains the EXEC and DD statements of the JCL to run IMS High Performance Unload:

EXEC

The EXEC statement must be in the following format:

```
//  EXEC PGM=FABHX034,PARM=('DFSRRC00/DLI',pgmname,psbname,...)
```

The program invoked by the PGM keyword on the EXEC statement is the IMS High Performance Unload runtime environment initializer named FABHX034.

The PARM parameters, except the first one, must be specified in the same format as the PARM parameters for the IMS DLIBATCH or DBBBATCH procedures. The first parameter must contain the name of the IMS region controller, DFSRRC00, followed by a slash (/) and the name of the IMS region in which the job is to run. The region name must be one of DBB, DLI, or ULU.

When processing a HALDB, the 14th parameter, which is the DBRC parameter, must be Y.

Tip: For each region type, IMS High Performance Unload provides a cataloged procedure. See "Preparing the basic JCL" on page 29.

STEPLIB DD

Points to the IMS High Performance Unload library, the IMS RESLIB library, and the user libraries that contain user programs and DFSMDA members.

Notes:

- You do not need to APF-authorize the STEPLIB libraries. If however, the STEPLIB is not authorized for example by having unauthorized libraries concatenated, you must specify the DFSRESLB DD statement.
- If all concatenations of the JOBLIB/STEPLIB are APF-authorized, Media Manager is used for reading VSAM ESDS database data sets and OSAM LDS database data sets, which saves the use of CPU time.
- If you do not want to authorize the library that contains the DFSMDA members, specify the IMSDALIB DD statement.

HSSROPT DD

This optional statement defines an input data set that contains optional control statements for HSSR Engine. For more information about HSSROPT control statements, see Chapter 11, "Options for HSSR Engine," on page 155.

HSSRCABP DD

This optional statement defines an input data set that contains the control statements that change buffering parameters for the CAB buffer handler provided by HSSR Engine. For more information about HSSRCABP control statements, see "Chained Anticipatory Buffer handler (CAB)" on page 208.

Note: Chained Anticipatory Buffering (CAB) is the default buffering method provided by HSSR Engine.

DFSVSAMP DD

This DD describes the data set that contains the buffer pool information required by the IMS DL/I buffer handler. If this data set is not specified, it is dynamically created by IMS High Performance Unload. For this reason, this data set should not usually be specified.

If you want to provide this data set, you should specify the minimum required number of buffers in the DFSVSAMP data set because IMS High Performance Unload does not use the buffers taken with the parameters specified in this data set. Otherwise, a larger region size is required in order to run IMS High Performance Unload.

You must however, specify DL/I buffers in DFSVSAMP in the following cases:

- When you specify the BLDLPCK control statement in HSSROPT data set.

  In this case, HSSR Engine issues DL/I calls internally to build LPCKs.
- When you do a migration unload of a database in which a logical child is defined.

  In this case, HSSR Engine issues DL/I calls internally to get information to be written in the header of the unload record of the logical child.
- When APISET 3 is specified in the HSSROPT data set or by the site default option table. In this case, if a call that is not supported in APISET 2 is issued, that call and all succeeding calls are passed to DL/I and are processed by DL/I action modules.

The usual DFSVSAMP definitions should be used in tuning the processing of these internal DL/I calls.

***ddname* DD**
These DDs define the database data set to be processed. One statement of this type must be present for each data set that appears in the DBD describing the database. The value of *ddname* must match the ddname in the DBD.

For an HIDAM database, DD statements must also exist for the data sets that represent the index. The DD statements that specify the index must contain *ddname*s specified in the DBD for the index database. No DD statements are required for the secondary indexes that are associated with this database unless the secondary indexes are used.

If the BLDLPCK control statement is specified in the HSSROPT data set, the database in which the logical parent is defined must be specified.

You must not code DCB=BUFNO=*n* or AMP='BUFND=*n*' options to request access method buffers for database data sets; HSSR Engine allocates its own buffer pool for each data set group.

Note: If you use dynamic allocation, do not use the DD statement for the database data sets.

For HALDBs, no DD statement needs to be specified for any database data set because the data set is always dynamically allocated.

HSSRLDEF DD
This optional statement defines the input data set that contains control statements for requesting the DB Record Length Distribution report with your own database record length ranges. For more information about HSSRLDEF control statements, see “Activating the Database Tuning Statistics” on page 307.

RECON*x* DD
This statement provides RECON1, RECON2, and RECON3 DDs under the same conditions as for standard IMS jobs. If RECON data sets need to be allocated dynamically, do not specify these DDs.

IMSDALIB DD
This optional statement defines the library that contains the DFSMDA members for dynamic allocation. Allocation of the database data sets and the RECON data sets is attempted in the following order:

1. The DD statements coded in the JCL stream
2. Dynamic allocation by data set definition that is contained in RECON (only for HALDB data sets)
3. Dynamic allocation members in the IMSDALIB concatenation or in the JOBLIB/STEPLIB concatenation

If you specify this library on the IMSDALIB DD statement, it is highly recommended that you remove the library from the STEPLIB concatenation.

IEFRDER DD
This required statement defines the IMS log data set.

HSSRSTAT DD
This optional statement defines the output data set for HSSR Engine to write statistical output after the termination of the application program. For the details, see “HSSRSTAT data set” on page 181.

HSSRLOUT DD
This optional statement defines the output sequential data set, in which a small record is written for each database record when the Database Tuning Statistics function is activated by coding the DBSTATS control statement in the HSSROPT data set. For details about this data set, see "Activating the Database Tuning Statistics" on page 307.

Note: The BLKSIZE and LRECL for the HSSRLOUT data set must not be specified in JCL. HSSR Engine determines them dynamically from the key length of the root segment.

HSSRTRAC DD
This optional statement defines an output data set in which the trace data is written. The DD statement is required whenever any of the following functions are activated by HSSROPT statements.

- The trace options TRHC and TRDB
- The compare option CO
- The diagnostic option DIAGG

For details about these control statements, see Chapter 11, "Options for HSSR Engine," on page 155. For details about this data set, see "HSSRTRAC data set" on page 190.

HSSRBUTR DD
This optional statement defines the output data set on which the buffer handler trace is written. It is ultimately used as input for a buffer simulation run with the FABHBSIM utility.

```
//HSSRBUTR DD DSN=xxx,DISP=(,KEEP),UNIT=tape,...
```

HSSRSNAP DD
This required statement defines the output data set on which snapshots are written when an error occurs during initialization of HSSR Engine. For details, see "HSSRSNAP data set" on page 202.

The remaining five DD statements (that is, IMS, IMSACB, PROCLIB, IMSMON, and DFSRESLB) are for IMS. See *IMS Database Utilities*.

Additional JCL requirements for each utility

When running one of the following utilities, see also the JCL requirement topic for that utility:

- For the FABHURG1 database unload utility, see "FABHURG1 JCL requirements" on page 39.
- For the FABHFSU database unload utility, see "FABHFSU JCL requirements" on page 54.

When using the Database Tuning Statistics functions, see the following topics:

- To customize the DB Record Length Distribution report (HSSRLDEF data set), see "JCL requirements for the Database Tuning Statistics" on page 308.
- To record the length of each database record (HSSRLOUT data set), see "JCL requirements for the Database Tuning Statistics" on page 308.
- To print long database records (FABHLDBR utility), see "FABHLDBR JCL requirements" on page 314.

Chapter 5. FABHURG1 unload utility

The FABHURG1 unload utility replaces the functions of the IMS HD Reorganization Unload utility (DFSURGU0) and the HISAM Reorganization Unload utility (DFSURUL0). By using HSSR Engine, FABHURG1 provides high speed unloading of databases with more control over the unload process.

FABHURG1 unloads databases faster than the IMS HD Reorganization Unload utility because it runs as an HSSR application program. The utility accepts user requests through the SYSIN data set. You can specify options for HSSR Engine by coding control statements in the HSSROPT data set and the HSSRCABP data set. For the output, in addition to the standard output reports produced by HSSR Engine, a Segment Statistics report is produced in the SYSPRINT data set.

Tip: To understand the system structure and the data flow in HSSR application jobs, see "IMS High Performance Unload system structure" on page 11.

You can also use the FABHURG1 utility to unload a corrupted database, perform migration unload, or fallback unload. For information about these tasks, see the following topics:

- To unload a corrupted database, see Chapter 9, "Utility options for unloading corrupted databases," on page 115.
- To perform migration unload or fallback unload, see "Migration unload and fallback unload" on page 107.

For the restrictions that apply to FABHURG1 jobs, see "Restrictions for IMS High Performance Unload" on page 22.

Topics:

Unloading a database with FABHURG1

To unload a database by using the FABHURG1 utility, you must select the format of the unload output and modify the basic JCL.

Procedure

To unload a database by using the FABHURG1 utility, complete the following steps:

1. Select a region type from ULU, DLI, or DBB, and prepare the basic JCL.

   For instructions for preparing the basic JCL, see "Preparing the basic JCL" on page 29.
2. Code the additional DD statements to meet the JCL requirements of FABHURG1.

   For additional JCL requirements, see "FABHURG1 JCL requirements" on page 39.
3. Select a format for the unload output.

   You can select from six formats. See "Unload output format supported by FABHURG1" on page 36 to determine the output format.
4. Code the required SYSIN control statements.

SYSIN control statements control the behavior of the FABHURG1 utility job. See “FABHURG1 SYSIN input data set” on page 41 for descriptions of the SYSIN control statements.

5. Optional: Code HSSROPT and HSSRCABP control statements to specify the options for HSSR Engine.

   These control statements are optional. For more information, see “FABHURG1 HSSROPT input data set” on page 45 and “FABHURG1 HSSRCABP input data set” on page 46.
6. Submit the JCL to run the FABHURG1 job.
7. Check the output reports and messages.

   In addition to the standard output reports produced by HSSR Engine, a Segment Statistics report is produced in the SYSPRINT data set. To interpret the report, see “FABHURG1 output: SYSPRINT output data set” on page 46.

Related reference

FABHURG1 JCL examples
Use the following JCL examples to prepare your FABHURG1 JCL.

Unload output format supported by FABHURG1

FABHURG1 can unload a database in any of six formats, *HD, *F1, *F2, *F3, *CS, and *CP.

These formats are called *standard formats* of FABHURG1. The *HD format is the default format. The output generated in this format is acceptable as input to the IMS HD Reorganization Reload utility or to a compatible utility. The other formats are used for processing by application programs.

Subtopics:

*HD format

If you use *HD format, you can replace the IMS HD Reorganization Unload utility with the faster FABHURG1 utility when reorganizing a database. The *HD format is the default format.

Considerations:

- If you create an unload data set by specifying the DECN control statement for a database that contains a compressed segment, the data set is not compatible with an unloaded data set that is created by the IMS HD Reorganization Unload utility. For details, see “DEC control statement” on page 42. You cannot reload such an unloaded data set by using the IMS HD Reorganization Reload utility (DFSURGL0), but you can reload it by using IMS High Performance Load (Load utility or PSSR utility) or the IPR Reload utility.
- FABHURG1 *HD output records can be compared against the output records created by the IMS HD Reorganization Unload utility. To do so, add the optional SYSUT1 statement to your JCL to define the IMS unloaded data set to be used in the comparison. See the description of SYSUT1 DD statement in “FABHURG1 JCL requirements” on page 39.

*F1 format

This section describes the *F1 format, which is a product-sensitive programming interface. See “Programming interface information” on page 518 to understand the restrictions associated with this type of material.

PSPI

This format is to be used for processing by application programs. When you use the *F1 Format, a variable-length record is written for each retrieved database segment that contains the following data fields:

- Segment code
- Segment level
- Segment data as returned by the HSSR call and as seen by HSSR application programs

```
           FORMAT OF *F1 RECORDS

STATEMENT            OFFSET    DESCRIPTION
                    DEC  HEX

REC1     DSECT
REC1LEN  DC   H'0'   0    0    RECORD LENGTH FIELD
REC1ZZ   DC   H'0'   2    2    ZZ   (RESERVED FOR MVS)
REC1SC   DC   X'00'  4    4    SEGMENT CODE
REC1LEV  DC   X'00'  5    5    SEGMENT LEVEL
REC1DATA EQU  *      6    6    SEGMENT DATA AS RETURNED
                                  BY HSSR
```

Considerations

- If the database contains segments shorter than 8 bytes, do not send the output to tape. (A block of less than 18 bytes is not acceptable for a tape.)
- If the database contains segments shorter than 12 bytes, the output cannot be processed by some Sort/Merge programs. (Some Sort/Merge programs require records with at least 18 bytes.)

PSPI

*F2 format

This section describes the *F2 format, which is a product-sensitive programming interface. See "Programming interface information" on page 518 to understand the restrictions associated with this type of material.

PSPI

This format is to be used for processing by application programs. When you use the *F2 format, a variable-length output record is written for each retrieved segment of the database. It contains the following data fields:

- Segment code
- Segment level
- Segment name
- Length of segment as returned by the HSSR call
- Offset of sequence field within segment data (zero if the segment has no sequence field)
- Length of sequence field (if the segment has no sequence field, zero is used.)
- A field that contains zeros for compatibility with *F3 format (in this field, the *F3 format contains the actual length of the concatenated PCB key feedback area.)
- The segment data as returned by the HSSR call and as seen by HSSR application programs

```
           FORMAT OF *F2 RECORDS

STATEMENT            OFFSET    DESCRIPTION
                    DEC  HEX

REC2     DSECT
REC2LEN  DC   H'0'   0    0    RECORD LENGTH FIELD
REC2ZZ   DC   H'0'   2    2    ZZ   (RESERVED FOR MVS)
REC2SC   DC   X'00'  4    4    SEGMENT CODE
REC2LEV  DC   X'00'  5    5    SEGMENT LEVEL
REC2SYM  DC   CL8' ' 6    6    SYMBOLIC SEGMENT NAME
REC2IOAL DC   H'0'   14   E    SEGMENT DATA LENGTH AS
                                  RETURNED BY HSSR
REC2KOFS DC   H'0'   16   10   KEY OFFSET WITHIN DATA
REC2KEYL DC   H'0'   18   12   KEY LENGTH OF THIS
                                  SEGMENT
REC2MKL  DC   H'0'   20   14   0 FOR COMPATIBILITY
                                  WITH *F3 FMT
REC2DATA EQU  *      22   16   SEGMENT DATA AS RETURNED
                                  BY HSSR
```

PSPI

*F3 format

This section describes the *F3 format, which is a product-sensitive programming interface. See “Programming interface information” on page 518 to understand the restrictions associated with this type of material.

PSPI

This format is to be used for processing by application programs. The *F3 format has the same format as the *F2 format, except the *F3 format also contains the concatenated PCB key feedback area after the segment data.

```
           FORMAT OF *F3 RECORDS

STATEMENT            OFFSET    DESCRIPTION
                    DEC  HEX

REC3     DSECT
REC3LEN  DC   H'0'   0    0    RECORD LENGTH FIELD
REC3ZZ   DC   H'0'   2    2    ZZ   (RESERVED FOR MVS)
REC3SC   DC   X'00'  4    4    SEGMENT CODE
REC3LEV  DC   X'00'  5    5    SEGMENT LEVEL
REC3SYM  DC   CL8' ' 6    6    SYMBOLIC SEGMENT NAME
REC3IOAL DC   H'0'   14   E    SEGMENT DATA LENGTH AS
                                  RETURNED BY HSSR
REC3KOFS DC   H'0'   16   10   KEY OFFSET WITHIN DATA
REC3KEYL DC   H'0'   18   12   KEY LENGTH OF THIS SEGMENT
REC3MKL  DC   H'0'   20   14   PCB KEY-FEED-BACK-LENGTH
REC3DATA EQU  *      22   16   SEGMENT DATA AS RETURNED
                                  BY HSSR
REC3KFD  EQU  *      22   16   PCB KEY-FEED-BACK-AREA
```

PSPI

*CS format

The communication industry standard format. This format is usable as input to the IPR Reload utility or IMS High Performance Load.

Consideration: The unloaded file in this format cannot be used for reorganization under the following conditions:

- If a logical relationship is defined in the database.
- If the database is a HALDB and a partitioned secondary index (PSINDEX) database is defined.

*CP format

The communication industry partitioned format. This format can be used as input to the IPR Reload utility or IMS High Performance Load. This format is useful if the database is a HALDB with partitioned secondary index (PSINDEX) databases.

Consideration: If a logical relationship is defined in the database and the database is unloaded in *CP format, you cannot use the unloaded file for reorganization.

FABHURG1 JCL requirements

FABHURG1 runs as an HSSR application program and, therefore, must meet the requirements for the basic JCL (FABHX034 JCL). In addition, FABHURG1 JCL requires other DD statements.

Prerequisite: See "Basic JCL requirements" on page 30 for the basic (FABHX034) JCL requirements.

The following table summarizes the additional JCL requirements for FABHURG1.

Table 3. FABHURG1 DD statements

DDNAME	Use	Format	Need
SYSIN	Input	LRECL=80	Optional
SYSPRINT	Output	LRECL=133	Required
SYSUT1	Input	HD Unload	Optional
SYSUT2	Output	-	Required
SYSUT3	Output	-	Optional

Note: You can also use JCL that is written for IMS HD Reorganization Unload (DFSURGU0) to run FABHURG1. For details, see "IMS HD Reorganization Unload JCL for running FABHURG1" on page 48. In this compatibility mode, Media Manager will not become effective for VSAM ESDS and OSAM LDS, regardless of whether JOBLIB or STEPLIB libraries are APF-authorized.

EXEC

This statement invokes procedures FABHULU, FABHDLI, or FABHDBB (see "Preparing the basic JCL" on page 29). The EXEC statement must be in one of the following formats:

```
//   EXEC FABHULU,MBR=FABHURG1,DBD=dbdname
//   EXEC FABHDLI,MBR=FABHURG1,PSB=psbname
//   EXEC FABHDBB,MBR=FABHURG1,PSB=psbname
```

SYSIN DD

This optional DD statement defines the input data set that contains control statements for FABHURG1.

For the description of the control statements, see "FABHURG1 SYSIN input data set" on page 41.

SYSPRINT DD

This required statement defines the output data set to which FABHURG1 writes error messages and segment statistics. The data set can be defined as:

```
//SYSPRINT DD SYSOUT=A
```

SYSUT1 DD

This optional statement defines an input data set created (in a previous step) by the IMS HD Reorganization Unload utility. Use it only for problem determination. It activates a comparison of *HD output records with the output records that were created by the IMS HD Reorganization Unload utility. FABHURG1 does not compare the lengths of the header or trailer records.

If a mismatch is detected, FABHURG1 ends abnormally with an error message.

The record that is created by FABHURG1 is compared with the record in SYSUT1 before any optional user exit routine is invoked. For records that contain a logical child with a logical parent's concatenated key that is specified as VIRTUAL on the SEGM statement of the DBD, the comparison is done as follows:

- If a BLDLPCK control statement is specified, the entire record is compared.
- If no BLDLPCK control statement is specified, the entire record except the LPCK field is compared.

If the utility detects that a segment is not sensitive, or if a user exit routine requests that one or more segments be skipped, the comparison is halted.

SYSUT2 DD

This required statement defines the primary output data set on which the database is unloaded. The data set can reside on a tape or a direct-access device. It can also be defined as DUMMY.

You can specify the number of buffers in the URG1BUFNO= option in the default option table (FABHOPT). For details, see Chapter 19, “Site default options,” on page 261. The buffers are obtained above the 16-MB line.

If neither the BUFNO subparameter of the DCB parameter on the DD statement nor the URG1BUFNO= option is given, the number of buffers is determined automatically. The buffers are approximately 1 MB in total size and are above the 16-MB line.

If a block size is not specified on the SYSUT2 DD statement, FABHURG1 uses the following block size:

- For device type 3380, the default block size is 23 K. If the database contains segments larger than 23 K, the maximum block size of 32 K is used.
- For device type 3390, the default block size is 28 K. If the database contains segments larger than 28 K, the maximum block size of 32 K is used.
- For device type 9345, the default block size is 22 K. If the database contains segments larger than 22 K, the maximum block size of 32 K is used.
- For device types other than the preceding, the default is the maximum capacity for the device types.

SYSUT3 DD

This optional statement defines a secondary output data set that contains a second copy of the SYSUT2 data set. If SYSUT2 is defined as DUMMY, IMS High Performance Unload ignores this statement. The same rules for BUFNO and BLKSIZE apply as for SYSUT2.

Notes:

The LRECL values for SYSUT2 and SYSUT3 data sets are determined as follows:

- For the unload formats *F1, *F2, and *F3:
  - If the LRECL value prepared in the JCL DD statement is used when the value is specified in either of the following ways:
    - Explicitly by the LRECL subparameter of DCB parameter
    - Implicitly, for example, by using a model data set or by using the DFSMS/MVS DATACLAS parameter.
  - If the LRECL value is not specified either explicitly or implicitly, the default record size (block size minus 4) is used.
- For *HD unload format, the LRECL value prepared in the JCL DD statement is always ignored, and the default record size (block size minus 4) is used.

The large format data set can be specified for SYSUT2 and SYSUT3.

FABHURG1 input

Input for the FABHURG1 utility includes three data sets: SYSIN data set (control statements for the FABHURG1 utility), HSSROPT data set (control statements for HSSR Engine), and HSSRCABP data set (control statements for the buffer handler of HSSR Engine).

FABHURG1 SYSIN input data set

The SYSIN data set for the FABHURG1 utility contains the control statements for the FABHURG1 utility.

Normally, unless a user record-formatting routine or an optional user exit routine is invoked, only two control statements are required for FABHURG1: the PCB control statement and the FRMT control statement. If the defaults are taken, neither of these control statements is required.

Other control statements for general use are:

- CHECKREC
- DEC
- FALLBACK
- MIGRATE
- PARTITION
- SEGSTAT

When the IMS management of ACBs is enabled, the FABHURG1 utility retrieves the DBDs from the IMS catalog directory data sets.

To obtain the pending database from the IMS catalog staging data set before the database is activated, specify the STAGING control statement in the SYSIN data set. If you specify the STAGING control statement, you must execute the FABHURG1 utility in a ULU region.

Tip: FABHURG1 provides a user exit interface for system programmers. System programmers can provide their own record-formatting routine to unload the database into their own format rather than one of the standard formats. For the SYSIN control statement intended mainly for system programmers to use, see “User record-formatting routine” on page 246.

Format

This data set contains 80-byte fixed-length records. The control statements can be coded in the input stream or accessed as members of a partitioned data set.

CHECKREC control statement

This optional control statement specifies whether the particular record is written just behind the header record in the unload data set.

This record is used by IMS High Performance Load for it to know certain status of the database at unloading time. This control statement is used only for the *HD unload format.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

CHECKREC ddd
```

Position
Description

1
Code the CHECKREC keyword to identify this statement as a CHECKREC statement.

10
Use one of the following keywords:

YES
The particular record is written.

NO
The particular record is not written. NO is the default.

Tip: The default of this control statement can be changed by replacing the default option table (FABHOPT). Specify the URG1CHKRC=YES parameter on the FABHTOPT macro statement. For details, see Chapter 19, “Site default options,” on page 261.

DEC control statement

The DEC control statement, which activates the decompress option, specifies whether FABHURG1 is to decompress database segments.

```
0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

DEC d
```

Position
Description

1
Code the DEC keyword to activate the decompress option.

4
The 1-character entry *d* specifies whether compressed segments be decompressed by FABHURG1. This entry is required.

Use one of the following keywords:

Y
Compressed segments are decompressed. Y is the default.

N
Compressed segments are not decompressed.

Code N only when you want to have compressed segments in the output data sets. If an unloaded data set is created by specifying this DECN option for a database that contains a compressed segment, the data set is not compatible with the unloaded data set created by the IMS HD Reorganization Unload utility. You cannot reload such an unloaded data set by using the IMS HD Reorganization Reload utility (DFSURGL0), but you can reload it by using the IMS High Performance Load (Load utility or PSSR utility) or the IPR Reload utility.

Notes:

- If there is a segment type for which a Segment Edit/Compression exit routine is specified and the use of a Data Conversion exit is designated, the DECN option is ignored and the process continues with DECY.
- If the DECN option is specified and one or more partitions of PHDAM or PHIDAM are in the HALDB OLR cursor-active status, FABHURG1 ends abnormally.
- Do not code DECN if you want to change the size of the segments.

Tip: The default of this control statement can be changed by replacing the default option table (FABHOPT). Specify the URG1DEC=NO parameter on the FABHTOPT macro statement. For details, see Chapter 19, “Site default options,” on page 261.

FALLBACK control statement

This optional control statement is used to initiate the fallback unload of a partitioned database.

```
0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

FALLBACK
```

Position
Description

1
Code the FALLBACK keyword to instruct FABHURG1 to perform the fallback unload of a partitioned database.

Restrictions:

- This control statement cannot be specified with a control statement of PARTITION or MIGRATE.
- If a FALLBACK control statement is specified, FABHURG1 must be executed in the ULU region.
- The input database must be either a PHIDAM or a PHDAM.

FRMT control statement

This optional control statement specifies the output format of the unloaded database. Use this statement only when specifying a format other than *HD.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

FRMT frmt
```

Position
Description

1
Code the FRMT keyword to identify the format control statement.

6
Code the output format name, which can be either of the following names:

- The name of one of the standard formats provided by the utility: *HD, *CS, *CP, *F1, *F2, or *F3. The default is the *HD unload format, for which you do not need this statement.
- The load module name of a user record-formatting routine (see “User record-formatting routine” on page 246).

Restrictions:

- If you specify a control statement such as MIGRATE or FALLBACK, you cannot specify any format other than *HD.
- If *CS is specified and one or more partitions of PHDAM are in the HALDB OLR cursor-active status, FABHURG1 ends abnormally.

MIGRATE control statement

This optional control statement is used to make FABHURG1 perform the migration unload of a nonpartitioned database.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

MIGRATE
```

Position
Description

1
Code the MIGRATE keyword to make FABHURG1 perform the migration unload of a nonpartitioned database.

Restrictions:

- This control statement cannot be specified with a control statement of PARTITION or FALLBACK.
- If MIGRATE control statement is specified, FABHURG1 must be executed in the ULU region.

- The input database must be either a HIDAM or an HDAM database.
- If PTR=H or PTR=HB is defined as the parent segment of virtual logical child, FABHURG1 does not support the migration unload of the database.
- Migration unload of a secondary index or HISAM database is not supported.

Tip: If a logical child is defined in the input database, and you want better performance, code an appropriate number of IMS buffer pools in DFSVSAMP DD of your JCL. For details, see "Basic JCL requirements" on page 30.

PARTITION control statement

When you restrict the partitions to be unloaded, use this optional control statement for a partitioned database.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

PARTITION partnme nnnn
```

Position
Description

1
Code the PARTITION keyword to identify this statement as a PARTITION statement.

11
Code the partition name *partnme* from which you want to start the unloading.

This parameter is a required parameter.

19
This optional parameter *nnnn* specifies the total number of partitions to be unloaded. The number *nnnn* must be left-aligned, and its length must be less than or equal to four.

If this parameter is not specified, the default of 1 is assumed.

Tip: If you want to change the high key of partitions, you can use the IBM provided exit routine FABHKEYX to distribute the unload records to multiple unload files according to the root key values. For the details, see "Migration unload: Exit routine FABHKEYX for distributing unload records" on page 109.

Consideration: When you specify the PARTITION control statement, you must pay attention to the sequence of partitions. If no partition selection exit routine is used, the sequence of partitions is determined from the sequence of high key values. If a partition selection exit routine is used, the sequence of partitions depends on the routine.

Restrictions:

- This control statement cannot be specified with a MIGRATE or FALLBACK control statement.
- If you want to unload all partitions, do not specify the PARTITION control statement.

PCB control statement

This optional control statement is used to select an HSSR PCB used for the database call in FABHURG1.

Use this statement only when you run FABHURG1 in a DLI or DBB region and you want to specify an HSSR PCB other than the first one. If you run FABHURG1 in the ULU region, you do not need to specify the statement.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

PCB pcbnbr     dbdname
```

Position
Description

1
Code the PCB keyword to identify this statement as a PCB statement.

5-14
Code an HSSR-PCB number. Specify 1 for the first HSSR PCB in the PSB. The number specified does not need to contain leading zeros, or it does not need to be aligned.

Note: HSSR-PCB number is the sequence number, within the PSB, that is assigned to each HSSR PCB. If, for example, the first and the third database PCBs are defined as HSSR PCBs, but not the second PCB, then the third database PCB has the HSSR-PCB number of 2.

16-23
Code a DBD name.

The HSSR PCB used for the database call in FABHURG1 is determined by the following rule:

- If no PCB control statement is provided, the first HSSR PCB is used.
- If *dbdname* is blank, the HSSR PCB identified by *pcbnbr* is used.
- If *dbdname* is not blank, the first HSSR PCB that refers to the named DBD is used.
- If no HSSR PCB that matches with the specification is defined, FABHURG1 issues an error message.

SEGSTAT control statement

This optional control statement instructs FABHURG1 to produce the segment statistics report for each partition that is processed.

See "FABHURG1 Segment Statistics report" on page 46 for the segment statistics report produced for a partition.

```
0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

SEGSTAT PART
```

Position
Description

1
Code the SEGSTAT keyword to identify this statement as a SEGSTAT statement.

9
Code the PART keyword to print the partition-wide segment statistics report for each partition that is processed and from which at least one segment is retrieved.

Restriction: In any one of the following cases, the partition-wide statistics are not printed:

- The SEGSTAT statement is specified for a nonpartitioned database.
- The PART keyword is not specified.
- A keyword other than PART is specified as the operand of the SEGSTAT statement.

FABHURG1 HSSROPT input data set

The HSSROPT data set for the FABHURG1 utility contains the control statements for HSSR Engine.

The use of this data set is optional. However, you must code this data set when unloading a corrupted database. The following HSSROPT control statements can help you unload corrupted databases:

- SKERROR causes incorrect HD pointers or HISAM records to be skipped.
- DIAGG generates diagnostic information about the incorrect pointers and records.
- KEYCHECK GG returns a GG status code to FABHURG1 rather than ending abnormally when a sequence error is detected.

For a complete description of the HSSROPT control statements, see Chapter 11, "Options for HSSR Engine," on page 155.

Related concepts

Utility options for unloading corrupted databases
The unload utilities of IMS High Performance Unload provide certain options for unloading corrupted databases.

FABHURG1 HSSRCABP input data set

The HSSRCABP data set for the FABHURG1 utility contains the control statements for the buffer handler of HSSR Engine.

For a complete description of the HSSRCABP control statements, see “HSSRCABP control statements” on page 217. Any of the HSSRCABP options that are appropriate to your task can be used.

FABHURG1 output: SYSPRINT output data set

Output from the FABHURG1 utility includes the FABHURG1 Unload Parameters report and the FABHURG1 Segment Statistics report. These reports are generated in the SYSPRINT data set.

In addition to the reports generated by the FABHURG1 utility, reports and statistics produced by HSSR Engine are written in HSSRSTAT and HSSRTRAC data sets. For the reports that are generated by HSSR Engine, see Chapter 12, “Reports and output from HSSR Engine,” on page 181.

Format

This data set contains 133-byte fixed-length records. When the block size is coded in the JCL, the block size must be a multiple of 133.

FABHURG1 Unload Parameters report

This report contains the parameters that were used by FABHURG1 for this run.

The following figure shows an example of the report.

```
IMS HIGH PERFORMANCE UNLOAD                          "FABHURG1 UNLOAD PARAMETERS"                          PAGE:     1
5655-E06                                          DATE: 06/01/2021  TIME: 01.01.01                   FABHURG1 - V1.R2


SYSIN CONTROL STATEMENTS FOLLOW:
    PARTITION PARTDB1 5
    SEGSTAT PART


UTILITY PARAMETERS USED FOR THIS EXECUTION ARE:
    REGION-TYPE=ULU
    PSBNAME=HDAMPART
    DBDNAME=HDAMPART
    PCB 1
    FRMT *HD
    NO EXIT
    OFFS 0
    ULEN 0
    USEGMAX 0
    DECY
    CHECKREC NO
```

Figure 2. FABHURG1 Unload Parameters report

FABHURG1 Segment Statistics report

This report contains a printed copy of the segment statistics.

The statistics provided for each segment are segment code, level, the number of segments retrieved for that segment type, and the number written to the output data set. If there is a difference between the number retrieved and the number written, that difference is shown. The totals and total errors for the database are provided. Also a message is printed stating that the database unload is complete.

The following figure shows an example of the report.

```
IMS HIGH PERFORMANCE UNLOAD                              "FABHURG1 SEGMENT STATISTICS"                          PAGE:     1
5655-E06                                                DATE: 06/01/2021  TIME: 13.05.34                    FABHURG1 - V1.R2

STATISTICS FOR DB=ORDHDAM
SEGMENT    S-C   LV      RETRIEVED         OUTPUT     DIFFERENCE

ORDER        1    1              6              6              0
ORDART       2    2              4              4              0
DELIVER      3    3              2              2              0
SCHEDULE     4    3              2              2              0
HISTORY      5    3              2              2              0
-------------------------------------------------------------------
*TOTAL                         16             16              0
*TOTAL ERRORS                   0
```

Figure 3. FABHURG1 Segment Statistics report

The following figure shows another example of the Segment Statistics report. This report shows statistics from a HALDB partition when the SEGSTAT PART control statement is specified.

```
IMS HIGH PERFORMANCE UNLOAD                              "FABHURG1 SEGMENT STATISTICS"                          PAGE:     1
5655-E06                                                DATE: 06/01/2021  TIME: 12.34.56                    FABHURG1 - V1.R2


STATISTICS FOR DB=HDAMMDSG; PARTITION=PARTDB1

SEGMENT    S-C   LV      RETRIEVED         OUTPUT     DIFFERENCE

ORDER        1    1              5              5              0
ORDART       2    2              5              5              0
DELIVER      3    3              3              3              0
SCHEDULE     4    3              3              3              0
HISTORY      5    3              5              5              0
-------------------------------------------------------------------
*TOTAL                         21             21              0
```

Figure 4. FABHURG1 Segment Statistics report for HALDB

FABHURG1 JCL examples

Use the following JCL examples to prepare your FABHURG1 JCL.

Subtopics:

- "Example 1: Running FABHURG1 in ULU region" on page 47
- "Example 2: Running FABHURG1 in DLI region" on page 48

For examples for processing a HALDB with the FABHURG1 utility, see Chapter 8, "Methods for processing High Availability Large Databases," on page 97.

Example 1: Running FABHURG1 in ULU region

To create an unloaded data set that can be used as input to the IMS HD Reorganization Reload utility, you can use the following JCL.

```
//    EXEC FABHULU,MBR=FABHURG1,DBD=USERDBD
//HDAM     DD DSN=IMSDB.HDAM,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUT2   DD DSN=IMSDB.HDUNLD,DISP=(,CATLG),UNIT=TAPE,
//            VOL=SER=xxxxxx
```

Figure 5. FABHURG1 JCL for unloading an HDAM database

The unloaded data set is specified by the SYSUT2 DD statement. In this example, the HDAM DD statement specifies the database data set to be unloaded.

In the ULU region, the PCB that is dynamically created by IMS from the specified DBD is always treated as an HSSR PCB. You do not need to code a PCB control statement in the SYSIN data set.

If you want to unload the database without decompressing the compressed segments, code the SYSIN DD as follows:

```
    //SYSIN DD *
    DECN
    /*
```

An unloaded data set created by this method can be reloaded by IMS High Performance Load (Load utility or PSSR utility) or the IPR Reload utility.

Example 2: Running FABHURG1 in DLI region

If you run FABHURG1 in the DLI region, you must code a PCB statement in the SYSIN data set, and you must define the PCB as an HSSR PCB by coding an HSSRPCB or HSSRDBD control statement in the HSSROPT data set. You can use the JCL shown in the following figure.

```
//    EXEC FABHDLI,MBR=FABHURG1,PSB=USERPSB,DBRC=Y
//SYSIN    DD *
PCB            USERDBD
FRMT *F1
/*
//HSSROPT  DD *
HSSRDBD *ALL
/*
//SYSPRINT DD SYSOUT=A
//SYSUT2   DD DSN=IMSDB.HDUNLD,DISP=(,CATLG),UNIT=TAPE,
//            VOL=SER=xxxxxx
```

Figure 6. FABHURG1 JCL specifying SYSIN control statements

In this example, an HSSRDBD statement is used to specify all PCBs that refer to the DBD as HSSR PCBs. The first database PCB that refers to the DBD is used to read the database, because only the DBD name is specified on the PCB control statement.

In this example, it is assumed that the RECON data sets and the database to be unloaded are dynamically allocated.

The unloaded data set is in the specified format because the FRMT control statement is coded to specify the unload format, which in this case, *F1 format.

IMS HD Reorganization Unload JCL for running FABHURG1

You can use a JCL that is written for IMS HD Reorganization Unload (DFSURGU0) to run FABHURG1, by adding the IMS High Performance Unload's SHPSLMD0 library ahead of the IMS RESLIB in the STEPLIB concatenation, and by using the DFSISVI0 exit of IMS.

Subtopics:

- “Preparing the DFSISVI0 exit” on page 48
- “JCL compatibility with IMS HD Reorganization Unload” on page 48
- “Restrictions” on page 49

Preparing the DFSISVI0 exit

To run FABHURG1 by using a JCL that was written for IMS HD Reorganization Unload utility, you must prepare the DFSISVI0 exit of IMS.

Also, you must create a copy of the load module FABHSVI0 with the name DFSISVI0 in a library that is concatenated to STEPLIB DD.

Tip: If you already created DFSISVI0 while preparing IMS High Performance Load, you do not need to copy the FABHSVI0 module again.

JCL compatibility with IMS HD Reorganization Unload

When the DFSISVI0 exit for FABHURG1 is prepared, you can use essentially the same JCL as the one used for the IMS HD Reorganization Unload utility (DFSURGU0) to run FABHURG1.

EXEC
: The parameters ULU and DFSURGU0 must be specified, otherwise the DFSISVI0 exit does not invoke FABHURG1.

STEPLIB DD
Add the SHPSLMD0 library of IMS High Performance Unload to this DD statement ahead of the IMS.SDFSRESL data set. The DFSISVI0 member must be placed in a library.

SYSIN DD
This data set can contain control statements for either DFSURGU0 or FABHURG1, but not both. The following table lists the control statements for DFSURGU0 that can be specified for SYSIN DD. The control statement for HSSR Engine can be specified in the HSSROPT data set.

Table 4. Supported control statements for DFSURGU0

Control statement for DFSURGU0	Interpreted control statement for FABHURG1
PARTITION=*partname* ,NUMBER=*nnn* ,STAT=*ccc*	PARTITION *partname nnn* SEGSTAT PART (for STAT=DET)
MIGRATE=YES	MIGRATE
FALLBACK=YES	FALLBACK
STAGING	STAGING

SYSPRINT DD
This data set contains the statistics reports produced by FABHURG1. The LRECL is 121. The statistics that are produced by HSSR Engine are printed in the HSSRSTAT DD and the HSSRTRAC DD if these DDs are specified.

DFSURGU1 DD
This required statement defines the primary output data set instead of SYSUT2 DD.

DFSURGU2 DD
This optional statement defines the secondary output data set instead of SYSUT3 DD.

***database* DD**
The DD statements for database data sets are optional. If the DD statements are not present for a non-partitioned database, dynamic allocation will be done by using the DFSMDA member if one is present in STEPLIB or IMSDALIB.

Other DD statements are described in *IMS Database Utilities*.

Restrictions

- The FABHURG1 restrictions that are listed in "Restrictions for IMS High Performance Unload" on page 22 apply to this processing.
- Even if JOBLIB or STEPLIB libraries are APF-authorized or HPIO YES is specified in the HSSROPT DD, Media Manager is not used to process VSAM ESDSs and OSAM LDSs.
- If any of the following options are specified, IMS HD Reorganization Unload (DFSURGU0) is forced to run instead of FABHURG1:
  - IMS batch region type other than ULU
  - DFSUCKPT DD for checkpoint function
  - DFSURSRT DD for restart function
  - The control statements for DFSURGU0 other than PARTITION, MIGRATE, FALLBACK, and STAGING.
  - MIGRATE=YES for HISAM databases
  - MIGRATE=YES for secondary indexes
  - MIGRATX=YES
  - FALLBACK=YES for PSINDEX databases
  - Running under IMS Utility Control Facility (DFSUCF00)
- SB (sequential buffering) is not used even if the following specifications are made:

  - DFSCTL DD statement
  - The PCB SB= statement in the PSBGEN
  - DFSSBUX0 (exit routine)
- The following control statements are ignored:
  - COMPAUTH YES in the HSSROPT data set
  - HPIO YES in the HSSROPT data set

Chapter 6. FABHFSU unload utility

The FABHFSU unload utility unloads IMS databases and provides compatibility with FSU II. FABHFSU can produce multiple unloaded output data sets. Each unloaded data set can contain different combinations of segments in different formats.

FABHFSU has two different modes in which the different types of unload operations are run: the standard mode and the Parallel Scan Facility (PSF) mode. In both modes, the FABHFSU runs as an HSSR application program, and makes full use of HSSR Engine.

Tip: To understand the system structure and the data flow in HSSR application jobs, see "IMS High Performance Unload system structure" on page 11.

FABHFSU accepts user requests through the CARDIN data set. HSSROPT options and HSSRCABP options can also be specified as input to the FABHFSU program. A Segment Statistics report is produced in addition to the standard outputs produced by HSSR Engine.

In the following topics, the basic functions of FABHFSU run in the standard mode are explained. For instructions to run FABHFSU in Parallel Scan Facility mode (PSF mode), see Chapter 10, "Parallel Scan Facility of FABHFSU," on page 121.

For information about compatibility with FSU II, see Chapter 33, "Compatibility with FSU II," on page 373.

You can also use FABHFSU to unload a corrupted database. For more information, see Chapter 9, "Utility options for unloading corrupted databases," on page 115.

For the restrictions that apply to FABHFSU jobs, see "Restrictions for IMS High Performance Unload" on page 22.

Topics:

Unloading a database with FABHFSU

To unload a database by using the FABHFSU utility in standard mode, you must select the format of the unload output and modify the basic JCL.

Procedure

To unload a database by using the FABHFSU utility, complete the following steps:

1. Select a region type from ULU and DLI, and prepare the basic JCL.

   For instructions for preparing the basic JCL, see "Preparing the basic JCL" on page 29.
2. Code the additional DD statements to meet the JCL requirements of FABHFSU.

   For additional JCL requirements, see "FABHFSU JCL requirements" on page 54.
3. Code DBD and PSB control statements in CARDIN data set to specify HSSR PCBs used for the database call and to specify the segment sensitivity.

The DD name and record format for each output data set can be specified in a PSB control statement. See "Unload output format supported by FABHFSU" on page 52 and "FABHFSU CARDIN input data set" on page 55.

4. Optional: If necessary, specify other control statements in CARDIN data set.

   See "FABHFSU CARDIN input data set" on page 55.

5. Optional: Code HSSROPT and HSSRCABP control statements to specify the options for HSSR Engine.

   These control statements are optional. For more information, see "FABHFSU HSSROPT input data set" on page 65 and "FABHFSU HSSRCABP input data set" on page 65.

6. Submit the JCL to run the FABHFSU job.
7. Check the output reports and messages.

   In addition to the standard output reports produced by HSSR Engine, several reports are produced in the PRNTOUT data set. To interpret the reports, see "FABHFSU output: PRNTOUT output data set" on page 66.

Related reference

FABHFSU JCL examples
Use the following JCL examples to prepare your FABHFSU JCL.

Unload output format supported by FABHFSU

FABHFSU unloads a database into one of four applicable formats: HS, UL, VB, and VN.

For each format, the output data set must be a sequential data set with RECFM=VB.

Subtopics:

HS format

When the HS format is selected for the unload format, the output data set must be consistent with HSAM requirements. BLKSIZE must be specified on the *ddname1* DD statement. If the maximum segment length is even, the minimum BLKSIZE must be equal to the maximum segment length plus 2. If the maximum segment length is odd, the minimum BLKSIZE must be equal to the maximum segment length plus 3. (For more information about the HSAM format, see *IMS Database Administration*.)

UL format

The UL format is the same as the IMS HD Reorganization Unload format. You can use it to replace the IMS HD Reorganization Unload utility with the faster FABHFSU utility when reorganizing a database. The UL format is acceptable to IMS HD Reorganization Reload utility or a compatible utility. (For more information, see *IMS Database Utilities*.)

Consideration:

If you create an unload data set by specifying the DECN control statement for a database that contains a compressed segment, the data set is not compatible with the unloaded data set that is created by the IMS HD Reorganization Unload utility. For details, see "DEC control statement" on page 59. You cannot reload such an unloaded data set by using IMS HD Reorganization Reload utility (DFSURGL0), but you can reload it by using IMS High Performance Load (Load utility or PSSR utility) or the IPR Reload utility.

Block size can be specified on the *ddname1* DD statement. Minimum BLKSIZE equals the length of the largest segment plus 43 bytes. Maximum BLKSIZE is determined by the output device.

If block size is not specified on the *ddname1* DD statement, FABHFSU uses one of the following block sizes:

- For device type 3380, the default block size is 23 K. If the database contains segments larger than 23 K, the maximum block size, 32 K, is used.
- For device type 3390, the default block size is 28 K. If the database contains segments larger than 28 K, the maximum block size, 32 K, is used.
- For device type 9345, the default block size is 22 K. If the database contains segments larger than 22 K, the maximum block size, 32 K, is used.
- For device types other than the preceding, the default is the maximum device capacity.

If some segments have been dropped, make sure that the resulting database is valid.

VB format

This section describes the VB format, which is a product-sensitive programming interface. See "Programming interface information" on page 518 to understand the restrictions associated with this type of material.

PSPI

This format is to be used for processing by application programs. When you use the VB Format, a variable-length record is written for each retrieved database segment that contains the following data fields:

- Segment code
- Segment level
- Segment data as returned by the HSSR call and as seen by HSSR application programs

```
           FORMAT OF VB RECORDS

STATEMENT            OFFSET    DESCRIPTION
                    DEC  HEX

RVB      DSECT
RVBLENN  DC   H'0'   0    0    RECORD LENGTH FIELD
RVBZZ    DC   H'0'   2    2    RESERVED FOR MVS
RVBSC    DC   X'00'  4    4    SEGMENT CODE
RVBLEV   DC   X'00'  5    5    SEGMENT LEVEL
RVBDATA  EQU  *      6    6    SEGMENT DATA AS RETURNED
                                  BY HSSR
```

Block size is specified on the *ddname1* DD statement. If the maximum segment length is even, the minimum BLKSIZE must be equal to the maximum segment length plus 10. If the maximum segment length is odd, the minimum BLKSIZE must be equal to the maximum segment length plus 11. If block size is not specified on the *ddname1* DD statement, FABHFSU uses one of the following block sizes:

- For device type 3380, the default block size is 23 K. If the database contains segments larger than 23 K, the maximum block size, 32 K, is used.
- For device type 3390, the default block size is 28 K. If the database contains segments larger than 28 K, the maximum block size, 32 K, is used.
- For device type 9345, the default block size is 22 K. If the database contains segments larger than 22 K, the maximum block size, 32 K, is used.
- For device types other than the preceding, the default is the maximum device capacity.

If some segments have been dropped, make sure that the resulting database is valid.

PSPI

VN format

This section describes the VN format, which is a product-sensitive programming interface. See “Programming interface information” on page 518 to understand the restrictions associated with this type of material.

PSPI

This format is to be used for processing by application programs.

The VN format has the same format as the VB format, except the VN format also contains the segment name in addition to the segment code.

```
           FORMAT OF VN RECORDS

STATEMENT            OFFSET    DESCRIPTION
                    DEC  HEX

RVN      DSECT
RVNLEN   DC   H'0'   0    0    RECORD LENGTH FIELD
RVNZZ    DC   H'0'   2    2    ZZ   (RESERVED FOR MVS)
RVNSYM   DC   CL8' ' 4    4    SEGMENT NAME
RVNSC    DC   X'00'  12   C    SEGMENT CODE
RVNLEV   DC   X'00'  13   D    SEGMENT LEVEL
RVNDATA  EQU  *      14   E    SEGMENT DATA AS RETURNED
                                  BY HSSR
```

DCB requirements are the same as the requirements for the VB. If the maximum segment length is even, the minimum BLKSIZE must be equal to the maximum segment length plus 18. If the maximum segment length is odd, the minimum BLKSIZE must be equal to the maximum segment length plus 19.

PSPI

FABHFSU JCL requirements

FABHFSU runs as an HSSR application program and, therefore, must meet the requirements for the basic JCL (FABHX034 JCL). In addition, FABHFSU JCL requires other DD statements.

Prerequisite: See “Basic JCL requirements” on page 30 for the basic (FABHX034) JCL requirements.

The following table summarizes additional JCL requirements for FABHFSU.

Table 5. FABHFSU DD statements

DDNAME	Use	Format	Need
CARDIN	Input	LRECL=80	Optional
PRNTOUT	Output	LRECL=133	Required
ddname1	Output	RECFM=VB	Required

EXEC

This required statement invokes either the FABHULU or FABHDLI procedure (see “Preparing the basic JCL” on page 29). Before using either procedure, modify the JCL to include the correct names of the PSBLIB, DBDLIB. Use one of the following formats:

```
//  EXEC FABHULU,MBR=FABHFSU,DBD=dbdname
//  EXEC FABHDLI,MBR=FABHFSU,PSB=psbname
```

Restriction: FABHFSU cannot be run in the DBB region.

CARDIN DD

This required DD statement defines the input data set that contains the control statements for FABHFSU. For details, see “FABHFSU CARDIN input data set” on page 55.

PRNTOUT DD

This required statement defines the output data set to which FABHFSU writes error messages and segment statistics. For details, see "FABHFSU output: PRNTOUT output data set" on page 66. The data set can be defined as:

```
//PRNTOUT DD SYSOUT=A
```

***ddname1* DD**

This DD statement specifies the primary output data set that contains the unloaded database. One DD statement is required for each PSB control statement (in the CARDIN data set) that specifies an output format other than NO. (*ddname1* must be the name specified in the *ddname1* field of the PSB control statement. See "PSB control statement" on page 61.) You can have up to three of those data sets.

The large format data set can be specified for those data sets.

The data set can reside either on tape or on a direct-access device, or it can be specified as DUMMY.

For an HSAM output data set, specify the BLKSIZE on the DD statement. For the BLKSIZE of other output formats, see "Unload output format supported by FABHFSU" on page 52 and the description of the PSB control statement.

You can specify the number of buffers in the FSUBUFNO= option in the default option table (FABHOPT). For details, see Chapter 19, "Site default options," on page 261. The buffers are obtained above the 16-MB line.

If neither the BUFNO subparameter of the DCB parameter on the DD statement nor the FSUBUFNO= option is given, the number of buffers is determined automatically. The buffers are approximately 1 MB in total size and are above the 16-MB line.

Note: The LRECL values for *ddname1* are determined as follows.

For VB and VN formats, the LRECL value prepared in the JCL DD statement is used when the value is specified either:

- Explicitly by the LRECL subparameter of the DCB parameter
- Implicitly, for example, by using a model data set or the DFSMS/MVS DATACLAS parameter

If the LRECL value is not specified either explicitly or implicitly, the default record size (block size minus 4) is used.

For UL format, the LRECL value prepared in the JCL DD statement is always ignored, and the default record size (block size minus 4) is used.

In addition to these DDs, the standard HSSRTRAC DD statement is required whenever HSSROPT options that produce trace and diagnostic data are specified. It is also required when the pointer bypass option is specified on a DBD control statement in the CARDIN data set (see "DBD control statement" on page 58). This specification causes the DIAGG option to be invoked automatically (see "DIAGG control statement" on page 167).

FABHFSU input

Input for the FABHFSU utility includes three data sets: CARDIN data set (control statements for the FABHFSU utility), HSSROPT data set (control statements for HSSR Engine), and HSSRCABP data set (control statements for the buffer handler of HSSR Engine).

FABHFSU CARDIN input data set

The CARDIN data set for the FABHFSU utility contains the control statements for the FABHFSU utility.

Normally, only three control statements are required for FABHFSU: the DBD control statement, the PSB control statement, and the END control statement.

Other control statements that can be used for FABHFSU in standard mode are:

- BLM and ELM
- DEC
- GOT
- PARTITION
- SEGSTAT

The following table provides brief explanation for each control statement.

Table 6. Control statements for FABHFSU CARDIN data set in standard mode

Control statement	Function
BLM/ELM	Specifies a portion of the database to be read.
DBD	Identifies the database to be processed and specifies some processing options.
DEC	Specifies whether to decompress compressed segments or not.
END	Specifies the end of CARDIN control statements.
GOT	Provides the support for PROCOPT=GOT.
PARTITION	Restricts the partitions to be processed.
PSB	Specifies the segment sensitivity and characteristics of the output data set to be created. Up to three PSB statements can be specified.
SEGSTAT	Specifies to produce the partition-wide segment statistics report.

Format

This data set contains 80-byte fixed-length records. The control statements can be coded in the input stream or accessed as a member of a partitioned data set.

BLM/ELM control statements

The optional BLM and ELM limit control statements define the portion of the database to be read.

For HIDAM or HISAM, the limits are specified as keys. For HDAM, the limits are defined as relative block numbers of the CIs or blocks in the Root Addressable Area.

Restriction: The BLM and ELM control statements cannot be specified for a PHDAM or PHIDAM database. Use the PARTITION control statement instead.

One BLM or one ELM limit control statement can be used. If no limit control statements are provided, the entire database is read. These statements must immediately follow the DBD control statement.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

BLMtlimitval
ELMtllkeyval
```

Position
Description

1

Code the BLM or ELM keyword to identify a limit control statement.

4

Specify the limit value type.

The 1-character entry *t* identifies the limit value type. It indicates the format of the starting or ending limit field.

Specify one of the following keywords:

R
Indicates that the limit field contains the relative block number of the CI or block. This specification is valid only for HDAM.

C
Indicates that the limit field contains keys in character format. This specification is valid only for HIDAM and HISAM.

X
Indicates that the limit field contains keys in hexadecimal display format. This specification is only valid for HIDAM and HISAM.

5-78
Specify the starting limit value or the ending limit value.

This 74-character field is a required field.

For BLM, this field is the beginning limit value. It contains the value at which the scan is to begin.

For ELM, this field is the ending limit value. It contains the value of the last database record to be included in the scan.

If the limit value type is R:
If you specify R for the limit value type, code the relative block number of the CI or block. The value is numeric with leading or trailing blanks; it must fall within the limits of the root addressable area.

- For BLM, the scan begins at the first RAP in the specified CI or block.
- For ELM, the scan ends after the last RAP in the specified CI or block has been processed.

If the limit value type is C:
If you specify C for the limit value type, code the length of the key value in positions 5 and 6 (maximum length is 74). Code the key value starting in position 7. You can specify a generic key by entering a key value length less than the key length of the root segment. FABHFSU pads the key value with X'00's up to the key length for the starting limit and with X'FF's up to the key length for the ending limit.

- For BLM, the scan begins at the first root segment with a key equal to or greater than the specified key value. If no starting limit value is specified, the scan begins at the logical beginning-of-data set.
- For ELM, the scan ends before processing the first root segment with a key greater than the indicated key value. If no ending limit value is specified, the scan proceeds to the logical end-of-data set.

If the limit value type is X:
If you specify X for the limit value type, code the hexadecimal string starting in position 5. Do not specify length. The hexadecimal string must contain an even number of characters (0-F) and be left-aligned in the field. Field length is determined by the first blank encountered in the field. When generic keys are entered, they are padded as described for character keys.

- For BLM, the scan begins at the first root segment with a key equal to or greater than the specified key value. If no starting limit value is specified, the scan begins at the logical beginning-of-data set.
- For ELM, the scan ends before processing the first root segment with a key greater than the indicated key value. If no ending limit value is specified, the scan proceeds to the logical end-of-data set.

Notes:

- If you are using a Data Conversion exit for the database and you want to specify a limit value by specifying a key value (that is, setting a limit value type of C or X) you must specify the key value in the stored form, not in the application form.

- If a secondary index is used to retrieve root segments of HIDAM or HDAM, you can specify the limits by the value of search field of the index that has a limit value type of C or X.

DBD control statement

The DBD control statement identifies the database to be scanned. Only one DBD control statement can be used.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

DBDdbname  index   sednnn  b
```

Position
Description

1
Code the DBD keyword to identify the DBD control statement.

4
Code the DBD name.

Code the name of the DBD that describes the physical database to be scanned. This required 8-character field must be left-aligned with trailing blanks. The DBD must specify ACCESS=HDAM, HIDAM, PHDAM, PHIDAM, HISAM, or SHISAM.

12
Code the index name.

The 8-character entry is optional and valid only for running in the ULU region. This entry specifies the name of index DBD that is either the primary index of the HIDAM database or a secondary index of the HIDAM or HDAM database if you want the root segments to be retrieved in the index sequence. For details, see “Considerations for using a secondary index” on page 27.

20
Code this field to activate the sequence check option.

This 1-character entry *s* determines whether to perform sequence check during the unload processing. Specify one of the following keywords:

Y
Perform sequence checking.

N | blank
Do not perform sequence checking (default).

21
Code this field to activate the sequence error option.

The 1-character entry *e* determines whether the output routines bypass or process sequence errors. Specify one of the following keywords:

A | blank
Accept sequence errors (default). A GX status code is returned.

B
Bypass sequence errors. The segment in error, and all of its children, are skipped. A GG status code is returned.

22
Code this field to activate the sequence error print option.

The 1-character entry *d* determines whether diagnostic information is printed for sequence errors in the HSSRTRAC data set. Specify one of the following keywords:

Y | blank
Print diagnostic data on sequence errors (default).

N
Do not print diagnostic data.

23
Code this field to specify the sequence error threshold.

The 3-digit numeric entry *nnn* indicates the number of sequence errors to be allowed before ending the run (the default value is 10). Any number up to 999, with leading or trailing blanks, can be used. If you use the number 999, sequence errors do not cause the run to end.

28
Code this field to activate the pointer bypass option.

The 1-character entry *b* is used to activate the pointer bypass option. The pointer bypass option allows FABHFSU to continue processing a database that contains bad pointers, instead of issuing an abend. The pointer bypass option automatically activates the DIAGG option.

Specify one of the following keywords:

Blank
The pointer bypass option is inactive.

1
This entry invokes the pointer bypass option.

2
This entry forces FABHFSU to use the index, rather than the root twin forward pointers, to unload an HIDAM or PHIDAM database.

See Chapter 9, “Utility options for unloading corrupted databases,” on page 115, for instructions and considerations for using the pointer bypass option.

DEC control statement

The DEC control statement, which activates the decompress option, specifies whether FABHFSU is to decompress database segments.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

DEC d
```

Position
Description

1
Code the DEC keyword to activate the decompress option.

4
The 1-character entry *d* specifies whether compressed segments are decompressed by FABHFSU. This entry is required.

Use one of the following keywords:

Y
Compressed segments are decompressed. Y is the default.

N
Compressed segments are not decompressed.

Code N only when you want to have compressed segments in the output data sets.

If an unloaded data set has been created by specifying this option for a database that contains a compressed segment, that data set is not compatible with the unloaded data set that is created by the IMS HD Reorganization Unload utility. You cannot reload such an unloaded data set by using the IMS HD Reorganization Reload utility (DFSURGL0), but you can reload it by using IMS High Performance Load (Load utility or PSSR utility) or the IPR Reload utility.

Notes:

- If there is a segment type for which a Segment Edit/Compression exit routine is specified and the use of a Data Conversion exit is designated, the DECN option is ignored and the process continues with DECY.
- If the DECN option is specified and one or more partitions of PHDAM or PHIDAM are in the HALDB OLR cursor-active status, FABHFSU ends abnormally.
- Do not code DECN if you want to change the size of the segments.

Tip: The default of this control statement can be changed by replacing the default option table (FABHOPT). Specify the FSUDEC=NO parameter on the FABHTOPT macro statement. For details, see Chapter 19, “Site default options,” on page 261.

END control statement

The END control statement specifies the end of the CARDIN control statements.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

END
```

Position
Description

1
Code the END keyword to identify the END statement as the last statement of the CARDIN data set.

GOT control statement

The optional GOT control statement is provided to request that the same function provided by HSSR Engine for PCBs with PROCOPT=GOT be activated for the PCB even if PROCOPT=GOT is not specified for the PCB used in the FABHFSU job.

If the GOT control statement is specified, FABHFSU ignores the PROCOPT of the PCB statement in PSBGEN and forces PROCOPT=GOT to be used.

The GOT control statement is effective only when DBRC is inactive for the FABHFSU job.

Note: For more information about the PROCOPT=GOT support, see “Support for processing options GON and GOT” on page 91.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

GOT
```

Position
Description

1
Code the GOT keyword to activate the support for PROCOPT=GOT.

PARTITION control statement

Use this optional control statement for HALDB to restrict the partitions to be processed.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

PARTITION partnme nnnn
```

Position
Description

1
Code the PARTITION keyword to identify the PARTITION statement.

11

Code a partition name *partnme* from which you want to start the unloading.

This parameter is a required parameter.

19

This optional parameter *nnnn* specifies the total number of partitions to be unloaded. The number *nnnn* must be left-aligned, and its length must be less than or equal to four.

If this parameter is not specified, the default of 1 is assumed.

Consideration: If no partition selection exit routine is used, the sequence of partitions is determined from the sequence of high key values. If a partition selection exit routine is used, the sequence of partitions depends on the routine.

PSB control statement

The PSB control statement for the standard mode identifies the characteristics of the output data sets to be created. From one to three PSB statements can be used for each execution of FABHFSU.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

PSBpsbname ddname1 p#ofuserxit mkxsc    extlnl
```

Position

Description

1

Code the PSB keyword to identify the PSB control statement.

4-11

Code the PSB name.

The 8-character field contains either an asterisk (*) or the name of the PSB that is coded on the EXEC statement of the JCL.

***

An asterisk (*) on this field indicates that all segments that are described in the DBD are to be processed for this output data set.

When running FABHFSU in ULU region, specify this value.

PSB_name

If the name of the PSB that is coded on the EXEC statement is specified, it indicates that segment types passed to the output routines are controlled through the segment sensitivity of the PCB specified in positions 20 - 21. The name of the PSB must match the name that is supplied in the PARM field of the EXEC statement. The name must be left-aligned with trailing blanks.

Coding the PSB name is valid only for FABHFSU running in the DLI region.

12-19

Code the output DD name.

This 8-character entry specifies the name of the DD statement that defines the data set to be created. An output DD name is required unless the output format specification in positions 22 – 23 is NO. *ddname1* must be left-aligned with trailing blanks.

20-21

Code the PCB number.

The 2-character entry *p#* specifies which database PCB within the PSB is to be used. This entry determines the segments written into the output data set that is identified by the output DD name field (column 12 - 19). The PCB referred to must specify the *dbdname* that is specified in positions 4 – 11 of the DBD control statement.

If the field is left blank, the first database PCB that refers to the DBD specified by the DBD control statement is used.

The number *p#*, if specified, must be the sequence number that starts from the first database PCB in the PSB; TP PCBs are not counted. Specify 1 for the first database PCB in the PSB.

Note: To assure compatibility with FABHFSU of DBT HSSR, it is processed as if 1 was specified when one of values '00', ' 0 ', or '0 ' is specified.

22-23
Code the output format.

The 2-character entry *of* specifies the output format of the data set to be created. Specify one of the following keywords:

HS
HSAM format

UL
IMS HD Reorganization Unload format

MI
IMS HD Reorganization Unload format for migration unload from an HDAM or HIDAM database to a PHDAM or PHIDAM database.

Restrictions:

- This format can be specified only in a ULU region.
- If PTR=H or PTR=HB is defined as the parent segment of virtual logical child, the database is not supported.
- Migration unload of a secondary index or HISAM database is not supported.
- When MI format is selected, two or more PSB control statements cannot be specified.

VB
Variable-length-blocked data set of the selected segments (one segment per logical record)

VN
The format is similar to VB except that the segment name is included in addition to the segment code.

NO
No output data set is created by FABHFSU. The output function can be handled in an exit routine.

For more information about these output formats, see "Unload output format supported by FABHFSU" on page 52.

24
Code the exit routine name.

The 8-character entry specifies the name of a user exit routine. The name must be left-aligned with trailing blanks. The load module must be in a STEPLIB library. For more information, see "FABHFSU user exit routine" on page 69.

Note: If a user exit routine is specified and one or more partitions are in the HALDB OLR cursor-active status, FABHFSU ends abnormally.

32
Code this field to activate the segment modification option.

The 1-character entry *m* indicates whether segments are to be modified by the user exit routine.

Specify one of the following keywords:

Y
Indicates that segments are to be modified by the user exit.

This option does not support a change of the database segment length. If you change the segment length with the Y option, the result is unpredictable. For details, see “Modifying segments in user exits” on page 71.

E
Indicates that segments are to be modified by the user exit.

This option supports a change of the database segment length. The option is valid only for HDAM, HIDAM, PHDAM, and PHIDAM databases.

An extra 100-byte field is added at the end of the segment data that is passed to the exit routine. This extra field can be used for segment extension. If the default length of this extra field is shorter than you require, you can change the length of the extra field by specifying the length in column 41 of the same PSB statement.

If this option has been selected, any request to activate the compare option used for problem determination is deactivated.

For details, see “Modifying segments in user exits” on page 71.

N | blank
Indicates that segments are not to be modified by the user exit (default).

33
Code this field to activate the concatenated key option.

The 1-character entry *k* indicates whether the fully concatenated key of each segment is to be built and passed to the exit routine.

Specify one of the following keywords:

Y
Build concatenated key.

N | blank
Do not build concatenated key (default).

34
Code this field to activate the exit routine control option.

The 1-character entry *x* indicates whether the user exit routine is given control before and after segments are processed (see “FABHFSU user exit routine” on page 69).

Specify one of the following keywords:

Y
Allow exit routine control.

N | blank
Do not allow exit routine control (default).

35
Code this field to activate the DBR skip option.

The 1-character entry *s*, the Database Record (DBR) skip option, indicates whether return code 12 or 16 is valid for the exit routine specified in columns 24 - 31 of this statement. A return code 12, if valid, causes FABHFSU to skip the remaining segments associated with the current root segment and begin processing at the next root segment.

Return code 16 causes a skip to a new root key value specified by the exit routine.

Use this control statement only when a single PSB control statement is defined. The skipping invoked by one PSB statement affects all others included in the same run.

For more information about return codes from the user exit routines, see “Contents of registers on exit” on page 75.

Specify one of the following codes:

Y
Allow DBR skip option.

N | blank
Do not allow DBR skip option (default).

36
Code this field to activate the data conversion option.

If a Data Conversion exit (DFSDBUX1 exit) is activated, the user exit routine specified by the exit routine name option in column 24 receives the segment data that has been converted from the stored form to the application form.

The 1-character entry *c* indicates whether the inverse conversions, that is, the conversion from the application form to the stored form, is done before the segment data edited in the exit routine is written into the output data set.

Specify one of the following keywords:

Y
Perform the conversion.

This option is valid only for UL and HS unload format, and is valid only when the option 'DATXEXIT YES' is specified in the HSSROPT data set.

N | blank
Do not perform the conversion. This keyword is the default.

37-40
The value specified in this 4-byte field is always ignored.

41-45
Code the extension length.

The 5-digit numeric entry *extln* specifies how many extra bytes are to be reserved for the segment extension in the exit routine. The length specified on this field, plus the maximum length of the segments in the database, will be used as the length of the work area for editing segments in the exit routine. This field is valid only when the option 'E' is specified in column 32 of the PSB statement. If the option 'E' is specified and this field is blank, the default value, 100, is used.

You can specify a value in the range of 00000 - 32767.

Notes:

- If the resulting length of the segment editing work area is more than 32,767 bytes long, 32,767 is used as the length of the work area.
- If option 'E' is not specified in column 32, the value specified on this field is ignored.

46
Language environment option

L
Indicates that the user exit routine runs in the Language Environment (LE) using the CEEPIPI invocation.

This option is effective when the user exit routine is written with Enterprise COBOL for z/OS. This option is not effective for user exit routines written in assembler or PL/I language.

This option is mutually exclusive with the RTEXIT control statement. If you specify this option, the runtime environment exit routine specified for the RTEXIT control statement is not invoked.

Example

The following figure shows an example of the PSB control statement for the FABHFSU program that runs in the ULU region and invokes the user exit routine:

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
PSB*       OUT1      ULMYCOBEXTY YY          L
```

- 'MYCOBEXT' in columns 24 - 31 is the name of the user exit routine.
- 'Y' in column 32 activates the segment modification option.
- 'Y' in column 34 activates the exit routine control option.
- 'Y' in column 35 activates the DBR skip option.
- 'L' in column 46 activates the language environment option.

SEGSTAT control statement

This optional control statement requests FABHFSU to create a Segment Statistical report for each partition that is processed.

See "FABHFSU Segment Statistics report" on page 67 for an example of the Segment Statistics report that is produced for a partition.

```
0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

SEGSTAT PART
```

Position
Description

1
Code the SEGSTAT keyword to identify the statement.

9
Code the PART keyword to print the partition-wide segment statistics report for each partition that is processed and from which at least one segment is retrieved.

Restriction: In any one of the following cases, the partition-wide statistics are not printed:

- The SEGSTAT statement is specified for a nonpartitioned database.
- The PART keyword is not specified.
- A keyword other than PART is specified as the operand of the SEGSTAT statement.

FABHFSU HSSROPT input data set

The HSSROPT data set for the FABHFSU utility contains the control statements for HSSR Engine.

For a complete description of the HSSROPT control statements, see Chapter 11, "Options for HSSR Engine," on page 155.

You can use any HSSROPT options that are appropriate for your task. When FABHFSU has the pointer bypass option specified on the DBD control statement, do not specify the SKERROR and DIAGG options. These options are automatically activated by FABHFSU. DIAGG output requires an HSSRTRAC DD statement.

FABHFSU HSSRCABP input data set

The HSSRCABP data set for the FABHFSU utility contains the control statements for the buffer handler of HSSR Engine.

For a complete description of the HSSRCABP control statements, see "HSSRCABP control statements" on page 217. Any of the HSSRCABP options that are appropriate to your task can be used.

FABHFSU output: PRNTOUT output data set

Output from the FABHFSU utility includes the FABHFSU Control Statement report, the FABHFSU Control Specification report, and the FABHFSU Segment Statistics report. These reports are generated in the PRNTOUT data set.

In addition to the reports generated by the FABHFSU utility, reports and statistics produced by HSSR Engine are written in HSSRSTAT and HSSRTRAC data sets. For the reports generated by HSSR Engine, see Chapter 12, “Reports and output from HSSR Engine,” on page 181.

Format

This data set contains 133-byte fixed-length records. When the block size is coded in the JCL, the block size must be a multiple of 133.

FABHFSU Control Statements report

This report contains the CARDIN control statements that were used as input in the FABHFSU job.

The following figure shows an example of the report.

```
IMS HIGH PERFORMANCE UNLOAD                              "FABHFSU CONTROL STATEMENTS"                              PAGE:     1
5655-E06                                               DATE: 06/01/2021  TIME: 12.13.14                       FABHB15  - V1.R2


0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

DBDHSHDP
PSBHSHDPGB OUT1      UL
PSBHSHDPGB OUT2      UL
END
```

Figure 7. FABHFSU Control Statements report

FABHFSU Control Specifications report

This report contains information about the parameters that were specified on the CARDIN control statements. This report is produced for each output data set that is defined by the PSB control statement in the CARDIN data set.

The following figure shows an example of the report.

```
IMS HIGH PERFORMANCE UNLOAD                          "FABHFSU CONTROL SPECIFICATIONS"                          PAGE:     1
5655-E06                                             DATE: 06/01/2021  TIME: 12.59.59                  FABHFSU  - V1.R2


**** SCAN CONTROL SPECIFICATIONS ****               **** FORMAT CONTROL SPECIFICATIONS ****
PARALLEL SCAN NAME              N.A                 FORMAT (CONTROL PSB) NUMBER     01
BASE DBD NAME                   HDAMMDSG            CONTROL PSB NAME                *
INDEX DBD NAME                                      CONTROL PCB NUMBER
SEQUENCE CHECK OPTION           N.A                 OUTPUT DDNAME                   SYSUT2
SEQUENCE ERROR OPTION           N.A                 OUTPUT FORMAT                   UL
SEQUENCE ERROR PRINT OPTION     N.A                 EXIT ROUTINE NAME               N.A
SEQUENCE ERROR THRESHOLD        N.A                 SEGMENT MODIFICATION OPTION     N.A
SEQUENCE ERROR ABEND OPTION     N.A                 CONCATENATED KEY OPTION         N.A
BEGINN LIMIT CONTROL            N.A                 EXIT OPEN/CLOSE CONTROL         N.A
END LIMIT CONTROL               N.A                 DBR SKIP OPTION                 N.A
POINTER BYPASS OPTION           N.A                 DATA CONVERSION OPTION          N.A
NO. PARALLEL SCAN               N.A                 FSU FILE TO BE COMPARED         N.A
DSN CHECK OPTION                N.A                 EXIT LE OPTION                  N.A
PSC WTO OPTION                  N.A

                                                     **** PARTITIONED DB OPTIONS ****
CONCATENATE SEGMENT AND PREFIX  N.A                 PARTITION SEGSTAT               YES
DECOMPRESS SEGMENTS             YES                 STARTING PARTITION              PART1
PROCOPT=GOT SUPPORT             N.A                 NUMBER OF PARTITIONS            0005
```

Figure 8. FABHFSU Control Specifications report

The content of the report is as follows:

SCAN CONTROL SPECIFICATIONS
Shows specifications or defaults from the DBD, BLM, and ELM control statements.

FORMAT CONTROL SPECIFICATIONS
Shows specifications or defaults from the PSB control statement.

FSU FILE TO BE COMPARED
Shows specification or default from the CO control statement.

See “FABHFSU control statements: CO and CON” on page 361.

CONCATENATE SEGMENT AND PREFIX
Shows specification or default from the CON control statement.

See “FABHFSU control statements: CO and CON” on page 361.

DECOMPRESS SEGMENTS
Shows specification or default from the DEC control statement.

PROCOPT=GOT SUPPORT
Shows specification or default from the GOT control statement.

PARTITIONED DB OPTIONS
Shows specification or default from the SEGSTAT and the PARTITION control statements for HALDB.

FABHFSU Segment Statistics report

This report provides statistics for each sensitive segment in the database. This report is produced for each output data set defined by the PSB control statement in the CARDIN data set.

The following figure shows an example of the report.

```
IMS HIGH PERFORMANCE UNLOAD                              "FABHFSU SEGMENT STATISTICS"                              PAGE:     1
5655-E06                                                DATE: 06/01/2021  TIME: 18.32.56                        FABHFSU  - V1.R2

**** FORMAT 01 OUTPUT STATISTICS FOR DB=DBHD1010 ****
PSB         SEGMENT  SEGMENT         TOTAL         TOTAL      SEQUENCE       MAXIMUM       MAXIMUM       AVERAGE       AVERAGE
NAME        NAME       LEVEL     RETRIEVED        OUTPUT        ERRORS         TWINS      CHILDREN         TWINS      CHILDREN
PDBHD101    ROOTLEV1       1            10            10             0           N.A           N.A          1.00          6.10
            DEP1LEV2       2            10            10             0           N.A           N.A          1.00           .00
            DEP2LEV2       2            11            11             0           N.A           N.A          1.10           .81
            DEP3LEV3       3             9             9             0           N.A           N.A           .81           .00
            DEP4LEV2       2            11            11             0           N.A           N.A          1.10          1.81
            DEP5LEV3       3             9             9             0           N.A           N.A           .81           .00
            DEP6LEV3       3            11            11             0           N.A           N.A          1.00           .00
          TOTAL RETRIEVED                  71
          TOTAL OUTPUT                     71
          TOTAL SEQUENCE ERRORS             0
```

Figure 9. FABHFSU Segment Statistics report

If the database is a HALDB, and the SEGSTAT PART control statement is specified in the CARDIN data set, the partition-wide segment statistics report that is shown in the following figure is also produced for each partition.

```
IMS HIGH PERFORMANCE UNLOAD                              "FABHFSU SEGMENT STATISTICS"                              PAGE:     1
5655-E06                                                DATE: 06/01/2021  TIME: 18.33.14                        FABHFSU  - V1.R2

**** FORMAT 01 OUTPUT STATISTICS FOR DB=PHDO0300, PARTITION=PHDO03A ****
PSB         SEGMENT  SEGMENT         TOTAL         TOTAL      SEQUENCE       MAXIMUM       MAXIMUM       AVERAGE       AVERAGE
NAME        NAME       LEVEL     RETRIEVED        OUTPUT        ERRORS         TWINS      CHILDREN         TWINS      CHILDREN
PHDO030A    ROOTLEV1       1             7             7             0           N.A           N.A          1.00         27.00
            DEP1LEV2       2             7             7             0           N.A           N.A          1.00           .00
            DEP2LEV2       2             7             7             0           N.A           N.A          1.00          6.00
            DEP3LEV3       3            14            14             0           N.A           N.A          2.00          2.00
            DEP4LEV4       4            28            28             0           N.A           N.A          2.00           .00
            DEP5LEV2       2             7             7             0           N.A           N.A          1.00         18.00
            DEP6LEV3       3            14            14             0           N.A           N.A          2.00          8.00
            DEP7LEV4       4            28            28             0           N.A           N.A          2.00           .00
            DEP8LEV4       4            28            28             0           N.A           N.A          2.00          2.00
            DEP9LEV5       5            56            56             0           N.A           N.A          2.00           .00
          TOTAL RETRIEVED                 196
          TOTAL OUTPUT                    196
          TOTAL SEQUENCE ERRORS             0
```

Figure 10. FABHFSU Segment Statistics report (Partition-wide)

The content of the report is as follows:

PSB NAME
Name of PSB

SEGMENT NAME
Name of segment

SEGMENT LEVEL
Level of segment

TOTAL RETRIEVED
The number of each segment type retrieved. This count does not include segments bypassed due to sequence errors.

TOTAL OUTPUT
The number of each segment type processed by FABHFSU. If the output format is specified as NO, the value is zero. Differences between TOTAL RETRIEVED and TOTAL OUTPUT represent the segments that were bypassed by the user exit routine.

SEQUENCE ERRORS
The number of sequence errors detected for this segment type. If the sequence check option is N, this field is zero.

MAXIMUM TWINS
The maximum number of this segment type that occurs under any one root segment. This field is always N.A. (that is, not applicable).

MAXIMUM CHILDREN
The maximum number of dependent children that occurs for this segment type. This field is always N.A. (that is, not applicable).

AVERAGE TWINS
The TOTAL RETRIEVED of this segment type divided by TOTAL RETRIEVED of this segment's parent. Average occurrences of this segment per parent occurrence.

AVERAGE CHILDREN
The sum of all segment occurrences dependent on this segment type divided by the TOTAL RETRIEVED of this segment type. (This value might be incorrect if sequence errors are bypassed or if the PCB is not sensitive to all dependents.)

TOTAL RETRIEVED
The total number of segments retrieved.

TOTAL OUTPUT
The total number of segments processed by FABHFSU.

TOTAL SEQUENCE ERRORS
The total number of sequence errors.

FABHFSU user exit routine

The user exit routine enables you to perform additional selectivity or modification of segments before the segment data is written into the output data set. You can also control FABHFSU using various return codes from your exit routine.

The following topics includes product-sensitive programming interface information. See "Programming interface information" on page 518 to understand the restrictions associated with this type of material.

PSPI

Note: The user exit interface is compatible with the user exit interface of FABHFSU in DBT HSSR and IPR Unload. You can use the exit routine that you wrote for these utilities if that routine does not depend on the CON option, that is, if the routine does not expect that the segment prefix and the segment data are concatenated in contiguous virtual storage.

The routine must be link-edited in 31-bit addressing mode (AMODE 31). If the routine does not refer to the segment prefix area that is pointed to by the first parameter, you can link-edit the routine in 24-bit addressing mode (AMODE 24).

FABHFSU calls the user exit routine as an Assembler subroutine. The linkage convention follows the MVS standard. Exit routines must be placed into a load module library to which access is provided with a STEPLIB JCL statement.

At initialization of FABHFSU, and before any HSSR call is issued to the database to be unloaded, FABHFSU examines each PSB control statement to determine whether an exit routine has been specified. If one has been, it is loaded from its resident library.

The exit routine is called for each segment retrieved from the database before the segment record is written into the output data set.

If your exit routine needs some kind of initialization and termination processing, specify Y for the exit routine control option on the PSB control statement. For more information, see "Initialization and termination processing in the exit routine" on page 72.

If you want to modify the content of segments, specify the option Y or E for the segment modification option of the PSB control statement. For more information, see "Modifying segments in user exits" on page 71.

For best performance, coding the exit routines in Assembler language is recommended. However, exit routines in COBOL and PL/I exit routines are also supported for application programmers.

PSPI

Considerations when writing user exit routines

Certain considerations apply when writing user exit routines.

Subtopics:

PSPI

Writing a user exit routine in COBOL

When you write an exit routine in COBOL, you should know that a COBOL exit is always called as a subroutine of FABHFSU. Because FABHFSU is not a COBOL program and is not running in the Language Environment, you must use some methods to get better performance.

Enterprise COBOL, COBOL for OS/390, COBOL for MVS, and COBOL/370

If your exit routine is compiled by Enterprise COBOL, COBOL for OS/390 and VM Version 2, COBOL for MVS and VM, or COBOL/370, you must use the runtime option module CEEUOPT to optimize the Language-Environment (LE) options.

Because FABHFSU is not an LE-conforming Assembler program, RTEREUS=(ON) must be specified in CEEUOPT. TRAP=(OFF) option must also be specified, to avoid the overhead of intercepting abends in the COBOL user exit.

VS COBOL II

If your exit routine was compiled by VS COBOL II or OS/VS COBOL and run with VS COBOL II runtime library, use one of the following methods:

- Generate the runtime option module IGZEOPT with RTEREUS=YES and STAE=NO; and link-edit it with the exit routine if that routine was compiled with VS COBOL II.

  If the routine was compiled with OS/VS COBOL, link-edit the IGZEOPT module with a copy of ILBONTR from the VS COBOL II subroutine compatibility library. At run time, this copy of ILBONTR must be available.
- Specify the control statement RTEXIT=HPSUCOB2 in the HSSROPT data set. HPSUCOB2 sets up and terminates the COBOL II runtime environment. This method is useful if RTEREUS=NO is specified in your current IGZEOPT module that is linked with the exit routine.

Writing a user exit routine in PL/I

If an exit routine is written in PL/I, a special Assembler interface routine must be linked to it. For details of the interface routine, see the sample HPSUXPA0 in the HPS.SHPSSAMP sample library.

When you use the interface routine, observe the following rules:

- The PL/I routine must be compiled as an external procedure (no OPTIONS on the procedure statement).
- The name on the procedure statement must be FSUEXIT.
- The parameters in the PL/I routine must be declared as pointer variables.
- Based variables must be used to access the data passed by FABHFSU to the exit routine.
- The built-in function PLIRETC must be used to set the return code expected by FABHFSU.
- The interface routine must be link-edited with the exit routine by use of the following link-edit control statement:

```
INCLUDE exitlib(FSUEXIT)   FSUEXIT - NORMALLY ON SYSLIN
INCLUDE somelib(FSUTOPLI)  INTERFACE ROUTINE
ENTRY FSUTOPLI             ENTRY POINT IN INTERFACE
NAME whatever              ANY NAME YOU CHOOSE
```

- The module name indicated as 'whatever' in this example is the name specified on the Exit Routine Name field of the PSB control statement.

Sample exit routines

The sample exit routines shown in the following table are provided as members in the HPS.SHPSSAMP sample library:

Table 7. Examples of exit routines

Member name	Description
HPSUXAA0	Sample exit routine written in Assembler language
HPSUXCA0	Sample exit routine written in COBOL
HPSUXPA0	Sample exit routine written in PL/I

PSPI

Modifying segments in user exits

If you want to modify the content of segments, activate the segment modification option by making the specification on the PSB control statement.

Procedure

If you want to modify the content of segments, specify the option Y or E for the segment modification option of the PSB control statement.

PSPI

Y

If you want to modify database segments in your exit routine, but you do not change the length of a segment, specify Y in column 32 of the PSB control statement.

If this option is selected, each segment will be moved to a work area so that modifications made by an exit routine will not affect the same segment when it is being written to another data set, defined by another PSB control statement. Though any part of the data (except the size field of a variable length segment) can be changed by the user exit, you are cautioned against modifying the segment prefix or sequence fields.

The Y option does not allow the exit routine to change the segment length. If you change the segment length with the Y option, the result is unpredictable. If you want to change the segment length, you must specify the E option instead of the Y option.

E

The E option notifies FABHFSU that segments will be modified by the user exit. The option is valid only for HDAM, HIDAM, PHDAM, or PHIDAM databases. If the E option is specified on a PSB control statement, an extra 100-byte field is added at the end of the segment data field that is passed to the exit routine. This additional area can be used to extend the segment passed. If you have changed the length of a fixed-length segment, you must update the SGLEN field in the segment table SGTBL. When the segment is a logical child and the logical parent's concatenated key (LPCK) is defined as virtual, the length passed through SGLEN does not include the length of the (virtual) LPCK field. In that case, the length of the virtual LPCK must not be included when SGLEN is updated. Furthermore, if you have changed the length of a variable-length segment, you must update the size field of the segment. However, you do not need to update the SGLEN field for a variable-length segment.

Note: You do not need to regenerate a new DBD before running FABHFSU so that the new segment lengths are reflected to the DBD.

PSPI

Related reference

PSB control statement

The PSB control statement for the standard mode identifies the characteristics of the output data sets to be created. From one to three PSB statements can be used for each execution of FABHFSU.

Initialization and termination processing in the exit routine

If the exit routine control option is coded Y (on the PSB control statement), the exit routine is called before the first segment is processed and again after the last segment is processed.

PSPI

The SGLEVEL field in SGTBL is shown in the following table:

Table 8. The SGLEVEL field in SGTBL

Value of SGLEVEL	Description
X'00'	This is the initialization call. It is issued before any segment processing begins.
X'FF'	This is the termination call. It is issued after the last segment processing.
X'01'–X'0F'	This is the normal segment processing call, and the value in SGLEVEL shows the segment level of the segment just retrieved.

PSPI

Related reference

PSB control statement

The PSB control statement for the standard mode identifies the characteristics of the output data sets to be created. From one to three PSB statements can be used for each execution of FABHFSU.

Information passed to the exit routine

FABHFSU invokes the user exit routine for each segment retrieved from the database before the segment record is written into the output data set.

PSPI

The following parameters are passed to the routine (see “Contents of registers” on page 75):

Parameter
Description

Word 1
Address of the segment prefix for the segment

Word 2
Address of the segment data

Word 3
Address of the segment-table (SGTBL) entry for the retrieved segment

Word 4
Address of the Key Area

You can refer to or edit the segment data by referring to the information in the SGTBL and Key Area. The address of the segment prefix area is passed to the exit routine for compatibility with the user exits for FABHFSU of DBT HSSR and IPR Unload.

Subtopics:

- “Segment prefix area” on page 73
- “Segment table (SGTBL)” on page 73
- “Key area” on page 74

Segment prefix area

The area is always above the 16-MB line; therefore, the address of the segment prefix is set into the parameter list as a 31-bit address.

The exit routine (link-edited in 31-bit addressing mode) can refer to the segment prefix area, but must not modify the data in the area.

If the segment code needs to be checked, the routine must refer to SGCDE in the SGTBL entry instead of referring to the segment prefix.

Segment table (SGTBL)

The segment table (SGTBL) is a control block built by FABHFSU. Each entry of SGTBL contains the information for the segment just retrieved. Your exit routine can refer to the information, but must not modify the data in SGTBL unless the length of a fixed-length segment is changed. See “Modifying segments in user exits” on page 71.

The Assembler mapping macro HPSUSGTB for an SGTBL entry is provided in HPS.SHPSMAC0 macro library.

The fields of interest in an SGTBL are summarized in the following table. For other fields, see HPSUSGTB macro.

Table 9. Fields in an SGTBL

Label	Offset (bytes)	Length (bytes)	Type	Description
SGNAME	0	8	Character	Segment name
SGCDE	8	1	Hex	Segment code
SGPARCDE	9	1	Hex	Parent segment code
SGLEVEL	10	1	Hex	Segment level
SGPCNUM	11	1	Hex	Number of physical children
SGSNSFLG	12	1	Flag byte	Sensitivity
SGPACOP	13	1	Flag byte	Compaction options
SGFDFLG	14	1	Flag byte	Sequence field flag
SGDSG	15	1	Hex	Data set group number
SGVLIND	16	1	Flag byte	Variable-length segment flag If the flag bit SGVL=X'04' is on, the segment has variable length.
SGHIER	17	1	Flag byte	Hierarchical pointer flag
SGMINLEN	18	2	Halfword	Minimum length of variable-length segment
SGFDLEN	20	2	Halfword	Length of sequence field
SGFDDISP	22	2	Halfword	Offset to sequence field

Table 9. Fields in an SGTBL (continued)

Label	Offset (bytes)	Length (bytes)	Type	Description
SGLEN	24	2	Halfword	Segment length for a fixed-length segment; maximum length for a variable-length segment The length does not include the length of the LPCK area if it is defined as VIRTUAL.
SGPFXLEN	26	2	Halfword	Prefix length

Key area

The key area is used for the following purposes:

- If Y is specified for the concatenated key option on the PSB control statement, the fully concatenated key of each segment is built and passed to the exit routine in this area.

  If a secondary index is used, the value in the search field of the index segment is considered as the key of the root segment.

  If the DECN option is specified in CARDIN data set, compressed segments are not decompressed. The concatenated key might not be complete if the segment being processed is key-compressed or has a parent that is key-compressed; FABHFSU considers those segments unsequenced. If you need a completely concatenated key for such a database, you must not specify DECN.

  Note: If you are using a Data Conversion exit (DFSDBUX1 exit) for the database processed in the FABHFSU job, the concatenated key is passed in the application form. If you are not using the DFSDBUX1 exit, the concatenated key is passed in the stored form.
- To skip the scan to a new root key value, the exit routine uses the concatenated key area to pass the new key back to FABHFSU. A generic key can be specified by returning a key length less than that required for a root key.

  If a secondary index is used, specify the value in the search field of the index segment for the next root.

  Set the return code of 16 to force FABHFSU to accept the skip. The DBR skip option is also required to be Y in the PSB control statement.

  The exit routine can skip to any key that is lower than the upper limit for the unload process. A skip to a key beyond the upper limit simulates end-of-the-database.

  Note: If you are using a Data Conversion exit (DFSDBUX1 exit) for the database that is processed in the FABHFSU job, you must specify the new key in the application form. If you are not using the DFSDBUX1 exit, you must specify the new root key in the stored form.

  The format of the key area is shown the following table.

Table 10. Format of the key area

Offset	Length (bytes)	Type	Description
0	2	Halfword	Length of key area
2	variable	-	Key value

PSPI

Contents of registers

The following information describes how to communicate with the main logic.

PSPI

Subtopics:

Contents of registers on entry

The following table shows register contents upon entry to the user exit routine:

Table 11. Register contents upon entry to an exit routine

Register	Content
1	Pointer to the parameter list
13	Pointer to the register save area
14	Return address
15	Entry-point address of the routine

The following table lists the parameters which are pointed to by register 1 upon entry.

Table 12. Parameter list pointed to by register 1 upon entry

Word	Content
1	Address of segment prefix area
2	Address of segment data (see Note)
3	Address of segment table (SGTBL) entry for the segment type
4	Address of key area

Note: The second word (segment-data pointer) points to the beginning of the data portion of the segment. If the segment is a variable-length segment, the pointer points to the size field. If the segment is a fixed-length segment, the pointer points to the beginning of the data, and the segment length is available from the SGLEN field of the SGTBL entry.

Contents of registers on exit

When the control is returned to the caller, the contents of all registers except for Register 15 must be restored. Register 15 must contain one of the return codes that are summarized in the following table.

Table 13. Return codes

Code	Meaning
0	FABHFSU writes this segment in the output data set.
4	FABHFSU does not write this segment in the output data set.
8	FABHFSU bypasses this segment and stops the processing for this output data set. If the processing for all output data sets is stopped, FABHFSU ends the job step.

Table 13. Return codes (continued)

Code	Meaning
12	FABHFSU bypasses this segment and skips the unload processing to the next root segment. **Note:** The DBR skip option must be coded as Y on the PSB control statement. See the description of the DBR skip option in "PSB control statement" on page 61.
16	FABHFSU bypasses this segment and skips the unload processing to the root segment with the sequence key returned in the Key Area. If a Data Conversion (DFSDBUX1) exit routine is used for the database, the key of the next root must be specified in the application form. **Note:** The DBR skip option must be coded as Y on the PSB control statement. See the description of the DBR skip option in "PSB control statement" on page 61.
256+ (any one of the preceding codes)	The segment work area contains a decompressed fixed-length segment that is now in a true fixed format. The length of the fixed-length segment is in the SGLEN field of the SGTBL. If this code is returned, FABHFSU gets the length of the decompressed fixed-length segment from the SGLEN field. These return codes are allowed only for fixed-length segments.

If a code other than the codes summarized in the preceding table is returned, FABHFSU ends abnormally.

PSPI

FABHFSU JCL examples

Use the following JCL examples to prepare your FABHFSU JCL.

Subtopics:

For examples of unloading a PHDAM or PHIDAM database, see Chapter 8, "Methods for processing High Availability Large Databases," on page 97.

Example 1: Running FABHFSU in ULU region

The FABHULU procedure can be used to run FABHFSU in the ULU region. The DBD name must be specified on the DBD= EXEC parameter and on the DBD control statement in CARDIN DD. The DBD must be a physical DBD.

Because the job runs in the ULU region, PSB name '*' is specified for PSB control statements. In this example, two PSB statements are coded.

The first PSB statement specifies the UL format and the DD name of OUT1 for the output data set. The output data set produced by the PSB statement can be reloaded by the IMS HD Reorganization Reload utility or a compatible utility.

The second PSB statement specifies the VN format and the DD name of OUT2 for the output data set. The output data set is intended to be processed by an application program.

In the following example, SKILHDAM DD specifies the database to be unloaded.

```
//    EXEC  FABHULU,MBR=FABHFSU,DBD=SKILLINV
//CARDIN    DD  *
DBDSKILLINV
PSB*       OUT1      UL
PSB*       OUT2      VN
END
/*
//PRNTOUT   DD  SYSOUT=A
//SKILHDAM  DD  DSN=IMSDB.SKILLINV.DB,DISP=SHR
//OUT1      DD  DSN=IMSDB.UNLOAD1,DISP=(,KEEP),UNIT=TAPE
//OUT2      DD  DSN=IMSDB.UNLOAD2,DISP=(,KEEP),UNIT=TAPE
```

Figure 11. Running FABHFSU in the ULU region

Example 2: Running FABHFSU in DLI region

The FABHDLI procedure can be used to run FABHFSU in the DLI region. The PSB name must be specified on the PSB= EXEC parameter. The DBD control statement must be coded in CARDIN DD and must specify the DBD name for the database to be processed. The DBD must be a physical DBD.

The three options, that is, sequence check option (Y), sequence error option (A), and sequence error print option (Y) which are specified in the DBD statement, mean:

- Y: sequence check is to be done.
- A: sequence errors are to be processed.
- Y: the diagnostic data is to be printed in the HSSRTRAC data set.

Any incorrect pointers are bypassed because the DBD statement also specifies the pointer bypass option (1).

In this example, two PSB statements are coded.

All segments defined in the DBD are sensitive to the output defined by the OUT1 DD, because '*' is specified in the PSB name field of the first PSB statement. If no sequence errors or pointer errors are found, the output data set produced by the PSB statement can be reloaded by the IMS HD Reorganization Reload utility or a compatible utility, because the unload format UL is specified.

The second PSB statement specifies the VB format and the DD name of OUT2 for the output data set. Only segments of the types defined in the first DB PCB of the PSB OUT2PSB for the SKILLINV database are unloaded, because the PSB statement specifies the PSB name and the PCB Number field is left blank. A user exit routine, OUT2EXIT, is specified on the PSB statement.

The OUT1 and OUT2 DD statements specify the data sets for unloading.

In the following example, it is assumed that the RECON data sets and the database data sets are dynamically allocated.

```
//    EXEC FABHDLI,MBR=FABHFSU,PSB=OUT2PSB,DBRC=Y
//CARDIN    DD  *
DBDSKILLINV         YAY     1
PSB*       OUT1       UL
PSBOUT2PSB OUT2       VBOUT2EXIT
END
/*
//PRNTOUT   DD  SYSOUT=A
//OUT1      DD  DSN=IMSDB.UNLOAD1,DISP=(,KEEP),UNIT=TAPE
//OUT2      DD  DSN=IMSDB.UNLOAD2,DISP=(,KEEP),UNIT=TAPE
```

Figure 12. Running FABHFSU in the DLI region

Chapter 7. Application programming interface for using HSSR Engine

IMS High Performance Unload provides an application programming interface (API) to run, by using HSSR Engine, an IMS DL/I batch application program.

Topics:

IMS High Performance Unload API overview

An application programming interface (API) in IMS High Performance Unload enables an IMS DL/I batch application program, which reads a database sequentially, to use HSSR Engine without being recompiled or relink-edited. Through this API, the application program can retrieve a large number of segments more efficiently than the native IMS DL/I and the elapsed time and CPI time can be reduced.

This API supports IMS DL/I calls and IMS EXEC DLI commands. GN and GNP calls with an unqualified segment search argument (SSA) are optimized. If the calls other than GU, GN, or GNP are included, do not run the application program with IMS High Performance Unload.

An overview of the system flow of an application program is shown in "IMS High Performance Unload system structure" on page 11.

An application program that uses HSSR Engine for processing DL/I database calls is called an *HSSR application program*. To run an HSSR application in the DLI or the DBB region, a PSB is needed. In the PSB, you have to select one or more DBPCBs by specifying them in the HSSRPCB or the HSSRDBD control statement in the HSSROPT data set. The selected DBPCB is called an *HSSR PCB*. A DL/I call or an EXEC DLI command to the HSSR PCB is called an *HSSR call*. The HSSR calls are processed by HSSR Engine and the calls to the other DBPCBs are processed by native IMS DL/I.

Requirement: An application program must have been assembled, compiled, and link-edited in the same way as IMS DL/I batch application programs. For details, see *IMS Application Programming*.

Notes:

- The specification of HSSR PCBs through the KEYLEN keyword of PCB statement is supported for the compatibility with DBT HSSR. See "Method for specifying an HSSR PCB through KEYLEN" on page 358.
- If the ULU region is used, you do not need to code the HSSRDBD or the HSSRPCB control statement. This region type is supported for the compatibility with DBT HSSR.

For the restrictions that apply to HSSR application program jobs, see "Restrictions for IMS High Performance Unload" on page 22.

HSSR PCB requirements

Each *HSSR PCB* that is specified in the HSSRPCB or the HSSRDBD control statement in the HSSROPT data set must satisfy certain requirements.

The following subtopics describe the requirements that must be satisfied:

- "PCB statement requirements" on page 80
- "Processing option (PROCOPT) requirements" on page 80
- "SENSEG statement requirements" on page 81

PCB statement requirements

The following PCB statement requirements must be satisfied:

- The DBDNAME (or NAME) parameter must specify the name of a physical DBD. The name must not be the name of a logical DBD.
- LIST=NO option must not be specified.

Processing option (PROCOPT) requirements

The PROCOPT parameter must specify one or certain combinations of the following codes (see *IMS System Utilities* for more information):

Code
Description

PROCOPT=G
Get function. If a database is shared at the database level when PROCOPT=G, DBRC grants access with read integrity. It prevents the concurrent execution of the application program and the updating by the IMS subsystem.

PROCOPT=O
Read only. (Use this code with G.) If a database is shared at the database level when PROCOPT=GO, DBRC grants access without read integrity. It enables the application program to run and the IMS subsystem to update the database concurrently.

PROCOPT=N
Read only. (Use this code with GO.) N allows the application program to avoid abends when referring to data that is being updated by another program. A GG status code is returned, and the program must decide whether to end or continue.

PROCOPT=T
Read Only. (Use this code with GO.) T works similarly to N, except that it causes DL/I to retry the operation automatically before returning the GG status code.

PROCOPT=E
Enables exclusive use of the database or segment. (Use this code with G.) If a database is shared at the database level when PROCOPT=GE, DBRC grants exclusive use to the database.

If you want to use PROCOPT=GON or GOT, see "Consideration for database sharing" on page 90.

Considerations:

- PROCOPT=R is also accepted for compatibility with DBT HSSR, however, is not recommended. See "Support of PROCOPT=R and replace calls" on page 359.
- PROCOPT=A is also accepted, however, is not recommended. Even if PROCOPT=A is specified, HSSR engine treats it as PROCOPT=G. The other functions I, R, and D are ignored.
- PROCOPT=P is also accepted, however, is not recommended. Even if PROCOPT=P is specified, D command code is not processed by HSSR engine.

SENSEG statement requirements

The following conditions must be satisfied:

- SENSEG statements must be coded in the same hierarchical sequence as is defined in the DBD.
- Field sensitivity must not be specified.
- Either the PROCOPT field must be omitted in the SENSEG statements; or, if it is coded, the processing option must be G, GE, or K.

Consideration: PROCOPT=R is also accepted for compatibility with DBT HSSR, but is, however, not recommended. See “Support of PROCOPT=R and replace calls” on page 359.

HSSR PCB feedback information

After a call is made, the PCB feedback in the HSSR PCB contains the same information as a DL/I PCB with some exceptions.

This topic describes the general-use programming interface. See “Programming interface information” on page 518 to understand the restrictions associated with this type of material.

Subtopics:

- “Interpreting PCB feedback” on page 81
- “Status codes” on page 82

Interpreting PCB feedback

GUPI

An application program should never modify any information in the HSSR PCB. The access to PCB entries by the application program is restricted to read-only. After a call is made, the PCB feedback in the HSSR PCB contains the same information as a DL/I PCB with some exceptions. For details, see “Status codes” on page 82.

GUPI

DMTI

Note: If PCBLIST HSSR which is the default, is specified, the PCB list that is passed to the application program is not the same as the PCB list built by IMS. If PCBLIST HSSR is specified, each list entry for a PCB that is defined as an HSSR PCB points to the corresponding HSSR PCB that has been built by HSSR Engine. An HSSR PCB that is passed to the application program is not the original IMS PCB, but the one rebuilt by IMS High Performance Unload's program controller. HSSR Engine describes the results of the calls your program issues in the HSSR PCB. The HSSR PCB is referred to in the call in the same way as in IMS.

DMTI

The following table shows the HSSR PCB mask, which is the same as IMS.

Table 14. HSSR PCB mask

Descriptor	Length	Note
Database Name	8 bytes	
Segment Level Number	2 bytes	
Status Code	2 bytes	See “Status codes” on page 82.

Table 14. HSSR PCB mask (continued)

Descriptor	Length	Note
Processing Options	4 bytes	If PCBLIST HSSR (the default) is specified, the first two bytes are always 'GX' and the last two bytes list additional processing options, such as 'O ', ' ON', ' OT', 'E ', or 'R '. If the options are defined as 'GONP' or 'GOTP' in the PSB, the last two bytes are 'NP' or 'TP'. If PCBLIST IMS is specified, the content is the same as for IMS.
Reserved	4 bytes	Do not use this field.
Segment Name	8 bytes	
Length of Key Feedback Area	4 bytes	
Number of Sensitive Segments	4 bytes	
Key Feedback Area	Variable length	

For the EXEC DLI command, the DL/I interface block (DIB) is used as the application programming interface. Every part except the status code is the same as the DIB of IMS. For details about the status codes, see “Status codes” on page 82.

The following table shows the format of the DIB.

Table 15. Format of the DIB

Descriptor	Length
Translator Version (DIBVER)	2 bytes
Status Code (DIBSTAT)	2 bytes
Segment name (DIBSEGM)	8 bytes
Segment Level Number (DIBSEGLV)	2 bytes
Key Feedback Length (DIBKFBL)	4 bytes
Database Description Name (DIBDBDNM)	8 bytes
Database Organization (DIBDBORG)	8 bytes

Status codes

The status codes listed in the following table are returned as the result of an HSSR call.

Table 16. Status codes returned from an HSSR call

Code	Meaning	I/O area	PCB feedback
blank	Normal return. No errors were detected.	Segment data is returned.	PCB has a segment name and a segment level, and the concatenated key is set in the key feedback area.
AI	Data management OPEN error	Unpredictable.	Unpredictable. You must not use the PCB feedback information.

Table 16. Status codes returned from an HSSR call (continued)

Code	Meaning	I/O area	PCB feedback
FM	This status code is returned when: • The randomizing routine returns a return code of 4. • The database is a HALDB, and the key supplied on the call was greater than the high key value for the last partition, or the user partition selection exit routine returned a return code of 4 after having been passed a key value with which to select a partition.	No segment was returned.	Unpredictable. You must not use the PCB feedback information.
GB	End of database (issued only for GN and GHN calls or for GN commands).	No segment was returned.	The segment name is filled with blanks, and the segment level is set to C'00'; the key feedback area is nullified, and the length of the key feedback area is set to zero. You must not use the PCB feedback information.
GE	This status code is returned when a segment that satisfies the segment search argument described in the call could not be found.	No segment was returned.	For a GU and GHU call or a GU command, the segment name is filled with blanks, and the segment level is set to C'00'; the key feedback area is nullified, and the length of the key feedback area is set to zero. The content of the key feedback area is unpredictable. You must not use the PCB feedback information. For a GN, GHN, GNP, or GHNP call or for a GN or GNP command, the content of the PCB feedback area is unpredictable. You must not use it.
GG	Pointer error, key sequence error, or other database error occurred. This code can be returned either when PROCOPT=GON or GOT is specified in the PCB, or when 'KEYCHECK GG' or SKERROR option is specified in the HSSROPT data set.	Unpredictable. **Note:** HSSR Engine might skip retrieving some segments while processing the first GN call or GN command that follows a GG status code.	Unpredictable. You must not use the PCB feedback information.

Table 16. Status codes returned from an HSSR call (continued)

Code	Meaning	I/O area	PCB feedback
GP	The program issued a GNP or GHNP call or a GNP command when there was no parentage established, or the segment level specified in the call was not lower than the level of the established parent.	No segment was returned.	Unpredictable. You must not use the PCB feedback information.
GX	Key sequence error occurred. (This code can be returned only when 'KEYCHECK GX' is specified in the HSSROPT data set.) For the EXEC DLI command, the status 'GX' is not returned to the application program, but message DFS1041 is issued in the EXEC DLI interface module (DFSEIPB0), and ends with an abend code of U1041.	Segment data is returned.	PCB has a segment name and a segment level, and the concatenated key is set in the key feedback area.

Notes:

- If there is an error in a call statement or a command statement, the application program receives abend U4013, instead of status code AC, AD, AJ, or AK.
- If hierarchic levels are changed after a successful GN, GHN, GNP, or GHNP call or a successful GN or GNP command, a status code of blank is returned instead of GA.
- If segment types are changed after a successful GN, GHN, GNP, or GHNP call or a successful GN or GNP command, a status code of blank is returned instead of GK.
- When a GN call or a GN command is issued after a GU call or a GU command that returned a GE status code, HSSR Engine might not return the same segment that DL/I would return.
- The status code BA is not returned even if an INIT call or an ACCEPT command has been issued with the character string STATUS GROUPA in the I/O area. If a PROCOPT other than GON and GOT is specified for a PCB, and if HSSR Engine encounters unavailable data regarding the database that is referred to by the PCB, HSSR Engine issues an abend.

DL/I calls and EXEC DLI command for HSSR PCB

An *HSSR call* is a DL/I call or an EXEC DLI command that is issued against an HSSR PCB. Each HSSR call is processed by HSSR Engine.

The HSSR call is issued through the DL/I language interface, but the call is transferred to the HSSR Call Analyzer. The call is finally processed by either HSSR call handler or native IMS DL/I, depending on the type of the call or the API set that is specified by the APISET control statement in the HSSROPT data set.

If APISET 1 (default) or 2 is specified and an unsupported call is issued against an HSSR PCB, the application processing ends abnormally. To avoid program termination, specify APISET 3 or remove the PCB from the list of HSSR PCBs that is specified by the HSSRPCB or the HSSRDBD control statement.

Subtopics:

- "DL/I calls supported by each API set" on page 85

- "EXEC DLI commands supported by each API set" on page 86

DL/I calls supported by each API set

APISET 1 is the system default. The following table shows the DL/I call types supported by each API set, and their effects with and without segment search arguments (SSAs):

Note: For restrictions that are related to API sets, see "Restrictions common to all HSSR applications" on page 23.

Table 17. DL/I call types supported by each API set

API set	Call types supported
APISET 1	The following call types are supported: • GU and GHU calls without SSA: the first root segment of the database is retrieved. • GU and GHU calls with an unqualified SSA that contains only the name of a root segment type. • GU and GHU calls with an SSA that is qualified on the root sequence key. In the SSA, any one of the following relational operators is available: – equal-to (=b,b=, or EQ. b represents a single blank) – greater-than-or-equal-to (>=, =>, or GE) • GN and GHN calls without SSAs. • GN and GHN calls with an unqualified SSA that contains the name of a root segment type. • GN and GHN calls with an SSA that is qualified on the root sequence key. In the SSA, any one of the following relational operators is available: – equal-to (=b,b=, or EQ) – greater-than-or-equal-to (>=,=>, or GE) – greater-than (>b,b>, or GT) These calls are not supported for HDAM or PHDAM database. • GNP and GHNP calls without SSAs.
APISET 2	The following call types are supported for HIDAM, HDAM, PHIDAM, and PHDAM in addition to the call types supported in APISET 1: • GN, GHN, GNP, and GHNP calls with an unqualified SSA that contains the name of a dependent segment type. • GN, GHN, GNP, and GHNP calls with a qualified SSA that contains the name of a dependent segment type at the second level. • GN, GHN, GNP, and GHNP calls with *N* SSAs to retrieve the segment at the *N*-th level (2<=*N*<=15). The following combinations of SSAs are supported: – *N* unqualified SSAs – *N* qualified SSAs – A qualified SSA, *N*-2 unqualified SSAs, and a unqualified or qualified SSA – *N*-1 qualified SSAs, and a unqualified SSA Here, the first SSA must contain the root segment name and only equal-to (=b,b=,or EQ) relational operator is allowed for each qualified SSA.

Table 17. DL/I call types supported by each API set (continued)

API set	Call types supported
APISET 3	The fully supported call types are the same as those in APISET 2. Once an unsupported call is issued for HIDAM, HDAM, PHIDAM, or PHDAM, the call and all the succeeding calls to the HSSR PCB are passed to the IMS DL/I call handler to continue the processing instead of ending it abnormally.

In APISET 1 or 2, if a user application issues a call that is not supported by the HSSR call handler, the call is printed in the Trace Output report in the HSSRTRAC data set even if the TRHC and TRDB control statements are not specified in the HSSROPT data set. By using this report, you can check which call is not supported.

EXEC DLI commands supported by each API set

Requirement: If the EXEC DLI command is used in your application, you must specify 'PCBLIST IMS' in the HSSROPT data set. For details of the PCBLIST IMS specification, see "PCBLIST control statement" on page 173.

IMS High Performance Unload supports three API sets. APISET 1 is the system default. The following table shows the EXEC DLI Command types supported by each API set, and their effects with and without SEGMENT options.

Table 18. EXEC DLI command types supported by each API set

API set	Command types supported
APISET 1	The following command types are supported: • GU command without SEGMENT options • GU command with a SEGMENT option that contains the name of a root segment type
APISET 2	The following command types are supported for HIDAM, HDAM, PHIDAM, and PHDAM in addition to the command types supported in APISET 1: • GN or GNP command with a SEGMENT option that contains the name of a dependent segment type • GN or GNP command with a SEGMENT option and a WHERE option qualified on the sequence key of a second level dependent segment type
APISET 3	The supported call types are the same as those in APISET 2. Once an unsupported call is issued for HIDAM, HDAM, PHIDAM, or PHDAM, the call and all the succeeding calls to the HSSR PCB are passed to the IMS DL/I call handler to continue the processing instead of ending it abnormally.

An HSSR call gives the same results as a DL/I call or an EXEC DLI command, with some minor differences. For details, see "HSSR PCB feedback information" on page 81.

For a complete description of the commands, options, and layout of the DIB and qualified commands, see *IMS Application Programming*.

JCL requirements for your HSSR application

The JCL stream that is used to run an application program on IMS High Performance Unload is not fully compatible with the JCL stream used for the native IMS DL/I batch.

For details, see "Basic JCL requirements" on page 30. You can also use the IBM-supplied catalog procedures. For details, see "Preparing the basic JCL" on page 29.

To run your application program, the HSSROPT DD statement is needed to specify at least an HSSRPCB or HSSRDBD control statement. Also, consider specifying the following output DD statements, which are used to print the HSSR call statistics and the HSSR call traces:

- HSSRSTAT DD statement
- HSSRTRAC DD statement
- HSSRSNAP DD statement

The following figure shows a JCL example to run the application program named as 'YOURAPPL' that issues IMS DL/I calls.

```
//APPLGO    EXEC PGM=FABHX034,
//            PARM=('DFSRRC00/DLI',YOURAPPL,YOURPSB)
//*                                        <--Change EXEC statement
//STEPLIB   DD DSN=IMSVS.SDFSRESL,DISP=SHR
//          DD DSN=IMSTOOL.SHPSLMD0,DISP=SHR <--Add HP Unload LIB
//          DD DSN=YOURPRJ.PGMLIB,DISP=SHR
//DFSRESLB  DD DSN=IMSVS.SDFSRESL,DISP=SHR
//IMS       DD DSN=IMSVS.PSBLIB,DISP=SHR
//          DD DSN=IMSVS.DBDLIB,DISP=SHR
//PROCLIB   DD DSN=IMSVS.PROCLIB,DISP=SHR
//IEFRDER   DD DSN=NULLFILE,
//             DISP=(,KEEP),VOL=(,,,99),UNIT=(TAPE,,DEFER),
//             DCB=(RECFM=VBS,LRECL=3964,BLKSIZE=3968,DEN=3)
//SYSUDUMP  DD SYSOUT=*
//IMSMON    DD DUMMY
//HSSRSTAT  DD SYSOUT=*     <--- Add OUTPUT DD
//HSSRTRAC  DD SYSOUT=*     <--- Add OUTPUT DD
//HSSRSNAP  DD SYSOUT=*     <--- Add OUTPUT DD
//DFSSTAT   DD SYSOUT=*     <--- Add OUTPUT DD
//HSSROPT   DD *            <--- Add INPUT DD
HSSRPCB *ALL
APISET 3
/*
//ddname    DD DSN=your_database_DSN,...
```

Figure 13. JCL to run an application program using the DL/I calls

In this JCL example, two control statements are specified in the HSSROPT data set:

- 'HSSRPCB *ALL' defines that all DBPCBs in the PSB with the name YOURPSB are HSSR PCBs. If you want to select one or more DBPCBs but not all, there are two ways to do so:
  - Specify the PCB numbers as in 'HSSRPCB 001,003'.
  - Specify as 'HSSRDBD *dbdname*' to select all DBPCBs that refer to a specific DBD with the name *dbdname*.
- 'APISET 3' enables all DL/I call types that are supported by this API. And if an unsupported call is issued once, the succeeding call processing to the HSSR PCB is taken over by native IMS DL/I. The statistics of the calls processed by HSSR Engine is logged in the HSSRSTAT data set. For details, see "HSSRSTAT data set" on page 181. And the statistics of the calls processed by native IMS DL/I is logged in the DFSSTAT data set. To tune the IMS DL/I performance, add the DFSVSAMP DD statement and specify IMS VSAM and OSAM buffers and options in the data set.

  If 'APISET 2' is specified instead of 'APISET 3', in the previous mentioned case of the unsupported call, the application processing ends abnormally. Check which call is unsupported in the Trace Output report in the HSSRTRAC data set. For details of the supported calls and restrictions, see "DL/I calls and EXEC DLI command for HSSR PCB" on page 84.

For other options, see Chapter 11, "Options for HSSR Engine," on page 155.

Requirement: If your application uses the EXEC DLI commands, you must add the 'PCBLIST IMS' control statement to the HSSROPT data set.

Considerations for Db2 DL/I Batch interface

IMS High Performance Unload supports the Db2 DL/I Batch interface, with some restrictions.

For DL/I Batch support of Db2, also read the *Db2 Application Programming and SQL Guide*.

By using Db2 DL/I Batch interface, an HSSR application program can issue:

- Any HSSR call, with the restrictions stated in "Restrictions common to all HSSR applications" on page 23.
- Any IMS batch call except a ROLS, SETS, or SYNC call, or any IMS batch command except a ROLS, SETS, or SYNC command. This is the same restriction as for Db2 DL/I Batch support. For further details about the restrictions on IMS batch calls, see *Db2 Application Programming and SQL Guide*.
- IMS system service calls or commands with the same restrictions. See "Considerations for checkpoint and restart" on page 89.
- Any SQL statements except COMMIT and ROLLBACK. The application program must use the IMS CHKP call or the IMS CHKP command to commit data, and the IMS ROLL or ROLB to roll back changes.
- Any call or command to a standard or conventional access method such as QSAM or VSAM.

Subtopics:

Program design considerations

The program design considerations for ordinary Db2 DL/I Batch support apply to IMS High Performance Unload. You should be familiar with the program design considerations for the Db2 DL/I Batch support, especially those related to checkpoint calls and application program synchronization, or to checkpoint commands and application program synchronization.

Restrictions on Db2 DL/I Batch support

The restrictions described in "Restrictions common to all HSSR applications" on page 23 apply also to HSSR application programs that use the Db2 DL/I Batch interface. The following restriction also applies:

- Db2 change data capture exit routine (DB2CDCEX) is supported, but a changed data capture exit routine cannot issue any HSSR call.

Requirements for using Db2 DL/I Batch support

You can use IBM-supplied cataloged procedure FABHDB2, which resides in the HPS.SHPSSAMP sample library.

The required changes to the application program and the job step JCL are basically the same as for Db2 DL/I Batch support, except that using DSNMTV01 on the MBR= parameter is not supported; the results obtained when it is used are unpredictable.

You must specify a subsystem member, using the SSM= parameter on the procedure FABHDB2. Also, specify your HSSR application program name, using the MBR= parameter, and the name of the IMS High Performance Unload's program controller, FABH000, as the ninth positional parameter PROG in the DDITV02 data set. For more information about how to specify the EXEC parameters and the DDITV02 DD on the Db2 DL/I Batch job step JCL, see the *Db2 Application Programming and SQL Guide*.

In the HSSR application program that uses Db2 DL/I Batch support, the following additional EXEC parameter must be provided:

SSM=
This required parameter specifies a 1- to 4-byte character identifier. When building the IMS.PROCLIB member that contains the information about each Db2 subsystem that IMS communicates with, you must generate the member name by concatenating this SSM identifier to the IMSID.

Carefully provide the following EXEC parameter:

MBR=
This required parameter specifies the name of the HSSR application program. The Db2 module name DSNMTV01 must not be specified.

Provide the following DD statement in your JCL:

DDITV02 DD
Specify the program controller name FABH000 as the ninth positional parameter PROG in this data set.

For example:

```
//DDITV02 DD *
DSN,SYS1,DSNMIN10,,R,-,BATCH001,DB2PLAN,FABH000
/*
```

The following DD statement is optional:

DDOTV02 DD
This optional DD statement defines the output data set in which the Db2 output information is written. If DCB is coded on the JCL, the specification must be RECFM=V or VB, LRECL=4092, and "BLKSIZE ≥ LRECL+4". If the DD statement is missing, message IEC130I is issued and processing continues without any output.

Considerations for checkpoint and restart

Certain considerations and restrictions apply when your application program uses MVS checkpoints, DL/I CHKP and XRST calls, or EXEC DLI CHKP and XRST commands.

Subtopics:

- “MVS checkpoints” on page 89
- “DL/I CHKP and XRST calls or EXEC DLI CHKP and XRST commands” on page 89

MVS checkpoints

MVS checkpoints can cause unpredictable results in HSSR application programs. Therefore, HSSR application programs must not issue:

- MVS checkpoints
- Those basic DL/I CHKP calls or EXEC DLI CHKP commands that request MVS checkpoints (For a detailed description of DL/I CHKP calls, see *MVS Application Programming: Design Guide*.)

DL/I CHKP and XRST calls or EXEC DLI CHKP and XRST commands

HSSR application programs can issue PCB DL/I CHKP and XRST calls or XRST commands to the I/O PCB. However, a DL/I CHKP call or an EXEC DLI CHKP command should not request an MVS checkpoint.

- If PCBLIST HSSR (the default value) is specified, HSSR application programs can issue XRST calls with APISET 1 (also the default value).
- If HSSR application programs issue XRST calls with PCBLIST IMS specified, you must specify APISET 3.
- If XRST command is to be issued, you must specify APISET 3.

HSSR Engine is not aware of CHKP and XRST calls or XRST commands. For HSSR PCBs, the behavior of HSSR Engine is not compatible with the behavior of DL/I. Be careful if you are concerned about compatibility between HSSR and DL/I.

The differences between HSSR Engine and DL/I are as follows:

- After a CHKP call or a CHKP command, the position of HSSR PCBs is not set to the beginning of the database. Therefore, the database position established after a CHKP call or a CHKP command is not compatible with HSSR Engine and DL/I. The HSSR Engine's compare option might become useless in this environment.

  If you are concerned about compatibility, issue GU database calls or GU database commands to the HSSR PCBs after a DL/I CHKP call or an EXEC DLI command.
- During a CHKP call or a CHKP command, the key feedback area of HSSR PCBs is not written to the IMS log.

  Instead, the IMS log might contain records for internal DL/I PCBs, which are unknown to the application program.
- During an XRST call or an XRST command, key feedback information is not restored in HSSR PCBs.

  Application programs that need to record database positioning information for HSSR PCBs can record it themselves by providing it in user areas during a CHKP call or a CHKP command. The XRST call or the XRST command restores this logged information in user areas. For more information about providing user calls, see *IMS Application Programming*.

Consideration for database sharing

In general, HSSR application programs, including FABHURG1 and FABHFSU, should be run with exclusive control of the database as any unload utility does during reorganization. In some cases, however, you might want to read the database while it is being updated by IMS. The update might be done either by IMS within the same program or by another IMS subsystem that is running concurrently.

This topic provides hints for such a case.

Subtopics:

Database sharing support

HSSR Engine supports the following database-sharing functions. However, it does not provide read integrity, if an HSSR application program reads a database that is being updated at the same time by IMS either within the same program or through concurrent execution of another IMS subsystem.

The capability to cope with data set extensions, which are created by concurrently updating IMS subsystems

See “Handling data set extensions” on page 91.

Support for processing options GON and GOT

See “Support for processing options GON and GOT” on page 91.

Support for database level sharing through the use of the standard database access authorization logic of DBRC

In a database-level sharing environment, an HSSR application program can read databases either with read integrity (with read processing intent; PROCOPT=G) or without read integrity (with read-only processing intent; PROCOPT=GO).

See “Support for database level sharing” on page 92.

Ability to an HSSR application program to run in a block-level sharing environment
In this case, HSSR Engine does not provide read integrity.

See "Considerations for block level sharing" on page 93.

Some problems might occur because the HSSR buffer handler does not synchronize the content of its buffers with the content of the IMS buffer handler. HSSR Engine is not aware of a modification to a block or CI unless IMS writes the modified block or CI from its buffers to DASD. It is also not aware of a modification if an HSSR buffer contains an older image of that block or CI. For example, the following problems that might occur with IMS in a database level sharing environment might also occur with HSSR Engine:

- The HSSR application program might read uncommitted data or might read data that is no longer up-to-date.
- The HSSR Engine might issue an abend or a GG status code due to what appears to be incorrect pointers in the segment prefix.
- Additional problems can occur with VSAM KSDS, if HSSR Engine reads a KSDS that is updated by an IMS program that might create CI splits or CA splits.

For methods of avoiding these problems, see "Avoiding problems caused by the lack of read integrity" on page 93.

Handling data set extensions

HSSR buffer handler can access OSAM blocks or VSAM CIs stored and update IMS subsystems in new extents of the data sets concurrently.

This support is provided in the following environments:

- For OSAM, in all environments, including database level sharing and block level sharing
- For VSAM, only when all the following conditions are met:
  - The updating IMS subsystem runs within the same MVS system.
  - VSAM SHAREOPTIONS (3,3) are used. (See "VSAM SHAREOPTIONS" on page 93.)

For OSAM, OSAM LDS, and ESDS, this support is provided by default. For KSDS, this support must be explicitly activated by a RETRY KSDS control statement of the HSSROPT data set (for more details, see "RETRY control statement" on page 174).

Support for processing options GON and GOT

For the processing options GON, HSSR Engine provides the same support as IMS. Upon encountering a database error, or what in a database sharing environment appears to be a database error, HSSR Engine returns a GG status code.

For the processing option GOT, HSSR Engine provides slightly different support from IMS. Upon encountering a database error situation, or what in a database-sharing environment appears to be a database error situation, HSSR Engine takes the following actions:

1. Retries many times to access the database, and waits for a specific number of seconds before each attempt (only for HDAM, HIDAM, PHDAM, and PHIDAM).

   The number of attempts and the waiting time can be specified on the GOTRETRY control statement.
2. Returns a GG status code if the attempts to access the database are not successful.
3. Invalidates all PCB-related buffers; database calls issued after a GG status code do not use old buffer copies but are satisfied with new database accesses. The buffer invalidation can increase the chances of an application program resuming its processing after a GG status code. The HSSR buffer handler invalidates buffers only for HDAM, HIDAM, PHDAM, and PHIDAM.

Notice the following additional points concerning processing options GON and GOT:

- The SKERROR option (see "SKERROR control statement" on page 175) can be considered a logical extension of the support that HSSR Engine provides for the processing options GON and GOT. When the SKERROR option has been activated, HSSR Engine does not reset the database position to the beginning of the database, but keeps the current database position. If the application program issues a GN call or a GN command after a GG status code, the database error is skipped (for example, the processing of an incorrect pointer is skipped) and HSSR Engine returns the next segment it can retrieve. Instead of using abends or retrying the database access by combining WAITs, GU calls or commands, and GN calls or commands, an application program can use SKERROR to continue the database retrieval by issuing GN calls or commands. The option can also be used by an application program that can afford to skip the retrieval of one or more database segments.
- The number of returned GG status code is reported in the HSSRSTAT data set.
- To get detailed diagnosis information about the GG situation, activate the DIAGG option. (See "DIAGG control statement" on page 167.)
- HSSR Engine does not have an interface to IRLM. Therefore, in a block level sharing environment with processing option GOT, HSSR Engine does not issue IRLM test enqueues when attempting to re-access the database during a GG situation. For HDAM, HIDAM, PHDAM, and PHIDAM databases, HSSR Engine tries to balance this lack of IRLM test enqueues by making many attempts to access the database, and by waiting before each attempt.

Support for database level sharing

IMS High Performance Unload provides the same support for database level sharing as IMS. This support requires that DBRC be installed and active, and that the database be registered in the DBRC RECON data sets.

Application program running in DLI or DBB region

For an HSSR application program running in DLI or DBB region, DBRC authorizes database access to the job step under the same conditions as a normal IMS batch job step.

Authorization for the database access depends on the following:

- The database processing intent as specified through PROCOPT during PSBGEN
- The database-sharing level defined in the RECON data sets
- The current status indicators of the database, as recorded in the RECON data sets

You have the following options during PSBGEN:

- You can specify a read processing intent (PROCOPT=G) in order to have read integrity. DBRC then prevents concurrent execution of the HSSR application program with an IMS subsystem that updates the database.
- You may specify a read-only processing intent (PROCOPT=GO) to be able to run your application program concurrently to other IMS subsystems that update the database. In this case, no read integrity is provided.

You can also specify an exclusive processing intent (PROCOPT=GE) in order to have exclusive usage of the database.

Restriction: If two or more database PCBs (DBPCBs) are defined in the PSB and the IMSDALIB DD statement specifies the library of DFSMDA members for dynamic allocation of the database data sets, HSSR Engine does not send the DBRC authorization request for database access. If two or more DBPCBs are defined in the PSB, you must concatenate the library of DFSMDA members to the STEPLIB DD statement.

Application program running in a ULU region

In a ULU region, DBRC authorizes database access to an HSSR application program under the same conditions as to the standard IMS HD Reorganization Unload utility. When an HSSR application program runs in the ULU region, HSSR Engine lets IMS and DBRC believe that it is the IMS HD Reorganization Unload utility that is being run.

Note: Databases that have been registered for block-level sharing are shared by a ULU region at the database level. This is what IMS does for the IMS HD Reorganization Unload utility.

Considerations for block level sharing

HSSR Engine does not provide any special support for block level sharing, and does not provide an interface to the IRLM. It is, however, possible to run an HSSR application program in a block level sharing environment if no replace processing option is specified for HSSR PCBs.

HSSR Engine behaves in the same way as in a database level sharing environment with read-only processing intent. Read integrity is not provided. When running in a block level sharing environment with read processing intent, HSSR Engine issues a warning message and runs with read-only processing intent.

If you need to read, with read integrity, a database that has been registered with SHARELEVL 2 or SHARELEVL 3 (block level sharing), you can consider specifying IRLM=N on the JCL and specifying a read processing intent during PSBGEN. In this case, sharing occurs at the database level, and DBRC prevents concurrent execution with an updating IMS subsystem.

Avoiding problems caused by the lack of read integrity

To avoid the problems that are caused by the lack of read integrity, you might use the following methods. Apply them with care.

Frequency of SYNC points or checkpoint calls and commands in updating programs
Updating IMS programs should rapidly write the content of modified buffers to DASD. This can be achieved by a high frequency of calls or commands, whether SYNC points or checkpoints, in the updating IMS application programs.

Avoiding CI splits and CA splits
With ESDS, OSAM, and OSAM LDS data sets, the exposure of reading inconsistent data is limited to the reading of database records that are modified by the updating IMS program. With KSDS data sets, the risk of reading inconsistent data is much higher if CI splits or CA splits occur.

Reading a CI that has been split might create a skip of all the root segments that have been shifted out of the split CI. Reading a CA that has been split might create a skip of all the roots that have been shifted out of the split CA. Other incorrect results might also occur.

To avoid these problems, allocate enough free space in the KSDS and re-create it frequently enough to reduce the number of CI splits and CA splits. For example, a KSDS can be re-created by restoring with the IMS Database Recovery utility.

Unless the occurrence of CI splits and CA splits can be kept very low while the HSSR application program is running, do not read an HISAM KSDS with HSSR Engine.

Free space for ESDS, OSAM, and OSAM LDS data sets
By specifying free space within the OSAM blocks, within the OSAM LDS CIs, or within the ESDS CIs, you can reduce the probability that insert and delete calls will create what may appear to HSSR buffer handler to be an incorrect pointer situation.

VSAM SHAREOPTIONS

This topic presents information about a product-sensitive programming interface. See “Programming interface information” on page 518 to understand the restrictions associated with this type of material.

PSPI

The considerations discussed here for the selection of appropriate VSAM SHAREOPTIONS are similar to the corresponding IMS considerations. You can select the VSAM SHAREOPTIONS (3,3), (2,3) or (1,3).

If the application program does not issue HSSR REPL calls, VSAM SHAREOPTIONS (1,3) can be used.

Note: Specifications of incorrect VSAM SHAREOPTIONS might result in OPEN errors.

VSAM SHAREOPTIONS

You can select either the VSAM SHAREOPTIONS (1,3), (2,3), or (3,3).

VSAM SHAREOPTIONS (1,3) allows concurrent access to the following:

- Multiple HSSR ACBs-that is, ACB, ACBs to multiple HSSR application programs issuing HSSR calls.
- Multiple-input-only IMS ACBs–that is, ACBs to multiple IMS subsystems that issue DL/I calls or EXEC DLI commands to the database with a processing intent read or read-only.

If an HSSR PCB has a replace processing option, SHAREOPTIONS (1,3) cannot be used.

VSAM SHAREOPTIONS (2,3) allows concurrent access to multiple HSSR ACBs (to multiple HSSR application programs issuing HSSR calls), to multiple-input-only IMS ACBs (to multiple IMS subsystems that issue DL/I calls or EXEC DLI commands to the database with a processing intent read or read-only), and to one single-output IMS ACB (to one IMS subsystem that issues DL/I calls or EXEC DLI commands to the database with a processing intent update). VSAM SHAREOPTIONS (2,3) does not allow HSSR Engine to get access to a control interval that has been stored in a new extent of the data set by a concurrently updating IMS subsystem.

VSAM SHAREOPTIONS (3,3) allow concurrent access to multiple HSSR ACBs (to multiple HSSR application programs issuing HSSR calls), and to multiple-input-only and output IMS ACBs (to multiple IMS subsystems that issue DL/I calls or EXEC DLI commands with read-only, read, and update processing intents). The VSAM SHAREOPTIONS (3,3) are one of the prerequisites that must be met in order for HSSR buffer handler to get access to control intervals that have been stored by concurrently updating IMS subsystems in new extents of the data set.

Technical explanations

The VSAM SHAREOPTIONS are used by the VSAM Open modules in order to control within one single operating system the concurrent access to a VSAM data set through input-only ACBs and through output ACBs.

- The SHAREOPTIONS (3,3) allow concurrent access through multiple-output ACBs or through multiple-input-only ACBs.
- The SHAREOPTIONS (2,3) allow concurrent access through one-single-output ACB and through multiple-input-only ACBs.
- The SHAREOPTIONS (1,3) allow concurrent access through multiple-input-only ACBs or through one-single-output ACB.

The following paragraphs explain which kinds of ACBs are used in an IMS environment and in an IMS High Performance Unload environment.

An IMS subsystem is either one online IMS system with multiple message regions and BMP regions, or one IMS batch region. Each IMS subsystem uses one ACB for each VSAM data set in order to perform the I/O resulting from DL/I calls or EXEC DLI commands. IMS opens the ACB either for input only (if the database processing intent is read or read-only) or for output (if the database processing intent is update).

In an IMS High Performance Unload job step, HSSR Engine uses its own read-only ACBs to do the I/O resulting from HSSR calls; ordinarily these are the only ACBs opened by the job step. In any of the following three conditions, IMS opens and uses one additional IMS:

- The HSSR application program gets access to the same database with both HSSR calls and DL/I calls, or with both HSSR calls and EXEC DLI commands.
- The compare option has been activated by coding the CO control statement in the HSSROPT data set, for problem determination. (The compare option internally reissues all HSSR calls to DL/I.)

  The LPCK-building option has been activated by coding the BLDLPCK control statement in the HSSROPT data set. (If the BLDLPCK option has been specified, HSSR Engine issues DL/I calls or EXEC DLI commands internally to get LPCKs.)
- The application program issues HSSR REPL calls.

The additional ACB used by IMS is opened by IMS either for input only, or for output (see the preceding description).

PSPI

Consideration for HALDB single partition processing

By specifying the HALDB control statement on the DFSHALDB DD statement, you can select a single HALDB partition to be processed. This function is the same as the IMS function with the same name *HALDB Single Partition Processing*.

The following figure is an example of the HALDB control statement. Here, '001' is the DB PCB number and 'PHDO01C' is the partition name that is to be processed.

```
//DFSHALDB DD *
HALDB PCB=(001,PHDO01C)
/*
```

Figure 14. Example of the HALDB control statement for single partition processing

For details of the DFSHALDB DD statement and the HALDB control statement, see *IMS System Definition*.

Note: If the application program issues GN calls repeatedly, and reaches the end of the partition, the status code GB is returned to the application program.

Chapter 8. Methods for processing High Availability Large Databases

You can use the FABHURG1 unload utility, the FABHFSU unload utility, or your HSSR application program to process High Availability Large Databases.

Topics:

Functions that support HALDBs

HD databases can be partitioned by use of the HALDB Partition Definition utility (DSPXPDDU) of IMS. HD databases partitioned in this way are called High Availability Large Databases (HALDBs).

If you partition an HDAM database, it becomes a partitioned hierarchical direct-access method (PHDAM) database. If you partition a HIDAM database, it becomes a partitioned hierarchical indexed direct-access method (PHIDAM) database. For details about PHDAM and PHIDAM databases, refer to the *IMS Database Administration*.

You can use FABHURG1 and FABHFSU to unload PHIDAM and PHDAM databases. You can also write your own HSSR application program to access these databases, with some restrictions.

Note: In this topic and in the subsequent topics, any reference to HALDB or a partitioned database means either a PHDAM or a PHIDAM database unless otherwise specified. The partitioned secondary index (PSINDEX) is not intended to be included.

Unloading of a PHDAM or PHIDAM database

You can use the following unload utilities to unload all partitions of a PHDAM or a PHIDAM database:

- FABHURG1
- FABHFSU in standard mode

You can also use these utilities to unload a particular partition or a sequence of partitions from a PHDAM or PHIDAM database.

If a HALDB is unloaded by FABHURG1 with *HD format or by FABHFSU with UL format, you can use the IMS HD Reorganization Reload utility or a compatible utility to reload it.

If a HALDB has been unloaded by FABHURG1 with *HD format or by FABHFSU with UL format, the IPR Reload utility and IMS High Performance Load regard it internally, and refer to it, as having PHD format.

Both FABHURG1 and FABHFSU provide a function for unloading a particular partition or a sequence of partitions from a partitioned database. Unloading partition data sets in parallel, using the multiple FABHURG1 or FABHFSU jobs, reduces the elapsed time required for unloading a partitioned database.

To process partitions that are defined as HALDB Online Reorganization (OLR) capable, see the considerations described in "Considerations for HALDB Online Reorganization capable partitions" on page 26.

In the FABHFSU utility, only the standard mode is supported for unloading a partitioned database, a partition of it, or a sequence of partitions of it. You cannot run FABHFSU in PSF mode for a partitioned database.

FABHURG1 also supports migration unload and fallback unload. The control statements that are used for migration and fallback are explained in “Migration unload and fallback unload” on page 107.

Restrictions for processing HALDBs

The following restrictions apply to the support that IMS High Performance Unload provides for HALDB.

FABHURG1 utility

- FABHURG1 does not support the migration unload of secondary indexes.
- FABHURG1 does not support the migration unload of HISAM databases.
- If PTR=H or PTR=HB is defined as the parent segment of virtual logical child, FABHURG1 does not support the migration unload of the database.
- FABHURG1 does not support the fallback unload of partitioned secondary indexes.

FABHFSU utility

- FABHFSU cannot HALDB unload in PSF mode.
- FABHFSU does not support the migration unload of secondary indexes.
- FABHFSU does not support the migration unload of HISAM databases.
- If PTR=H or PTR=HB is defined as the parent segment of a virtual logical child, FABHFSU does not support the migration unload of the database.
- FABHFSU does not support fallback unload.

User HSSR application program

A user HSSR application program cannot process only the selected partitions; HALDB is processed as an entire database. However, by specifying the HALDB control statement on the DFSHALDB DD statement, you can select a single HALDB partition to be processed.

Types of processing for unloading a HALDB

By using FABHURG1 or FABHFSU (in standard mode), you can unload an entire database, a selected partition, or a sequence of partitions of a HALDB.

Subtopics:

Unloading the entire database

You can unload a partitioned database as a whole database with one FABHURG1 job step or with one FABHFSU job step.

If you unload a partitioned database in *HD format (FABHURG1) or in UL format (FABHFSU), you can use the IMS HD Reorganization Reload utility or a compatible utility to reload the unloaded database data set.

The following figure shows the data flow for unloading and reloading an entire database.

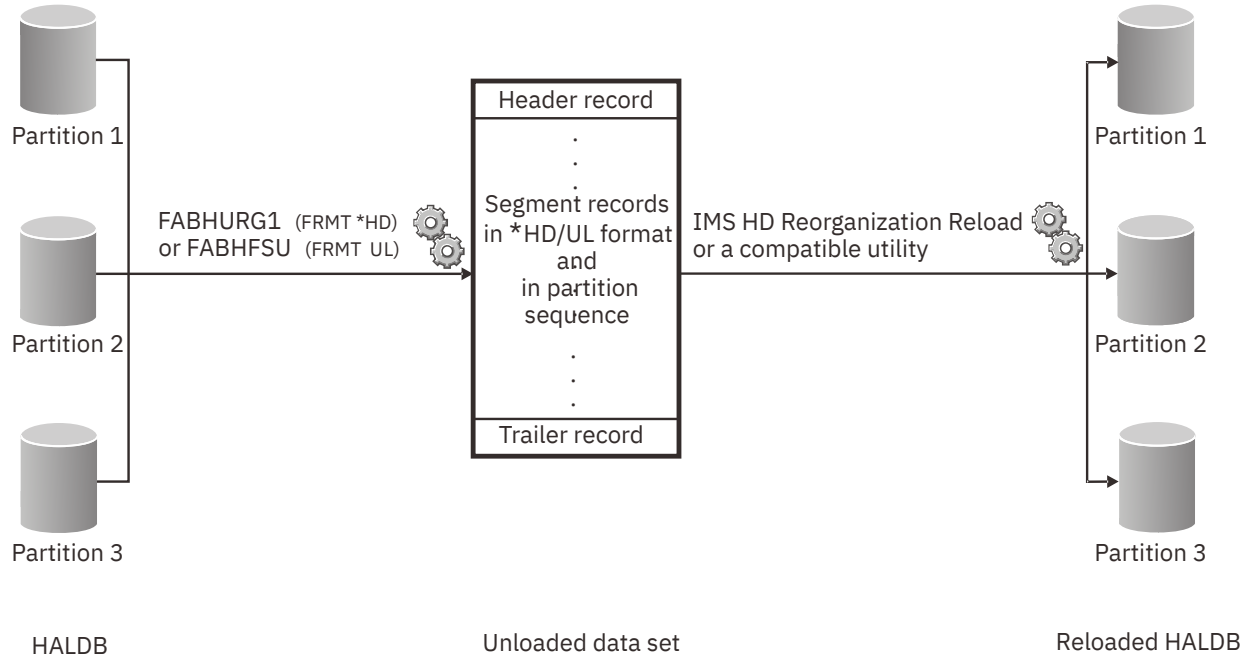

Figure 15. Unloading an entire HALDB

Unloading a partition

FABHURG1 or FABHFSU (in standard mode) can also be used to unload a single partition from a partitioned database. The PARTITION control statement in the SYSIN data set is used to specify a partition for FABHURG1, and the PARTITION control statement in the CARDIN data set is used to specify a partition for FABHFSU. HSSR Engine allocates the buffers only for a selected partition.

When a single partition is unloaded, the unloaded data set contains a header record, all segment records in the selected partition, and a trailer record. You can use the IMS HD Reorganization Reload utility or a compatible utility to reload the *HD-format data set that is unloaded by the FABHURG1 utility or the UL-format data set that is unloaded by the FABHFSU utility.

The following figure shows the data flow for unloading and reloading each partition.

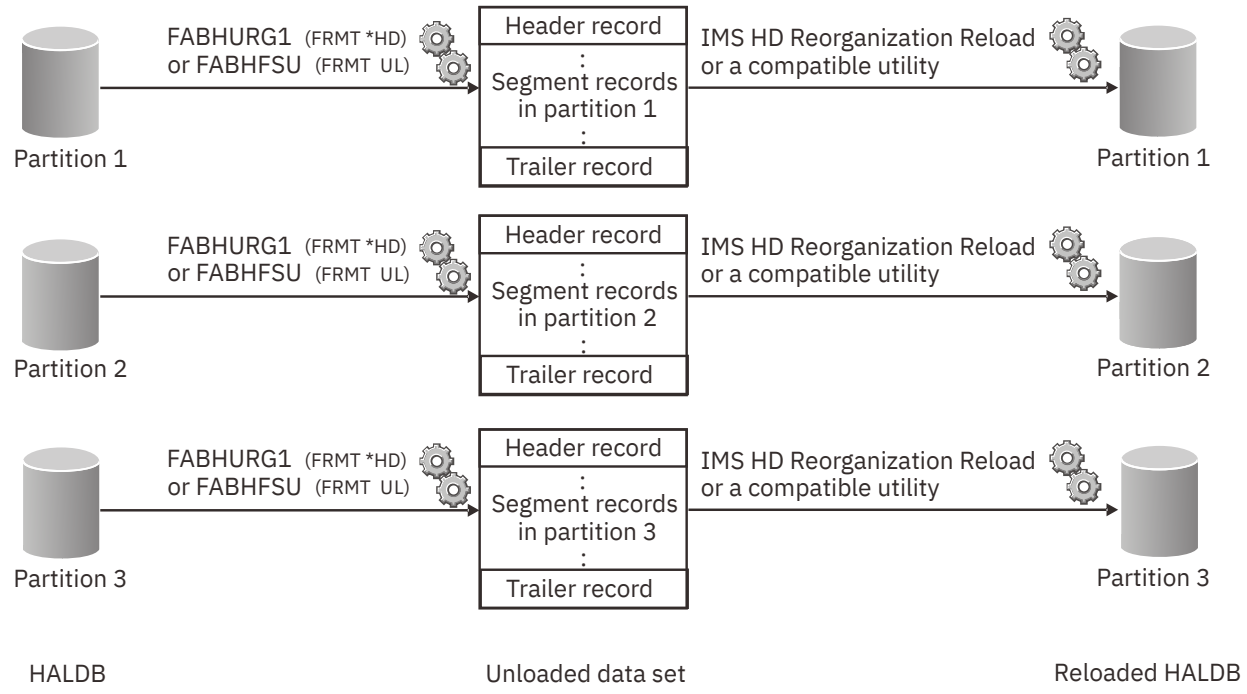

Figure 16. Unloading a partition

Unloading a sequence of partitions

FABHURG1 or FABHFSU (in standard mode) can also unload a sequence of partitions that is specified on a PARTITION control statement. HSSR Engine allocates buffers only for the specified partitions. When a sequence of partitions is unloaded, the unloaded data set contains the following records in the following sequence:

1. A header record
2. Segment records of all selected partitions in the partition sequence determined by the IMS partition selection logic
3. A trailer record

You can use the IMS HD Reorganization Reload utility or a compatible utility to reload the *HD-format data set that is unloaded by the FABHURG1 utility or the UL-format data set that is unloaded by the FABHFSU utility.

The following figure shows the data flow for unloading and reloading a sequence of partitions.

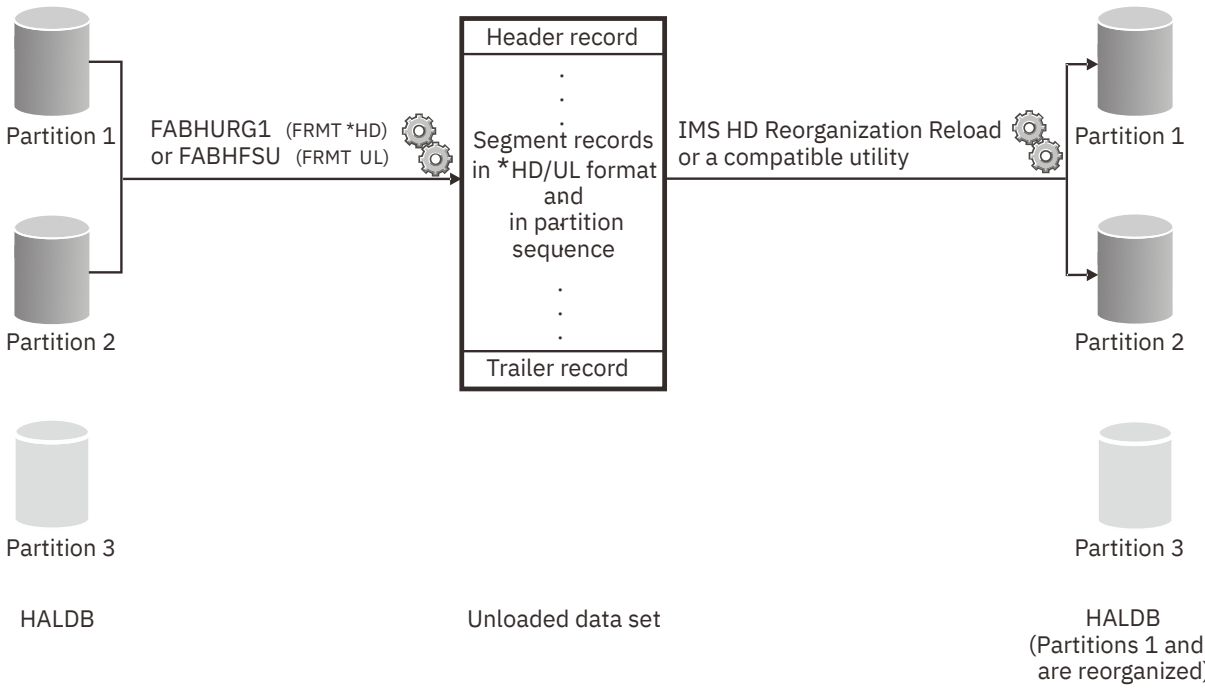

Figure 17. Unloading a sequence of partitions

Unloading a partitioned database with FABHURG1

To unload a partitioned database by using FABHURG1, you must modify the FABHURG1 JCL that is used for a nonpartitioned database.

Procedure

1. Prepare FABHURG1 JCL.

   In FABHURG1 JCL that is used for a nonpartitioned database, make the following modifications to process a HALDB:

   - Specify that DBRC will be used.
   - Specify the DD statements for RECON data sets or make sure that RECON data sets will be allocated dynamically.
   - Ensure that DD statements for database data sets are not specified.
   - If you want to unload a particular partition or a sequence of partitions, code a PARTITION control statement in the SYSIN data set (explained in the next step).
2. Code the PARTITION control statement in the SYSIN data set.

   To process a partition or a sequence of partitions, you must specify the partition by coding the PARTITION control statement. For details of the control statement, see "PARTITION control statement" on page 44.

   Note: To unload an entire database, you do not need to code a PARTITION control statement in your FABHURG1 JCL.
3. Optional: If you want to print partition-wide statistics reports, code additional control statements.

   The following control statements can be used to print partition-wide statistics reports:

   - SEGSTAT control statement in the SYSIN data set
   - CALLSTAT control statement in the HSSROPT data set

- PARTINFO control statement in the HSSROPT data set

Examples

Example 1: Unloading an entire database

Use the following JCL example to unload an entire database.

```
//    EXEC FABHULU,MBR=FABHURG1,DBD=USERDBD,DBRC=Y
//RECON1   DD DSN=IMSVS.RECON1,DISP=SHR
//RECON2   DD DSN=IMSVS.RECON2,DISP=SHR
//RECON3   DD DSN=IMSVS.RECON3,DISP=SHR
//HSSROPT  DD *
PARTINFO DEF
/*
//SYSIN    DD *
SEGSTAT PART
/*
//SYSPRINT DD SYSOUT=A
//SYSUT2   DD DSN=IMSDB.HDUNLD,DISP=(,CATLG),UNIT=TAPE,
//            VOL=SER=VOL001
```

In this example:

- The unloaded data set is defined by the SYSUT2 DD statement.
- The database that is unloaded is specified on the DBD parameter.
- DBRC=Y is specified so that DBRC is used.
- The DD statements for RECON data sets are specified.
- All partition data sets are dynamically allocated by HSSR Engine.
- The 'PARTINFO DEF' statement produces the HALDB Partition Definition report.
- The 'SEGSTAT PART' statement produces the partition-wide Segment Statistics report.

Example 2: Unloading a partition

Use the following JCL example to unload a partition of a partitioned database.

```
//    EXEC FABHULU,MBR=FABHURG1,DBD=USERDBD,DBRC=Y
//SYSIN    DD *
PARTITION PART10
/*
//SYSPRINT DD SYSOUT=A
//SYSUT2   DD DSN=IMSDB.HDUNLD,DISP=(,CATLG),UNIT=TAPE,
//            VOL=SER=VOL001
```

In this example:

- The unloaded data set is defined by the SYSUT2 DD statement.
- The database that is unloaded is specified on the DBD parameter.
- DBRC=Y is specified so that DBRC is used.
- The partition to be unloaded is specified by the PARTITION control statement in the SYSIN DD statement. All data sets for the selected partition PART10 are dynamically allocated by HSSR Engine.

In this example, it is assumed that the RECON data sets are dynamically allocated.

Example 3: Unloading a sequence of partitions

Use the following JCL example to unload a sequence of partitions of a partitioned database.

```
//    EXEC FABHULU,MBR=FABHURG1,DBD=USERDBD,DBRC=Y
//RECON1   DD DSN=IMSVS.RECON1,DISP=SHR
//RECON2   DD DSN=IMSVS.RECON2,DISP=SHR
//RECON3   DD DSN=IMSVS.RECON3,DISP=SHR
//HSSROPT  DD *
PARTINFO DEF,ACC
/*
//SYSIN    DD *
SEGSTAT PART
PARTITION PART10  5
/*
//SYSPRINT DD SYSOUT=A
//SYSUT2   DD DSN=IMSDB.HDUNLD,DISP=(,CATLG),UNIT=TAPE,
//            VOL=SER=VOL001
```

In this example:

- The unloaded data set is defined by the SYSUT2 DD statement.
- The database that is unloaded is specified on the DBD parameter.
- DBRC=Y is specified so that DBRC is used.
- The DD statements for RECON data sets are specified.
- The partitions to be unloaded are five consecutive partitions. The first is partition PART10, which is specified on the PARTITION control statement in the SYSIN DD statement. All data sets for the selected partitions are dynamically allocated by HSSR.
- The 'PARTINFO DEF,ACC' statement produces the HALDB Partition Definition report and the HALDB Partitions Accessed report.
- The 'SEGSTAT PART' statement produces the partition-wide Segment Statistics report.

Unloading a partitioned database with FABHFSU

To unload a partitioned database by using FABHFSU, you must modify the FABHURG1 JCL that is used for nonpartitioned database.

Procedure

1. Prepare FABHFSU JCL.

   In standard mode FABHFSU JCL that is used for a nonpartitioned database, make the following modifications to process a HALDB:

   - Specify that DBRC will be used.
   - Specify the DD statements for RECON data sets or make sure that RECON data sets will be allocated dynamically.
   - Ensure that DD statements for database data sets are not specified.
   - If you want to unload a particular partition or a sequence of partitions, code a PARTITION control statement in the SYSIN data set (explained in the next step).
2. Code the PARTITION control statement in the CARDIN data set.

   To process a partition or a sequence of partitions, you must specify the partition by coding the PARTITION control statement. For details of the control statement, see “PARTITION control statement” on page 60.

   Note: To unload an entire database, you do not need to code a PARTITION control statement in your FABHFSU JCL.
3. Optional: If you want to print partition-wide statistics reports, code additional control statements.

   The following control statements can be used to print partition-wide statistics reports:

   - SEGSTAT control statement in the CARDIN data set

- CALLSTAT control statement in the HSSROPT data set
- PARTINFO control statement in the HSSROPT data set

Examples

Example 1: Unloading an entire database

Use the following JCL example to unload an entire database.

```
//       EXEC FABHULU,MBR=FABHFSU,DBD=SKILLINV,DBRC=Y
//RECON1   DD DSN=IMSVS.RECON1,DISP=SHR
//RECON2   DD DSN=IMSVS.RECON2,DISP=SHR
//RECON3   DD DSN=IMSVS.RECON3,DISP=SHR
//CARDIN   DD *
DBDSKILLINV
PSB*       OUTPUT    UL
END
/*
//HSSROPT  DD *
PARTINFO DEF
/*
//PRNTOUT  DD SYSOUT=A
//OUTPUT   DD DSN=IMSDB.UNLOADDS,DISP=(,KEEP),UNIT=TAPE
```

In this example:

- The database that is unloaded is specified on the DBD parameter.
- DBRC=Y is specified so that DBRC is used.
- The DD statements for RECON data sets are specified.
- All partition data sets are dynamically allocated by HSSR Engine.
- The DBD statement specifies the DBD name.
- The PSB statement shows that the unloaded data set is specified by the OUTPUT DD statement and that the output records are written in the UL format.
- The 'PARTINFO DEF' statement produces the HALDB Partition Definition report.

Example 2: Unloading a partition

Use the following JCL example to unload a partition of a partitioned database.

```
//       EXEC FABHULU,MBR=FABHFSU,DBD=SKILLINV,DBRC=Y
//CARDIN   DD *
DBDSKILLINV
PSB*       OUTPUT    UL
PARTITION SKINVP1
SEGSTAT PART
END
/*
//PRNTOUT  DD SYSOUT=A
//OUTPUT   DD DSN=IMSDB.UNLOADDS,DISP=(,KEEP),UNIT=TAPE
```

In this example:

- The database that is unloaded is specified on the DBD parameter.
- DBRC=Y is specified so that DBRC is used.
- The DBD statement specifies the DBD name.
- The PSB statement shows that the unloaded data set is specified by the OUTPUT DD statement and that the output records are written in the UL format.
- The partition to be unloaded is specified by the PARTITION control statement in the CARDIN DD statement. All data sets for the selected partition SKINVP1 are dynamically allocated by HSSR.
- The 'SEGSTAT PART' statement produces the partition-wide Segment Statistics report.

In this example, it is assumed that the RECON data sets are dynamically allocated.

Example 3: Unloading a sequence of partitions

Use the following JCL example to unload a sequence of partitions from a partitioned database.

```
//       EXEC FABHULU,MBR=FABHFSU,DBD=SKILLINV,DBRC=Y
//RECON1   DD DSN=IMSVS.RECON1,DISP=SHR
//RECON2   DD DSN=IMSVS.RECON2,DISP=SHR
//RECON3   DD DSN=IMSVS.RECON3,DISP=SHR
//CARDIN   DD *
DBDSKILLINV
PSB*       OUTPUT    UL
PARTITION SKINVP1 3
SEGSTAT PART
END
/*
//HSSROPT  DD *
PARTINFO DEF,ACC
/*
//PRNTOUT  DD SYSOUT=A
//OUTPUT   DD DSN=IMSDB.UNLOADDS,DISP=(,KEEP),UNIT=TAPE
```

In this example:

- The database that is unloaded is specified on the DBD parameter.
- DBRC=Y is specified so that DBRC is used.
- The DD statements for RECON data sets are specified.
- The DBD statement specifies the DBD name.
- The PSB statement shows that the unloaded data set is specified by the OUTPUT DD statement and that the output records are written in the UL format.
- The partitions to be unloaded are three consecutive partitions. The first is the partition SKINVP1, which is specified on the PARTITION control statement in the CARDIN DD statement. All data sets for the selected partitions are dynamically allocated by HSSR Engine.
- The 'SEGSTAT PART' statement produces the partition-wide Segment Statistics report.
- The 'PARTINFO DEF,ACC' statement in HSSROPT DD produces the HALDB Partition Definition report and the HALDB Partitions Accessed report.

Processing HALDBs with your HSSR application program

You can use your HSSR application program to process a partitioned database.

Restriction: A user HSSR application program cannot process only the selected partitions; HALDB is processed as an entire database. However, by specifying the HALDB control statement on the DFSHALDB DD statement, you can select a single HALDB partition to be processed.

Subtopics:

JCL requirements

In an HSSR JCL for processing HALDBs, the following requirements must be met:

- DBRC must be used.
- The DD statements for RECON data sets must be specified or dynamically allocated.
- No DD statements for HALDB data sets must be specified.

Control statements

The following control statements, in addition to the standard control statements, can be used when you are processing a partitioned database:

- PARTINFO control statement in the HSSROPT data set
- CALLSTAT control statement in the HSSROPT data set

- PARTPROC control statement in the HSSRCABP data set

Examples

Example 1: Coding JCL for your HSSR application program

You can use FABHDLI, FABHDBB, or FABHULU as the cataloged procedure to run your application program. In the following example, FABHDLI is used.

Here, assume that the application program HSSRAPPL processes a PHDAM database and the second DB PCB in the PSB PSBAPPL1 is used to access the PHDAM database from the application program through HSSR calls.

Assume also that the partitions are accessed sequentially.

DBRC=Y is specified so that DBRC is used. Also, the DD statements for RECON data sets are specified. No DD statement for database data sets for the PHDAM to be processed is specified, because all data sets are dynamically allocated by HSSR Engine.

Because no HSSRCABP DD statement is coded, the default CAB buffering parameters are used for the job. Because partitions are accessed sequentially, no PARTPROC control statement needs to be specified in HSSRCABP DD.

```
//       EXEC FABHDLI,MBR=HSSRAPPL,PSB=PSBAPPL1,DBRC=Y
//RECON1   DD DSN=IMSVS.RECON1,DISP=SHR
//RECON2   DD DSN=IMSVS.RECON2,DISP=SHR
//RECON3   DD DSN=IMSVS.RECON3,DISP=SHR
//HSSROPT  DD *
HSSRPCB 002
/*
//OUTPUT   DD DSN=TESTDS.HSSRAPPL.OUTPUT,DISP=OLD
```

Example 2: Coding HSSRCABP for sequential access for HALDB

Assume that the same application program and the same PSB as in Example 1 are used.

Assume also that the data set group A of the partitions PART3 and PART5 of the PHDAM database are extremely disorganized and you want to allocate more sequential buffers than the default for these data sets. Then, for example, you would code the CABDD and NBRSRAN control statements in HSSRCABP DD as follows:

```
//       EXEC FABHDLI,MBR=HSSRAPPL,PSB=PSBAPPL1,DBRC=Y
//RECON1   DD DSN=IMSVS.RECON1,DISP=SHR
//RECON2   DD DSN=IMSVS.RECON2,DISP=SHR
//RECON3   DD DSN=IMSVS.RECON3,DISP=SHR
//HSSROPT  DD *
HSSRPCB 002
/*
//OUTPUT   DD DSN=TESTDS.HSSRAPPL.OUTPUT,DISP=OLD
//HSSRCABP DD *
CABDD PART3A
NBRSRAN 10
CABDD PART5A
NBRSRAN 20
/*
```

If you want to change NBRSRAN for all data sets of data set group A, you can code as follows:

```
//       EXEC FABHDLI,MBR=HSSRAPPL,PSB=PSBAPPL1,DBRC=Y
//RECON1   DD DSN=IMSVS.RECON1,DISP=SHR
//RECON2   DD DSN=IMSVS.RECON2,DISP=SHR
//RECON3   DD DSN=IMSVS.RECON3,DISP=SHR
//HSSROPT  DD *
HSSRPCB 002
/*
//OUTPUT   DD DSN=TESTDS.HSSRAPPL.OUTPUT,DISP=OLD
//HSSRCABP DD *
CABDD 'PART*A'
NBRSRAN 10
/*
```

Remember to specify the single quotation marks on both side of the operand of the CABDD statement so that the operand can be treated as a wildcard specification.

For details of HSSRCABP control statements, see "HSSRCABP control statements" on page 217.

Example 3: Coding HSSRCABP for random access to HALDB

Although not recommended, random access to multiple partitions of an HALDB is supported.

Assume that the same PSB as in Example 1 is used; but assume here that the application HSSRAPPL accesses the partitions of the PHDAM database PARTDB1 in random. You need to code the PARTPROC control statement for the database in HSSRCABP DD.

If no more than two partitions are accessed at a time, you should code the PARTPROC statement as follows:

```
//       EXEC FABHDLI,MBR=HSSRAPPL,PSB=PSBAPPL1,DBRC=Y
//RECON1   DD DSN=IMSVS.RECON1,DISP=SHR
//RECON2   DD DSN=IMSVS.RECON2,DISP=SHR
//RECON3   DD DSN=IMSVS.RECON3,DISP=SHR
//HSSROPT  DD *
HSSRPCB 002
/*
//OUTPUT   DD DSN=TESTDS.HSSRAPPL.OUTPUT,DISP=OLD
//HSSRCABP DD *
PARTPROC PARTDB1 R
CABDD 'PART%A'
RANSIZE 8
NBRSRAN 4
NBRDBUF 16
REFT4 12
/*
```

If no more than three partitions are accessed at a time, you must code the third operand of the PARTPROC statement. See the following example.

```
//       EXEC FABHDLI,MBR=HSSRAPPL,PSB=PSBAPPL1,DBRC=Y
//RECON1   DD DSN=IMSVS.RECON1,DISP=SHR
//RECON2   DD DSN=IMSVS.RECON2,DISP=SHR
//RECON3   DD DSN=IMSVS.RECON3,DISP=SHR
//HSSROPT  DD *
HSSRPCB 002
/*
//OUTPUT   DD DSN=TESTDS.HSSRAPPL.OUTPUT,DISP=OLD
//HSSRCABP DD *
PARTPROC PARTDB1 R 3
CABDD 'PART%A'
RANSIZE 8
NBRSRAN 4
NBRDBUF 16
REFT4 12
/*
```

For details of HSSRCABP control statements, see "HSSRCABP control statements" on page 217.

Migration unload and fallback unload

Unloading an HDAM or HIDAM database in order to change the DL/I access method of the database to PHDAM or PHIDAM is called *migration unload*. The process of restoring HDAM or HIDAM databases that were migrated to PHDAM or PHIDAM is known as *fallback*. Unloading a PHDAM or PHIDAM database for fallback is called *fallback unload*.

FABHURG1 can be used for both purpose. FABHFSU can be used only for migration unload.

Migration unload

You can use the FABHURG1 or FABHFSU unload utility to migrate nonpartitioned databases to HALDBs.

The procedure for migrating nonpartitioned databases to HALDBs is described in *IMS Database Administration*. The FABHURG1 and the FABHFSU unload utilities can be used to replace the IMS

HD Reorganization Unload utility (DFSURGU0) in the migration scenario. Note, however, the following considerations:

- FABHURG1 and FABHFSU can be used for the migration unload of HDAM or HIDAM databases.
- FABHURG1 and FABHFSU do not support the migration unload of secondary indexes. Secondary indexes must be unloaded by the IMS HD Reorganization Unload utility.
- FABHURG1 and FABHFSU do not support the migration unload of HISAM databases. HISAM database must be unloaded by the IMS HD Reorganization Unload utility.

If you want to unload a database by several job steps that run in parallel, the FABHFSU Parallel Scan Facility (PSF) can be used.

Subtopics:

Requirements for FABHURG1

To unload an HDAM or HIDAM database for a migration, specify the MIGRATE control statement in SYSIN DD. For more information about the control statement, see "MIGRATE control statement" on page 43.

FABHURG1 must be run in a ULU region when migration unload is designated.

Requirements for FABHFSU

To unload an HDAM or HIDAM database for a migration, specify the MI option for the PSB control statement in CARDIN DD. For more information about the control statement, see "PSB control statement" on page 135.

FABHFSU must be run in a ULU region when migration unload is designated.

Restrictions

- Migration unload of the secondary index is not supported.
- Migration unload of the HISAM database is not supported.
- If PTR=H or PTR=HB is defined as the parent segment of virtual logical child, migration unload of the database is not supported.

Considerations

DFSVSAMP DD
If you want to get better performance, when a logical child is defined in the input database, code an appropriate number of IMS buffer pools in DFSVSAMP DD of your JCL. For details, see Chapter 4, "Basic job control language," on page 29.

Database Tuning Statistics
Statistics for the segment length and the number of I/Os for virtual logical segment types are not produced.

Hard-copy trace
For the virtual logical child, the segment prefix reported in a call trace is the segment prefix of the paired real logical child.

JCL example for migration unload

The following is a JCL example to create an unloaded data set that can be used in migrating an HDAM or HIDAM database. The example uses the IBM-supplied FABHULU cataloged procedure.

Assume that the database is an HDAM database and consists of three data set groups.

The database data sets that are to be unloaded are defined by HDAMDD1, HDAMDD2, and HDAMDD3 DD statements; the unloaded data set is defined by the SYSUT2 DD statement.

The MIGRATE control statement is specified in the SYSIN DD, which indicates that this is a migration unload.

```
//    EXEC FABHULU,MBR=FABHURG1,DBD=HDAMDBD
//HDAMDD1  DD DSN=TESTDS.HDAMDS1,DISP=SHR
//HDAMDD2  DD DSN=TESTDS.HDAMDS2,DISP=SHR
//HDAMDD3  DD DSN=TESTDS.HDAMDS3,DISP=SHR
//SYSIN    DD *
MIGRATE
/*
//SYSPRINT DD SYSOUT=A
//SYSUT2   DD DSN=MIGDS1.MIGULDS,DISP=(,CATLG),UNIT=TAPE,
//            VOL=SER=MIGDS1
```

Migration unload: Exit routine FABHKEYX for distributing unload records

In the migration unload, you can use the IBM-provided exit routine FABHKEYX to distribute the unload records to multiple unload files according to the root key values.

The multiple unload files enable the Reload and the optional PSSR SORT for two or more HALDB partitions to run in parallel processes, which will reduce the elapsed time for migration.

Note: The exit routine FABHKEYX is used for unloading the HALDB partitions and changing the high key.

To use this exit, you must specify FABHKEYX as an exit routine name on the EXIT control statement in the SYSIN data set and prepare a list of the DD names and the high key values for the unload files in the FABHKEYX data set.

In each unload file, the correct header and trailer records are added.

Restriction: If the root key is compressed, the control statement EXIT FABHKEYX cannot be specified with the DECN control statement.

Subtopics:

- “FABHKEYX data set” on page 109
- “JCL example for migration unload with FABHKEYX exit” on page 110

FABHKEYX data set

The FABHKEYX data set contains 80-byte fixed-length records. The FABHKEYX exit routine reads this data set that contains a list of the DD names and the high key values to distribute the unload records.

The entries of the list must be in the following format and must be listed in ascending order of the high key values.

```
0........1.........2.........3.........4.........5.........6...
12345678901234567890123456789012345678901234567890123456789012345678901234567890...
DDname     KeyString
```

Position
Description

1-8
Code the output DD name.

This 8-character entry specifies the name of the DD statement for each unload files. The format of each data set must be same as the SYSUT2 DD.

10-

Code the key string.

This variable-length string specifies a high key value. The key values must be enclosed by quotation marks and preceded by the letter C or X: C indicates the character values, and X indicates the hexadecimal values. When the key string is long, you can specify it on multiple lines as follows:

```
DDNAME01 C'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
C'AAAAAAAAAAAA'
```

Hexadecimal values must be even-length. If the high key is longer than the root key, the later extra will be ignored. If the high key is shorter, the high key value is padded with X'FF's up to the defined root key length.

JCL example for migration unload with FABHKEYX exit

The following is a JCL example to distribute the unload records for migration to HALDB to four unload files by using the FABHKEYX exit routine.

```
//    EXEC FABHULU,MBR=FABHURG1,DBD=HDAMDBD
//HDAMDD1  DD DSN=TESTDS.HDAMDS1,DISP=SHR
//HDAMDD2  DD DSN=TESTDS.HDAMDS2,DISP=SHR
//HDAMDD3  DD DSN=TESTDS.HDAMDS3,DISP=SHR
//SYSIN    DD *
MIGRATE
EXIT FABHKEYX
//SYSPRINT DD SYSOUT=A
//SYSUT2    DD DUMMY
//FABHKEYX DD *
ULFPART1 C'2999999'
ULFPART2 C'5999999'
ULFPART3 C'9999999'
ULFPART4 X'FF'
//ULFPART1   DD DSN=MIGDS1.MIGULDS.ULFPART1, ...
//ULFPART2   DD DSN=MIGDS2.MIGULDS.ULFPART2, ...
//ULFPART3   DD DSN=MIGDS3.MIGULDS.ULFPART3, ...
//ULFPART4   DD DSN=MIGDS3.MIGULDS.ULFPART4, ...
```

Tips:

- You can define the SYSUT2 DD as DUMMY to reduce the elapsed time for the I/O operations.
- It is recommended that you specify X'FF' for the last DD name in the FABHKEYX data set not to throw away segments.

Parallel migration unload

The FABHFSU Parallel Scan Facility (PSF) can reduce the elapsed time of migration unload of a large database that has external logical relationships by scanning the predefined portions of the database separately. This method is called *parallel migration unload*.

For more information about the FABHFSU Parallel Scan Facility, see Chapter 10, “Parallel Scan Facility of FABHFSU,” on page 121.

Subtopics:

Example of parallel migration unload of HIDAM

In the following JCL examples, an HIDAM database USRHIDAM is migrated to a PHIDAM database. The HIDAM database has three partitions with high keys, C'2999999999', C'5999999999', and C'9999999999'.

Figure 18 on page 111 shows a JCL to run the FABHPSFC program to create a scan control that is named SCANCNTL. The CARDIN statements specify the control of three phases of the migration unload. MI on

the PSB control statement specifies to run migration unload. Columns 19 and 20 of the CTL control statement specify the total number of the unload phases, and the HKY control statements specify the high keys of the partition as node points of unload phases.

The LRECL for the CNTLDD data set must be greater than (834 * *the number of phases*) + 432

Figure 19 on page 111 shows a JCL for phase 1 of the unload step. For phases 2 and 3, modify the phase number 01 to 02 or 03. Each FABHFSU job produces a unload data set that is specified by the OUT1 DD statement. Each unload data set contains a header record, and the unload data set can be used as an input for IMS High Performance Load.

Notes:

- If a logical child is defined in the input database, and you want better performance of unload, code an appropriate number of IMS buffer pools in DFSVSAMP DD of the JCL. For details, see "Basic JCL requirements" on page 30.
- The FABHFSU job steps, except for the last phase, return the code of 08 when the processes end successfully.
- The load utility of IMS High Performance Load returns the code of 04 because a trailer record is not contained.

```
//*-------------------------------------------------------------------------
//*    FABHPSFC - CREATE SCAN CONTROL DATA SET
//* -------------------------------------------------------------------------
//PSFCTL   EXEC PGM=FABHPSFC
//STEPLIB  DD DISP=SHR,DSN=HPS.SHPSLMD0
//         DD DISP=SHR,DSN=IMSVS.SDFSRESL
//IMS      DD DISP=SHR,DSN=USER.DBDLIB
//CNTLDD   DD DSN=TEMPDS.FSCNTL,
//            DISP=(NEW,CATLG,DELETE),
//            UNIT=SYSDA,SPACE=(TRK,(4,1)),VOL=SER=TSTVOL,
//            DCB=(BLKSIZE=4096,LRECL=4092,RECFM=VB)
//PRNTOUT  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//CARDIN   DD *
*........1.........2.........3.........4.........5.........6.........7.........8
*2345678901234567890123456789012345678901234567890123456789012345678901234567890
DBDUSRHIDAM
PSB*       OUT1      MI
CTLSCANCNTL209936503Y
HKYC'29999999'
HKYC'59999999'
HKYC'99999999'
END
/*
```

Figure 18. FABHPSFC JCL for parallel migration unload of HIDAM

```
//*-------------------------------------------------------------------------
//*    FABHFSU  - UNLOAD DATABASE (PHASE 1 OF 3)
//* -------------------------------------------------------------------------
//HPULFSU EXEC FABHULU,MBR=FABHFSU,DBD=USRHIDAM,DBRC=N,
//             COND=EVEN,
//             DBTLMD=HPS.SHPSLMD0,
//             IMSDSN=IMSVS.SDFSRESL,
//G.IMS    DD DISP=SHR,DSN=USER.DBDLIB
//DFSVSAMP DD DISP=SHR,DSN=USER.DFSVSAMP
//DFSSTAT  DD SYSOUT=*
//PRNTOUT  DD SYSOUT=*
//CNTLDD   DD DISP=SHR,DSN=TEMPDS.FSCNTL,
//OUT1     DD DSN=TEMPDS.USRHIDAM.DFSURGU0.PHASE01,          <---specify phase
//            DISP=(NEW,CATLG),UNIT=SYSDA,
//            SPACE=(CYL,(100,25)),VOL=SER=TSTVOL
//CARDIN   DD *
*........1.........2.........3.........4.........5.........6.........7.........8
*2345678901234567890123456789012345678901234567890123456789012345678901234567890
PSCSCANCNTLUSRHIDAM0301                                      <---specify phase
END
/*
```

Figure 19. FABHFSU JCL for parallel migration unload of HIDAM

Example of parallel migration unload of HDAM

In the following JCL example, an HDAM database USRHDAM is migrated to a PHDAM database. The HDAM database has three partitions with high keys, X'2FFFFFFF', X'5FFFFFFF', and X'FFFFFFFF'.

The secondary index USRSIX01, whose search field is the root key of USRHDAM, is used to unload the database in the ascending order of the root key.

The following figure shows JCL to run the FABHPSFC program. The secondary index name is specified in the DBD control statement. If a secondary index is not defined, the relative block number of the CI or block can be specified as a node point in the NPT control statement.

For the JCL of each phase in the unload step, see Figure 19 on page 111.

```
//*-------------------------------------------------------------------------
//*    FABHPSFC - CREATE SCAN CONTROL DATA SET
//* -------------------------------------------------------------------------
//PSFCTL   EXEC PGM=FABHPSFC
:
//CARDIN   DD *
*........1.........2.........3.........4.........5.........6.........7.........8
*23456789012345678901234567890123456789012345678901234567890123456789012345678
DBDUSRHDAM USRSIX01
PSB*       OUT1      MI
CTLSCANCNTL209936503Y
HKYX'2FFFFFFF'
HKYX'5FFFFFFF'
HKYX'FFFFFFFF'
END
/*
```

Figure 20. FABHPSFC JCL for parallel migration unload of HDAM

Step 1 and step 4 that are described in "Unloading a database with FABHFSU in PSF mode" on page 122 are optional. For information about each control statement, see "FABHPSFC CARDIN input data set" on page 129.

Fallback unload

You can use the FABHURG1 unload utility to restore HDAM or HIDAM databases that were migrated to PHDAM or PHIDAM.

The procedure for the fallback unload of HALDBs is described in *IMS Database Administration*. The FABHURG1 unload utility can be used as a replacement of the IMS HD Reorganization Unload utility (DFSURGU0) in the fallback scenario. Note, however, the following considerations:

- FABHURG1 can be used for the fallback unload of PHDAM or PHIDAM databases.
- FABHURG1 does not support the fallback unload of partitioned secondary indexes (PSINDEXs). PSINDEXs must be unloaded by the IMS HD Reorganization Unload utility.
- FABHFSU does not support the fallback unload.

Subtopics:

- "Requirements" on page 112
- "JCL example for fallback unload" on page 113

Requirements

Specify the FALLBACK control statement in SYSIN DD to unload a PHDAM or PHIDAM database for a fallback. For more information about the control statement, see "FALLBACK control statement" on page 42.

When fallback unload is designated, FABHURG1 must be executed in the ULU region.

JCL example for fallback unload

To create an unloaded data set that can be used for a fallback of a PHDAM or PHIDAM database, you can use the JCL shown in the following figure. The example uses the IBM-supplied FABHULU cataloged procedure.

Assume that the database is a PHDAM database and consists of three data set groups.

The DBRC=Y option must be specified on the EXEC statement. The unloaded data set is defined by the SYSUT2 DD statement. You do not need to code any DD statements for database data sets, because all data sets are dynamically allocated by HSSR Engine.

The FALLBACK control statement is specified in the SYSIN DD, which indicates that this is a fallback unload.

```
//    EXEC FABHULU,MBR=FABHURG1,DBD=PHDAMDBD,DBRC=Y
//RECON1   DD DSN=IMSVS.RECON1,DISP=SHR
//RECON2   DD DSN=IMSVS.RECON2,DISP=SHR
//RECON3   DD DSN=IMSVS.RECON3,DISP=SHR
//SYSIN    DD *
FALLBACK
/*
//SYSPRINT DD SYSOUT=A
//SYSUT2   DD DSN=FBKDS1.FBKULDS,DISP=(,CATLG),UNIT=TAPE,
//            VOL=SER=FBKDS1
```

Chapter 9. Utility options for unloading corrupted databases

The unload utilities of IMS High Performance Unload provide certain options for unloading corrupted databases.

The following topics discuss the problems you might encounter with corrupted databases, and the options available for repairing the databases.

Topics:

Rules for unloading corrupted databases

To unload corrupted databases, you can use the SKERROR option of FABHURG1 or the pointer bypass option of FABHFSU.

Before unloading corrupted databases, learn the differences between the two methods to determine which method to use.

Subtopics:

DIAGG option for FABHFSU and FABHURG1

The DIAGG option is automatically invoked by FABHFSU when the pointer bypass option is specified. For FABHURG1, DIAGG is specified on a control statement in the HSSROPT data set.

The DIAGG option provides the following diagnosis information (in addition to technical information addressing highly skilled database specialists):

- Information about the last segment that was successfully unloaded before encountering the database error:
  - The segment name
  - The concatenated PCB key feedback area
- Information about the first segment that was successfully unloaded after the database error:
  - The segment name
  - The concatenated PCB key feedback area
  - The segment data
- The name of those segment types, for which some segment occurrences might be missing on the unloaded data set.

The preceding information can be used to locate the error and to see which segment types might be missing from a segment. Using this information, you can then determine which lost database segments should be inserted after a successful reload into the database.

For each incorrect pointer, the DIAGG option might write more than 4000 print lines on the HSSRTRAC data set. To avoid S722 or SB37 system abends, allocate the HSSRTRAC data set on a tape or allow the job to produce a large number of printed lines. (One example would be through the use of a OUTLIM JCL parameter.)

Sequence check option and sequence error option for FABHFSU

For FABHFSU, it is recommended that you specify (on the DBD control statement) the sequence check option as Y and the sequence error option as B. With these options, HSSR Engine performs more extensive error-checking while unloading a corrupted database.

When the default sequence error option of A is used, the unloaded database data set might contain segments that are not in sequence. This can prevent a successful reload.

KEYCHECK GG option for FABHURG1

For FABHURG1, the SKERROR option requires that you consider activating the KEYCHECK GG option. The KEYCHECK GG option lets HSSR Engine perform more error-checking while unloading a corrupted database. However, do not use the KEYCHECK GG option if you are going to reload the unloaded database and if you do not want to lose any segments in the original database.

Note: When SKERROR option is used without KEYCHECK GG option, the unloaded database might contain segments that are not in key sequence.

Before using the new database

Before using the new database created through the unload and reload process, determine how many segments were skipped or lost during the unloading process.

This can be achieved by:

1. Reloading the database without destroying the original corrupted database.
2. Running HD Pointer Checker utility of IBM IMS High Performance Pointer Checker for z/OS (5655-K53) on the original corrupted database.
3. Comparing the number of occurrences of each segment type in the original corrupted database (shown by HD Pointer Checker utility) to the segment statistics (provided in HSSRSTAT and in PRNTOUT). PRNTOUT shows how many segments have been successfully unloaded.

By combining information about the number of skipped or lost segments with the DIAGG key feedback information, you can decide whether the new database is acceptable. Be sure that you run the HD Pointer Checker utility on the new database to confirm that the new database is free of IMS technical errors.

What you must do after a "crash"

After a "crash," the corrupted databases should be recovered with standard IMS recovery procedures such as Emergency Restart, Database Backout, and Database Recovery utility. An IMS High Performance Unload's unload utility with the SKERROR or pointer bypass option should be run only after these recovery procedures are completed.

Neither the SKERROR nor pointer bypass option copes with DASD I/O errors. The use of either option requires a prior recovery from DASD I/O errors through standard recovery procedures.

Databases involved in logical relationships

The following concerns apply to databases involved in logical relationships:

- Unloading and reloading of huge databases that are heavily involved in logical relationships might require many hours for the scanning, unloading, reloading, prefix resolution, prefix update, and image copy.

For such databases, consider using the Database Repair Facility of IMS High Performance Pointer Checker as an alternative to the unload and reload approach.

- If you suspect that logical pointers (logical parent pointers, logical child pointers, logical twin pointers, or counter fields) are incorrect, run the Prereorganization utility, using the DBIL= control statement. Unload and reload all related databases. (Refer to the description of the HD Reorganization Unload utility DFSURGU0 in *IMS Database Utilities*.)

  When you unload a corrupted database with a logical pointer error in a logical child segment with the virtual LPCK defined, you must specify the BLDLPCK control statement in the HSSROPT data set.
- If the corrupted database is involved in a logical relationship, FABHFSU cannot be used (without postprocessing of its output by user programs) for a successful unload/reload/prefix resolution and prefix update process if one of the following types of segment occurrence is skipped or lost during the unload:
  - A logical parent segment that has one or more logical children
  - A logical child segment that is physically paired

  For such cases, investigate the use of the Database Repair Facility of IMS High Performance Pointer Checker.

Using the SKERROR option for FABHURG1

You can use the SKERROR option of the FABHURG1 utility to unload databases even if the database contains incorrect physical pointers or incorrect HISAM records.

If a database contains incorrect physical HD pointers or HISAM records that you cannot correct by standard IMS recovery procedures such as database backout or database recovery, you can correct the errors with the Database Repair Facility of IMS High Performance Pointer Checker or AMASPZAP. The procedure requires a database specialist who understands how ZAPs can repair the database.

As an alternative, IMS High Performance Unload's HSSR Engine offers the SKERROR option. SKERROR allows FABHURG1 to unload HIDAM, HDAM, PHIDAM, PHDAM, or HISAM databases, even if they contain incorrect physical pointers or incorrect HISAM records. SKERROR skips the processing of incorrect pointers or records, which omits some occurrences of segments on the unloaded database data set. You can then use the unloaded database as input to the standard IMS HD Reorganization Reload utility or a compatible utility to reconstruct a new database. Unless the corrupted database involves logical relationships, your new database is now error-free from a technical IMS standpoint.

Example: Using FABHURG1 to unload a corrupted database

To unload a corrupted database, you can use the following JCL. The unloaded data set is defined by the SYSUT2 DD statement; the database that is unloaded is defined by the HDAM DD statement. HSSROPT options SKERROR, KEYCHECK GG, and DIAGG are specified. OUTLIM is specified on the HSSRTRAC DD statement.

```
//   EXEC  FABHULU,MBR=FABHURG1,DBD=USERDBD
//HSSROPT  DD *
SKERROR 10
KEYCHECK GG
DIAGG
/*
//HDAM     DD DSN=TESTDS.HDAM,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUT2   DD DSN=TESTDS.HDUNLD,DISP=(,CATLG),UNIT=TAPE,
//            VOL=SER=xxxxxx
//HSSRTRAC DD SYSOUT=A,OUTLIM=10000000
```

Figure 21. FABHURG1 JCL to unload a corrupted database

Using the pointer bypass option for FABHFSU

You can use the pointer bypass option, instead of SKERROR, for FABHFSU to unload a corrupted database.

You can select either one of two available options for pointer bypass option. Both options enable FABHFSU to continue processing a database that contains bad pointers. Because most of the data is good (and if it is possible to obtain an unload of the database), its reconstruction can be aided by reorganizing the database.

The logic invoked by this option treats an incorrect pointer as X'00000000' (end of chain), and FABHFSU proceeds with the next logical pointer path. This means that one or more segments are bypassed and are not contained in any output data set.

If the database contains logical relationships, prefix resolution might fail on a subsequent reload because a logical parent segment has been bypassed. This can be corrected by locating the logical child records that refer to the logical parent and eliminating them from the unloaded data set. After reload, the relationships can then be reestablished with maintenance programs.

Option 1

Option 1 (indicated by a 1 in column 28 of the DBD control statement) invokes the pointer bypass option using normal retrieval techniques. The usual retrieval methods are as follows:

HIDAM and PHIDAM with twin backward on root segment
The primary index is used to find the first root segment. The root twin forward pointer is then followed. If a pointer error is encountered in the root twin forward pointer, HSSR Engine attempts to locate the next root via the primary index.

HDAM and PHDAM
Retrieval begins at the first RAP and follows the root twin forward until a "0" pointer or pointer error is encountered. The retrieval then continues at the next RAP.

Option 2

Option 2 (indicated by a 2 in column 28 of the DBD control statement) is applicable only to HIDAM and PHIDAM. It forces FABHFSU to use the index (rather than the root twin forward pointers) to unload the database.

Note: Specifying a FABHFSU pointer bypass option automatically activates both the SKERROR option and the DIAGG option for HSSR Engine.

Example: Using FABHFSU to unload a corrupted database

To unload a corrupted database by using FABHFSU in the standard mode, you can use the JCL shown in the following figure. The FABHDLI procedure used specifies the PSB named OUT2PSB.

The OUT1 and OUT2 DD statements define two unloaded data sets. The database that is unloaded is defined by the SKILHDAM DD statement.

The CARDIN DBD statement specifies the IMS database to FABHFSU. It specifies that sequence checking is to be done, sequence errors are to be processed, and diagnostic data is to be printed. The pointer bypass option is to be activated. Incorrect segments are not unloaded.

The CARDIN PSB statements specify the two output data sets to FABHFSU. OUT1 specifies a data set unloaded in the IMS HD Reorganization Unload format. Asterisk (*) indicates that all segments defined in the DBD are unloaded. OUT2 specifies the VB-format output data set. Only segment types defined in the first DB PCB of the PSB OUT2PSB for the SKILLINV database are unloaded. A user exit routine, OUT2EXIT, is specified.

```
//    EXEC  FABHDLI,MBR=FABHFSU,PSB=OUT2PSB
//CARDIN    DD  *
DBDSKILLINV         YAY     1
PSB*       OUT1       UL
PSBOUT2PSB OUT2       VBOUT2EXIT
END
/*
//SKILHDAM  DD  DSN=SKIL.INV.DB,DISP=SHR
//PRNTOUT   DD  SYSOUT=A
//OUT1      DD  DSN=TESTDS.UNLOAD1,DISP=(,KEEP),UNIT=TAPE
//OUT2      DD  DSN=TESTDS.UNLOAD2,DISP=(,KEEP),UNIT=TAPE
```

Figure 22. FABHFSU JCL to unload a corrupted database

Chapter 10. Parallel Scan Facility of FABHFSU

By using the FABHFSU Parallel Scan Facility (PSF), you can significantly reduce the amount of elapsed time that is required to scan multivolume databases.

FABHFSU provides this facility for the compatibility with JCL written for FSU II.

The following topics describe how to use the Parallel Scan Facility of FABHFSU.

Topics:

Overview of Parallel Scan Facility

The FABHFSU Parallel Scan Facility (PSF) makes possible a significant reduction in the amount of elapsed time required to scan multivolume databases. It does so by scanning separate, predefined portions of the database simultaneously.

A PSF run results in multiple output data sets that must be combined to represent the output of a complete scan. PSF supports only HDAM and HIDAM databases; PHDAM and PHIDAM databases are not supported.

Note: For the explanation of how to unload partitions of a PHDAM or PHIDAM database in parallel, see Chapter 8, "Methods for processing High Availability Large Databases," on page 97.

The FABHFSU utility in PSF mode consists of the following four programs:

FABHPSFM
: Provides assistance to the user in determining how to divide the database into portions to be scanned by PSF phases.

FABHPSFC
: Processes the control statements that define the scan and builds a scan control data set that is used by the remaining functions to obtain and record information about the scan.

FABHFSU
: In the PSF mode, FABHFSU performs individual scan phases, which can run separately or concurrently to unload a multivolume database. In the PSF mode, the information is obtained from the scan control data set created by FABHPSFC.

FABHPSFS
: Creates summarized statistics and data set concatenation sequences. For an unload format, prepares the header and trailer records required. This is the last program to be run.

Because all facilities provided for a typical FABHFSU run are available for individual PSF scan phases, PSF can benefit as follows:

- Earlier detection of severe errors that must not be bypassed. This is especially advantageous when the error is toward the end of the database or if there are multiple errors.
- Reduced recovery time from database errors. This is possible because only the affected scan phases need to be rerun. Therefore, other scan phases (even those in which the beginning node is located beyond the point of error in the database) can continue to run while the errors are being analyzed or corrected.

- Reduced recovery time from a permanent write error at unload time or a permanent read error at reload time. This is possible because only affected scan phases need to be rerun. If dual outputs are being created, this would not be relevant unless both formats encountered permanent errors or damages.

Restrictions

The restrictions that apply to FABHFSU in standard mode also apply to FABHFSU in PSF mode. For those restrictions, see “Restrictions for IMS High Performance Unload” on page 22.

Additionally, the following restrictions apply to PSF mode:

- FABHFSU does not support the unloading of an HISAM database under PSF mode.
- PSF mode does not support IMS-managed ACBs.

Unloading a database with FABHFSU in PSF mode

To unload a database with FABHFSU in PSF mode, you must run four programs.

Procedure

1. Optional: Run the FABHPSFM program as an MVS batch job to obtain information about the database extents.

   This step is optional. The FABHPSFM program is provided as an aid in determining phase limit definitions.
2. Run the FABHPSFC program as an MVS batch job to define the scan parameters and the phase data in the scan control data set.
3. Run the specified number of FABHFSU jobs or job steps in any sequence or, preferably, in parallel.

   Specify the number of FABHFSU jobs to run as an input for FABHFSU.
4. Run the FABHPSFS program as an MVS batch job to obtain the summarized statistics, output data set concatenation sequences, and header and trailer data set (if applicable).

Example

The following figure illustrates the steps that are necessary to run a 3-phase PSF run.

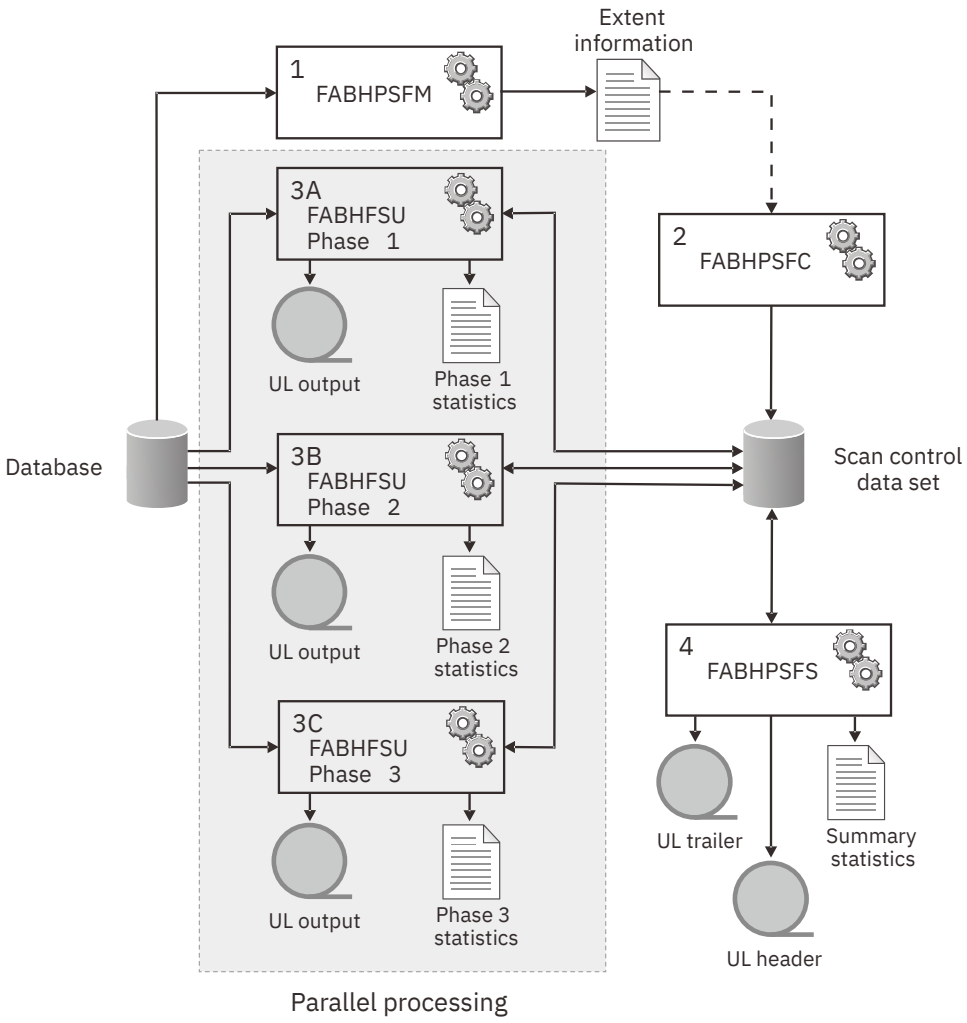

Figure 23. Execution of a 3-phase PSF job

The statistics generated by each phase in step 3 apply only to the portion of the data set processed by that phase. The summarized statistics generated in step 4 apply to the entire database. Step 4 usually does not run until all of the step 3 phases have completed, but it does provide a status report indicating where the individual phases stand. Step 4 can be appended to each step 3 phase as a conditional job step such that it runs only when the last phase completes.

FABHPSFM program

The FABHPSFM program provides information to assist you in determining how to divide the database into portions to be scanned by PSF phases. This information is useful as input for FABHPSFC.

FABHPSFM supplies the following extent information for the primary data set group of an HDAM or HIDAM database:

- Volume serial number of each extent
- Starting relative block for each extent
- Number of relative blocks for each extent
- High allocated block
- First overflow block (HDAM only)
- High used CI (VSAM only)

FABHPSFM JCL requirements

The FABHPSFM program is an MVS batch program. The execution of this program is optional. A FABHPSFM JCL job must satisfy the JCL requirements for the FABHPSFM program.

The following table summarizes the DD statements for FABHPSFM.

Table 19. FABHPSFM DD statements

DDNAME	Use	Format	Need
IMS	Input	Partitioned data set (DSORG=PO)	Required
CARDIN	Input	LRECL=80	Required
ddname3	Input	-	Required
ddname4	Input	-	Optional
PRNTOUT	Output	LRECL=133	Required
SYSUDUMP	Output	-	Optional

EXEC
The EXEC statement must be in the following format:

```
//  EXEC PGM=FABHPSFM
```

IMS DD
This required DD statement defines the library that contains the DBD that describes the database to be scanned.

CARDIN DD
This required DD statement defines the input data set that contains control statements for FABHPSFM (see "FABHPSFM CARDIN input data set" on page 124).

***ddname3* DD**
This required DD statement defines the primary data set of the DBD that is specified in the CARDIN DD statement.

***ddname4* DD**
If the database is an HIDAM database, this DD statement is required to define the index data set of the DBD that is specified in the CARDIN DD statement.

PRNTOUT DD
This required statement defines the output data set to which FABHPSFM writes error messages and segment statistics (see "FABHPSFM PRNTOUT output data set" on page 126). The data set can be defined as:

```
//PRNTOUT DD SYSOUT=A
```

SYSUDUMP DD
This optional DD statement defines the dump data set for this program. The data set can reside on a printer, tape, or direct-access device, or be routed through the output stream.

Related reference
JCL examples for FABHFSU PSF mode
To unload a multivolume IMS database in PSF mode, you can use the JCL jobs shown in the following figures.

FABHPSFM CARDIN input data set

The FABHPSFM CARDIN input data set contains the control statements for FABHPSFM.

You can specify the following control statements for the FABHPSFM CARDIN data set.

Table 20. FABHPSFM control statements

Control statements	Function	Mode
MAP	Directs the FABHPSFM program.	PSF
END	Specifies the end of FABHPSFM CARDIN control statements.	PSF

Format

This data set contains 80-byte fixed-length records. The control statements can be coded in the input stream or accessed as a member of a partitioned data set.

MAP control statement

FABHPSFM is directed by the MAP control statement for the PSF.

Only one MAP control statement can be used.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

MAPdbdname dlidbd  k  oxxxxxxxxyy
```

Position
Description

1
Code the MAP keyword to identify the MAP control statement.

4
Code the database name that describes the physical database to be scanned. This required 8-character field is left-aligned with trailing blanks. The DBD must specify ACCESS=HDAM or HIDAM.

12
This 8-character entry *dlidbd* specifies the database name that describes the primary index of the HIDAM database. This entry is required for HIDAM databases.

A secondary index which points to the HIDAM or the HDAM root segments can be specified instead of the primary index. In this case, the values of the search field of the index are shown in the FABHFSU PSF Extent Mapping report.

20
This 1-character entry *k* indicates the key type when the printed report is necessary. Keys longer than 80 bytes print in hex format regardless of the type specified. Specify one of the following keywords:

C
For character keys (default)

X
For all types other than character keys

23
The 1-character entry *o* selects the key option. Specify one of the following keywords:

Y
All keys within the search delta of the specified relative block are listed.

A
All keys are listed.

N
The select key option is not used.

D
All keys within the search delta of the first block in the first extent on each volume beyond the first volume are listed. (This option scans the entire index once for each volume of the database.)

E
Up to 10 keys that fall within the first 10 blocks of each extent are listed.

V
Up to 20 keys that fall within the first 10 blocks of the first extent of each volume beyond the first volume are listed (default).

24
The 8-digit numeric entry *xxxxxxxx* indicates the relative block for which all keys of that block plus any keys in the blocks within the specified search delta are listed. This value can be in the range of 00000001 - 99999999. This entry is required if position 23 is specified with Y.

32
The 2-digit numeric entry *yy* specifies the number of blocks on either side of the base relative block for which keys are listed. This value can be in the range of 00 - 99. This entry is required if the position 23 is specified with Y or D. For key option Y, the base relative block is indicated by the number in columns 24 - 31; for key option D, each base relative block is the first block of the first extent of each volume beyond the first volume.

END control statement

The END control statement specifies the end of the CARDIN control statements.

```
0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

END
```

Position
Description

1
Code the END keyword to identify the END statement as the last statement of the CARDIN data set.

FABHPSFM PRNTOUT output data set

The FABHPSFM program produces the FABHFSU PSF Control Statement report and the FABHFSU PSF Extent Mapping report for the primary data set group of HIDAM or HDAM database.

Format

The format is 133-byte fixed-length records. When the block size is coded in the JCL, the block size must be a multiple of 133.

FABHFSU PSF Control Statements report

This report contains the CARDIN control statements that were used as input to FABHPSFM.

The following figure shows an example of this report.

```
IMS HIGH PERFORMANCE UNLOAD                          "FABHFSU PSF CONTROL STATEMENTS"                          PAGE:     1
5655-E06                                             DATE: 06/01/2021  TIME: 12.21.37                  FABHPSFM - V1.R2

0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

MAPMSHDP
END
```

Figure 24. FABHFSU PSF Control Statements report

FABHFSU PSF Extent Mapping report

This report helps you in specifying the node point value in the NPT control statement as input for the FABHPSFC program. This report is produced for each output data set that is defined in the CARDIN data set.

The following figure shows an example of the report for an HDAM database.

```
IMS HIGH PERFORMANCE UNLOAD                              "FABHFSU PSF EXTENT MAPPING"                              PAGE:     1
5655-E06                                               DATE: 06/01/2021  TIME: 12.21.37                        FABHPSFM - V1.R2


DDNAME = MSHDP

  VOL          START  NO      START          NO    TRK/  BLK/       HIGH       VSAM HI      HDAM FST
  SER  EXTENT  CYL  CYLS     REL  CI        CIS   CYL   TRK      ALLOC CI     USED CI      OFLO CI    BLK/CI

IMS22C    0   1089   003           0         810   15    18                                              1

IMSDBT    1    159   003         810         810   15    18                                              1

IMSDBT    2    153   003       1,620         810   15    18                                              1

IMSDBT    3   1661   003       2,430         810   15    18                                              1

IMSDBT    4   1664   003       3,240         810   15    18                                              1

IMSDBT    5   1677   003       4,050         810   15    18                                              1

IMSDBT    6   1680   003       4,860         810   15    18        5,669        4,993        5,000       1
```

Figure 25. FABHFSU PSF Extent Mapping report (HDAM database)

The following figure shows another example of the report. This report is for a HIDAM database.

```
IMS HIGH PERFORMANCE UNLOAD                              "FABHFSU PSF EXTENT MAPPING"                              PAGE:     1
5655-E06                                               DATE: 06/01/2021  TIME: 14.48.06                        FABHPSFM - V1.R2


DDNAME = ESDSDATA

  VOL          START  NO      START          NO    TRK/  BLK/       HIGH       VSAM HI      HDAM FST
  SER  EXTENT  CYL  CYLS     REL  CI        CIS   CYL   TRK      ALLOC CI     USED CI      OFLO CI    BLK/CI

IMS31B    0   1609   002           0         540   15    18          539            4                   1


IMS HIGH PERFORMANCE UNLOAD                              "FABHFSU PSF EXTENT MAPPING"                              PAGE:     2
5655-E06                                               DATE: 06/01/2021  TIME: 14.48.06                        FABHPSFM - V1.R2

  VOL
  SER  EXTENT   REL CI       NODE POINT KEY CANDIDATES

IMS31B    0           0           KEY1004000
IMS31B    0           0           KEY106C000
IMS31B    0           0           KEY1084000
IMS31B    0           0           KEY113C000
IMS31B    0           0           KEY1174000
IMS31B    0           0           KEY120C000
IMS31B    0           0           KEY1274000
IMS31B    0           0           KEY129C000
IMS31B    0           0           KEY1344000
IMS31B    0           0           KEY138C000
IMS31B    0           0           KEY1414000
IMS31B    0           0           KEY147C000
IMS31B    0           0           KEY1484000
```

Figure 26. FABHFSU PSF Extent Mapping report (HIDAM database)

The content of the report is as follows:

VOL SER
Volume serial number of each extent

EXTENT
Sequential number of extent

START CYL
Starting cylinder for each extent

NO CYLS
Number of cylinders

START REL BLOCK or START REL CI
Starting relative block number or CIs

NO BLOCKS or NO CIS
Number of blocks or CIs

TRK/CYL
Number of tracks per cylinder

BLK/TRK
Number of blocks per track

HIGH ALLOC BLK or HIGH ALLOC CI
High-allocated block or CI

VSAM HI USED CI
High-used CI (VSAM only)

HDAM FST OFLO BLK or HDAM FST OFLO CI
First overflow block or CI (HDAM only)

BLK/CI
Number of blocks per CI (VSAM only)

REL BLOCK or REL CI
Relative block number or CI number (HIDAM only)

NODE POINT KEY CANDIDATES
Key candidates for the node point (HIDAM only)

For an HDAM database without a secondary index, you should pick the relative block numbers as node point values from the preceding information. Normally the relative block numbers of the lowest extent of each new volume (beyond the first volume and up to the volume that contains the beginning of the overflow area) should be specified as node points.

For a HIDAM database, you can specify the key value from NODE POINT KEY CANDIDATES as a node point value. The number of key candidates listed here depends on how you specify the options of the positions 23, 24 - 31, and 32 - 33 in the MAP control statement.

FABHPSFC program

The Control Data Set Creation program, FABHPSFC, must be run before any scan phase in the PSF mode. FABHPSFC processes all the FABHFSU control statements that define the scan. FABHPSFC also creates a scan control data set to contain information about the scan and control the execution of the multiple scan phases that follow.

The scan is defined in the following terms:

- The database to be scanned (DBD control statement)
- The PSF specifications (CTL control statement)
- The subareas to be scanned in parallel (NPT control statement)
- The output formats to be produced (PSB control statement)

FABHPSFC analyzes the input, creates the necessary information in the scan control data set, and reports the scan specifications and the limits of the indicated phases.

FABHPSFC JCL requirements

The FABHPSFC program is an MVS batch program. A FABHPSFC JCL job must satisfy the JCL requirements for the FABHPSFC program.

The following table summarizes the DD statements for FABHPSFC.

Table 21. FABHPSFC DD statements

DDNAME	Use	Format	Need
IMS	Input	Partitioned data set (DSORG=PO)	Required
CARDIN	Input	LRECL=80	Required

Table 21. FABHPSFC DD statements (continued)

DDNAME	Use	Format	Need
CNTLDD	Output	When the NPT control statement is specified: LRECL >= (484 * *the number of phases*) + 432 When the HKY control statement is specified: LRECL >= (834 * *the number of phases*) + 432 BLKSIZE=LRECL+4 RECFM=VB	Required
PRNTOUT	Output	LRECL=133	Required
SYSUDUMP	Output	-	Optional

EXEC

The EXEC statement must be in the following format:

```
//  EXEC PGM=FABHPSFC
```

IMS DD

This required DD statement defines the library that contains the DBD that describe the database to be scanned.

CARDIN DD

This required DD statement defines the input data set contains control statements for FABHPSFC (see “FABHPSFC CARDIN input data set” on page 129).

CNTLDD DD

This required DD statement defines the output scan control data set, which has the following format:

```
 //CNTLDD DD DSN=fsuscancntl,...
```

DSN is the user data set name assigned when the scan control data set is created by FABHPSFC.

PRNTOUT DD

This required DD statement defines the output data set to which FABHPSFC writes error messages and segment statistics (see “FABHPSFC PRNTOUT output data set” on page 138). The data set can be defined as:

```
//PRNTOUT DD SYSOUT=A
```

SYSUDUMP DD

This optional DD statement defines the dump data set for this program. The data set can reside on a printer, tape, or direct-access device, or be routed through the output stream.

Related reference

JCL examples for FABHFSU PSF mode
To unload a multivolume IMS database in PSF mode, you can use the JCL jobs shown in the following figures.

FABHPSFC CARDIN input data set

The FABHPSFC CARDIN input data set contains the control statements for FABHPSFC.

The following table lists the FABHPSFC control statements.

Table 22. FABHPSFC control statements

Control statements	Function	Mode
CTL	Directs the FABHPSFC program.	PSF
DBD	Identifies database to be scanned.	PSF

Table 22. FABHPSFC control statements (continued)

Control statements	Function	Mode
END	Specifies end of FABHPSFC CARDIN control statements.	PSF
HKY	Specifies the high key value of each unload data set.	PSF
NPT	Defines area of database to be scanned by the PSF.	PSF
PSB	Identifies characteristics of output data sets to be created.	PSF

Format

This data set contains 80-byte fixed-length records. The control statements can be coded in the input stream or accessed as a member of a partitioned data set.

CTL control statement

The CTL control statement directs the FABHPSFC program.

The CTL control statement is required to run FABHPSFC for the PSF. Only one CTL control statement is used.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

CTLpsname  yyyydddxxwcnnh
```

Position
Description

1
Code the CTL keyword to identify the CTL control statement.

4
Code a 1- to 8-character name of the scan control data set for the parallel scan operation. This name is defined in the CNTLDD DD statement, and is used in the entire PSF mode. All scan phases must specify this name in order to gain access to the scan control data set. This 8-character entry is left-aligned with trailing blanks.

12
This required 7-digit numeric entry *yyyyddd* allows the Julian date to be overridden. *yyyy* is the year and *ddd* is the day of the year. This entry specifies the last date that FABHFSU scan phases will be allowed access to the scan control data set unless the date is specifically overridden in the PSC control statement.

19
This required 2-digit numeric entry *xx* indicates the total number of scan phases participating in this parallel scan operation. This value must equal the number of NPT control statements plus 1. The maximum value is shown in the following table.

Table 23. Maximum value for the total number of scan phases

Device type	Value when NPT control statement is specified	Value when HKY control statement is specified	Limiting factor
3380	66	38	Max BLKSIZE (32760)
3390	66	38	Max BLKSIZE (32760)
9345	66	38	Max BLKSIZE (32760)

21

The 1-character entry *w* determines whether messages are written to the operator console when all scan phases have been started or all scan phases have been completed. Use one of the following keywords:

Y
Messages are written.

N
No operator messages are written (default).

22

The 1-character entry *c* determines whether FABHFSU will verify the data set name of the DD statement, which is specified in the PSB control statement for the output data set. The data set name should match the following FABHFSU standard naming convention:

FABHFSU
DSN= *nnnnnnnn*.*scanname*.*wwxx*. PHASE*yy*

FABHPSFS
DSN= *nnnnnnnn*.*scanname*.UL*xx*.HEADER

DSN= *nnnnnnnn.scanname*.UL*xx*.TRAILER

where:

***nnnnnnnn*.**
Optional high-level qualifiers supplied by the user. If the length of the qualifiers is N, N+1 must be specified in position 21.

scanname
Name assigned for the parallel scan operation in positions 4 - 11.

ww
The output format as specified in the respective PSB control statement (HS, UL, VB, VN).

vxx
Relative occurrence number of the PSB control statement (01, 02, or 03) that defines this output.

yy
Number of the phase being run.

Use one of the following keywords:

Y
The data set names used must conform to the naming standard. If not, the execution ends.

N
No checking is performed on data set names (default).

23

This 2-digit numeric entry *nn* specifies the position (relative to 1) where the parallel scan name begins in the data set name of the DD statement. This value can be in the range of 01 - 31. No checking is done before this position. This entry is required if the position 22 is specified with Y.

25

The 1-character entry *h* determines whether the UL header record is to be written to a separate output data set by the FABHPSFS program. Use one of the following keywords:

Y
The UL header record is written.

N
No UL header record is written (default).

DBD control statement

The DBD control statement identifies the database to be scanned. Only one DBD control statement can be used.

The DBD control statement specified in FABHPSFC CARDIN data set applies to all phases of FABHFSU.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

DBDdbname  index   sednnn  b
```

Position
Description

1
Code the DBD keyword to identify the DBD control statement.

4
Code the DBD name.

Code the name of the DBD that describes the physical database to be scanned. This required 8-character field must be left-aligned with trailing blanks. The DBD must specify ACCESS=HDAM, HIDAM, PHDAM, PHIDAM, HISAM, or SHISAM.

12
Code the index name.

The 8-character entry is optional and valid only for running in the ULU region. This entry specifies the name of index DBD that is either the primary index of the HIDAM database or a secondary index of the HIDAM or HDAM database if you want the root segments to be retrieved in the index sequence. For details, see "Considerations for using a secondary index" on page 27.

20
Code this field to activate the sequence check option.

This 1-character entry *s* determines whether to perform sequence check during the unload processing. Specify one of the following keywords:

Y
Perform sequence checking.

N | blank
Do not perform sequence checking (default).

21
Code this field to activate the sequence error option.

The 1-character entry *e* determines whether the output routines bypass or process sequence errors. Specify one of the following keywords:

A | blank
Accept sequence errors (default). A GX status code is returned.

B
Bypass sequence errors. The segment in error, and all of its children, are skipped. A GG status code is returned.

22
Code this field to activate the sequence error print option.

The 1-character entry *d* determines whether diagnostic information is printed for sequence errors in the HSSRTRAC data set. Specify one of the following keywords:

Y | blank
Print diagnostic data on sequence errors (default).

N
Do not print diagnostic data.

23
Code this field to specify the sequence error threshold.

The 3-digit numeric entry *nnn* indicates the number of sequence errors to be allowed before ending the run (the default value is 10). Any number up to 999, with leading or trailing blanks, can be used. If you use the number 999, sequence errors do not cause the run to end.

28
Code this field to activate the pointer bypass option.

The 1-character entry *b* is used to activate the pointer bypass option. The pointer bypass option allows FABHFSU to continue processing a database that contains bad pointers, instead of issuing an abend. The pointer bypass option automatically activates the DIAGG option.

Specify one of the following keywords:

Blank
The pointer bypass option is inactive.

1
This entry invokes the pointer bypass option.

2
This entry forces FABHFSU to use the index, rather than the root twin forward pointers, to unload an HIDAM or PHIDAM database.

See Chapter 9, “Utility options for unloading corrupted databases,” on page 115, for instructions and considerations for using the pointer bypass option.

END control statement

The END control statement specifies the end of the CARDIN control statements.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

END
```

Position
Description

1
Code the END keyword to identify the END statement as the last statement of the CARDIN data set.

HKY control statement

The HKY control statement specifies the high key value of each unload data set that is to be produced in each scan phase. This control statement is used in the parallel migration unload.

For information about parallel migration unload, see “Parallel migration unload” on page 110.

Note: The key value in the *n*-th NPT control statement specifies the low key of (*n*+1)-th unload phase. For parallel migration unload, specify the high key value of each HALDB partition in the HKY control statement.

The HKY control statement must immediately follow the CTL control statement. If no HKY or NPT control statements are provided, the entire database will be scanned in a single phase.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

HKYt'keyvalue'
HKYt'keyvalue.....................................
.....................................................................'
```

Position
Description

1
Code the HKY keyword to identify the HKY control statement.

4
This required 1-character entry *t* identifies the value type. It indicates the format of the node point value field. Use one of the following keywords:

C
Indicates that the high key value field contains keys in character format.

X
Indicates that the high key value field contains keys in hexadecimal display format.

5 - 80
This high key value field is a required field.

- Specify a character value up to 256 characters or a hexadecimal value up to 512 characters on one or more lines.
- The high key value must be enclosed by single quotation marks.
- The hexadecimal string must contain an even number of characters (0-F) and be left-aligned in the field.
- FABHFSU pads the key value with X'FF's up to the key length of the root segment.

Note: If you are using a Data Conversion exit for the database and you want to specify a node point value by using a key value (that is, setting a node point value type of 'C' or 'X'), you must specify the key value in the stored form, not in the application form.

NPT control statement

The NPT node point control statement defines the portion of the database to be scanned for the PSF.

For an indexed database, the node points are specified as keys. For an HDAM database with no secondary index, the node points are defined as relative block numbers of the CIs or blocks in the root addressable area.

The NPT control statement must immediately follow the CTL control statement. If no NPT control statements are provided, the entire database will be scanned in a single phase. One or more NPT control statements can optionally be used.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

NPTtnodepointvalue
NPTtllkeyvalue
```

Position
Description

1
Code the NPT keyword to identify the NPT control statement.

4
This required 1-character entry *t* identifies the node point value type. It indicates the format of the node point value field. Use one of the following keywords:

R
Indicates that the node point value field contains the relative block number of the CI or block.

C
Indicates that the node point value field contains keys in character format.

X
Indicates that the node point value field contains keys in hexadecimal display format.

Note: C and X are valid for the HIDAM database or the HDAM database that is associated with an index. If the secondary index is used to retrieve the root segments, specify the value in the search field of the secondary index instead of the root key.

5–80
This node point value field (76-character field) is required.

- If type R is specified, code the relative block number of the CI or block. The value is up to 8-digit numeric with leading or trailing blanks; it must fall within the limits of the root addressable area.
- If type C is specified, code the length of the key value in positions 5 and 6 (maximum length is 74). Code the key value starting in position 7. You can specify a generic key by entering a key value length less than the key length of the root segment. FABHFSU pads the key value with X'00's up to the key length.
- If type X is specified, code the hexadecimal string starting in position 5. Do not specify length. The hexadecimal string must contain an even number of characters (0-F) and be left-aligned in the field. Field length is determined by the first blank encountered in the field. When generic keys are entered, they are padded as described for character keys.

Note: If you are using a Data Conversion exit for the database and you want to specify a node point value by using a key value—that is, setting a node point value type of 'C' or 'X'—you must specify the key value in the stored form, not in the application form.

PSB control statement

The PSB control statement for the standard mode identifies the characteristics of the output data sets to be created. From one to three PSB statements can be used for each execution of FABHFSU.

The PSB control statement applies to all phases of FABHFSU.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

PSBpsbname ddname1 p#ofuserxit mkxsc    extln
```

Position
Description

1
Code the PSB keyword to identify the PSB control statement.

4-11
Code the PSB name.

The 8-character field contains either an asterisk (*) or the name of the PSB that is coded on the EXEC statement of the JCL.

***
An asterisk (*) on this field indicates that all segments that are described in the DBD are to be processed for this output data set.

When running FABHFSU in ULU region, specify this value.

PSB_name
If the name of the PSB that is coded on the EXEC statement is specified, it indicates that segment types passed to the output routines are controlled through the segment sensitivity of the PCB specified in positions 20 - 21. The name of the PSB must match the name that is supplied in the PARM field of the EXEC statement. The name must be left-aligned with trailing blanks.

Coding the PSB name is valid only for FABHFSU running in the DLI region.

12-19
Code the output DD name.

This 8-character entry specifies the name of the DD statement that defines the data set to be created. An output DD name is required unless the output format specification in positions 22 – 23 is NO. *ddname1* must be left-aligned with trailing blanks.

20-21
Code the PCB number.

The 2-character entry *p#* specifies which database PCB within the PSB is to be used. This entry determines the segments written into the output data set that is identified by the output DD name field (column 12 - 19). The PCB referred to must specify the *dbdname* that is specified in positions 4 – 11 of the DBD control statement.

If the field is left blank, the first database PCB that refers to the DBD specified by the DBD control statement is used.

The number *p#*, if specified, must be the sequence number that starts from the first database PCB in the PSB; TP PCBs are not counted. Specify 1 for the first database PCB in the PSB.

Note: To assure compatibility with FABHFSU of DBT HSSR, it is processed as if 1 was specified when one of values '00', ' 0 ', or '0 ' is specified.

22-23
Code the output format.

The 2-character entry *of* specifies the output format of the data set to be created. Specify one of the following keywords:

HS
HSAM format

UL
IMS HD Reorganization Unload format

MI
IMS HD Reorganization Unload format for migration unload from an HDAM or HIDAM database to a PHDAM or PHIDAM database.

Restrictions:

- This format can be specified only in a ULU region.
- If PTR=H or PTR=HB is defined as the parent segment of virtual logical child, the database is not supported.
- Migration unload of a secondary index or HISAM database is not supported.
- When MI format is selected, two or more PSB control statements cannot be specified.

VB
Variable-length-blocked data set of the selected segments (one segment per logical record)

VN
The format is similar to VB except that the segment name is included in addition to the segment code.

NO
No output data set is created by FABHFSU. The output function can be handled in an exit routine.

For more information about these output formats, see "Unload output format supported by FABHFSU" on page 52.

24
Code the exit routine name.

The 8-character entry specifies the name of a user exit routine. The name must be left-aligned with trailing blanks. The load module must be in a STEPLIB library. For more information, see "FABHFSU user exit routine" on page 69.

Note: If a user exit routine is specified and one or more partitions are in the HALDB OLR cursor-active status, FABHFSU ends abnormally.

32
Code this field to activate the segment modification option.

The 1-character entry *m* indicates whether segments are to be modified by the user exit routine.

Specify one of the following keywords:

Y
Indicates that segments are to be modified by the user exit.

This option does not support a change of the database segment length. If you change the segment length with the Y option, the result is unpredictable. For details, see "Modifying segments in user exits" on page 71.

E
Indicates that segments are to be modified by the user exit.

This option supports a change of the database segment length. The option is valid only for HDAM, HIDAM, PHDAM, and PHIDAM databases.

An extra 100-byte field is added at the end of the segment data that is passed to the exit routine. This extra field can be used for segment extension. If the default length of this extra field is shorter than you require, you can change the length of the extra field by specifying the length in column 41 of the same PSB statement.

If this option has been selected, any request to activate the compare option used for problem determination is deactivated.

For details, see "Modifying segments in user exits" on page 71.

N | blank
Indicates that segments are not to be modified by the user exit (default).

33
Code this field to activate the concatenated key option.

The 1-character entry *k* indicates whether the fully concatenated key of each segment is to be built and passed to the exit routine.

Specify one of the following keywords:

Y
Build concatenated key.

N | blank
Do not build concatenated key (default).

34
Code this field to activate the exit routine control option.

The 1-character entry *x* indicates whether the user exit routine is given control before and after segments are processed (see "FABHFSU user exit routine" on page 69).

Specify one of the following keywords:

Y
Allow exit routine control.

N | blank
Do not allow exit routine control (default).

35
Code this field to activate the DBR skip option.

The 1-character entry *s*, the Database Record (DBR) skip option, indicates whether return code 12 or 16 is valid for the exit routine specified in columns 24 - 31 of this statement. A return code 12, if valid, causes FABHFSU to skip the remaining segments associated with the current root segment and begin processing at the next root segment.

Return code 16 causes a skip to a new root key value specified by the exit routine.

Use this control statement only when a single PSB control statement is defined. The skipping invoked by one PSB statement affects all others included in the same run.

For more information about return codes from the user exit routines, see "Contents of registers on exit" on page 75.

Specify one of the following codes:

Y
Allow DBR skip option.

N | blank
Do not allow DBR skip option (default).

36
Code this field to activate the data conversion option.

If a Data Conversion exit (DFSDBUX1 exit) is activated, the user exit routine specified by the exit routine name option in column 24 receives the segment data that has been converted from the stored form to the application form.

The 1-character entry *c* indicates whether the inverse conversions, that is, the conversion from the application form to the stored form, is done before the segment data edited in the exit routine is written into the output data set.

Specify one of the following keywords:

Y
Perform the conversion.

This option is valid only for UL and HS unload format, and is valid only when the option 'DATXEXIT YES' is specified in the HSSROPT data set.

N | blank
Do not perform the conversion. This keyword is the default.

37-40
The value specified in this 4-byte field is always ignored.

41-45
Code the extension length.

The 5-digit numeric entry *extln* specifies how many extra bytes are to be reserved for the segment extension in the exit routine. The length specified on this field, plus the maximum length of the segments in the database, will be used as the length of the work area for editing segments in the exit routine. This field is valid only when the option 'E' is specified in column 32 of the PSB statement. If the option 'E' is specified and this field is blank, the default value, 100, is used.

You can specify a value in the range of 00000 - 32767.

Notes:

- If the resulting length of the segment editing work area is more than 32,767 bytes long, 32,767 is used as the length of the work area.
- If option 'E' is not specified in column 32, the value specified on this field is ignored.

FABHPSFC PRNTOUT output data set

The FABHPSFC program produces the FABHFSU PSF Control Statement report and the FABHFSU PSF Scan Control Data Set report.

Format

The format is 133-byte fixed-length records. When the block size is coded in the JCL, the block size must be a multiple of 133.

FABHFSU PSF Control Statements report

This report contains the CARDIN control statements that were input to FABHPSFC.

The following figure shows an example of this report.

```
IMS HIGH PERFORMANCE UNLOAD                              "FABHFSU PSF CONTROL STATEMENTS"                          PAGE:     1
5655-E06                                                 DATE: 06/01/2021  TIME: 12.52.06                       FABHPSFC - V1.R2


0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

DBDMSHDP           NB 999  2
CTLSCANCNTL8936505NN
NPTR810
NPTR1620
NPTR3240
NPTR4860
PSB*       OUT1      UL        NNNN
END
```

Figure 27. FABHFSU PSF Control Statements report

FABHFSU PSF Scan Control Data Set report

This report contains information about the parameters specified on the CARDIN control statements that were input to the scan control data set. This report is produced for each output data set defined by the PSB control statement in the CARDIN data set.

The following figure shows an example of this report.

```
IMS HIGH PERFORMANCE UNLOAD                          "FABHFSU PSF SCAN CONTROL DATA SET REPORT"                     PAGE:     1
5655-E06                                                  DATE: 06/01/2021  TIME: 12.52.06                      FABHPSFC - V1.R2


          *** SCAN CONTROL SPECIFICATIONS ***                     *** FORMAT CONTROL SPECIFICATIONS ***

          PARALLEL SCAN NAME            SCANCNTL                  FORMAT (CONTROL PSB) NUMBER   01

          BASE DBD NAME                 MSHDP                     CONTROL PSB NAME              *

          INDEX DBD NAME                                          CONTROL PCB NUMBER

          SEQUENCE CHECK OPTION         NO                        OUTPUT DDNAME                 OUT1

          SEQUENCE ERROR OPTION         N.A                       OUTPUT FORMAT                 UL

          SEQUENCE ERROR PRINT OPTION   N.A                       EXIT ROUTINE NAME             N.A

          SEQUENCE ERROR THRESHOLD      N.A                       SEGMENT MODIFICATION OPTION   N.A

          SEQUENCE ERROR ABEND OPTION   N.A                       CONCATENATED KEY OPTION       N.A

                                                                  EXIT OPEN/CLOSE CONTROL       N.A

          LIMIT CONTROL                 YES                       DBR SKIP OPTION               N.A

          POINTER BYPASS OPTION         NO                        EXIT LE OPTION                N.A

          NO. PARALLEL SCANS            05

          DSN CHECK OPTION              NO

          PSC WTO OPTION                NO

          SEP HEADER OPTION             NO
IMS HIGH PERFORMANCE UNLOAD                          "FABHFSU PSF SCAN CONTROL DATA SET REPORT"                     PAGE:     2
5655-E06                                                  DATE: 06/01/2021  TIME: 12.52.06                      FABHPSFC - V1.R2


**** PHASE CONTROL SPECIFICATIONS ****

PHASE 01 STARTING BLOCK            BEGINNING OF FILE
          ENDING BLOCK                     809

PHASE 02 STARTING BLOCK                    810
          ENDING BLOCK                   1,619

PHASE 03 STARTING BLOCK                  1,620
          ENDING BLOCK                   3,239

PHASE 04 STARTING BLOCK                  3,240
          ENDING BLOCK                   4,859

PHASE 05 STARTING BLOCK                  4,860
          ENDING BLOCK             END OF FILE
```

Figure 28. FABHFSU PSF Scan Control Data Set report

SCAN CONTROL SPECIFICATIONS
Shows specifications or defaults from CTL and DBD control statements.

FORMAT CONTROL SPECIFICATIONS
Shows specifications or defaults from PSB control statement.

PHASE CONTROL SPECIFICATIONS
Shows starting and ending block values of each scan phase based on specifications from NPT control statement.

FABHFSU program (PSF mode)

In the PSF mode, FABHFSU performs individual scan phases, which can run separately or concurrently to unload a multivolume database. In the PSF mode, the information is obtained from the scan control data set that is created by FABHPSFC.

FABHFSU JCL requirements (PSF mode)

FABHFSU in PSF mode runs as an HSSR application program and, therefore, must meet the requirements for the basic JCL (FABHX034 JCL). In addition, FABHFSU JCL for PSF mode requires other DD statements.

Prerequisite: See "Basic JCL requirements" on page 30 for the basic (FABHX034) JCL requirements.

The following table summarizes additional JCL requirements for FABHFSU. In PSF mode, CNTLDD DD statement is also required.

Table 24. FABHFSU DD statements for PSF mode

DDNAME	Use	Format	Need
CARDIN	Input	LRECL=80	Optional
PRNTOUT	Output	LRECL=133	Required
ddname1	Output	RECFM=VB	Required
CNTLDD	Input and Output	LRECL=80	Required for PSF

The functions of CARDIN DD, PRNTOUT DD, and *ddname1* DD are the same in the FABHFSU standard mode and in the PSF mode. See "FABHFSU JCL requirements" on page 54 for details.

CNTLDD DD
This DD statement defines the input and output scan control data set, which has the following format:

```
//CNTLDD DD DSN=fsuscancntl,DISP=SHR
```

DSN is the user data set name assigned when scan control data set is created by FABHPSFC.

Related reference
JCL examples for FABHFSU PSF mode
To unload a multivolume IMS database in PSF mode, you can use the JCL jobs shown in the following figures.

FABHFSU CARDIN input data set (PSF mode)

The FABHFSU CARDIN input data set contains the control statements for FABHFSU.

When running in the PSF mode, the CARDIN data set contains the PSC and the END control statements. The DEC and GOT control statements can optionally be used in the PSF mode, but the DBD, PSB, BLM, ELM, PARTITION, and SEGSTAT control statements are not allowed as input to PSF.

The following table lists the FABHFSU control statements used in the PSF mode.

Table 25. FABHFSU control statements for PSF mode

Control statements	Function
DEC	Decompresses any compressed segments.

Table 25. FABHFSU control statements for PSF mode (continued)

Control statements	Function
END	Specifies the end of CARDIN control statements.
GOT	Provides support for PROCOPT GOT.
PSC	Identifies the scan control data set that directs the phases of the scan.

DEC control statement

The DEC control statement, which activates the decompress option, specifies whether FABHFSU is to decompress database segments.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

DEC d
```

Position
Description

1
Code the DEC keyword to activate the decompress option.

4
The 1-character entry *d* specifies whether compressed segments are decompressed by FABHFSU. This entry is required.

Use one of the following keywords:

Y
Compressed segments are decompressed. Y is the default.

N
Compressed segments are not decompressed.

Code N only when you want to have compressed segments in the output data sets.

If an unloaded data set has been created by specifying this option for a database that contains a compressed segment, that data set is not compatible with the unloaded data set that is created by the IMS HD Reorganization Unload utility. You cannot reload such an unloaded data set by using the IMS HD Reorganization Reload utility (DFSURGL0), but you can reload it by using IMS High Performance Load (Load utility or PSSR utility) or the IPR Reload utility.

Notes:

- If there is a segment type for which a Segment Edit/Compression exit routine is specified and the use of a Data Conversion exit is designated, the DECN option is ignored and the process continues with DECY.
- If the DECN option is specified and one or more partitions of PHDAM or PHIDAM are in the HALDB OLR cursor-active status, FABHFSU ends abnormally.
- Do not code DECN if you want to change the size of the segments.

Tip: The default of this control statement can be changed by replacing the default option table (FABHOPT). Specify the FSUDEC=NO parameter on the FABHTOPT macro statement. For details, see Chapter 19, “Site default options,” on page 261.

END control statement

The END control statement specifies the end of the CARDIN control statements.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

END
```

Position
Description

1
Code the END keyword to identify the END statement as the last statement of the CARDIN data set.

GOT control statement

The optional GOT control statement is provided to request that the same function provided by HSSR Engine for PCBs with PROCOPT=GOT be activated for the PCB even if PROCOPT=GOT is not specified for the PCB used in the FABHFSU job.

If the GOT control statement is specified, FABHFSU ignores the PROCOPT of the PCB statement in PSBGEN and forces PROCOPT=GOT to be used.

The GOT control statement is effective only when DBRC is inactive for the FABHFSU job.

Note: For more information about the PROCOPT=GOT support, see "Support for processing options GON and GOT" on page 91.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

GOT
```

Position
Description

1
Code the GOT keyword to activate the support for PROCOPT=GOT.

PSC control statement

The PSC control statement activates the PSF mode.

PSF mode differs from standard mode in that the data required by FABHFSU is obtained from the scan control data set instead of the CARDIN data set. Make sure that you have a CNTLDD DD statement and refer to the PSF operating instructions.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

PSCpsname  dbdname xxnnbryyyyddd
```

Position
Description

1
Code the PSC keyword to activate the PSF mode to unload large databases using multiple scans.

4
Code the name of the scan control data set as specified in the CTL control statement as input for FABHPSFC, which creates the scan control data set. This name is defined in the CNTLDD DD statement, and is used in the entire PSF mode. This 8-character entry is left-aligned with trailing blanks.

12
Code the database name as specified in the DBD control statement as input for FABHPSFC, which creates the scan control data set. This required 8-character entry is left-aligned with trailing blanks.

20
The required 2-digit entry *xx* specifies the expected number of scan phases in the PSF job. This value is the same as the value specified in the CTL control statement as input for FABHPSFC, which creates the scan control data set.

22
The required 2-digit entry *nn* specifies the phase number for this particular scan phase. This value can range from 01 to the expected number of scan phases specified in positions 20–21 of this statement.

24
The 1-character entry *b* allows overriding of the pointer bypass option specified in the DBD control statement as input for FABHPSFC, which creates the scan control data set. This can be specified for a particular scan phase or all scan phases.

Use one of the following keywords:

Blank
This entry accepts the pointer bypass option specified in the scan control data set.

1
The entry invokes the pointer bypass option.

2
The entry forces FABHFSU to use the index, rather than the root twin forward pointers, to unload an HIDAM database.

25
The 1-character entry *r* allows a particular scan phase to be rerun regardless of prior completion status. Use one of the following keywords:

Y
Rerun the scan phase.

N|blank
Do not rerun the scan phase (default).

26
The 7-digit entry *yyyyddd* allows the Julian date to be overridden. *yyyy* is the year and *ddd* is the day of the year. This value overrides the value specified in the CTL control statement as input for FABHPSFC, which creates the scan control data set.

FABHFSU PRNTOUT output data set (PSF mode)

Output from the FABHFSU utility includes the FABHFSU Control Statement report, the FABHFSU Control Specification report, and the FABHFSU Segment Statistics report. These reports are generated in the PRNTOUT data set.

The reports that are generated in the FABHFSU PSF mode are the same as the reports that are generated in the FABHFSU standard mode. See “FABHFSU output: PRNTOUT output data set” on page 66 for more information about these reports.

FABHPSFS program

The FABHPSFS program generates summarized statistics, output data set concatenation sequences, and header and trailer data set (if applicable). FABHPSFS is the last program to be run in PSF mode.

FABHPSFS JCL requirements

The FABHPSFS program is an MVS batch program. A FABHPSFS JCL job must satisfy the JCL requirements for the FABHPSFS program.

The following table summarizes the DD statements for FABHPSFS.

Table 26. FABHPSFS DD statements

DDNAME	Use	Format	Need
IMS	Input	Partitioned data set (DSORG=PO)	Required
CARDIN	Input	LRECL=80	Required

Table 26. FABHPSFS DD statements (continued)

DDNAME	Use	Format	Need
CNTLDD	I/O	Created by FABHPSFC	Required
ddname5	Output	-	Required
ddname6	Output	-	Optional
PRNTOUT	Output	LRECL=133	Required
SYSUDUMP	Output	-	Optional

EXEC

The EXEC statement must be in the following format:

```
//  EXEC PGM=FABHPSFS
```

IMS DD

This required DD statement defines the library that contains the DBD that describe the database to be scanned.

CARDIN DD

This required DD statement defines the input data set that contains control statements for FABHPSFS (see "FABHPSFS CARDIN input data set" on page 145).

CNTLDD DD

This required DD statement defines the input and output scan control data set, which has the following format:

```
 //CNTLDD DD DSN=fsuscancntl,...
```

DSN is the user data set name assigned when the scan control data set is created by FABHPSFS.

***ddname5* DD**

This DD statement defines the data set that contains the trailer record of the unloaded database. One DD statement is required for each PSB control statement in the CARDIN DD statement defined by FABHPSFS. (The name of *ddname5* must be the same as the one specified in the *ddname1* field of the PSB control statement.)

***ddname6* DD**

This DD statement defines the data set that contains the header record of the unloaded database. When the separate header record option is specified on the CTL control statement of CARDIN DD statement defined by FABHPSFS, this DD statement is required. One DD statement is required for each PSB control statement in the CARDIN DD statement defined by FABHPSFS. (The name of *ddname6* must be the same as the one of *ddname5*, except the first two characters must be 'XH'.)

PRNTOUT DD

This required DD statement defines the output data set to which FABHPSFS writes error messages and segment statistics (see "FABHPSFS PRNTOUT output data set" on page 147). The data set can be defined as:

```
//PRNTOUT DD SYSOUT=A
```

SYSUDUMP DD

This optional DD statement defines the dump data set for this program. The data set can reside on a printer, tape, or direct-access device, or be routed through the output stream.

Related reference

JCL examples for FABHFSU PSF mode

To unload a multivolume IMS database in PSF mode, you can use the JCL jobs shown in the following figures.

FABHPSFS CARDIN input data set

The control statements for the FABHPSFS program are specified in the CARDIN data set.

The FABHPSFS program provides the following two basic functions:

- Report the status of an in-progress parallel scan
- Wrap up a "completed" parallel scan.

These two purposes are performed by the specification of blank/STATUS/ RERUN/FORCE option in the position 20 of the SUM control statement.

The STATUS option, which can be run anytime after the FABHPSFC program has run, reports the current status of the scan from the scan control data set.

The "wrap-up" function has three options, normal (the default), RERUN, and FORCE.

Normal wrap-up must be run after all phases have completed (if they have not, it defaults to a STATUS run). It can be run only once and ends abnormally if run multiple times. It produces the summarized statistics for the entire scan. It also produces for each output format a concatenation sequence of the output data sets created by each phase. If any of the output formats is called for an Unload (UL) output, FABHPSFS creates one additional output data set that contains the "Trailer Label" required by HD Reorganization Reload and optionally another data set containing a separate "Header Label." All these data sets must be concatenated in the proper sequence by JCL for the input of any following job (that is, IMS HD Reorganization Reload).

The RERUN option can be used to re-create the output of the normal wrap-up function after a normal wrap-up has been run.

The FORCE option can be used to force normal wrap-up when one or more phases are not intended to be run. The following table lists the FABHPSFS control statements.

Table 27. FABHPSFS control statements

Control statements	Function	Mode
SUM	Directs the FABHPSFS program.	PSF
END	Specifies end of FABHPSFS CARDIN control statements.	PSF

Format

This data set contains 80-byte fixed-length records. The control statements can be coded in the input stream or accessed as a member of a partitioned data set.

SUM control statement

FABHPSFS is directed by the SUM control statement for the PSF.

Only one SUM control statement can be used.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

SUMpsname  dbname opt1  123
```

Position
Description

1
Code the SUM keyword to identify the SUM control statement.

4
Code a 1- to 8-character name for the parallel scan operation as specified in the CTL control statement as input for FABHPSFC.

12
This required 8-character entry *dbdname* indicates the database name as specified in the DBD control statement as input for FABHPSFC.

20
This 6-character entry *opt1* determines the following optional keywords:

blank
Normal wrap-up function runs and the full summary report is provided (default). This option must be run after all phases have been completed.

STATUS
Only the current status of the scan is reported. This option can be run at anytime after the FABHPSFC program has run.

RERUN
The full summary report is re-created. The unloaded trailer data set can also be re-created as specified in position 26, 27, or 28.

FORCE
This option allows only some of the PSF phases are to be run.

Note: When this option is selected, and when the first phase is not to be run, the "Separate Header" option is required to be used for reload. (Without specifying the "Separate Header" option, the header is not created when the first phase is not run.)

26
The 1-character entry *1* is applicable only if the RERUN option is specified in position 20. This option allows for the re-creation of the header or trailer data set defined by the first PSB control statement. Use one of the following optional keywords:

Y
Both header and trailer are re-created.

N
Neither header nor trailer is re-created (default).

H
Only header is re-created.

T
Only trailer is re-created.

27
This is the same option as the option of position 26, except that it applies to the output data set defined by the second PSB control statement.

28
This is the same option as the option of position 26, except that it applies to the output data set defined by the third PSB control statement.

END control statement

The END control statement specifies the end of the CARDIN control statements.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

END
```

Position
Description

1
Code the END keyword to identify the END statement as the last statement of the CARDIN data set.

FABHPSFS PRNTOUT output data set

FABHPSFS generates the FABHFSU PSF Control Statements report and the FABHFSU PSF Summary report in the PRNTOUT data set.

Format

The format is 133-byte fixed-length records. When the block size is coded in the JCL, the block size must be a multiple of 133.

FABHFSU PSF Control Statements report

This report contains the CARDIN control statements that were used as input to FABHPSFS.

The following figure shows an example of this report.

```
IMS HIGH PERFORMANCE UNLOAD                          "FABHFSU PSF CONTROL STATEMENTS"                          PAGE:     1
5655-E06                                             DATE: 06/01/2021  TIME: 13.09.36                      FABHPSFS - V1.R2


0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

SUMSCANCNTLMSHDP
END
```

Figure 29. FABHFSU PSF Control Statement report

FABHFSU PSF Summary report

This report provides the summarized statistics for the entire scan. This report is produced for each output data set that is defined by the PSB control statement in the CARDIN data set.

The following figures show an example of the report.

```
IMS HIGH PERFORMANCE UNLOAD                          "FABHFSU PSF SUMMARY REPORT"                          PAGE:     1
5655-E06                                          DATE: 06/01/2021  TIME: 13.09.36                    FABHPSFS - V1.R2


          *** SCAN CONTROL SPECIFICATIONS ***                  *** FORMAT CONTROL SPECIFICATIONS ***

          PARALLEL SCAN NAME            SCANCNTL               FORMAT (CONTROL PSB) NUMBER   01

          BASE DBD NAME                 MSHDP                  CONTROL PSB NAME              *

          INDEX DBD NAME                                       CONTROL PCB NUMBER

          SEQUENCE CHECK OPTION         NO                     OUTPUT DDNAME                 OUT1

          SEQUENCE ERROR OPTION         N.A                    OUTPUT FORMAT                 UL

          SEQUENCE ERROR PRINT OPTION   N.A                    EXIT ROUTINE NAME             N.A

          SEQUENCE ERROR THRESHOLD      N.A                    SEGMENT MODIFICATION OPTION   N.A

          SEQUENCE ERROR ABEND OPTION   N.A                    CONCATENATED KEY OPTION       N.A

                                                               EXIT OPEN/CLOSE CONTROL       N.A

          LIMIT CONTROL                 YES                    DBR SKIP OPTION               N.A

          POINTER BYPASS OPTION         NO                     EXIT LE OPTION                N.A

          NO. PARALLEL SCANS            05

          DSN CHECK OPTION              NO

          PSC WTO OPTION                NO

          SEP HEADER OPTION             NO
IMS HIGH PERFORMANCE UNLOAD                          "FABHFSU PSF SUMMARY REPORT"                          PAGE:     2
5655-E06                                          DATE: 06/01/2021  TIME: 13.09.36                    FABHPSFS - V1.R2

**** FORMAT 01 OUTPUT STATISTICS ****

 PSB        SEGMENT  SEGMENT      TOTAL       TOTAL       SEQUENCE    MAXIMUM    MAXIMUM     AVERAGE     AVERAGE
 NAME        NAME     LEVEL     RETRIEVED     OUTPUT       ERRORS      TWINS     CHILDREN     TWINS      CHILDLEN

*           HSHDPRT     1          2,000       2,000          0          1          21        1.00        21.00

            HSHDPDR     2          2,000       2,000          0          1           0        1.00          .00

            HSHDPSR     2          4,000       4,000          0          2           0        2.00          .00

            HSHDPMC     2          6,000       6,000          0          3           0        3.00          .00

            HSHDPIR     2          2,000       2,000          0          1           0        1.00          .00

            HSHDPMG     2          4,000       4,000          0          2           0        2.00          .00

            HSHDPSF     2          6,000       6,000          0          3           0        3.00          .00

            HSHDPCB     2          2,000       2,000          0          1           0        1.00          .00

            HSHDPSD     2          4,000       4,000          0          2           0        2.00          .00

            HSHDPAX     2          6,000       6,000          0          3           0        3.00          .00

            HSHDPSA     2          2,000       2,000          0          1           0        1.00          .00

            HSHDPCR     2          4,000       4,000          0          2           0        2.00          .00
          TOTAL RETRIEVED                  44,000

          TOTAL OUTPUT                     44,000

          TOTAL SEQUENCE ERRORS                0
```

Figure 30. FABHFSU PSF Summary report (Part 1 of 2)

```
IMS HIGH PERFORMANCE UNLOAD                              "FABHFSU PSF SUMMARY REPORT"                              PAGE:     3
5655-E06                                                DATE: 06/01/2021  TIME: 13.09.36                      FABHPSFS - V1.R2


**** FORMAT 01 DSN CONCATENATION SEQUENCE ****

        1. IMS31B.SCANCNTL.UL01.PHASE01

        2. IMS31B.SCANCNTL.UL01.PHASE02

        3. IMS31B.SCANCNTL.UL01.PHASE03

        4. IMS31B.SCANCNTL.UL01.PHASE04

        5. IMS31B.SCANCNTL.UL01.PHASE05

        6. IMS31B.SCANCNTL.UL01.TRAILER
IMS HIGH PERFORMANCE UNLOAD                              "FABHFSU PSF SUMMARY REPORT"                              PAGE:     4
5655-E06                                                DATE: 06/01/2021  TIME: 13.09.36                      FABHPSFS - V1.R2


**** FORMAT 01 OF 01 PHASE STATUS ****


OPER  H E A D E R  L A S T   T R A I L E R  L A S T    TOTAL PHASE     P H A S E  L A S T     PHASE   SEQ    PNTRS SYNAD
TYPE  STATUS  DATE   TIME     STATUS  DATE    TIME    PHASES NUMBER  DATE    TIME     STATUS  RERUNS  ERRORS BYPSD ERROR

 UL                          COMPL    892  13:09:37     5     1    00252  13:02:53   COMPLETE    0       0    NO

                                                               2    00252  12:56:59   COMPLETE    0       0    NO

                                                               3    00252  12:58:59   COMPLETE    0       0    NO

                                                               4    00252  12:58:58   COMPLETE    0       0    NO

                                                               5    00252  12:56:31   COMPLETE    0       0    NO
```

Figure 31. FABHFSU PSF Summary report (Part 2 of 2)

Page 1 of this report contains the summary information about the parameters that were specified on the CARDIN control statements.

SCAN CONTROL SPECIFICATIONS
Shows specifications or defaults from CTL and DBD control statements

FORMAT CONTROL SPECIFICATIONS
Shows specifications or defaults from PSB control statement

Page 2 of this report provides statistics for each sensitive segment in the database.

PSB NAME
Name of PSB

SEGMENT NAME
Name of segment

SEGMENT LEVEL
Level of segment

TOTAL RETRIEVED
This count shows the number of each segment type retrieved. This count does not include segments bypassed due to sequence errors.

TOTAL OUTPUT
This shows the number of each segment type processed by FABHFSU. If the output format is specified as NO, the value is zero. Differences between TOTAL RETRIEVED and TOTAL OUTPUT represent segments bypassed by the user exit routine.

SEQUENCE ERRORS
This value represents the number of sequence errors detected for this segment type. If the sequence check option is N, this field is zero.

MAXIMUM TWINS
This is the maximum number of this segment type that occurs under any one root segment.

MAXIMUM CHILDREN
This is the maximum number of dependent children that occurs for this segment type.

AVERAGE TWINS
This is the TOTAL RETRIEVED of this segment type divided by TOTAL RETRIEVED of this segment's parent. Average occurrences of this segment per parent occurrence.

AVERAGE CHILDLEN
This value is the sum of all segment occurrences dependent on this segment type divided by the TOTAL RETRIEVED of this segment type. (This value might be incorrect if sequence errors are bypassed or if the PCB is not sensitive to all dependents.)

TOTAL RETRIEVED
This is the total number of the segments retrieved.

TOTAL OUTPUT
This is the total number of the segments processed by FABHFSU.

TOTAL SEQUENCE ERRORS
This is the total number of sequenced errors.

Page 3 of this report contains information on the DFSUINPT DD statements when you reload the unloaded database data sets. It shows the sequence of the concatenation of the header data set, unloaded data sets, and trailer data set.

Page 4 of this sample report contains the following information:

ORDER TYPE
Operation type. This field shows the output format type.

HEADER LAST
The following three fields all pertain to the latest header label: `Status`, `Date`, and `Time`.

TRAILER LAST
The following three fields all pertain to the latest trailer label: `Status`, `Date`, and `Time`.

TOTAL PHASES
The total number of scan phases in the PSF job.

PHASE NUMBER
The phase number of a particular scan phase.

PHASE LAST
The following three fields all pertain to the latest scan phase: `Date`, `Time`, and `Status`.

PHASE RERUNS
The total number of the FABHPSFS jobs rerun by the rerun option of the SUM control statement.

SEQ ERRORS
The number of sequence errors.

PNTRS BYPSD
This shows whether the pointer bypass option was specified or not.

SYNAD ERROR
This shows whether a synad error was detected or not.

JCL examples for FABHFSU PSF mode

To unload a multivolume IMS database in PSF mode, you can use the JCL jobs shown in the following figures.

In this example, three individual parallel scan phases are utilized to unload an HDAM database. In each phase, FABHFSU scans and unloads only a predefined portion of the database controlled by a scan control data set.

Step 1 gets a primary data set extent information of the database. Based on the information, step 2 creates a scan control data set to define scan parameters and the number of parallel scan phases. Step 4 generates a set of summarized statistics and concatenation sequence of the output data sets unloaded by each parallel scan phase. Step 4 creates two data sets that contain "Header record" and "Trailer record" required by the IMS HD Reorganization Reload utility.

Subtopics:

Step 1: Example of FABHPSFM

The TESTDB DD statement identifies the primary data set of the DBD specified by the MAP control statement in the CARDIN DD statement. The extent information of the primary data set is reported to SYSOUT class A defined by the PRNTOUT DD statement. This information helps to determine the phase limit definition in step 2.

```
//*-------------------------------------------------------------------------
//*    PSF STEP 1 - GET DATABASE EXTENTS INFORMATION
//*-------------------------------------------------------------------------
//FSUMAP  EXEC PGM=FABHPSFM
//IMS       DD DSN=IMSVS.DBDLIB,DISP=SHR
//TESTDB    DD DSN=TESTDS.HDAM.VSAM,DISP=OLD
//PRNTOUT   DD SYSOUT=A
//SYSUDUMP  DD SYSOUT=A
//CARDIN    DD *
MAPTESTDB
END
/*
```

Figure 32. FABHPSFM JCL for PSF mode

Step 2: Example of FABHPSFC

The CNTLDD DD statement identifies the scan control data set created by this job step.

The CARDIN DBD control statement specifies the DBD name of the database.

The CARDIN CTL control statement defines a parallel scan name (SCANCNTL) for use in getting access to the scan control data set; defines the Julian date as the last day of 2000; and sets the number of parallel scan phases to 03. Y in column 22 specifies that the data set names match the naming standard. The beginning position of the parallel scan name in the data set is 08 in columns 23–24. Y in column 25 specifies that the header record is written to the separate output data set in step 4.

Two CARDIN NPT control statements define the scan limit of each parallel scan phases. The first phase scans the database from the beginning of the database to the relative CI number 1000. The second phase scans the database from the relative CI number 1001 to the number 4000, and the third phase scans it from the number 4001 to the end of the database.

The CARDIN PSB control statement specifies the name of the DD statement (OUTDATA) that defines the data set to be created.

```
//*-------------------------------------------------------------------------
//*    PSF STEP 2 - CREATE SCAN CONTROL DATA SET
//* -------------------------------------------------------------------------
//FSUCTRL EXEC PGM=FABHPSFC
//IMS       DD DSN=IMSVS.DBDLIB,DISP=SHR
//CNTLDD    DD DSN=TESTDS.CONTROL,DISP=(NEW,CATLG),
//             UNIT=SYSDA,VOL=SER=TESTVOL,SPACE=(TRK,(4,1)),
//             DCB=(BLKSIZE=4096,LRECL=4092,RECFM=VB)
//PRNTOUT   DD SYSOUT=A
//SYSUDUMP  DD SYSOUT=A
//CARDIN    DD *
DBDTESTDB
CTLSCANCNTL200036603NY08Y
NPTR1001
NPTR4001
PSB*       OUTDATA 00UL
END
/*
```

Figure 33. FABHPSFC JCL for PSF mode

Steps 3A, 3B, 3C: Example of FABHFSU

The CNTLDD DD statement identifies the scan control data set created by FABHPSFC in a previous job step.

The CARDIN PSC control statement specifies that there are three expected scan phases. A portion of the database with the *dbdname* TESTDB is unloaded in this scan phase. The scan control data set created by FABHPSFC is named SCANCNTL.

The OUTDATA DD statement specifies a data set on which this portion of the IMS database is unloaded.

Two other job steps in PSF mode would be required to complete the unloading of this database. The JCL for the job steps is the same, except for modifying the PSC control statement and the *ddname* DD (OUTDATA) statement to specify parameters required for phase 2 and phase 3.

```
//*-------------------------------------------------------------------------
//*    PSF STEP 3A - UNLOAD DATABASE (FABHFSU) - PHASE 1 OF 3
//* -------------------------------------------------------------------------
//UNLOAD1 EXEC FABHULU,MBR=FABHFSU,DBD=TESTDB,DBRC=N,IRLM=N
//CNTLDD    DD DSN=TESTDS.CONTROL,DISP=SHR
//TESTDB    DD DSN=TESTDS.HDAM.VSAM,DISP=SHR
//OUTDATA   DD DSNAME=UNLOAD.SCANCNTL.UL01.PHASE01,
//             DISP=(NEW,KEEP),UNIT=SYSDA,
//             SPACE=(CYL,(100,25),RLSE)
//PRNTOUT   DD SYSOUT=A
//CARDIN    DD *
PSCSCANCNTLTESTDB  0301
END
/*
```

Figure 34. FABHFSU JCL for PSF mode

Step 4: Example of FABHPSFS

The XHTDATA DD statement defines a data set that contains the header record of the unloaded data sets. The name of this DD statement must begin with 'XH'.

The OUTDATA DD statement defines a data set that contains the trailer record of the unloaded data sets. The name of this DD statement (OUTDATA) must be equal to *ddname1* of CARDIN PSB control statement in step 2.

The CNTLDD DD statement identifies the scan control data set. The CARDIN SUM control statement specifies the parallel scan name (SCANCNTL) to access the scan control data set.

```
//*-------------------------------------------------------------------------
//*    PSF STEP 4 - SUMMARIZE STATISTICS AND DATA SETS SEQUENCE
//* -------------------------------------------------------------------------

//FSUSUMM EXEC PGM=FABHPSFS
//IMS       DD DSN=IMSVS.DBDLIB.DISP=SHR
//CNTLDD    DD DSN=TESTDS.CONTROL,DISP=OLD
//XHTDATA   DD DSNAME=UNLOAD.SCANCNTL.UL01.HEADER,DISP=(NEW,KEEP)
//             UNIT=SYSDA,SPACE=(TRK,(4,1)),
//             DCB=(BLKSIZE=4096,LRECL=4092,RECFM=VB)
//OUTDATA   DD DSNAME=UNLOAD.SCANCNTL.UL01.TRAILER,DISP=(NEW,KEEP)
//             UNIT=SYSDA,SPACE=(TRK,(4,1)),
//             DCB=(BLKSIZE=4096,LRECL=4092,RECFM=VB)
//PRNTOUT   DD SYSOUT=A
//SYSUDUMP  DD SYSOUT=A
//CARDIN    DD *
SUMSCANCNTLTESTDB
END
/*
```

Figure 35. FABHPSFS JCL for PSF mode

When you reload the unloaded database data sets with the IMS HD Reorganization Reload utility or an equivalent program, you must specify the concatenation of the header data set, unloaded data sets, and trailer data set on the DFSUINPT DD statement. The sequence of the concatenation is reported in the PRNTOUT data set of step 4. The following is an example to specify DD statements:

```
  //DFSUINPT DD DSN=UNLOAD.SCANCNTL.UL01.HEADER,DISP=OLD
              DD DSN=UNLOAD.SCANCNTL.UL01.PHASE01,DISP=OLD
              DD DSN=UNLOAD.SCANCNTL.UL01.PHASE02,DISP=OLD
              DD DSN=UNLOAD.SCANCNTL.UL01.PHASE03,DISP=OLD
              DD DSN=UNLOAD.SCANCNTL.UL01.TRAILER,DISP=OLD
```

Chapter 11. Options for HSSR Engine

You can specify options for HSSR Engine by coding control statements in the HSSROPT data set.

Topics:

Overview of HSSROPT control statements

Control statements for HSSR Engine (HSSROPT control statements) are specified in the HSSROPT data set.

The options that can be specified for the HSSROPT data set include:

- Options for call analyzer and call handler components of HSSR Engine
- Options for buffer handler component of HSSR Engine
- Options for reports and outputs produced by HSSR Engine
- Options for trace function provided by HSSR Engine
- Options for problem determination

You can tune HSSR Engine's Chained Anticipatory Buffering (CAB) buffer handler by coding control statements in the HSSRCABP data set. For the details, see “Chained Anticipatory Buffer handler (CAB)” on page 208.

You can specify HSSRLDEF DD to change the range of lengths of database records in Database Tuning Statistics. For the details, see “HSSRLDEF input data set for Database Tuning Statistics” on page 310.

Subtopics:

Summary of HSSROPT control statements

The control statements shown in the following table are provided to help you specify options for HSSR Engine. Control statements for options that control report output are also specified in the HSSROPT data set.

Table 28. List of HSSROPT control statements for HSSR Engine

Used for	Keyword	Function	Description
Call analyzer and call handler	APISET	Specifies a set of call types that can be processed.	See “APISET control statement” on page 159.
	BLDLPCK	Whether to retrieve logical parent's concatenated keys (LPCKs) or not.	See “BLDLPCK control statement” on page 160.
	BYINDEX	Unloads HIDAM root segments through the HIDAM index.	See “BYINDEX control statement” on page 162.
	DATXEXIT	Specifies whether the data conversion exit is to be activated.	See “DATXEXIT control statement” on page 165.
	HSSRDBD	Specifies HSSR PCBs by using DBD names.	See “HSSRDBD control statement” on page 169.
	HSSRPCB	Specifies HSSR PCBs by using PCB numbers.	See “HSSRPCB control statement” on page 169.
	KEYCHECK	Specifies the action for key sequence errors.	See “KEYCHECK control statement” on page 170.
	SKERROR	Sets error skip count.	See “SKERROR control statement” on page 175.
	SKIPAUTH	Specifies whether to bypass IMS DBRC database authorization for HALDB partitions.	See “SKIPAUTH control statement” on page 175.
	SKIPVLC	Specifies to ignore a sensitive virtual logical child segment in the DLI or the DBB region.	See “SKIPVLC control statement” on page 176.

Table 28. List of HSSROPT control statements for HSSR Engine (continued)

Used for	Keyword	Function	Description
Buffer handler	BUF	Specifies the database for which the BB buffer handler is to be used and the number of buffers to allocate for a buffer pool.	See "BUF control statement" on page 161.
	GOTRETRY	Change the default retry parameters for PROCOPT=GOT.	See "GOTRETRY control statement" on page 168.
	LSR	Specifies whether to share the LSR pool with DL/I.	See "LSR control statement" on page 172.
	NOFIX	Specifies not to page-fix CAB buffer pools.	See "NOFIX control statement" on page 172.
	NOVSAMOPT	Prevents the override of the read-ahead threshold values used by VSAM.	See "NOVSAMOPT control statement" on page 172.
	PCBLIST	Specifies a type of PCB list to be passed to the application program.	See "PCBLIST control statement" on page 173.
	RETRY	Specifies that a failing KSDS I/O operation is to be retried.	See "RETRY control statement" on page 174.
Reports and outputs	CABSTAT	Controls the amount of CAB Statistics report.	See "CABSTAT control statement" on page 163.
	CALLSTAT	Specifies that partition-wide database call statistics are to be produced for each partition of HALDB.	See "CALLSTAT control statement" on page 164.
	DBSTATS	Provides a Database Tuning Statistics report.	See "DBSTATS control statement" on page 166.
	DIAGG	Generates diagnosis information for status GG.	See "DIAGG control statement" on page 167.
	LOUT	Specifies the record-selection criteria for HSSRLOUT data set.	See "LOUT control statement" on page 171.
	PARTINFO	Produces Partition Definition reports and Partitions Accessed report.	See "PARTINFO control statement" on page 173.

Table 28. List of HSSROPT control statements for HSSR Engine (continued)

Used for	Keyword	Function	Description
Trace function	BUTR	Activates the buffer trace function.	See "BUTR control statement" on page 161
	TRDB	Specifies the databases to be traced.	See "TRDB control statement" on page 176.
	TRHC	Specifies the information to be reported by the trace function.	See "TRHC control statement" on page 177.
	TRXC	Specifies the number of trace entries for wrap-around core trace.	See "TRXC control statement" on page 178.
Problem determination	CO	Specifies that the result of an HSSR call is to be compared with that of a DL/I call.	See "CO control statement" on page 164.
	DBDL1	Specifies that HSSR calls are to fall back to DL/I calls.	See "DBDL1 control statement" on page 166.
Others	HPIO	Specifies whether Media Manager is used for reading VSAM ESDS data sets or OSAM LDS data sets.	See "HPIO control statement" on page 168.
	RTEXIT	Specifies the name of the runtime exit routine.	See "RTEXIT control statement" on page 174.
	ZIIPMODE	Specifies whether to offload eligible workloads to zIIP processors.	See "ZIIPMODE control statement" on page 178.

Syntax for HSSROPT control statements

To code HSSROPT control statements, follow these format rules:

- A control statement keyword must always begin in column 1.
- A control statement keyword must be followed by a blank.
- Additional control information is coded after the blank.
- Continuation control statements are not allowed.

APISET control statement

The APISET control statement specifies a set of HSSR call types that can be used in your HSSR application program.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

APISET 1
       2
       3
```

Position
Description

1
Code the APISET keyword to specify a set of HSSR call types.

8
Code 1, 2, or 3. For details about what each APISET supports, see "DL/I calls supported by each API set" on page 85.

APISET 1 is the default. Consider specifying APISET 2 or 3 only if your application program issues a call that APISET 1 does not support.

Notes:

- If you specify APISET 3, you cannot specify any of the following control statements: BYINDEX, DBSTATS, KEYCHECK, or SKERROR.
- If you specify APISET 3, the BLDLPCK control statement is always activated.
- If you specify APISET 3, the disposition of the database data sets must be DISP=SHR.

Tip: The default of this control statement can be changed by replacing the default option table (FABHOPT). For details, see Chapter 19, "Site default options," on page 261.

BLDLPCK control statement

The BLDLPCK control statement specifies whether to retrieve logical parent's concatenated keys (LPCKs).

By default, for a logical child segment with a logical parent's concatenated key (LPCK) that is specified as *virtual* on the SEGM statement of the DBD, HSSR call handler returns blanks in the I/O area that would usually hold the LPCK. For compatibility with FSU II, FABHFSU returns binary zeros to the I/O area instead of blanks. You can use the BLDLPCK control statement to have HSSR call handler build the LPCK and return it in the I/O area.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

BLDLPCK
```

Position
Description

1
Code the BLDLPCK keyword to activate the BLDLPCK option.

Notes:

- If the LPCK is defined as *physical* (that is, if the LPCK is physically stored as a part of the logical child segment in the database), HSSR call handler ignores this option.
- If the BLDLPCK statement is specified and there are a large number of virtual LPCKs in the database, the performance of IMS High Performance Unload could be degraded. The BLDLPCK statement is necessary, however, if the control statement for the Pre-reorganization utility specifies that the DBIL for the logical child database is to be unloaded by FABHURG1 or FABHFSU. If the DBIL is specified for the database, symbolic keys in the unloaded database are used to match up logical children and parents.
- If the BLDLPCK statement is specified for a database that has a logical child whose LPCK is defined as *virtual*, the database data sets for all logical parent databases of such logical children must be specified on the JCL. If the DD statement for one of logical parent databases is not specified, HSSR call handler returns the status code of AI to the application program when the logical child segment is processed, and the part of the I/O area that should contain the LPCK is filled with blanks. FABHURG1 and FABHFSU issue the message FABH0560E, and the programs end processing.
- The BLDLPCK statement is not supported for HISAM databases.
- The BLDLPCK statement is ignored for a PHDAM or PHIDAM database.

- The BLDLPCK statement is ignored if either MIGRATE or FALLBACK control statement is specified in the SYSIN data set for a FABHURG1 job.
- If APISET 3 is specified, the BLDLPCK statement is ignored.

BUF control statement

The optional BUF control statement specifies a database for which the BB buffer handler is to be used, and to override the default number of buffers for BB.

The BB buffer handler allocates a separate buffer pool for each data set group of the database. It also allocates a certain number of buffers to each of these buffer pools—either the default or a specified number.

The default number of buffers depends on how the PCB is defined as an HSSR PCB. If an HSSR PCB is defined by use of either an HSSRPCB or an HSSRDBD control statement, and the value to the KEYLEN keyword for the PCB is less than 200, BB allocates six basic buffers for each data set as the default. If the KEYLEN value for the PCB is greater than 200, the number of basic buffers to be allocated is determined as explained in "Number of Basic Buffers for an HSSR PCB" on page 359.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

BUF dbdname ,nbrbuffers
```

Position
Description

1
Code the BUF keyword to specify a database for which the BB buffer handler is to be used and to override the default number of buffers for BB.

5
Code the 8-byte *dbdname* to specify the database for which the default number of buffers will be overridden (if the database name is not 8 bytes long, include trailing blanks).

13
Add a comma (,) to separate the database name from the number of buffers.

14
nbrbuffers is the number of buffers that you want BB to allocate for a buffer pool.

BUTR control statement

The BUTR control statement directs HSSR Engine to create a file containing a machine-readable trace of internal calls to the buffer handler.

This trace, which is written to the HSSRBUTR data set, can be used as input to FABHBSIM, the HSSR Buffer Handler Simulation utility.

Note: For instructions for using the buffer handler simulation utility, see Chapter 17, "Buffer handler simulation utility (FABHBSIM)," on page 239.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

BUTR
```

Position
Description

1
Code the BUTR keyword to instruct HSSR Engine to create a file that contains a machine-readable trace of internal calls to the buffer handler.

Restriction: No buffer trace is taken for HALDBs.

BYINDEX control statement

The BYINDEX control statement instructs HSSR Engine to retrieve HIDAM or PHIDAM root segments sequentially via the HIDAM or PHIDAM index instead of via the root-twin chain.

If the root segments have no twin backward or hierarchical backward pointer, the BYINDEX control statement has no effect.

A secondary index can also be used for HIDAM or HDAM root segments.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

BYINDEX
BYINDEX index
```

Position
Description

1
Code the BYINDEX keyword to activate the BYINDEX option.

9
Specifies a secondary index of the HIDAM or HDAM database if you want the root segments to be retrieved in the secondary index sequence. The target segment of the index must be the root segment. For details, see “Considerations for using a secondary index” on page 27.

Restrictions:

- If APISET 3 is specified, this statement cannot be specified.
- If one or more partitions of PHDAM or PHIDAM are in the HALDB OLR cursor-active status, BYINDEX=YES is ignored.
- FABHFSU ignores this control statement. Specify the index name in the DBD control statement in the CARDIN data set.

CABBASE control statement

This optional control statement specifies, in number of tracks, the basic size of I/O buffers that the CAB buffer handler allocates.

The RANSIZE and NBRDBUF values are determined from the CABBASE control statement value and the CI/block size of the database data set. If the RANSIZE control statement or the NBRDBUF control statement is specified in the HSSRCABP data set, the RANSIZE control statement value or the NBRDBUF control statement value has precedence over the CABBASE control statement value.

```
0.......1.........2.........3.........4.........5..
12345678901234567890123456789012345678901234567890123456789012
CABBASE trk dbam typ
```

Position
Description

1
Code the CABBASE keyword.

9
Code a numeric value for *trk*. The value must be a left-aligned decimal number in the range of 1 - 255.

For VSAM sequential buffering, the value must be in the range of 1 - 15.

13

ALL
Indicates that this statement applies to all ESDS, OSAM, or OSAM LDS data sets.

ALL is the default.

VSAM
Indicates that this statement applies to all ESDS data sets or OSAM LDS data sets.

OSAM
Indicates that this statement applies to all OSAM data sets.

18

ALL
Indicates that this statement applies to both sequential buffering and direct buffering.

ALL is the default.

SEQ
Indicates that this statement applies to sequential buffering.

DIR
Indicates that this statement applies to direct buffering.

The default value is determined based on the access method as follows:

- For OSAM sequential buffering: 1 track (CABBASE 001 OSAM SEQ)
- For OSAM direct buffering: 2 tracks (CABBASE 002 OSAM DIR)
- For VSAM sequential buffering: 8 tracks (CABBASE 008 VSAM SEQ)
- For VSAM direct buffering: 15 tracks (CABBASE 015 VSAM DIR)

If the VSAM CI size is smaller than 2 KB, the CAB buffer handler applies the following values:

- For sequential buffering: 4 tracks (CABBASE 004 VSAM SEQ)
- For direct buffering: 8 tracks (CABBASE 008 VSAM DIR)

Tips:

- The following statement is recommended to improve the performance of CAB sequential I/O buffering for OSAM data sets. This statement changes the basic size of OSAM sequential buffers from one track to eight tracks: CABBASE 008 OSAM SEQ
- The default of this control statement can be changed by replacing the default option table (FABHOPT). For details, see Chapter 19, “Site default options,” on page 261.

CABSTAT control statement

Use this control statement to control the amount of CAB statistics that HSSR Engine prints on the HSSRSTAT data set.

```
0.......1........2........3........4........5........6........7........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

CABSTAT NO
        YES
```

Position
Description

1
Code the CABSTAT keyword to control the printing of the CAB statistics.

9
Specify one of the following keywords:

NO
Requests to limit CAB statistics to the main summary report. NO is the default.

YES
Requests the printing of the detailed CAB statistics report on the HSSRSTAT data set.

Tip: This default can be changed by replacing the default option table (FABHOPT). See Chapter 19, “Site default options,” on page 261.

CALLSTAT control statement

Use this control statement to produce the partition-wide database call statistics for each partition.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

CALLSTAT PART
```

Position
Description

1
Code the CALLSTAT keyword.

10
Code the PART keyword to print the partition-wide database call statistics reports for each partition that was processed and from which at least one segment was retrieved.

Restrictions: In any one of the following cases, the partition-wide statistics are not printed:

- If the CALLSTAT statement is specified for a nonpartitioned database.
- If the PART keyword is not specified.
- If a keyword other than PART is specified.

CO control statement

The CO control statement specifies that the result of an HSSR call is to be compared with the result of a DL/I call.

The compare (CO) control statement causes each HSSR call issued by the application program to be preceded by a DL/I call issued with the same SSA as the HSSR call. The resultant I/O areas and PCBs of both calls are compared internally. If a mismatch occurs, the I/O areas and the PCBs are printed on the HSSRTRAC data set. HSSR Engine either abends or returns to the application program.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

CO
CO N
```

Position
Description

1
Code the CO keyword to activate the compare option.

4
Specify the action of HSSR Engine when a difference is found in HSSR and DL/I calls.

Blank
HSSR Engine abends with a dump.

This option should normally be used when a mismatch is detected between HSSR and DL/I calls.

N
HSSR Engine returns control to the application program when a mismatch is detected between HSSR and DL/I calls.

Notes:

- To use the CO control statement, the database data sets must be allocated with DISP=SHR.
- If the database is being updated concurrently, the results of DL/I calls and HSSR calls might differ. In such a case, do not use the CO control statement.
- If APISET 3 is specified, the comparison is not done for the calls that are finally processed by DL/I.

- If one or more partitions of PHDAM or PHIDAM are in the HALDB OLR cursor-active status, the blank keyword is ignored.

COMPAUTH control statement

The COMPAUTH control statement specifies whether to call the segment compression exit in supervisor state.

If you also specify the DECN control statement, this control statement is ignored because HSSR Engine does not call the segment compression exit.

If you use an encryption exit of InfoSphere® Guardium® Data Encryption for Db2 and IMS Databases as the segment compression exit, specifying COMPAUTH YES reduces performance degradation.

```
0.......1.........2.........3.........4.........5..
12345678901234567890123456789012345678901234567890123456789012
COMPAUTH YES
         NO
```

Position
Description

1
Code the COMPAUTH keyword.

10
Specify one of the following keywords:

YES
Specifies to call the segment compression exit in supervisor state.

To enable COMPAUTH YES, the following conditions must be met:

- All of the STEPLIB libraries are APF-authorized.
- The first load module is link-edited with an authorization code of AC=1. This is the same as how FABHX034 load module is link-edited.

NO
Specifies to call the segment compression exit in problem state. NO is the default.

Tip: The default of this control statement can be changed by replacing the default option table (FABHOPT). Specify the COMPAUTH=YES parameter on the FABHTOPT macro statement. For details, see Chapter 19, “Site default options,” on page 261.

DATXEXIT control statement

The DATXEXIT control statement specifies whether the Data Conversion exit is to be activated for HSSR calls and DL/I calls.

Restriction: HSSR Engine support of Data Conversion exit routines is restricted to the Data Conversion exit routine provided by Year 2000 Exit Tool. No other Data Conversion exit routine is supported.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

DATXEXIT YES
         NO
```

Position
Description

1
Code the DATXEXIT keyword to specify the treatment of the Data Conversion exit routine DFSDBUX1.

10
Enter one of the following keywords:

NO
Requests HSSR Engine not to call DFSDBUX1 even if it is in a library concatenated to STEPLIB DD. The option DATXEXIT NO has the same effect; the module DFSDBUX1 is removed from STEPLIB libraries. NO is the default.

YES
Requests that HSSR Engine treat DFSDBUX1 in the same way that IMS does through IMS's Data Conversion exit. If DATXEXIT YES is specified, the module DFSDBUX1 exists in a STEPLIB library, and the use of the Data Conversion exit is designated for a database, then DFSDBUX1 is called each time a DL/I call or an HSSR call is issued against the database.

DBDL1 control statement

The DBDL1 control statement specifies that HSSR calls are to fall back to DL/I calls.

Specifically, the DBDL1 control statement has the following functions:

- Forces IMS High Performance Unload program controller to define PCBs as DL/I PCBs.
- Enables users to temporarily bypass software errors in HSSR Engine.
- Enables users to make performance comparisons between HSSR and DL/I calls.
- Enables an application using a particular PSB to run on one occasion with HSSR Engine, and on another occasion with DL/I.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

DBDL1 dbdname1,dbdname2,dbdname3,dbdname4
      *ALL
```

Position
Description

1
Code the DBDL1 keyword to force HSSR PCBs to be read as DL/I PCBs.

7
Enter in this 8-character field either multiple *dbdname*s or the keyword *ALL. If *dbdname*s are specified, only the PCBs referring to those names are considered DL/I PCBs. If the keyword *ALL is specified, all PCBs are considered DL/I PCBs.

Code *dbdname*s left-aligned, followed by trailing blanks, if necessary, and separated by commas. A maximum of eight *dbdname*s can be coded.

Note: If multiple DBDL1 statements are provided, only the last is used.

DBSTATS control statement

The DBSTATS control statement activates the Database Tuning Statistics function.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

DBSTATS nnnnn
```

Position
Description

1
Code the DBSTATS keyword to instruct HSSR Engine to provide the Database Tuning Statistics report.

9
Specify the number of buffers to be simulated with this entry. Enter any number up to five digits in the range of 1 - 32767, left-aligned, and followed by a blank. If this entry is left blank, the default value of 4 is used.

HSSR Engine simulates the LRU algorithms of the IMS OSAM buffer pool and the IMS VSAM buffer pool to provide statistics of the number of I/O operations. If the application program uses multiple HSSR PCBs, HSSR Engine assumes that each PCB has its own dedicated buffer pool containing the user-specified or default number of buffers.

Restrictions:

- If APISET 3 is specified, this statement cannot be specified.
- If one or more partitions of PHDAM or PHIDAM are in the HALDB OLR cursor-active status, this statement is ignored.

For a complete description of the Database Tuning Statistics function, see Chapter 26, “Obtaining statistics for database tuning,” on page 307.

DIAGG control statement

The DIAGG control statement specifies to generate diagnosis information for status GG.

The DIAGG control statement is used to request HSSR Engine to write diagnostic information to the HSSRTRAC data set whenever a GG or a GX status code is returned. This option is most important when you are unloading a database with the SKERROR option. The diagnosis information documents the location of the database errors and indicates which segment types might be missing in the unloaded database data set.

Important: When you activate the SKERROR control statement to unload a corrupted database, be sure to activate the DIAGG control statement, too.

For the details of diagnosis information, see “Trace Output report with diagnostics information” on page 193; especially, see Type of GG error for a discussion of types of error cases.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

DIAGG
DIAGG DIAGONLY
DIAGG [CB|BUF|CB,BUF|BUF,CB]
DIAGG NOINT
```

Position
Description

1

Code the DIAGG keyword to request that HSSR Engine write diagnosis information to the HSSRTRAC data set whenever a GG or a GX status code is returned.

7

Enter one of the following keywords. You can specify both CB and BUF in a single statement, separated from each other with a comma.

Blank
Interpreted as DIAGONLY.

Tip: You can change this interpretation by replacing the default option table (FABHOPT). See Chapter 19, “Site default options,” on page 261.

DIAGONLY
Writes diagnosis information only. You cannot use this keyword with other keywords.

CB
Writes HSSR control blocks in addition to the diagnosis information.

BUF
Writes buffer handler information in addition to the diagnosis information.

NOINT
Writes buffer handler information in addition to the diagnosis information. You cannot use this keyword with other keywords.

Notes:

- The trace of control blocks of HSSR Engine, produced by specifying CB or BUF option for DIAGG control statement, is not intended to be reviewed by users, but might be needed by IBM Software Support to analyze a problem.
- The DIAGG control statement can write more than 4000 print lines to the HSSRTRAC data set for each GG status code returned. For example, if a segment prefix contains 10 bad pointers, this could yield more than 40,000 print lines for the single bad segment prefix. Therefore, when the DIAGG option is active, the HSSRTRAC data set should be allocated in a way to avoid S722 or SB37 abends. For example, you can specify through the OUTLIM parameter a large number of SYSOUT print lines or you can allocate HSSRTRAC on tape.

GOTRETRY control statement

The GOTRETRY control statement changes the default retry parameters for PROCOPT=GOT.

The GOTRETRY control statement is used to retry database accesses for PROCOPT=GOT. When encountering an incorrect pointer with HIDAM, HDAM, PHIDAM, and PHDAM databases, HSSR Engine, by default, attempts to access the database access four times, with 5-second intervals between attempts. GOTRETRY can be used to change the default value for both the number of access attempts and the number of seconds to wait between these access attempts.

```
0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

GOTRETRY NBR=nnn,WAIT=sss
```

Position
Description

1
Code the GOTRETRY keyword to instruct HSSR Engine to override the default number of re-accesses and the default number of seconds to wait before each re-access attempt.

10
This entry can contain one or both of the following keywords in any order:

NBR=*nnn*
This entry denotes the number of times that HSSR Engine attempts to re-access a database. *nnn* is a left-aligned number in the range of 1 - 999.

WAIT=*sss*
This entry denotes the number of seconds that HSSR Engine waits before it attempts to re-access a database. *sss* is any left-aligned number in the range of 0 - 999.

HPIO control statement

The HPIO control statement specifies whether Media Manager is used for reading VSAM ESDS data sets or OSAM LDS data sets. The use of Media Manager can improve performance, especially the CPU time.

```
0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

HPIO YES
     NO
```

Position
Description

1
Code the HPIO keyword.

6
Specify YES or NO to activate Media Manager or not.

YES
Media Manager is used for reading VSAM ESDS or OSAM LDS. All concatenations of the JOBLIB or the STEPLIB must be APF-authorized.

NO
Media Manager is not used.

When this control statement is not coded and all concatenations of the JOBLIB or the STEPLIB are APF-authorized, Media Manager is used.

Restriction: When IMS High Performance Unload is started by the JCL that is written for IMS HD Reorganization Unload (DFSURGU0), Media Manager is not used to process VSAM ESDSs or OSAM LDSs. If HPIO YES is specified in the HSSROPT DD, the specification is ignored.

HSSRDBD control statement

The HSSRDBD control statement specifies the HSSR PCBs by using DBD names.

The HSSRDBD control statement has the following functions:

- Defines all PCBs that refer to the specified DBDs as HSSR PCBs.
- Enables an application issuing DL/I calls to run on one occasion with HSSR Engine, and on another occasion with DL/I.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

HSSRDBD dbdname1,dbdname2,dbdname3,dbdname4,...
        *ALL
```

Position
Description

1
Code the HSSRDBD keyword to force all PCBs that refer to the specified DBDs to be treated as HSSR PCBs.

9
Enter in this 8-character field either multiple *dbdname*s delimited by a comma or the keyword *ALL. If the length of a *dbdname* is less than 8 bytes, it must be left-aligned and padded with blanks. If *dbdname*s are specified, those PCBs that refer to specified DBD are considered to be HSSR PCBs. If the keyword *ALL is specified, all PCBs that refer to the DBD are considered to be HSSR PCBs.

Notes:

- A maximum of 500 DBD names can be coded.
- HSSRDBD control statement cannot be specified with an HSSRPCB control statement.
- If the list of DBD names cannot fit into one line, the DBD names must be specified as multiple HSSRDBD statements, each of which must fit into a line.
- If both an HSSRDBD statement that has the *ALL operand and an HSSRDBD statement that has the DBD list operand are specified, the *ALL specification has priority over all others.
- If a DBD name is specified on an HSSRDBD statement and is also specified on a DBDL1 control statement, the specification by the DBDL1 statement has priority and all PCBs that refer to the DBD are treated as DL/I PCBs.

HSSRPCB control statement

The HSSRPCB control statement specifies the HSSR PCBs by using PCB numbers.

The HSSRPCB control statement has the following functions:

- Defines the specified DL/I PCBs as HSSR PCBs.

- Enables an application issuing DL/I calls to run on one occasion with HSSR Engine, and on another occasion with DL/I.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

HSSRPCB pcbnum1, pcbnum2, pcbnum3, pcbnum4,...
        *ALL
```

Position
Description

1
Code the HSSRPCB keyword to force the specified PCBs to be treated as HSSR PCBs.

9
Enter in this 3-digit field either multiple database PCB numbers delimited by a comma or the keyword *ALL. The PCB number for the first database PCB is 001. If the PCB numbers are specified, those PCBs are treated as HSSR PCBs. If the keyword *ALL is specified on this field, all database PCBs are considered to be HSSR PCBs.

Notes:

- A maximum of 500 PCBs can be coded.
- The HSSRPCB control statement cannot be specified with the HSSRDBD statement.
- If the list of PCB numbers cannot fit into one line, the PCB numbers must be specified as multiple HSSRPCB statements, each of which must fit into a line.
- If both an HSSRPCB statement that has *ALL operand and an HSSRPCB statement that has PCB number operands are specified, the *ALL specification has priority over all others.
- If a PCB specified on an HSSRPCB statement refers to a DBD that is specified on a DBDL1 control statement, the specification by DBDL1 statement has priority and the PCB is treated as a DL/I PCB.

KEYCHECK control statement

The KEYCHECK optional control statement activates the key sequence check option to ensure that the segment key fields are in ascending sequence.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

KEYCHECK ABEND
         GX
         GG
```

Position
Description

1
Code the KEYCHECK keyword to activate the option that checks the key sequence. It must be used with one of the following three code options. (Separate the words with a space.)

10
Code one of the following optional keywords:

ABEND
Performs key checking and ends abnormally if a sequence error is detected.

GX
Performs key checking and returns a warning GX status code if a sequence error is detected. The segment with the incorrect key is returned normally to the calling application program or utility.

For the EXEC DLI command, the status GX is not returned to the application program. Instead, message DFS1041 is issued in the EXEC DLI interface module (DFSEIPB0). It ends with a code of U1041.

GG
Performs a key check and returns a GG status code if a sequence error is detected. No segment will be returned to the calling application program or utility.

This option requires PROCOPT=GON, GOT, or an active SKERROR; if none of these conditions is specified, HSSR Engine abends instead of returning a GG status code. If the SKERROR control statement is active, HSSR Engine does not retrieve the incorrect segment or other related segments during the processing of the next GN call. If the SKERROR control statement is inactive, HSSR Engine resets the current position to the beginning of the database.

Restrictions:

- If APISET 3 is specified, this statement cannot be specified.
- If one or more partitions of PHDAM or PHIDAM are in the HALDB OLR cursor-active status, KEYCHECK=ABEND/GX/GG is ignored.

LOUT control statement

The LOUT control statement requests that, when the DBSTATS control statement activates the Database Tuning Statistics function, the HSSRLOUT data set contains only the records that satisfy certain conditions.

The records that satisfy one or both of the following conditions are written in the HSSRLOUT data set:

- Database records that are longer than the specified limit.
- Database records that require more than the specified number of database I/Os.

This option is ignored if the DBSTATS control statement is not specified at the same time.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

LOUT LENGTH=llllllll
LOUT IO=nn
```

Position
Description

1
Code the LOUT keyword to request an optional record selection for the HSSRLOUT data set.

6
Code one of the following optional keywords for each LOUT statement:

LENGTH=*llllllll*
Requests that only the records for database records whose length is greater than *llllllll* bytes are written in the HSSRLOUT data set.

IO=*nn*
Requests that only the records for database records that require more than *nn* database I/Os for retrieval are written in the HSSRLOUT data set.

You can specify both LENGTH= and IO= parameters, either on a single LOUT control statement or separately (on multiple LOUT statements). If you specify both parameters on a single LOUT statement, separate them with a comma, as follows:

```
LOUT LENGTH=100,IO=15
```

If you specify both LENGTH= and IO= parameters, both conditions are effective (that is, database records that satisfy both conditions are written in the HSSRLOUT data set.)

If an incorrect parameter keyword or an incorrect parameter value is detected on an LOUT statement, the rest of the parameter specification on that statement is ignored. For example, the IO= parameter specification on the following statement is ignored because the LENGTH= parameter value is incorrect:

```
LOUT LENGTH=1A,IO=10
```

For a complete description of the Database Tuning Statistics function, see Chapter 26, “Obtaining statistics for database tuning,” on page 307.

LSR control statement

The optional LSR control statement is used to request that HSSR Engine share the local shared resource (LSR) pool with DL/I while processing primary index of HIDAM or PHIDAM databases.

For the details of this option and how to code DFSVSAMP DD when LSR YES is specified, see “VSAM LSR option for primary index databases” on page 210.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

LSR NO
LSR YES
```

Position
Description

1
Code the LSR keyword to specify the LSR option.

5
Enter one of the following keywords:

NO
Requests that primary indexes of HIDAM and PHIDAM databases be processed with the NSR option. NO is the default.

Tip: This default can be changed by replacing the default option table (FABHOPT). See Chapter 19, “Site default options,” on page 261.

YES
Requests that primary indexes of HIDAM and PHIDAM databases be processed with the LSR option, to improve the performance (through better look-aside buffering) of programs that issue numerous GU calls to HIDAM or PHIDAM databases.

NOFIX control statement

By default, buffer pools are page-fixed. To avoid page-fixing, you can use a NOFIX control statement.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

NOFIX
```

Position
Description

1
Code the NOFIX keyword to activate the NOFIX option.

NOVSAMOPT control statement

The NOVSAMOPT control statement prevents the override of the read-ahead threshold value used by VSAM.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

NOVSAMOPT
```

Position
Description

1
Code the NOVSAMOPT keyword to activate the NOVSAMOPT option.

PARTINFO control statement

This optional control statement specifies whether any optional information is to be reported when a HALDB is processed.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

PARTINFO DEF,ACC
```

Position
Description

1
Code the PARTINFO keyword to initiate HSSR Engine to produce the partition information report.

10
Specify which report to produce.

These parameters are not positional and can be specified in any order. If two or more parameters are specified, put a comma between parameters. Information specified on this statement affects the number of lines written in HSSRSTAT data set.

At least one parameter must be specified.

DEF
Initiate HSSR Engine to produce the HALDB Partition Definition report.

ACC
Initiate HSSR Engine to produce the HALDB Partitions Accessed report.

PCBLIST control statement

The PCBLIST control statement specifies the type of PCB list that is to be passed to the application program.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

PCBLIST HSSR
        IMS
```

Position
Description

1
Code the PCBLIST keyword to specify a type of PCB list.

9
Code one of the following keywords:

HSSR
The PCB list that is built by HSSR Engine and that can contain an entry that points to HSSR PCB is passed to the application program.

IMS
The PCB list that is built by IMS is passed to the application program.

PCBLIST HSSR is the default.

Notes:

- If your application program issues a DL/I system call that gets access to the IMS PCB list, it is recommended that you specify PCBLIST IMS. One such call is the INQY call with the FIND subfunction, which returns the PCB address of the requested PCB name.
- If your application program issues an HSSR call as an EXEC DLI command, you must specify PCBLIST IMS.

Tip: The default of this control statement can be changed by replacing the default option table (FABHOPT). For details, see Chapter 19, “Site default options,” on page 261.

RETRY control statement

The RETRY control statement specifies to retry a failing KSDS I/O operation.

In a database-sharing environment, HSSR Engine might encounter VSAM logical errors if it reads a KSDS data set that is being concurrently updated and is also subject to CI splits. By default, HSSR Engine abends in such a situation. You can use the optional RETRY control statement. However, to instruct HSSR Engine to refresh its buffers and retry the failing KSDS I/O operation, HSSR Engine accesses data that has been inserted, during a control area split, into a new control area at the end of the KSDS.

Note: When HSSR Engine reads a KSDS data set that is subject to CI splits, any KSDS records that shifted out of the CI being split might be skipped. CI splits might also cause some records to be read twice.

```
0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

RETRY KSDS
```

Position
Description

1
Code the RETRY keyword.

7
Code the KSDS keyword to activate the retry operation of HSSR Engine.

RTEXIT control statement

The RTEXIT control statement is used to specify the name of a runtime environment exit routine that can initialize and terminate processing required by the user's environment, such as a COBOL II runtime environment.

The FABHRTEX module is provided as the default runtime environment exit routine, which is a dummy exit routine that returns to HSSR Engine without any processing. If any initialization or termination processing is required by your environment, you can modify this routine or write your own exit routine.

Note: IBM Enterprise COBOL Version 5 or later does not support the IGZERRE interface for the runtime environment setup. If the interface is used in your runtime environment exit routine, remove the RTEXIT control statement. Instead, specify a language environment option to invoke CEEPIPI provided by Language Environment not only for initialization and termination calls but also for each segment call of your user exit routine for unload utilities. For more information, see “EXIT control statement” on page 253 for FABHURG1 or “PSB control statement” on page 61 for FABHFSU.

The runtime environment exit routine is called before and after an HSSR application program is invoked. If no RTEXIT control statement is specified, the IBM-supplied FABHRTEX module or the user-written FABHRTEX module is called as the runtime environment exit routine. For more information about FABHRTEX, see “Runtime Environment exit (FABHRTEX)” on page 243.

```
0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

RTEXIT xxxxxxxx
```

Position
Description

1
Code the RTEXIT keywords.

8
The left-aligned load module name of the runtime environment exit routine.

SKERROR control statement

The SKERROR control statement allows database errors to be bypassed. This option enables retrieval from a database even if the database contains incorrect HD pointers or incorrect HISAM records.

This bypass is achieved by skipping the processing either of the incorrect pointer or of the remainder of the incorrect HISAM record. A GG status code is returned. Thus, HSSR Engine cannot process segments with segment codes in error.

SKERROR is used when a corrupted HIDAM, HDAM, PHIDAM, PHDAM, or HISAM database is unloaded with FABHURG1. You can then reload this database through the IMS HD Reorganization Reload utility or a compatible utility, to create a new version of the database. Although the new version meets the technical requirements for a valid IMS database, some segments might be omitted. For more details, see the following topics:

- Chapter 9, "Utility options for unloading corrupted databases," on page 115, for procedures to follow when recovering a corrupted database.
- "Trace Output report with diagnostics information" on page 193, for a discussion of the types of error cases encountered and printed on the Trace Output report.

Notes:

- If you specify APISET 3, you cannot specify this statement.
- If one or more partitions of PHDAM or PHIDAM are in the HALDB OLR cursor-active status, SKERROR=*nnnnnnnn* is ignored.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

SKERROR nnnnnnnnn
```

Position
Description

1
Code the SKERROR keyword to activate the error-skipping option.

9
Specify the maximum number of incorrect records to be skipped. Enter up to 9 digits, left-aligned and followed by a blank. If the field is left blank, HSSR Engine skips up to 9999 incorrect records. If more than this number of GG status codes are returned, HSSR Engine issues an abend.

When you include the SKERROR control statement to unload a corrupted database, be sure to include a DIAGG control statement.

Note: The SKERROR control statement can be used for HSSR PCBs that do not have an update PROCOPT specified. For example, it can be used for HSSR PCBs that have PROCOPT equal to G, GE, GO, GON, or GOT; but it is ignored for all HSSR PCBs that have PROCOPT=R.

SKIPAUTH control statement

The SKIPAUTH control statement specifies whether to bypass IMS DBRC database authorization to avoid DBRC authorization failure (DFS047A) for HALDB partitions.

SKIPAUTH YES is effective only when the HALDB is processed in the ULU region. Otherwise, it is ignored.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

SKIPAUTH YES
         NO
```

Position
Description

1
Code the SKIPAUTH keyword to bypass IMS DBRC database authorization.

10
Specify one of the following keywords:

YES
Requests HSSR Engine to bypass IMS DBRC database authorization.

NO
Requests HSSR Engine not to bypass IMS DBRC database authorization. NO is the default value.

SKIPVLC control statement

The SKIPVLC control statement requests HSSR Engine to ignore a sensitive virtual logical child segment in the DLI or the DBB region.

In the ULU region, the virtual logical child segment type is always ignored, therefore this control statement is not used.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

SKIPVLC YES
        NO
```

Position
Description

1
Code the SKIPVLC keyword to ignore a sensitive virtual logical child segment in the HSSR PCB.

9
Specify one of the following keywords:

YES
Requests HSSR Engine to ignore a sensitive virtual logical child segment in the HSSR PCB.

NO
Requests HSSR Engine not to ignore the sensitive virtual logical child. The HSSR Engine does not treat the PCB, in which the logical child segment is sensitive, as an HSSR PCB. NO is the default.

- IMS High Performance Unload utilities such as FABHURG1, FABHFSU, and FABHTEST issue a user abend because the PCB is not an HSSR PCB.
- In a user application program that uses the IMS High Performance Unload API, all HSSR calls to the PCB fall back to DL/I calls.

TRDB control statement

The TRDB control statement activates the trace of HSSR Engine. Specify this control statement to obtain the snap-like print of database segments.

The TRDB control statement must be used with the TRHC control statement. After the TRHC indicates the type of trace to be run, the TRDB control statement then identifies the specific databases on which these traces are run.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

TRDB *ALL
     dbdname1,dbdname2,dbdname3,dbdname4
```

Position
Description

1
Code the TRDB keyword to activate the hardcopy tracing for specified database.

6
Enter either *dbdnames* separated by commas or the keyword *ALL.

Each *dbdname* must meet the following requirements:

- It must occupy eight bytes and be left-aligned.
- If the name is less than eight characters, trailing blanks must be specified.
- The last *dbdname* must be followed by a blank.

If multiple TRDB statements are provided, only the last statement is used.

TRHC control statement

The TRHC control statement activates the hardcopy trace option of HSSR Engine. Use this control statement to obtain the snap-like print of database segments.

This control statement is always used in combination with the TRDB control statement. When you activate this option, HSSR Engine traces calls that are issued against the databases named in the TRDB control statement. It writes data about these calls on the HSSRTRAC data set.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

TRHC CALL,CB,CBX,BUF,BUFCB,START=n,NPF
```

Position
Description

1
Code the TRHC keyword to activate the hardcopy tracing option.

6
Enter one or more of the following keywords, separated by commas, specified in any order. The last keyword must be followed by a blank.

CALL
Traces call information such as call function, PCB, IOAREA, SSA, and segment prefix.

Does a trace for the EXEC DLI command, just the same as for the DL/I call. This is because the command is converted to the format of a DL/I call and made to the HSSR call handler.

CB or CBX
Traces the control blocks of HSSR Engine. If CBX is specified, traces are also done for the extended control blocks of HSSR Engine.

If both CB and CBX are specified, CBX is taken.

BUF
Traces buffer handler information.

BUFCB
Traces the CAB buffer handler control blocks.

START=*n*
Specifies the call number *n*. Trace begins at the *n*-th HSSR call issued by the application program. Enter any number up to nine digits, left-aligned, and followed by a blank.

NPF
Excludes the segment prefix information from the trace.

Only HSSR calls issued against the databases that are specified on the TRDB control statement are traced.

Note: The trace of the control blocks of HSSR Engine, which is called for by specifying the CB, CBX, BUF, or BUFCB option for the TRHC control statement, is not intended to be reviewed by users, but might be needed by the IBM Software Support to analyze a problem.

TRXC control statement

The TRXC control statement activates the wrap-around core tracing option. HSSR Engine traces all HSSR calls issued by the application program and stores the information in a table maintained in storage.

In the *user subpools* portion of the dump, table entries show the following information about the last *n* calls for each PCB:

- Eye-catcher 'TRAC'
- Last byte of application program call function
- Last byte of PCB status code
- Segment name
- PCB key feedback area
- Address within the buffers of the returned segment
- Internal information about HSSR Engine

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

TRXC nnnnnnnnn
```

Position
Description

1
Code TRXC to activate the wrap-around core trace of HSSR calls.

6
Enter a number to designate the number of entries allocated for each HSSR PCB for wrap-around tracing.

Enter up to 9 digits, left-aligned and followed by a blank. If this position is left blank, the default value of 10 is used.

ZIIPMODE control statement

The ZIIPMODE control statement specifies whether HSSR engine offloads eligible VSAM ESDS I/O or OSAM LDS I/O workloads to zIIP processors.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

ZIIPMODE NEVER
         COND
```

Position
Description

1
Code the ZIIPMODE keyword.

10
Specify one of the following keywords:

NEVER
Does not offload any workload to zIIP processors. NEVER is the default to avoid unexpected performance degradation.

COND
Offloads VSAM ESDS I/O or OSAM LDS I/O workloads to zIIP processors when all of the following conditions are met:

- zIIP processors are available.
- The SGLXLOAD library of IMS Tools Base is specified to the STEPLIB or JOBLIB.
- VSAM ESDS is to be read using Media Manager. For details, see “HPIO control statement” on page 168.

If any of these conditions is not satisfied, the job runs using the main CPs.

Chapter 12. Reports and output from HSSR Engine

HSSR Engine generates statistical reports in the HSSRSTAT data set. It also generates other information in certain data sets according to the options that were used in the job.

The following reports are generated:

- The HSSRSTAT data set contains printed statistical reports.
- The HSSRTRAC data set contains printed trace reports.
- The HSSRSNAP data set contains, when an initialization error occurs, a snapshot of control blocks.
- The HSSRLOUT data set contains, for each database record, a record that can be used for tuning database.
- The HSSRBUTR data set contains buffer handler trace data that can be used for input to the FABHBSIM utility.

Topics:

HSSRSTAT data set

This output data set contains statistical reports that are generated by HSSR Engine.

In accordance with the control statements that were specified for the job, this data set contains one or more of the following reports:

- HSSROPT Control Statements report
- HALDB Partition Definition report
- HALDB Partitions Accessed report
- DB Call Statistics report
- DB Statistics report
- Randomizing Statistics report
- DB Record Length Distribution report
- Data Set I/O Statistics report
- CAB Statistics report

When you use the Sequential Subset Statistics (SS-STATS) function of the Sequential Subset Randomizer, the HSSRSTAT data set also contains the reports of the SS-STATS routine. For information about these reports, see "HSSRSTAT output data set when SS-STATS routine is applied" on page 295.

Format

This data set contains 133-byte fixed-length records. When the block size is coded in the JCL, the block size must be a multiple of 133.

HSSROPT Control Statements report

This report contains the parameters used by HSSR Engine for this job execution.

The following figure shows an example of the report.

```
IMS HIGH PERFORMANCE UNLOAD                              "HSSROPT CONTROL STATEMENTS"                              PAGE:     1
5655-E06                                               DATE: 06/01/2021  TIME: 10.14.01                        FABH015A - V1.R2


 HSSROPT CONTROL STATEMENTS FOLLOWS:

   0........1.........2.........3.........4.........5.........6.........7.........8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890

   HSSRDBD DBDNAME1,DBX
   HSSRDBD DBDNAME2,DBDNAME3
   DBSTATS
   RTEXIT FABHRRRR
 X DIAGG AAA
   SKERROR 500
   NOFIX

   NOTE: "X" ON THE LEFT SIDE OF A CONTROL STATEMENT INDICATES THAT THE STATEMENT HAS AN ERROR AND IS IGNORED.

 HSSR-ENGINE PARAMETERS USED FOR THIS EXECUTION:

   KEYWORD          VALUES
   ---------------- ----------------------------------------------------------------------------------------------
   APISET           1
   CABBASE          1,OSAM,SEQ
   CABBASE          2,OSAM,DIR
   CABBASE          8,VSAM,SEQ
   CABBASE          15,VSAM,DIR
   CABSTAT          NO
   COMPAUTH         NO
 * DBSTATS          4
 * DIAGG            DIAGONLY
   GOTRETRY         NBR=4,WAIT=5
 * HSSRDBD          DBDNAME1,DBDX    ,DBDNAME2,DBDNAME3
   LSR              NO
 * NOFIX
   PCBLIST          HSSR
 * RTEXIT           FABHRRRR
 * SKERROR          500
   SKIPVLC          NO
   SKIPAUTH         NO
   DATXEXIT         NO
   ZIIPMODE         NEVER

   NOTE: '*' INDICATES THAT THE PARAMETER IS SPECIFIED IN THE HSSROPT.
```

Figure 36. HSSROPT Control Statements report

HALDB Partition Definition report

This report contains definition information about the HALDB partitions.

If PARTINFO DEF is specified in the HSSROPT data set, this report is generated.

The following figure shows an example of this report.

```
IMS HIGH PERFORMANCE UNLOAD                            "HALDB PARTITION DEFINITION"                              PAGE:     1
5655-E06                                               DATE: 06/01/2021  TIME: 12.34.56                        FABHP00  - V1.R2

*** DBDNAME: PARTDB   ***

  DATABASE ORGANIZATION.............    PHDAM
  ACCESS METHOD.....................     VSAM
  NUMBER OF PARTITIONS DEFINED.....         5
  PARTITION SELECTION EXIT.........    (N/A)

*** PARTITIONS LISTED IN ORDER OF PARTITION HIGH KEY ***

  SEQ   NAME     ID     HIGH KEY

  0001  PART1        1  C'19999999'
  0002  PART2        2  C'29999999'
  0003  PART3        3  C'39999999'
  0004  PART4        4  C'49999999'
  0005  PART5        5  C'59999999'
```

Figure 37. HALDB Partition Definition report

HALDB Partitions Accessed report

This report summarizes information about HALDB partitions that were accessed.

If PARTINFO ACC is specified in the HSSROPT data set, this report is generated.

The following figure shows an example of this report.

```
IMS HIGH PERFORMANCE UNLOAD                              "HALDB PARTITIONS ACCESSED"                              PAGE:     1
5655-E06                                               DATE: 06/01/2021  TIME: 12.34.56                       FABHP00  - V1.R2

*** DATABASE REFERENCED BY THE PCB #0001 ***

  DBDNAME.......................... PARTDB
  NUMBER OF PARTITIONS DEFINED.....        5
  NUMBER OF PARTITIONS PROCESSED....       3

*** LIST OF PARTITIONS ACCESSED ***
  PART1  , PART2  , PART3
```

Figure 38. HALDB Partitions Accessed report

DB Call Statistics report

This report contains statistics about the number of HSSR calls.

Subtopics:

- "Example report: DB Call Statistics report" on page 183
- "Example report: DB Call Statistics report for HALDB" on page 183
- "Report field descriptions" on page 184

Example report: DB Call Statistics report

The following figure shows an example of this report.

```
IMS HIGH PERFORMANCE UNLOAD                                  "DB CALL STATISTICS"                                  PAGE:     1
5655-E06                                                   DATE: 06/01/2021  TIME: 16.40.07                       FABH070  - V1.R2

JOBNAME=OEFH2005     STEPNAME=G       .HPUL         PGMNAME=FABHURG1     PSBNAME=PHDV0100
-------------------------------------------------------------------------------------------------------------------------------
DBDNAME   SEGNAME       CALLS WITH BLANK/GX STATUS CODES                          OTHER STATUS CODES
                      ------------------------------------- ------------------------------------------------------------------
                            GN/GHN        GU/GHU          REPL            GB            GE            GG            VX         NI/NE

PHDV0100  ROOTLEV1             10
          DEP1LEV2             10
          DEP2LEV2             10
          DEP3LEV3             20
          DEP4LEV4             40
          DEP5LEV2             10
          DEP6LEV3             20
          DEP7LEV4             40
          DEP8LEV4             40
          DEP9LEV5             80
        PCB-TOTALS            280                                    1
                      -------------------------------------------------------------------------------------------------------------
TOTAL HSSR CALLS              280                                    1
                      =============================================================================================================
```

Figure 39. DB Call Statistics report

Example report: DB Call Statistics report for HALDB

If CALLSTAT PART is specified in the HSSROPT data set, the partition-wide DB Call Statistics report is printed for each HALDB partition that is processed, and from which at least one segment was retrieved. This report is printed in addition to the database-wide DB Call Statistics report. The PARTITION=*xxxxxxx* shows the partition name.

The following figure shows an example of this report.

```
IMS HIGH PERFORMANCE UNLOAD                                  "DB CALL STATISTICS"                                  PAGE:     1
5655-E06                                                   DATE: 06/01/2021  TIME: 16.40.07                       FABH070  - V1.R2

JOBNAME=OEFH2005     STEPNAME=G       .HPUL         PGMNAME=FABHURG1     PSBNAME=PHDV0100     PARTITION=PHDV01A
-------------------------------------------------------------------------------------------------------------------------------
DDNAME    SEGNAME       CALLS WITH BLANK/GX STATUS CODES                          OTHER STATUS CODES
                      ------------------------------------- ------------------------------------------------------------------
                            GN/GHN        GU/GHU          REPL            GB            GE            GG            VX         NI/NE

PHDV01AA  ROOTLEV1              2
          DEP1LEV2              2
          DEP2LEV2              2
          DEP3LEV3              4
          DEP4LEV4              8
          DEP5LEV2              2
          DEP6LEV3              4
          DEP7LEV4              8
          DEP8LEV4              8
          DEP9LEV5             16
        PART-TOTALS            56
                      -------------------------------------------------------------------------------------------------------------
```

Figure 40. DB Call Statistics report for HALDB

Report field descriptions

The following information is provided in the report:

JOBNAME
Name of the job.

STEPNAME
Name of the step for which DB Call Statistics are logged. If you use a cataloged procedure (that is, FABHDBB, FABHDLI, or FABHULU), this STEPNAME is shown as `G.` *xxxxxxxxx* (where *xxxxxxxx* is the name specified for your EXEC statement).

PGMNAME
Name of application program.

PSBNAME
Name of PSB containing the PCBs that are used.

DBDNAME
Dbdname of the database referred to by the HSSR PCB.

SEGNAME
All segment names that are accessed.

CALL WITH BLANK/GX STATUS CODES
Number of HSSR GN or GHN, GU or GHU, and REPL calls for each segment type.

Note: GNP and GHNP calls are listed under "GN/GHN."

OTHER STATUS CODES
Number of GB, GE, GG, VX, NI, and NE status codes returned for each segment type. NI and NE status codes together are listed under "NI/NE".

PCB TOTALS
Total number of calls and status codes for a single PCB.

TOTAL HSSR CALLS
Total number of calls and status codes.

The following fields are reported only when the status code FM is received.

STATUS:FM
Number of FM status codes for a single PCB.

TOTAL STATUS:FM
Total number of FM status codes.

The following fields are reported only when the status code GP is received:

STATUS:GP
Number of GP status codes for a single PCB.

TOTAL STATUS:GP
Total number of GP status codes.

DB Statistics report

This report contains information about the average number of I/Os that are required to randomly read all database segments of one database record, and the length of database records.

You can use this report to tune your databases. For a complete description of this report, see "DB Statistics report" on page 323.

Randomizing Statistics report

This report contains the statistics with key indicators that represent the quality of randomizing.

You can use this report to tune your databases. For a complete description of this report, see "Randomizing Statistics report" on page 328.

DB Record Length Distribution report

This report contains information about the distribution of database record lengths.

You can use this report to tune your databases. For a complete description of this report, see "DB Record Length Distribution report" on page 330.

Data Set I/O Statistics report

This report contains statistics about I/O and buffer handler activities.

Subtopics:

Example report: Data Set I/O Statistics report

The following figure shows an example of this report.

```
IMS HIGH PERFORMANCE UNLOAD                                  "DATA SET I/O STATISTICS"                                PAGE:     1
5655-E06                                                   DATE: 06/01/2021  TIME: 16.16.15                     FABH070  - V1.R2


JOBNAME=THFH0601     STEPNAME=G       .URG1HD       PGMNAME=FABHURG1     PSBNAME=PHDV030H

DDNAME   SIZE OF     NUM OF         IO         IO        RBA     PCT     PCT
           A BUF    BUFFERS     DIRECT       SEQU   REQUESTS  IO/REQ  IO/BLK
-------- ------- ---------- ---------- ---------- ---------- ------- -------
PHDV03AA     512        143          0          8        106    7.54    5.44
   OVFLW     512        143          0          1          9   11.11     .68
PHDV03AB   4,096         66          0          1          5   20.00    8.33
PHDV03AC   2,048         66          0          2         10   20.00    9.52
PHDV03BA     512        143          0          8        102    7.84    5.44
   OVFLW     512        143          0          1          7   14.28     .68
PHDV03BB   4,096         66          0          1          4   25.00    8.33
PHDV03BC   2,048         66          0          2          9   22.22    9.52
PHDV03CA     512        143          0          8        107    7.47    5.44
   OVFLW     512        143          0          1          9   11.11     .68
PHDV03CB   4,096         66          0          1          4   25.00    8.33
PHDV03CC   2,048         66          0          2          9   22.22    9.52
PHDV03DA     512        143          0          8        103    7.76    5.44
   OVFLW     512        143          0          1          7   14.28     .68
PHDV03DB   4,096         66          0          1          3   33.33    8.33
PHDV03DC   2,048         66          0          2          9   22.22    9.52
PHDV03EA     512        143          0          8        111    7.20    5.44
   OVFLW     512        143          0          2         18   11.11    1.36
PHDV03EB   4,096         66          0          1         12    8.33    8.33
PHDV03EC   2,048         66          0          3         25   12.00   14.28
-------- ------- ---------- ---------- ---------- ---------- ------- -------
TOTAL                                0         62        669    9.26

TOTAL NUMBER OF BYTES IN BUFFER POOL:        551,936
```

Figure 41. Data Set I/O Statistics report

Report field descriptions

JOBNAME
Name of the job.

STEPNAME
Name of the step for which Data Set I/O Statistics are logged. If you use a cataloged procedure (that is, FABHDBB, FABHDLI, or FABHULU), this STEPNAME is shown as `G.` *xxxxxxxxx* (where *xxxxxxxx* is the name specified for your EXEC statement). In this example report, the cataloged procedure is used, and the name of the EXEC statement is URG1HD.

PGMNAME
Name of application program.

PSBNAME
Name of PSB containing the PCBs that are used.

For each PCB data set combination, the following information is printed:

DDNAME
ddname as coded on the DD1=*keyword* in the DBD that is referred to by the HSSR PCB.

SIZE OF A BUF
Buffer size.

NUM OF BUFFERS
Total number of buffers.

IO DIRECT
Number of issued direct I/O requests.

IO SEQU
Number of chained sequential I/O requests.

RBA REQUESTS
Number of buffer handler requests.

PCT IO/REQ
Percentage of issued I/O operations per buffer handler request.

PCT IO/BLK
Percentage of issued I/O operations per number of blocks or CIs contained in the data set.

The TOTAL line appears once on the report:

TOTAL
This line contains totals of the above fields.

TOTAL NUMBER OF BYTES IN BUFFER POOL
The total number of bytes of the storage allocated for the buffer pool.

Note: If one or more partitions of PHDAM or PHIDAM are in the HALDB OLR cursor-active status, each count number is 0 for each data set of the partitions.

CAB Statistics report

This report contains detailed CAB information for the data set that is referred to by each PCB.

The following statistics are produced for each PCB for which CAB is used:

- HALDB buffering statistics (only for HALDBs)
- CAB environment (statistics)
- I/O summary (statistics)
- Timing Statistics

If the CABSTAT YES control statement is specified in the HSSROPT input data set, the following statistics are also produced:

- Event statistics
- Distribution of look-aside at *n*th position
- Distribution of HRAN steal/delete per reference count value
- Distribution of reference-count-difference
- Distribution of distance from current Seq HRAN

These statistics contain detailed technical information beyond what is usually required for CAB users. To use them, you need to have a good knowledge of CAB logic.

For a database of multiple data set groups, these statistics, except the HALDB buffering statistics, are produced for each data set.

For a HALDB, the statistics are produced for each partition that has been processed; also, if the database consists of multiple data set groups, the statistics are produced for each data set group for each partition.

For an HDAM or PHDAM database, the statistics are produced for the root addressable area and for the overflow area.

The DDNAME or the pair of partition name and DSGROUP name is printed on each page of the report. For a nonpartitioned database, the *ddname* as coded on the DD1= keyword in the DBD that is referred to by the HSSR PCB is printed. For a HALDB, the partition name and the character that identifies the data set group are printed.

Subtopics:

- “Example report: HALDB Buffering Statistics report for PHDAM” on page 187
- “Example report: CAB Statistics report for a nonpartitioned database” on page 187
- “Report field descriptions: CAB Statistics report” on page 188

Example report: HALDB Buffering Statistics report for PHDAM

The following report shows a sample of HALDB buffering statistics report. These statistics are produced only for HALDB. The CAB parameters used for each data set are reported. The total storage size for CAB buffers for each data set group is also reported. Column 'A' of the table, titled "CAB PARAMETERS FOR EACH PARTITION", indicates whether the data set was accessed ('Y') or not ('N').

```
IMS HIGH PERFORMANCE UNLOAD                                    "CAB STATISTICS"                                    PAGE:     1
5655-E06                                                 DATE: 06/01/2021  TIME: 22.15.13                    FABHAP0  - V1.R2


                                                           HALDB BUFFERING STATISTICS

DB=PHDV0300  PCB#=    1
-------------------------------------------------------------------------------------------------------------------------------------

*** CAB PARTITION PROCESSING INTENT

          PARTPROC PHDV0300 S


*** BUFFERING SUMMARY

    THE NUMBER OF PARTITIONS DEFINED.............     5
    THE NUMBER OF PARTITIONS ACCESSED............     5
    MAX NUMBER OF PARTITIONS ACCESSED AT A TIME...     1

    DSGROUP=A

       CAB PARAMETERS FOR THIS DSGROUP
          OVERFLOW   CAB

       CAB PARAMETERS FOR EACH PARTITION (RAA)
          PARTITION A   CI SIZE   CI'S/CA RANSIZE NBRSRAN NBRDBUF REFT4 INTER
          --------- - --------- --------- ------- ------- ------- ----- -----
          PHDV03A   Y       512        49      13       8      26    13    NO
          PHDV03B   Y       512        49      13       8      26    13    NO
          PHDV03C   Y       512        49      13       8      26    13    NO
          PHDV03D   Y       512        49      13       8      26    13    NO
          PHDV03E   Y       512        49      13       8      26    13    NO

       THE SIZE OF CAB BUFFER FOR THIS DSGROUP (RAA)
          NUMBER OF BYTES IN SEQUENTIAL BUFFER...      66,560
          NUMBER OF BYTES IN DIRECT BUFFER.......      13,312

       CAB PARAMETERS FOR EACH PARTITION (OVERFLOW)
          PARTITION A   CI SIZE   CI'S/CA RANSIZE NBRSRAN NBRDBUF REFT4 INTER
          --------- - --------- --------- ------- ------- ------- ----- -----
          PHDV03A   Y       512        49      13       8      26    13    NO
          PHDV03B   Y       512        49      13       8      26    13    NO
          PHDV03C   Y       512        49      13       8      26    13    NO
          PHDV03D   Y       512        49      13       8      26    13    NO
          PHDV03E   Y       512        49      13       8      26    13    NO

       THE SIZE OF CAB BUFFER FOR THIS DSGROUP (OVERFLOW)
          NUMBER OF BYTES IN SEQUENTIAL BUFFER...      66,560
          NUMBER OF BYTES IN DIRECT BUFFER.......      13,312
```

Figure 42. HALDB Buffering Statistics report for PHDAM

Example report: CAB Statistics report for a nonpartitioned database

The following figure shows an example of CAB statistics report for a non-partitioned database.

```
IMS HIGH PERFORMANCE UNLOAD                                    "CAB STATISTICS"                                    PAGE:     1
5655-E06                                                 DATE: 06/01/2021  TIME: 12.14.27                    FABHA90  - V1.R2

DDNAME=HSHDP
-------------------------------------------------------------------------------------------------------------------------------

*** CAB ENVIRONMENT
          OPERATING SYSTEM   z/OS 01.07.00 HBB7720
          ACCESS METHOD      BSAM
          BUFFER FIXING      YES
          RANSIZE               2
          NBRSRAN               4
          NBRDRAN               4
          NBRDBUF               3
          OVERLAP            YES
          REFT1                 0
          REFT2                 0
          REFT3                 0
          REFT4                 3
          NBHSIZE               3
          INTER              NO
          ANYNEXT            NO


*** I/O SUMMARY:
          NBR OF BLOCKS WITHIN DATASET...       2,441
          NBR OF CALLS TO BUFFERHANDLER..       4,165     100.00  PCT OF CALLS     170.62  PCT OF BLOCKS

          LOOKASIDE TOTAL................       1,240      29.77  PCT OF CALLS      50.79  PCT OF BLOCKS
             INTRA-PCB: SEQ BUF..........       1,199      28.78  PCT OF CALLS      49.11  PCT OF BLOCKS
             INTRA-PCB: DIRECT BUF.......          41        .98  PCT OF CALLS       1.67  PCT OF BLOCKS
             INTER-PCB...................           0          0  PCT OF CALLS          0  PCT OF BLOCKS

          NBR OF I/O
              TOTAL......................       3,200      76.83  PCT OF CALLS     131.09  PCT OF BLOCKS
              SEQUENTIAL TOTAL...........         982      23.57  PCT OF CALLS      40.22  PCT OF BLOCKS
                 SEQU IMMEDIATE..........         707      16.97  PCT OF CALLS      28.96  PCT OF BLOCKS
                 SEQU OVERLAPPED.........         275       6.60  PCT OF CALLS      11.26  PCT OF BLOCKS
              DIRECT.....................       2,218      53.25  PCT OF CALLS      90.86  PCT OF BLOCKS

          NBR OF BLOCKS READ
              TOTAL......................       4,182     100.40  PCT OF CALLS     171.32  PCT OF BLOCKS
              SEQUENTIAL TOTAL...........       1,964      47.15  PCT OF CALLS      80.45  PCT OF BLOCKS
                 SEQU IMMEDIATE..........       1,414      33.94  PCT OF CALLS      57.92  PCT OF BLOCKS
                 SEQU OVERLAPPED.........         550      13.20  PCT OF CALLS      22.53  PCT OF BLOCKS
              DIRECT.....................       2,218      53.25  PCT OF CALLS      90.86  PCT OF BLOCKS


*** NBR OF UNREFERENCED SEQ BUFFERS               490      24.94  PCT SEQ BUFFERS


*** TIMING  STATISTICS                      MEAN WAIT TIME    TOTAL WAIT TIME
                                                (MILLIS)        (SECONDS)
              IMMEDIATE SEQ I/O..........          24                17
              OVERLAPPED SEQ I/O.........           6                 1
              IMMEDIATE DIRECT I/O.......          20                45
              *TOTAL.....................                            64
```

Figure 43. CAB Statistics report for a nonpartitioned database

Report field descriptions: CAB Statistics report

The following information is provided in the CAB Statistics report:

CAB ENVIRONMENT
Description of the environment and CAB parameter values specified in the HSSRCABP data set.

I/O SUMMARY
Summary of CAB I/O.

NBR OF BLOCKS WITHIN DATASET
Number of OSAM blocks, OSAM LDS CIs, or ESDS CIs in the data set. This value is calculated by (high-used RBA)/(CI size).

NBR OF CALLS TO BUFFER HANDLER
Number of buffer handler requests.

If basic buffering is used for the overflow area specified by the OVERFLOW control statement, the number of calls to the buffer handler for the overflow area is not included in this value.

Buffer handler is called when the requested RBA is not in the current buffer.

LOOKASIDE TOTAL
Number of buffer handler requests satisfied through look-aside buffering.

INTRA-PCB: SEQ BUF
The number of requested RBAs found in the sequence buffer of the intra-PCB (my PCB).

INTRA-PCB: DIRECT BUF
The number of requested RBAs found in the direct buffer of the intra-PCB (my PCB).

INTER-PCB:
The number of requested RBAs found in the buffers of the inter-PCB (PCB other than my PCB). The buffers for the inter-PCB are referred to only when the requested RBA was not found in the intra-PCB.

NBR OF I/O
Summary of number of I/O requests. For VSAM ESDS or OSAM LDS, one GET macro is used by one I/O request. For OSAM, the READ macro is used and the number of READ macros is:

- 1 when direct I/O
- The value of RANSIZE when chained sequential I/O

TOTAL
Total number of I/O requests.

SEQUENTIAL TOTAL
Total number of chained sequential I/O requests.

SEQU IMMEDIATE
Number of immediate chained sequential I/O requests. This I/O request is made when the requested RBA is higher than the highest RBA in the current sequential buffer and the requested RBA is not found in both of the sequential buffers and the direct buffers.

SEQU OVERLAPPED
Number of overlapped chained sequential I/O requests. This I/O request is made when the requested RBA is found in the sequential buffers and the next block or CI is not yet read in the next sequential buffer.

DIRECT
Number of immediate direct I/O requests. This I/O request is made when:

- The requested RBA is less than the highest RBA in the current sequential buffer, and is not found in both of the sequential buffers and the direct buffers.
- Chained sequential I/O request is made and the sequential buffers cannot be filled with blocks or CIs because the number of blocks or CIs that remain in the data set is less than the RANSIZE value.

NBR OF BLOCKS READ
Summary of number of blocks or CIs read.

TOTAL
Total number of blocks or CIs read.

SEQUENTIAL TOTAL
Total number of blocks or CIs with chained sequential I/O.

SEQU IMMEDIATE
Number of blocks or CIs with immediate chained sequential I/O. This value is calculated by: (Number OF I/O requests for SEQU IMMEDIATE) multiplied by (RANSIZE)

SEQU OVERLAPPED
Number of blocks or CIs with overlapped chained sequential I/O. This value is calculated by: (Number of I/O requests for SEQU OVERLAPPED) multiplied by (RANSIZE)

DIRECT
Number of blocks or CIs with immediate direct I/O.

NBR OF UNREFERENCED SEQ BUFFERS
Number of blocks or CIs that were deleted from sequential buffers although they were never referred to. With this number, whether the CAB decisions to perform chained sequential I/O is reasonable or not can be measured.

TIMING STATISTICS
Wait time for immediate chained sequential I/O, overlapped chained sequential I/O, and immediate direct I/O.

MEAN WAIT TIME
Average wait time for I/O request, reported by type of I/O.

TOTAL WAIT TIME
Total wait time for I/O request, reported by type of I/O.

The rest of the statistics is obtained when the CABSTAT YES control statement is specified in the HSSROPT data set. These statistics contain detailed technical information beyond what is usually required for users of IMS High Performance Unload. The statistics are:

Event Statistics
Detailed data on CAB I/O.

Distribution of look-aside at *n* th Position
Detailed data on look-aside buffering.

Distribution of HRAN Steal/Delete per Reference Count Value
Detailed data on CAB I/O in ranges.

Distribution of Reference-Count-Difference
Detailed data on reference threshold parameters.

Distribution of Distance from Current Seq HRAN
Detailed statistics on HRAN distribution.

HSSRTRAC data set

This output data set contains trace reports that are generated by HSSR Engine.

The HSSROPT control statements, TRHC, TRDB, and DIAGG, cause trace information to be produced in the Trace Output report.

Note: The trace data of control blocks of HSSR Engine, which is produced by specifying CB or BUFCB option for TRHC control statement, is not intended to be reviewed by users, but might be needed by IBM Software Support to analyze a problem.

Format

This data set contains 133-byte fixed-length records. When the block size is coded in the JCL, the block size must be a multiple of 133.

Trace Output report

This hardcopy trace shows trace call data, control blocks data, buffer handler data, and CAB data. The content of this report depends on the trace options that you specified for the job.

Subtopics:

Example report: Trace Output report when 1 SSA is specified

The following figure shows an example of the Trace Output report when 1 SSA is specified.

```
IMS HIGH PERFORMANCE UNLOAD                                    "TRACE OUTPUT"                                    PAGE:     1
5655-E06                                                 DATE: 06/01/2021  TIME: 12.34.16                      FABH310  - V1.R2


*** DB CALL ***  FUNC=GN   DBD=DBHD0070 LEV=01 STAT=   PROC=GX   SEGN=SG001LV1  PCB#=0001

  KEYFEEDB 000E5C84   F0F1F0F0 F0F0F0F0 F0F1                                                  *0100000001                    *

  IO-AREA  000E9968   F0F1F0F0 F0F0F0F0 F0F14040 40404040   40404040 40404040 40404040 40404040   *0100000001                    *
           000E9988   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                              *
           000E99A8   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                              *
           000E99C8   40404040                                                                   *                              *

  SSA-1    000E9D59   E2C7F0F0 F1D3E5F1 40                                                       *SG001LV1                      *

  SEG-PREF 0734C818   01000000 0B1E0000 088E0000 08F80001   9804                               *            8                 *
```

Figure 44. Trace Output report when 1 SSA is specified

Example report: Trace Output report when 15 SSAs and APISET 3 are specified

The following figure shows an example of the Trace Output report when 15 SSAs and APISET 3 are specified.

```
IMS HIGH PERFORMANCE UNLOAD                                    "TRACE OUTPUT"                                    PAGE:     1
5655-E06                                                 DATE: 06/01/2021  TIME: 12.33.27                      FABH310  - V1.R2


*** DB CALL ***  FUNC=GN   DBD=DBHD0070 LEV=15 STAT=   PROC=GX   SEGN=SG016LVF  PCB#=0001

  KEYFEEDB 000E5C84   F0F1F0F0 F0F0F0F0 F0F1F0F1 F0F2F0F0   F0F0F0F1 F0F1F0F3 F0F0F0F0 F0F1F0F1   *0100000001010200000101030000010 1*
           000E5CA4   F0F4F0F0 F0F0F0F1 F0F1F0F5 F0F0F0F0   F0F1F0F1 F0F6F0F0 F0F0F0F1 F0F1F0F7   *0400000101050000010106000001010 7*
           000E5CC4   F0F0F0F0 F0F1F0F1 F0F8F0F0 F0F0F0F1   F0F1F0F9 F0F0F0F0 F0F1F0F1 F0F0F0F0   *0000010108000001010900000101000 0*
           000E5CE4   F0F0F0F1 F0F1F0F1 F0F0F0F0 F0F1F0F1   F0F2F0F0 F0F0F0F1 F0F1F0F3 F0F0F0F0   *0001010100000101020000010103000 0*
           000E5D04   F0F1F0F1 F0F4F0F0 F0F0F0F1 F0F1F0F5   F0F0F0F0 F0F1                       *0101040000010105000001          *

  IO-AREA  000E9968   F0F1F0F5 F0F0F0F0 F0F14040 40404040   40404040 40404040 40404040 40404040   *0105000001                    *
           000E9988   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                              *
           000E99A8   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                              *
           000E99C8   40404040                                                                   *                              *

  SSA-1    000E9D59   E2C7F0F0 F1D3E5F1 40                                                       *SG001LV1                      *

  SSA-2    000E9D62   E2C7F0F0 F3D3E5F2 40                                                       *SG003LV2                      *

  SSA-3    000E9D6B   E2C7F0F0 F4D3E5F3 40                                                       *SG004LV3                      *

  SSA-4    000E9D74   E2C7F0F0 F5D3E5F4 40                                                       *SG005LV4                      *

  SSA-5    000E9D7D   E2C7F0F0 F6D3E5F5 40                                                       *SG006LV5                      *

  SSA-6    000E9D86   E2C7F0F0 F7D3E5F6 40                                                       *SG007LV6                      *

  SSA-7    000E9D8F   E2C7F0F0 F8D3E5F7 40                                                       *SG008LV7                      *

  SSA-8    000E9D98   E2C7F0F0 F9D3E5F8 40                                                       *SG009LV8                      *

  SSA-9    000E9DA1   E2C7F0F1 F0D3E5F9 40                                                       *SG010LV9                      *

  SSA-10   000E9DAA   E2C7F0F1 F1D3E5C1 40                                                       *SG011LVA                      *

  SSA-11   000E9DB3   E2C7F0F1 F2D3E5C2 40                                                       *SG012LVB                      *

  SSA-12   000E9DBC   E2C7F0F1 F3D3E5C3 40                                                       *SG013LVC                      *

  SSA-13   000E9DC5   E2C7F0F1 F4D3E5C4 40                                                       *SG014LVD                      *

  SSA-14   000E9DCE   E2C7F0F1 F5D3E5C5 40                                                       *SG015LVE                      *

  SSA-15   000E9DD7   E2C7F0F1 F6D3E5C6 40                                                       *SG016LVF                      *

  SEG-PREF 071D0374   10000000 0000                                                           *                              *
```

Figure 45. Trace Output report when 15 SSAs and APISET 3 are specified

Report field descriptions

If the CALL option is specified for the TRHC control statement, the report contains a section that describes the following fields and data areas for database calls:

FUNC
The type of call requested for HSSR Engine.

DBD
The *dbdname* of the database referred to by the HSSR PCB.

LEV
Segment level.

STAT
Status code returned in the PCB.

PROC
Processing option coded in the PROCOPT field of the DBD.

SEGN
The name of the segment that is accessed.

PCB
The PCB number. The number for the first PCB is 0001.

PARTITION
Partition name (only for HALDB).

KEYFEEDB
Data returned in the key feedback area. This field provides the virtual storage addresses of the data traced on the right. There might be multiple lines, each showing up to 32 bytes of data in hexadecimal and EBCDIC format.

IO-AREA
Data returned in the I/O area. This field provides the virtual storage addresses of the data traced on the right. There might be multiple lines, each showing 32 bytes of data in hexadecimal and EBCDIC format.

SSA or SSA-n
Shows the Segment Search Argument. This field provides the virtual storage address of the SSA traced on the right. There might be multiple lines, each showing 32 bytes of the SSA in hexadecimal and EBCDIC format.

If APISET 1 or 2 is specified, the trace log shows the field name 'SSA'. If APISET 3 is specified, the trace log shows up to of 15 SSA fields, showing their names as 'SSA-1', 'SSA-2', 'SSA-3', and so on.

SEG-PREF
Shows the prefix of the returned segment. Gives the virtual storage address of the segment prefix traced on the right. There might be multiple lines, each showing 32 bytes of the prefix in hexadecimal and EBCDIC format.

If NPF option is specified for the TRHC control statement, the SEG-PREF information is not printed.

Notes:

- If a Data Conversion exit routine (DFSDBUX1) is used for the segment accessed, the data shown in the KEYFEEDB, IO-AREA, and SSA fields is presented in the application form, not in the stored form
- If a user application that is run by APISET 1 or 2 issues a DL/I call that is not supported by the HSSR call handler, the call data is printed in this report even if the TRHC and the TRDB control statements are not specified. Ignore LEV, STAT, SEGN and IO-AREA fields in the data.

If the BUF option is specified for the TRHC control statement, the report contains a section of information that describes the following control blocks and areas showing buffer handler trace data:

DBDNAME
Name of the DBD referred to by the HSSR PCB.

DSG
Data set group number.

PARTITION
Partition name (only for HALDB).

CALL-TYPE
Type of buffer handler call.

RC
Return code returned by the buffer handler.

HDMB
Snap dump of "Communication area" of HDMB control block of HSSR Engine.

DATA
Data returned by buffer handler. This field provides the virtual storage addresses of the data traced on the right. There might be multiple lines, each showing up to 32 bytes of data in hexadecimal and EBCDIC format. This data can be:

- A KSDS record
- An OSAM block
- An ESDS control interval

If the BUFCB option is specified for the TRHC control statement, and CAB buffering is used, the trace of the following control blocks of HSSR buffer handler is also printed:

HDMB
HDMC
SEQ HRAN
DIR HRAN

If CB option is specified for the TRHC control statement, the traces of the following control blocks immediately before completion of an HSSR call are printed:

HJCB
HDMB
HSDB'S
HPTR'S

Trace Output report with diagnostics information

The following diagnostic information appears on the report whenever you use the DIAGG option in conjunction with the SKERROR option when unloading a corrupted database and a GG or a GX status code is returned.

The diagnostic information that is printed on the Trace Output report varies with the database organization and the type of error detected by HSSR Engine.

Subtopics:

Example report: Trace Output report with diagnostics information (CASE 1-A, CASE 1-B, and CASE 2)

The following figures show an example of this report with diagnostics information when the error cases are CASE 1-A, CASE 1-B, and CASE 2.

```
IMS HIGH PERFORMANCE UNLOAD                                    "TRACE OUTPUT"                                    PAGE:     1
5655-E06                                              DATE: 06/01/2021  TIME: 18.57.16                      FABH310  - V1.R2

********************* BEGINNING OF GG DIAGNOSIS *******************************
********************* BEGINNING OF GG DIAGNOSIS *******************************
********************* BEGINNING OF GG DIAGNOSIS *******************************
GG STATUS CODE FOR PCB HDAM0010
THE NAME OF THE PREVIOUSLY RETRIEVED SEGMENT IS:ORDER
    THE SNAP OF ITS CONCATENATED KEY FOLLOWS:
  KEY      0006FB64   F0F0F0F0 F0F0F0F9 F0F0                                          *0000000900                    *
------------------ TYPE OF GG ERROR ----------------------------------------
CASE 1-A: CHILD-POINTER POINTS OUTSIDE OF DATASET
         THE BAD POINTER IS AT THE DECIMAL OFFSET 0006 WITHIN THE PREFIX OF
         THE SEGMENT=ORDER    AND SHOULD POINT TO THE SEGMENT=ORDART
RETRIEVAL OF FOLLOWING SEGMENTS MAY BE SKIPPED:
          ONE ORDART   ,ANY TWIN ON TWIN- OR HIERARCHICAL- CHAIN,
          AND ALL OF THEIR DEPENDENTS
------------------ "HD-FROM INFO" FOLLOWS -----------------------------------
RBA OF SEGMENT-PREFIX CONTAINING THE BAD POINTER AND LOW+HIGH RBA OF ITS CI:
  RBA'S    0000BEDC   00001C10 00001C00 00001FFF                                  *                              *
THE SEGMENT-PREFIX CONTAINING THE BAD POINTER
  PREFIX   071DF810   0100011C 00470FA3 0000                                      *                              *
------------------ "HD-TO INFO" FOLLOWS -------------------------------------
RBA OF POINTER-TARGET FOLLOWS:
  RBA'S    0000BEDC   0FA30000                                                    *                              *
THE BAD POINTER POINTS OUTSIDE OF THE DATASET
********************** END OF GG DIAGNOSIS ********************************
********************** END OF GG DIAGNOSIS ********************************
********************** END OF GG DIAGNOSIS ********************************
*** DB CALL ***  FUNC=GN   DBD=HDAM0010 LEV=01 STAT=GG PROC=GX   SEGN=ORDER
  KEYFEEDB 0006FB64   F0F0F0F0 F0F0F0F9 F0F0                                      *0000000900                    *
  IO-AREA
```

Figure 46. Trace Output report with diagnostics information (CASE 1-A, CASE 1-B, and CASE 2) (Part 1 of 3)

```
********************* BEGINNING OF GG DIAGNOSIS ******************************
********************* BEGINNING OF GG DIAGNOSIS ******************************
********************* BEGINNING OF GG DIAGNOSIS ******************************
GG STATUS CODE FOR PCB HDAM0010
PREVIOUS CALL WAS NOT SUCCESSFUL (OR NO PREVIOUS CALL)
------------------ TYPE OF GG ERROR ----------------------------------------
CASE 1-B: TWIN-POINTER POINTS OUTSIDE OF DATASET
         THE BAD POINTER IS AT THE DECIMAL OFFSET 0002 WITHIN THE PREFIX OF
         THE SEGMENT=ORDER    AND SHOULD POINT TO THE SEGMENT=ORDER
RETRIEVAL OF FOLLOWING SEGMENTS MAY BE SKIPPED:
          ONE ROOT, ANY OTHER ROOT ON SAME RAP CHAIN, AND ALL OF THEIR DEPENDENTS
------------------ "HD-FROM INFO" FOLLOWS -----------------------------------
RBA OF SEGMENT-PREFIX CONTAINING THE BAD POINTER AND LOW+HIGH RBA OF ITS CI:
  RBA'S    0000BEDC   00001C10 00001C00 00001FFF                                  *                              *
THE SEGMENT-PREFIX CONTAINING THE BAD POINTER
  PREFIX   071DF810   0100011C 00470FA3 0000                                      *                              *
------------------ "HD-TO INFO" FOLLOWS -------------------------------------
RBA OF POINTER-TARGET FOLLOWS:
  RBA'S    0000BEDC   011C0047                                                    *                              *
THE BAD POINTER POINTS OUTSIDE OF THE DATASET
********************** END OF GG DIAGNOSIS **********************************
********************** END OF GG DIAGNOSIS **********************************
********************** END OF GG DIAGNOSIS **********************************
*** DB CALL ***  FUNC=GN   DBD=HDAM0010 LEV=01 STAT=GG PROC=GX   SEGN=ORDER
  KEYFEEDB 0006FB64   F0F0F0F0 F0F0F0F9 F0F0                                      *0000000900                    *
  IO-AREA
*** DB CALL ***  FUNC=GN   DBD=HDAM0010 LEV=01 STAT=   PROC=GX   SEGN=ORDER
  KEYFEEDB 0006FB64   F0F0F0F0 F0F0F1F6 F0F0                                      *0000001600                    *
  IO-AREA  00105EFF   F0F0F0F0 F0F0F1F6 F0F0D6D9 C4C5D940   C4C5E2C3 D9C9D7E3 C9D6D540 D7D7D7D7   *0000001600ORDER DESCRIPTION PPPP*
           00105F1F   D7D7D7D7 D7D7D740 F9F6F1F2 F3F14040   40404040 40404040 40404040 40404040   *PPPPPPP 961231                *
           00105F3F   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                              *
           00105F5F   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                              *
           00105F7F   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                              *
           00105F9F   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                              *
           00105FBF   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                              *
           00105FDF   40404040 40404040 40404040 40404040   40404040 40404040 4040               *                              *
  SEG-PREF 07267978   01000000 00000000 2E7C                                      *                              *
```

Figure 47. Trace Output report with diagnostics information (CASE 1-A, CASE 1-B, and CASE 2) (Part 2 of 3)

```
********************* BEGINNING OF GG DIAGNOSIS *******************************
********************* BEGINNING OF GG DIAGNOSIS *******************************
********************* BEGINNING OF GG DIAGNOSIS *******************************
GG STATUS CODE FOR PCB HDAM0010
THE NAME OF THE PREVIOUSLY RETRIEVED SEGMENT IS: ORDART
    THE SNAP OF ITS CONCATENATED KEY FOLLOWS:
  KEY      0006FB64   F0F0F0F0 F0F0F1F8 F0F0E3F0 F1F8F0F2   40F0                                *0000001800T01802 0             *
------------------ TYPE OF GG ERROR ----------------------------------------
CASE 2: HDAM ROOT-ANCHOR-POINT DOES NOT POINT TO ROOT
RETRIEVAL OF FOLLOWING SEGMENTS MAY BE SKIPPED:
ONE ROOT, ANY OTHER ROOT ON SAME RAP CHAIN, AND ALL OF THEIR DEPENDENTS
------------------ "HD-FROM INFO" FOLLOWS -----------------------------------
RBA OF RAP CONTAINING THE BAD POINTER AND LOW+HIGH RBA OF ITS CI:
  RBA'S    0000BEDC   00002008 00002000 000023FF                                *                               *
THE BAD RAP
  RAP      071E0008   00002010                                                  *                               *
------------------ "HD-TO INFO" FOLLOWS -------------------------------------
RBA OF POINTER-TARGET AND LOW+HIGH RBA OF ITS CI:
  RBA'S    0000BEDC   00002010 00002000 000023FF                                *                               *
THE TARGET OF THE BAD POINTER FOLLOWS
  TARGET   071E0010   0200                                                      *                               *
THE "TO" BUFFER IS THE SAME AS THE "FROM" BUFFER
********************** END OF GG DIAGNOSIS ***********************************
********************** END OF GG DIAGNOSIS ***********************************
********************** END OF GG DIAGNOSIS ***********************************
*** DB CALL ***  FUNC=GN   DBD=HDAM0010 LEV=02 STAT=GG PROC=GX   SEGN=ORDART
  KEYFEEDB 0006FB64   F0F0F0F0 F0F0F1F8 F0F0E3F0 F1F8F0F2   40F0                *0000001800T01802 0             *
  IO-AREA
*** DB CALL ***  FUNC=GN   DBD=HDAM0010 LEV=01 STAT=   PROC=GX   SEGN=ORDER
  KEYFEEDB 0006FB64   F0F0F0F0 F0F0F0F8 F0F0                                    *0000000800                     *
  IO-AREA  001063B9   F0F0F0F0 F0F0F0F8 F0F0D6D9 C4C5D940   C4C5E2C3 D9C9D7E3 C9D6D540 C8C8C8C8   *0000000800ORDER DESCRIPTION HHHH*
           001063D9   C8C8C8C8 C8C8C840 F9F6F1F2 F3F14040   40404040 40404040 40404040 40404040   *HHHHHHHH 961231                *
           001063F9   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                               *
           00106419   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                               *
           00106439   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                               *
           00106459   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                               *
           00106479   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                               *
           00106499   40404040 40404040 40404040 40404040   40404040 40404040 4040               *                               *
  SEG-PREF 071DF184   01000000 00000000 1688                                    *                               *
```

Figure 48. Trace Output report with diagnostics information (CASE 1-A, CASE 1-B, and CASE 2) (Part 3 of 3)

Example report: Trace Output report with diagnostics information (CASE 19-A)

The following figure shows an example of this report with diagnostics information when the error case is CASE 19-A.

```
IMS HIGH PERFORMANCE UNLOAD                                    "TRACE OUTPUT"                                    PAGE:     1
5655-E06                                                DATE: 06/01/2021  TIME: 12.00.09                       FABH310  - V1.R2

********************* BEGINNING OF GG DIAGNOSIS *******************************
********************* BEGINNING OF GG DIAGNOSIS *******************************
********************* BEGINNING OF GG DIAGNOSIS *******************************
GG STATUS CODE FOR PCB DBHD0010
THE NAME OF THE PREVIOUSLY RETRIEVED SEGMENT IS:SG001LV1
    THE SNAP OF ITS CONCATENATED KEY FOLLOWS:
  KEY      001131E4   F0F2F0F0 F0F0F0F0 F0F1                                                *0200000001                    *
------------------ TYPE OF GG ERROR ----------------------------------------
CASE 19-A: THE DATABASE POSITION IS UNAVAILABLE
        INFORMATION OF INTERNAL RBA CALL
          STATUS CODE: GG
          THE SNAP OF SSA FOLLOWS:
  SSA-1    0000A238   E2C7F0F0 F1D3E5F1 5CE34D00 0138185D                                *SG001LV1*T(    )              *
************************ END OF GG DIAGNOSIS **********************************
************************ END OF GG DIAGNOSIS **********************************
************************ END OF GG DIAGNOSIS **********************************
*** DB CALL ***  FUNC=GN   DBD=DBHD0010 LEV=00 STAT=GG PROC=GXOT SEGN=
  KEYFEEDB
  IO-AREA
  SSA-1    07164AEE   E2C7F0F0 F1D3E5F1 40                                                *SG001LV1                      *
  SSA-2    07164AF7   E2C7F0F0 F4D3E5F2 40                                                *SG004LV2                      *
```

Figure 49. Trace Output report with diagnostics information (CASE 19-A)

Example report: Trace Output report with diagnostics information (CASE 19-B)

The following figure shows an example of this report with diagnostics information when the error case is CASE 19-B.

```
IMS HIGH PERFORMANCE UNLOAD                                    "TRACE OUTPUT"                                    PAGE:     1
5655-E06                                                DATE: 06/01/2021  TIME: 12.07.42                       FABH310  - V1.R2

********************* BEGINNING OF GG DIAGNOSIS *******************************
********************* BEGINNING OF GG DIAGNOSIS *******************************
********************* BEGINNING OF GG DIAGNOSIS *******************************
GG STATUS CODE FOR PCB DBHI0010
THE NAME OF THE PREVIOUSLY RETRIEVED SEGMENT IS:SG001LV1
    THE SNAP OF ITS CONCATENATED KEY FOLLOWS:
  KEY      0008B244   F1F0F0F0 F0F0F0F0 F0F1                                                *1000000001                    *
------------------ TYPE OF GG ERROR ----------------------------------------
CASE 19-B: THE DATABASE POSITION IS UNAVAILABLE
        INFORMATION OF DL/I BUFFER HANDLER REQUEST
          PSTRTCDE: 18; PSTFNCTN: F2; DDNAME: DDHX0010
          THE SNAP OF REQUESTED KEY FOLLOWS:
  KEY      00022569   F1F0F0F0 F0F0F0F0 F0F1                                                *1000000001                    *
************************ END OF GG DIAGNOSIS **********************************
************************ END OF GG DIAGNOSIS **********************************
************************ END OF GG DIAGNOSIS **********************************
*** DB CALL ***  FUNC=GN   DBD=DBHI0010 LEV=00 STAT=GG PROC=GXOT SEGN=
  KEYFEEDB
  IO-AREA
  SSA-1    07125CAE   E2C7F0F0 F1D3E5F1 40                                                *SG001LV1                      *
  SSA-2    07125CB7   E2C7F0F0 F4D3E5F2 40                                                *SG004LV2                      *
```

Figure 50. Trace Output report with diagnostics information (CASE 19-B)

Example report: Trace Output report with diagnostics information (CASE 20)

The following figure shows an example of this report with diagnostics information when the error case is CASE 20.

```
IMS HIGH PERFORMANCE UNLOAD                                    "TRACE OUTPUT"                                    PAGE:     1
5655-E06                                                DATE: 06/01/2021  TIME: 12.11.14                      FABH310  - V1.R2

********************* BEGINNING OF GG DIAGNOSIS ******************************
********************* BEGINNING OF GG DIAGNOSIS ******************************
********************* BEGINNING OF GG DIAGNOSIS ******************************
GG STATUS CODE FOR PCB DBHD0160
THE NAME OF THE PREVIOUSLY RETRIEVED SEGMENT IS:SG005LV3
    THE SNAP OF ITS CONCATENATED KEY FOLLOWS:
  KEY      000721E4   F0F4F0F0 F0F0F0F0 F0F1F0F4 F0F3F0F0   F0F0F0F1 F0F4F0F4 F0F0F0F0 F0F1       *040000000104030000010404000001  *
------------------ TYPE OF GG ERROR -------------------------------------------
CASE 20: STATUS CODE=GG RETURNED ON THE INTERNAL DL/I CALL
************************ END OF GG DIAGNOSIS *********************************
************************ END OF GG DIAGNOSIS *********************************
************************ END OF GG DIAGNOSIS *********************************
*** DB CALL ***  FUNC=GN   DBD=DBHD0160 LEV=00 STAT=GG PROC=GXOT SEGN=
  KEYFEEDB
  IO-AREA
  SSA-1    071A7848   E2C7F0F0 F1D3E5F1 40                                                *SG001LV1                       *
  SSA-2    071A7851   E2C7F0F0 F4D3E5F2 40                                                *SG004LV2                       *
  SSA-3    071A785A   E2C7F0F0 F5D3E5F3 40                                                *SG005LV3                       *
```

Figure 51. Trace Output report with diagnostics information (CASE 20)

Report field descriptions

The following information is provided in the report:

********** BEGINNING OF GG DIAGNOSIS **********
: Eye-catcher identifying the beginning of the DIAGG diagnosis. The line is printed three times.

GG STATUS CODE FOR PCB *psbname*
: *psbname* of PSB containing PCB used; if run in the ULU region, this is the *dbdname* as specified on the EXEC statement.

THE NAME OF THE PREVIOUSLY RETRIEVED SEGMENT IS: *segname*
: Segment name returned by the previous successful HSSR call.

THE SNAP OF ITS CONCATENATED KEY FOLLOWS:
: Concatenated key of the segment. There might be one or more lines each showing up to 32 bytes of data in hexadecimal and EBCDIC format.

The report contains a section that starts with the following line:

---------- TYPE OF GG ERROR ----------
: The section contains information describing the type of GG error that was detected. The next line, "CASE *nn*:", identifies a type of GG error.

Pointer or Key Sequence Errors
: For a pointer error or a key sequence error, the report tells which kind of pointer or key is incorrect and which type of segments may be skipped by the SKERROR option.

  The following is a description of each error case:

  CASE 1-A: Incorrect segment prefix pointer other than root twin pointer.
  : HSSR Engine handles the incorrect pointer as if it contained zero, and continues processing with the next appropriate pointer. In this case, HSSR Engine skips the retrieval of one or more dependent segments of the current database record.

    If the incorrect pointer is a physical child first pointer, the following segments are skipped:

- Child segment
- All twins on the physical twin chain or on the hierarchical chain of the above child
- All dependents of the above segments

If the incorrect pointer is a physical twin forward pointer, the following segments are skipped:

- Twin segment
- All further twins on the incorrect physical twin forward chain
- All dependents of the above segments

If the incorrect pointer is a hierarchical forward pointer, the following segments are skipped:

- All further segments on the incorrect hierarchical twin forward chain
- All dependents of the above segments

CASE 1-B: Incorrect HDAM root twin pointer.
HSSR Engine handles the incorrect pointer as if it contained zero, and continues processing by skipping to the next HDAM root anchor point (RAP).

In this case, HSSR Engine skips the retrieval of the following database segments:

- All further roots on the incorrect twin chain (incorrect RAP synonym chain)
- All dependents of the above roots

CASE 1-C: Incorrect HIDAM root twin pointer.
HSSR Engine attempts to access the HIDAM root segment via the HIDAM index.

If the pointer in the index record is correct, HSSR Engine does not skip the retrieval of any database segments. If the pointer in the index record is also incorrect, HSSR Engine skips to the next record, with the result that one root and all of its dependents are not retrieved.

CASE 2: Incorrect HDAM root anchor pointer.
HSSR Engine handles the incorrect RAP as if it contained zero, and continues processing by skipping to the next RAP.

In this case, HSSR Engine skips the retrieval of:

- All roots on the incorrect RAP synonym chain
- All dependents of the above roots

CASE 3: Incorrect HIDAM root index pointer.
HSSR Engine continues processing by accessing the next Index record.

In this case, HSSR Engine skips the retrieval of one root segment and all its dependents.

CASE 4: Incorrect pointer to split data portion of a segment.
HSSR Engine skips the retrieval of the following segments of the current database record:

- Segment with the incorrect data pointer
- All further segments on the remainder of a hierarchical forward pointer chain, if the segment is on such a chain
- All dependents of the above segments

CASE 5: KEYCHECK option detected sequence error.
HSSR Engine uses the same logic as in case 4.

This case includes the sequence error in the sequence key for a virtual logical child when a migration unload is being done. In that case, HSSR Engine skips the retrieval of the following occurrences of the logical child:

- The occurrence at which the sequence error is detected
- All further occurrences of the logical child on the remainder of a logical child forward pointer chain

CASE 9: Incorrect value for segment length field.
HSSR Engine skips the retrieval of the following segments of the current database record:

- Segment with the incorrect segment length field
- All dependents of the above segments

CASE 10: HISAM segment code is not 01 at beginning of root record.
If error is due to an incorrect pointer used to chain ESDS records of a secondary index database, HSSR Engine resumes its retrieval with the logical next KSDS record; hence the retrieval of one or more database records is skipped.

For other errors, HSSR Engine resumes its retrieval with the next root segment, skipping the retrieval of the remaining segments of the current database record.

CASE 11: HISAM record has incorrect segment code or delete byte.
HSSR Engine uses the same logic as in case 10.

CASE 12: HISAM pointer to overflow record points outside of data set.
HSSR Engine uses the same logic as in case 10.

CASE 13: Key sequence error detected for a segment in an HISAM or secondary index database.
HSSR Engine uses the same logic as in case 10.

For HDAM and HIDAM databases, the report contains a section of information describing the incorrect pointer:

----------"HD-FROM INFO" FOLLOWS ----------
An eye-catcher identifying the beginning of the "HD-FROM" information:

RBA OF SEGMENT PREFIX CONTAINING THE BAD POINTER AND LOW+HIGH RBA OF ITS CI:
Snap of one area containing three RBAs:

- RBA of segment prefix (or RBA of the RAP) that contains the incorrect pointer.
- Beginning RBA and ending RBA of the block or CI that contains the incorrect pointer.

THE SEGMENT PREFIX CONTAINING THE BAD POINTER
Snap of segment prefix that contains the incorrect pointer. This field provides the virtual storage addresses of the data traced on the right. There might be multiple lines, each showing up to 32 bytes of data in hexadecimal and EBCDIC format.

THE SEGMENT BUFFER
Snap of the block, CI, or index record that contains the incorrect pointer. This field provides the virtual storage addresses of the data traced on the right. There might be multiple lines, each showing up to 32 bytes of data in hexadecimal and EBCDIC format.

For HDAM and HIDAM databases, the report contains a section of information describing the portion of the database "pointed to" by the incorrect pointer.

---------- "HD-TO INFO" FOLLOWS ----------
An eye-catcher identifying the beginning of the "HD-TO" information:

RBA OF POINTER TARGET AND LOW+HIGH RBA OF ITS CI:
Snap of one area containing three RBAs:

- RBA contained in the incorrect pointer (RBA "pointed to" by the incorrect pointer).
- Beginning RBA and ending RBA of the block or CI that is "pointed to."

THE TARGET OF THE BAD POINTER FOLLOWS
Snap of the first 2 bytes "pointed to" by the incorrect pointer. If the incorrect pointer does not point outside of the data set, this area is snapped.

THE TARGET BUFFER
Snap of the block or CI "pointed to" by the incorrect pointer. This field provides the virtual storage addresses of the data traced on the right. There might be multiple lines, each showing up to 32 bytes of data in hexadecimal and EBCDIC format.

This snap is only provided if the "TO-" block or CI snapped in the "HD-TO" is not the same as the "FROM-" block or CI snapped in the "HD-FROM INFO". Otherwise, the message `THE "TO" BUFFER IS THE SAME AS THE "FROM" BUFFER` appears in this area.

For HISAM and secondary index databases, the report contains data on the last KSDS record accessed by HSSR Engine.

---------- "HS ISAM/KSDS RECORD" FOLLOWS ----------
An eye-catcher identifying the beginning of the "HS ISAM/KSDS RECORD". A snap of the last KSDS record accessed by HSSR Engine. This field provides the virtual storage addresses of the data traced on the right. There might be multiple lines, each showing up to 32 bytes of data in hexadecimal and EBCDIC format.

For HISAM and secondary index databases, the report contains information about the current ESDS/KSDS record.

---------- "HS CURRENT RECORD" FOLLOWS ----------
An eye-catcher identifying the beginning of the "HS CURRENT RECORD".

VSAM RBA OF BAD RECORD AND LOW+HIGH RBA.S OF ITS CI FOLLOWS ON SNAPS:
Snap of one area containing three ESDS RBAs or three OSAM RRNs.

- RBA of the ESDS record
- Beginning RBA and ending RBA of the CI that contains the record.

SNAP OF BAD SEGMENT CODE/DELETE BYTE AND OF BAD RECORD FOLLOWS:
Snap of the first 2 bytes of the segment prefix and of the current ESDS/KSDS record. This field provides the virtual storage addresses of the data traced on the right. There might be multiple lines, each showing up to 32 bytes of data in hexadecimal and EBCDIC format.

IMS call handler or IMS buffer handler errors
If APISET 3 is specified, and a call to IMS causes an error, one of the following reports is issued. CASE 15 to CASE 18 are deleted.

CASE 19-A: The current database position is not valid.
This might happen if there is something that updates the database concurrently. The error causes the current position to be reset to the beginning. If status code GG or GE is returned in response to the INTERNAL RBA call, the following:

- status code
- the SNAP DUMP of SSA specified at the time of RBA call

A sample of this report is provided in Figure 49 on page 197.

CASE 19-B: The current database position is not valid.
This might happen if there is a something that updates the database concurrently. If an error occurs when a DLI buffer handler is called for a DBDS, the following information is written in the report:

- DDNAME of DSG
- the return code from the buffer handler (PSTRTCDE)
- the function code (PSTFNCTN) sent to the buffer handler
- the key value (for key request) sent to the buffer handler

A sample of this report is provided in Figure 50 on page 197.

CASE 20: Status 'GG' is returned when a call is made from an application to IMS.
An error occurred in IMS after a call was made to IMS. The error resets the position of the database to the beginning. HSSR Engine does not generate any error report, because it cannot identify the reason for the error. The PCB information at the call time is reported in the CALL TRACE report that is generated after the DIAGG report. A sample of this report is provided in Figure 51 on page 198.

The following line indicates the end of a DIAGG output:

***************** END OF GG DIAGNOSIS *****************

The line is printed three times.

After printing a DIAGG output information, the DIAGG option forces a temporary activation of the hardcopy trace option, TRHC. The TRHC option remains in effect until the next database call is completed. The TRHC option enables you to determine whether the next call has been completed successfully, and to see the next retrieved segment. If the TRHC option is active, the Trace Output report contains the following information:

- Trace of the HSSR call that ended with a GG or a GX status code.
- Trace of the first successful HSSR call after a GG or a GX status code was returned. This contains the name of the returned segment, a snap of the PCB key feedback area, and a snap of the returned segment.
- Snaps of the buffer handler calls, if the keyword BUF is specified in the DIAGG statement.
- Snaps of the control blocks of HSSR Engine, if the keyword CB is specified in the DIAGG statement.

HSSRSNAP data set

This data set provides a snapshot of control blocks of HSSR Engine whenever HSSR Engine detects abnormalities while the control blocks are being initialized.

An error message is also issued. The initialization ends, and HSSR Engine *falls back* to DL/I. All HSSR PCBs are considered DL/I PCBs, and the application program runs using DL/I action modules instead of HSSR Engine.

If the HSSRSNAP DD statement is omitted, HSSR Engine abends instead of falling back to DL/I. If you want HSSR Engine to fall back to DL/I without a snap, code this data set as DUMMY.

Format

This data set contains 133-byte fixed-length records. When the block size is coded in the JCL, the block size must be a multiple of 133.

HSSR SNAPs

When a snap is taken, the control blocks of HSSR Engine are provided. The snap gives the virtual storage addresses of the data traced on the right. There might be multiple lines, each showing up to 32 bytes of data in hexadecimal and EBCDIC format.

HSSRLOUT data set

This output data set contains, for each database record, a record that can be used for tuning databases.

Each record consists of:

- Length of the database record (fullword)
- The number of database I/Os required to read all database segments of the database record (fullword)
- Key of the database root segment

You can use this report to tune your databases. For a description about this data set, see “HSSRLOUT output data set for Database Tuning Statistics” on page 311.

HSSRBUTR data set

This data set provides buffer handler trace information that is generated by an unload utility or application program, and that is used as the input for the HSSR Buffer Handler Simulation utility, FABHBSIM.

Format

This data set contains variable-length records that are read by FABHBSIM.

Part 3. Tuning and customizing HSSR application jobs

You can diagnose, tune, and customize your IMS High Performance Unload jobs to meet your requirements.

Topics:

Chapter 13. Overview of the buffer handlers

HSSR Engine provides two buffer handlers, called *HSSR buffer handlers*, that provide buffering services for ESDS, OSAM database, and OSAM LDS: Chained Anticipatory Buffer handler (CAB) and Basic Buffer handler (BB).

CAB provides better buffer handling than BB for well-organized HIDAM, HDAM, PHIDAM, and PHDAM databases, and well-organized HISAM overflow area. CAB uses more buffer space than BB; however, CAB uses virtual and real storage for a shorter elapsed time. The performance of both buffer handlers depends on how well the databases are organized.

Either CAB or BB can replace the DL/I buffer handler. A major advantage of the two HSSR buffer handlers over the DL/I buffer handler is that HSSR buffer handlers reduce the path length of database calls. Both HSSR buffer handlers provide one buffer pool for each PCB. Within a PCB, a buffer pool is provided for each data set group. By contrast, the DL/I buffer handler provides common buffer pools, which are shared by multiple PCBs and multiple data sets.

Note: For HALDBs, a buffer pool is provided for each data set group and each buffer pool is shared by all partitions.

The following table summarizes the buffering functions available to HSSR application programs.

Table 29. Overview of buffering services

	Chained Sequential I/O	Anticipatory Overlapped I/O	Look-aside Buffering	Reference Pattern Analysis
ESDS (CAB)	Y	Y	Y	Y
OSAM (CAB)	Y	Y	Y	Y
OSAM LDS (CAB)	Y	Y	Y	Y
ESDS (BB)	-	-	Y	-
OSAM (BB)	-	-	Y	-
OSAM LDS (BB)	-	-	Y	-
KSDS (see Note)	Y	Y	Y	-

Note: KSDS is buffered by VSAM, not by HSSR buffer handlers. VSAM provides the following buffering functions:

- Direct I/O
- Immediate-chained sequential I/O
- Anticipatory overlapped chained sequential I/O
- Look-aside buffering

Topics:

Chained Anticipatory Buffer handler (CAB)

The Chained Anticipatory Buffer handler (CAB) is a highly efficient HSSR buffer handler for ESDSs, OSAM databases, and OSAM LDSs. It is designed to reduce the I/O time, I/O wait time, and elapsed time for HSSR application programs.

CAB is the more advanced of the two HSSR buffer handlers. CAB uses the following buffering methods to improve buffering performance:

Direct I/O
Enables direct access to a single ESDS CI, OSAM LDS CI, or OSAM block.

Immediate (non-overlapped) chained sequential I/O
Enables reading of multiple consecutive ESDS CIs, OSAM LDS CIs, or OSAM blocks (called ranges) through Channel Command Word (CCW) chaining.

Anticipatory (overlapped) chained sequential I/O
Allows CAB to perform overlapped "look-ahead" I/O. This involves the overlapping of CAB I/O with processing and other I/O of the same job step. (Overlapped I/O is also performed with CCW chaining.) CAB decides dynamically when to start overlapped I/O, based on the reference pattern analysis.

Reference pattern analysis
Forecasts and decides whether chained sequential I/O of multiple blocks or CIs or direct I/O of one single block or CI is preferable.

The forecast is based on statistics about reference patterns within the most recently referred-to relative byte address (RBA) ranges of the data set. For example, assume that a segment was inserted after the last database load or reload, and that this segment was stored in an OSAM block, OSAM LDS CI, or ESDS CI far away from the blocks or CIs containing the other segments of either the same or neighboring database records. In such a case, CAB might forecast that direct I/O is superior to chained sequential I/O for reading the block or CI containing the inserted segment.

Look-aside buffering
Is an efficient buffering method also used by BB and DL/I buffer handlers.

Inter-PCB look-aside buffering (optional)
Inter-PCB look-aside is a method that enables CAB to attempt to find a requested RBA in buffers of other HSSR PCBs. If HSSR PCB 1 and HSSR PCB 2 refer to the same database, and if look-aside buffering for HSSR PCB 1 is unsuccessful, HSSR buffer handler tries to find the requested data in the CAB buffers of HSSR PCB 2. If the data is found, the buffer of HSSR PCB 2 is copied into a buffer of HSSR PCB 1.

Restrictions when using CAB for ESDS data sets and OSAM LDS data sets

When CAB is used for ESDS data sets or OSAM LDS data sets:

- The number of VSAM buffers for the ESDS data set or OSAM LDS data set is chosen by CAB. This number must not be overridden by means of JCL specifications or IDCAMS specifications. Do not code the BUFND, BUFSP, and STRNO operands in the AMP parameter of the DD statements. Avoid coding the BUFFERSPACE parameter on a DEFINE or ALTER command.
- If multiple HSSR PCBs defined in the PSB refer to the same ESDS database or OSAM LDS database, CAB can be used to buffer only one of these PCBs. Use the BB to buffer the other PCBs referring to this ESDS database or OSAM LDS database. Inter-PCB look-aside buffering option can be activated between CAB buffer and BB buffer for the same ESDS database or OSAM LDS database.

Basic Buffer handler

The Basic Buffer handler (BB) provides ESDS, OSAM data sets, and OSAM LDS with look-aside buffering services similar to those services provided by the DL/I buffer handler.

BB might yield better performance than the DL/I buffer handler, because HSSR Engine reduces the path length for database calls and uses direct I/O and look-aside buffering methods. However, BB does not

provide the immediate-chained sequential I/O or the anticipatory overlapped chained sequential I/O, which are provided by CAB.

BB is used for ESDS, OSAM data sets, and OSAM LDS if a BUF control statement is specified, in the HSSROPT data set, for the database.

To reduce the path length of an HSSR call, each HSSR PCB and each data set group within each PCB has its own buffers for OSAM, OSAM LDS, and ESDS, which is the same as CAB. If HSSR PCB 1 and HSSR PCB 2 refer to the same database, and look-aside buffering for HSSR PCB 1 is unsuccessful, the buffer handler tries to find the requested data in the BB buffers of HSSR PCB 2. If it finds the data, the buffer of HSSR PCB 2 is copied into a buffer of HSSR PCB 1. Buffers are maintained on a last-referred-to basis.

An output statistical report is provided to the user on the HSSRSTAT data set.

Buffering service for KSDS

By default, HSSR Engine uses "native" VSAM (VSAM with the No Shared Resource Pool Option) to read a KSDS.

Native VSAM provides both immediate chained sequential I/O and anticipatory overlapped chained sequential I/O. In batch processing, DL/I uses the Local Shared Resource (LSR) Pool option. This does not provide immediate chained sequential I/O, but provides more efficient look-aside capabilities for random database processing.

To improve the performance (through better look-aside buffering) of programs that issue a large number of GU calls to HIDAM or PHIDAM databases, HSSR Engine makes it possible, as an option, to process the primary index of an HIDAM and PHIDAM database to be processed with the LSR option.

For an HSSR application program, VSAM provides the following buffering services:

- Immediate chained sequential I/O
- Anticipatory overlapped chained sequential I/O
- Direct I/O
- Look-aside buffering

Subtopics:

- “"Native" VSAM” on page 209
- “VSAM LSR option for primary index databases” on page 210

"Native" VSAM

HSSR Engine does not provide statistics about VSAM KSDS buffering. Your installation might be able to provide SMF statistics that can be used to determine the optimum number of buffers.

The number of buffers for the processing of the index and data components of the KSDS cluster can be specified separately on the JCL DD statement.

The DD statement must be coded as follows:

```
//KSDS  DD  DSN=IMSVS.VSAM,DISP=SHR,
//          AMP=('BUFND=8,BUFNI=4')
```

BUFND is used to specify the buffer number for the data component and BUFNI for the index component of the KSDS cluster.

If the AMP parameter is not coded, the HSSR Engine sets the number of the two buffers for NSR as follows:

- BUFNI = Number of levels
- BUFND = One-fifth of the number of CIs per CA

VSAM LSR option for primary index databases

If primary index databases are to be processed with VSAM LSR, specify the LSR option in the HSSROPT data set. For details of the LSR control statement, see “LSR control statement” on page 172.

Usually, you do not need to take any other step to use LSR YES option. However, if a VSAM logical error occurs because all VSAM place holders are used in the DLI or DBB region, specify the appropriate STRINGNM parameter on the POOLID control statement or the STRINGMX parameter on the OPTIONS statement in the DFSVSAMP data set. The appropriate number is shown by the following formula:

```
2 x (the number of KSDS data sets that are to be accessed
     from your application program) + 2
```

Note: This option has no effect if the root segment of an HIDAM or PHIDAM database has no physical twin backward or hierarchical backward pointers. If the PSB has at least one DB PCB that has a PROCOPT allowing database update (PROCOPT=A, R, I, L, or D), this option is canceled internally.

If you specify the DFSSTAT data set, IMS prints statistics about LSR buffering in it at the end of the job step. HSSR Engine does not print any statistics report on its own.

To specify the number of buffers for the Local Shared Resource Pools, define it in the DL/I DFSVSAMP data set.

Chapter 14. Tuning the Chained Anticipatory Buffer handler

CAB buffering parameters and options are determined automatically based on the characteristics of the database data sets. Generally, you do not need to tune CAB. For severely fragmented databases, however, allocating more buffers than the default might improve the performance of your job.

Topics:

Considerations before tuning CAB

Before tuning CAB, learn about the factors that influence the performance of CAB.

What you need to know before tuning CAB

The most crucial CAB performance factor is that a database should be well organized. Such databases allow more chained sequential I/O and result in improved performance for HSSR application programs.

Avoid performance degradation due to inserts

CAB performs best with recently loaded or reorganized databases. The performance degrades when the database becomes older, because segments that have been inserted cannot be stored in the same OSAM block, OSAM LDS CI, or ESDS CI. This is especially true if the inserted segments of a given database key range (HIDAM and PHIDAM) or RAP range (HDAM and PHDAM) are scattered in multiple blocks or CIs. The performance degradation is encountered not only with CAB, but also with different kinds of processing, including online, and more specifically with almost any kind of sequential database processing.

To reduce performance degradation, two solutions can be investigated:

- Reorganize the database more frequently.
- Specify free space within the database in such a way that the database is better protected from performance degradations. This solution might require more DASD space for the database; however, most jobs including online jobs benefit from the free space specifications.

The impact of free space specifications on performance occurs when initially loading or reloading the database. If database load or reload occurs infrequently, you should plan for free space specifications before initially loading the database.

Here is an example of a free space implementation for a database:

- Try to store inserted segments into the same block or CI as the hierarchical neighbor segments by means of standard DBDGEN specifications, which reserve free space within each block or CI.
- Segments that cannot be inserted in the same block as the hierarchical neighbors should not be scattered into numerous blocks, but should be grouped by database key range or RAP range into a few blocks. The user achieves this by means of standard DBDGEN specifications that reserve entirely free blocks and by taking care of the IMS HD bit map blocks.

Without such a grouping, segments of a given database key range or RAP range that are inserted after database reorganization could be scattered in a number of different blocks. Retrieval of each of these segments could require up to one I/O per segment.

To implement these free space specifications, consider the following actions that can be taken:

- The FRSPC *fspf* operand on the DATASET statement of a DBD can be used to specify free space within each block. This free space improves the performance of most programs, including online programs, that access the database.
- The FRSPC *fbff* operand on the DATASET statement of a DBD can be used to leave each *n*th block entirely free for future segment insertion.

  Note: For HALDBs, use the HALDB Partition Definition Utility to specify FRSPC parameters.
- A dummy segment can be defined in the DBD. This dummy segment must be long enough to achieve the following goal: the IMS bit map blocks should indicate that only entirely free blocks contain enough free space to contain the longest database segment.
- The DBDGEN must be performed, and the database must be reloaded.

To decide where to store segments inserted after a database reload, the HD space search algorithm used by IMS uses the following criteria:

1. Same block
2. On the same track: any blocks that were originally left entirely free
3. On the same cylinder, then within *n* cylinders: any blocks that were originally left entirely free
4. Any block at the end of data set, based on the bit map
5. Any blocks that were originally left entirely empty.

First, search criterion 1 tries to insert the segment in the same block as its neighbors. If this fails, search criteria 2 and 3 try to group segments of a key range or RAP range into a few blocks of the same track, the same cylinder, or neighbor cylinders. Only if search criteria 1, 2, and 3 fail, scattering can occur during segment insertion.

Note: The list of search criteria has been simplified for the sake of demonstration. In reality, the search criteria used by IMS also consider CIs or blocks actually in the IMS buffer pools to reduce the amount of grouping.

For a more detailed description of the HD space search algorithm, see *IMS Database Administration*.

Control statements that affect performance

Use the HSSRCABP control statements to allocate more buffers to tune the performance of your job. Some control statements that are specified in the HSSROPT data set can also affect the performance of CAB.

Subtopics:

- "HSSRCABP control statements" on page 212
- "CAB buffer sharing for HALDB" on page 214
- "HSSROPT control statements" on page 214

HSSRCABP control statements

HSSRCABP data set contains the CAB control statements that are used to change CAB buffering parameters and options for a specified data set or for a specified group of data sets. The statements make it possible to override the default values of the CAB parameters. The following table lists the HSSRCABP control statements.

For an instruction for tuning buffers, see "Determining the appropriate CAB parameters" on page 216. For a detailed description of each control statement, see "HSSRCABP control statements" on page 217.

Table 30. HSSRCABP control statements

Control statement	Description
CABDD	This control statement defines a data set or a group of data sets to which the succeeding CAB control statements apply.
OCCURRENCE	This optional control statement specifies the HSSR PCB to which the specifications of the control statement set apply. It is used when multiple HSSR PCBs refer to the same database.
RANSIZE	This optional control statement specifies the size of a range, which is the fixed number of ESDS CIs, OSAM LDS CIs, or OSAM blocks read together in one chained sequential I/O operation. The default value for RANSIZE is determined from the characteristics of each database data set; the default value is determined as follows: • For OSAM, the value is equal to the number of blocks per track • For ESDS and OSAM LDS, the value is equal to: – Half of the number of CIs per CA if the CI size is greater than 2,048 bytes – One-fourth of the number of CIs per CA if the CI size is less than or equal to 2,048 bytes This default is efficient for relatively well organized databases. A RANSIZE value smaller than the default can be specified to reduce the buffer space. (In this case, the REFT4 parameter value, if it is coded, must also be reduced.)
NBRSRAN	This optional control statement specifies the number of whole ranges to be buffer-resident for look-aside buffering. The NBRSRAN parameter affects the number of successful look-aside operations, as well as buffer space. With well-organized databases, NBRSRAN can be reduced to the minimum value of 3 (because look-aside operations are less important with well organized databases) to reduce the buffer space. With disorganized databases, increasing NBRSRAN can sometimes increase the number of successful look-aside operations. This decreases the number of direct and chained sequential I/Os.
NBRDBUF	This optional control statement specifies the number of single blocks or CIs read with direct I/O to be buffer-resident for look-aside buffering.
OVERFLOW	This optional control statement affects the buffering of the prime data set group of an HDAM or PHDAM database and describes how the overflow area should be buffered. The OVERFLOW parameter affects the buffering of HDAM and PHDAM databases. If a large number of I/O operations are performed in the overflow area, either the OVERFLOW CAB option (default) or the OVERFLOW SHR option is recommended, because either of them allows chained sequential I/O in the overflow area. Note that the OVERFLOW CAB option requires more buffer space than OVERFLOW SHR. The OVERFLOW BB is reasonable if only a few I/O operations are performed in the overflow area. However, it does not allow chained sequential I/O in the overflow area.

Table 30. HSSRCABP control statements (continued)

Control statement	Description
REFT4	This optional control statement is used as a reference threshold value to help determine whether chained sequential I/O or direct I/O should be performed. The default value of REFT4 equals to that of RANSIZE. This value can be decreased or increased, to regulate the number of direct and chained sequential I/O operations. This parameter does not affect the number of buffers to be allocated by CAB.
INTER	This optional control statement activates the CAB inter-PCB look-aside, which enables CAB to attempt to find a requested RBA within buffers of other HSSR PCBs that refer to the same database.
PARTPROC	This optional control statement, valid only for HALDB, specifies the access intent for the database or the databases specified on the statement. For a HALDB, CAB buffers are shared among data sets that belong to the same data set group. If only one partition is accessed at a time, as in the unload utilities such as FABHURG1 or FABHFSU utility, you do not need to use this control statement. If more than one partition is accessed randomly, you must specify the number of partitions that are accessed at a time. For the detailed description of PARTPROC statement and the buffer sharing for HALDB, refer to “CAB buffer sharing for HALDB” on page 214.

CAB buffer sharing for HALDB

For HALDBs, CAB buffers are shared among data sets that belong to the same data set group.

If only one partition is accessed at a time, as in the unload utilities such as FABHURG1 or FABHFSU utility, do not code the PARTPROC control statement. In this case, the CAB buffer for each data set group is used exclusively for the partition that HSSR buffer handler accesses at that time.

If more than one partition is accessed randomly at a time from your HSSR application program, you must specify a PARTPROC control statement for the HALDB, with R as the second operand and the number of partitions that are accessed at a time as the third operand. CAB buffer handler then calculates the required number of buffers based on the CAB parameters specified for each data sets. It allocates enough buffer space so that CAB buffering requirements specified by the CAB control statement are met for each data set as long as partitions more than the number specified in the PARTPROC statement are not accessed.

The total amount of storage used for the CAB buffer space for the HALDB is reported in the HALDB Buffering Statistics report (see HALDB Buffering Statistics).

HSSROPT control statements

This input data set provides control statements that affect CAB buffering. The following table shows a brief review of the HSSROPT control statements that you might want to consider using with CAB. For a detailed description of these control statements, see Chapter 11, “Options for HSSR Engine,” on page 155.

Table 31. HSSROPT control statements for CAB

Control statements	Description
BUTR	This optional control statement activates a trace of CAB buffering activities. The trace is written to the data set defined by the HSSRBUTR DD statement. This data set is used as input to FABHBSIM to simulate CAB buffering in order to tune CAB buffers. The buffer trace is not taken for HALDBs.
NOFIX	This optional control statement prevents page-fixing. Since a noticeable amount of paging might occur when CAB buffers are not page-fixed, this statement is usually omitted.
NOVSAMOPT	This optional control statement prevents CAB from using the default read-ahead threshold value used by VSAM.

Trade-off decisions between elapsed time and buffer space

In some cases, you must make trade-off decisions between elapsed time and buffer space.

The RANSIZE parameter has the greatest effect on both elapsed time and buffer space. The number of buffers allocated by CAB to each PCB/data set combination, except for the prime data set group of HDAM, is given by:

```
      (RANSIZE x (NBRSRAN + 1)) + NBRDBUF
```

Additional CAB buffers of the same number are allocated for the overflow area of the prime data set group of HDAM and PHDAM if 'OVERFLOW CAB', which is the default, is specified.

CAB default parameters are:

```
RANSIZE=(the value determined from data set characteristics)
NBRSRAN=8
NBRDBUF=RANSIZE x 2
OVERFLOW=CAB
REFT4=RANSIZE
```

The use of these default parameter values should greatly reduce elapsed time. The default values result in the allocation of buffers of RANSIZE x 11 (and additional CAB buffers for the overflow area of the prime data set group of HDAM or PHDAM).

Although CAB uses a fair amount of storage for buffer space, it normally uses virtual and real storage for less time than BB. By default, the CAB buffers are page-fixed for better performance, but the page-fixing can be disabled by coding the NOFIX control statement in the HSSROPT DD.

In cases where CAB performance is not superior to BB performance (for example, with disorganized databases), use BB rather than CAB. This saves both virtual and real storage.

Size of OSAM blocks and ESDS/OSAM LDS control intervals

The optimal size of OSAM blocks and ESDS/OSAM LDS control intervals depends on the type of processing.

Smaller blocks or CIs are often beneficial to random processing (including online processing). When, during random processing, segments of a database record are accessed, it seldom makes sense to perform I/O for huge blocks or CIs containing much data belonging to many other database records. If, during direct processing, the blocks or CIs are smaller, the data transfer time often decreases. Therefore, the use of DASD string controllers, DASD control units, and channels can drop. The buffer space might also decrease.

Larger blocks or CIs are often beneficial to sequential processing since, with conventional buffer handlers, one full DASD rotational delay is lost between each I/O for two consecutive blocks or CIs. Increasing

the block size or CI size decreases the number of blocks or CIs and, consequently, the number of lost rotational delays and the elapsed time.

Therefore, you must sometimes make trade-off decisions in favor of random processing or sequential processing.

With CAB, this problem can sometimes be solved by using small block sizes or CI sizes for random processing. Sequential processing with CAB does not suffer, since CAB may chain CCWs in order to read many of these smaller blocks together as if they were parts of one single larger block.

SMF EXCP statistics

Statistics from the System Management Facilities (SMF) about the Execute Channel Program (EXCP) can sometimes be confusing.

The SMF EXCP statistics for data sets used by HSSR Engine should be used for reference only. "CAB Statistics" does not report the EXCP counts, but does report the number of I/O requests. For VSAM ESDS or OSAM LDS, one GET macro is used by one I/O request. For OSAM, the READ macro is used and the number of READ macros is:

- 1 for direct I/O
- The value of RANSIZE for chained sequential I/O

Determining the appropriate CAB parameters

Determine the appropriate CAB parameters to improve the CAB performance.

About this task

The CAB performance can be considered satisfactory when the following conditions are met:

- The number of OSAM blocks, OSAM LDS CIs, or ESDS CIs read in chained sequential mode is much greater than the number of blocks or CIs read in direct mode.
- The percentage of unreferred-to sequential buffers is below 15 or 20%.
- Elapsed time is substantially smaller with CAB than with BB.
- The amount of buffer space that is used is acceptable.

If you did not get satisfactory results by running your IMS High Performance Unload job with the default CAB parameters, you can attempt tuning.

Procedure

The following steps describe a possible approach to tuning CAB's reference pattern analysis logic:

1. Run your application program or the FABHTEST utility on a dedicated system with CAB and with the default values for the CAB parameters.

   Activate the machine-readable buffer handler trace for future FABHBSIM runs. (This increases the processor time slightly; bear this fact in mind when you compare processor times.)

   Note: Performance tests with databases that contain fewer than 300,000 segments are not reasonable because CAB buffer handler has a high initialization overhead.

   If you code 'CABSTAT YES' in the HSSROPT data set, extensive buffer handler statistics are printed at the end of the job step on the HSSRSTAT data set. See “CAB Statistics report” on page 186 and “Data Set I/O Statistics report” on page 185.

   Tip: After you have tuned the buffer handler, you can remove the CABSTAT control statement to reduce the content of the report. If no CABSTAT control statement is coded, or 'CABSTAT NO' is specified in the HSSROPT data set, only the summary page of the CAB Statistics is printed for each buffer pool.
2. Run the same program with BB on a dedicated system by coding the BUF control statement in the HSSROPT DD.

3. Check the results of both runs. First compare the elapsed time.

   Note the number of sequential buffers not referred to, and the number of blocks read in chained sequential mode and direct mode.

   If your database is well organized, CAB should show excellent elapsed time figures. Unless you want to reduce the buffer space, no further tuning is needed.

   If your database is poorly organized, CAB might show moderate or disappointing results.

   If the improvements are moderate—that is, the elapsed time of the CAB run is between 65% and 80% of the BB run—you should investigate whether the database reorganization interval is appropriate and whether the free space specifications are appropriate.

   You can also investigate the increase of the NBRSRAN parameter and the tuning of the REFT4 parameter to decrease a little more the elapsed time.

   If the results are disappointing—that is, if the elapsed time of the CAB run is not below 80% of the first run—the best thing, probably, is to reorganize the database more frequently to improve the specification of free space.
4. If you decide to rerun with modified CAB parameters, you can use the FABHBSIM utility to compare the HSSRSTAT statistics of the original run with those of the new run.

   The FABHBSIM utility enables you to observe the effect that changes to parameters on HSSRCABP control statements have on buffer handler performance. FABHBSIM can be used for simulations of both CAB and BB.

   The statistics to be compared are the number of direct I/Os, the number of chained sequential I/Os, the number of sequential buffers note referred to, and the timing statistics.

   Restriction: FABHBSIM cannot be used for HALDBs.

HSSRCABP control statements

The HSSRCABP data set contains the CAB control statements that are used to change CAB buffering parameters and options for a specified data set or for a specified group of data sets.

HSSRCABP control statements can be included in the input stream or stored in a partitioned data set (PDS). To centralize all CAB buffering specifications, consider storing HSSRCABP control statements in a PDS. Each member of the PDS must contain a set of HSSRCABP control statements for one given data set.

The HSSRCABP DD statement of the IMS High Performance Unload jobs can then refer to the members of this PDS. Multiple members can be referred to through concatenated DD statements.

Syntax rules for HSSRCABP control statements

Before you code CAB control statements, you should be familiar with the following general coding rules:

- CAB control statements other than the PARTPROC control statement must be provided in groups in the HSSRCABP data set. There must be one group of control statements for each set of data sets to be buffered by CAB.
- Each group of CAB control statements must begin with a CABDD statement that identifies (by *ddname*) a set of data sets to be buffered by CAB. The group ends when the next CABDD statement or the end-of-file is encountered.
- A group can contain additional control statements after the CABDD statement.
- The first control statement entry must be a keyword that identifies the type of control statement being defined. It is often the name of a CAB parameter.
- The keyword must always be followed by a single blank.
- The blank is followed by variable data. This variable data is often the value to be assigned to a CAB parameter.

CABDD control statement

This required control statement defines the data sets to be buffered by CAB.

This statement must be the first statement in a group of statements.

The CABDD control statement can have one of the following formats:

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

CABDD ddname
      'pattern'
      *ALL
      *HD
      *HS
      *PHD
```

Position
Description

1
Code the CABDD keyword to change the CAB buffering parameters for the set of data sets determined from the value specified in the operand of the statement.

7
This required entry identifies a set of data sets to which the succeeding CAB control statements apply. Code one of the following keywords:

ddname
Indicates that the succeeding CAB control statements apply to the data set whose DD name is *ddname*.

'*pattern*'
Indicates that the succeeding CAB control statements apply to the data sets whose DD names match the wildcard string *pattern*.

The *pattern* is a string of alphanumeric characters, including wildcard characters. You can use two kinds of wildcard character: an asterisk (*) and a percent (%) symbol. An asterisk is treated as a sequence of 0 to 8 characters; and, a percent symbol is treated as a single character. If two or more asterisks are specified sequentially, only the first asterisk is recognized. You cannot specify asterisks and percent symbols simultaneously.

The *pattern* string must be enclosed by single quotation marks.

***ALL**
Indicates that the succeeding CAB control statements apply to all ESDSs, OSAM data sets, or OSAM LDSs.

***HD**
Indicates that the succeeding CAB control statements apply to all ESDSs, OSAM data sets, or OSAM LDSs of all HD databases, including PHDAM and PHIDAM databases.

***HS**
Indicates that the succeeding CAB control statements apply to all ESDS data sets of all HISAM databases.

***PHD**
Indicates that the succeeding CAB control statements apply to all ESDS, OSAM data sets, or OSAM LDSs of all PHIDAM and PHDAM databases.

INTER control statement

This optional control statement activates the CAB Inter-PCB Look-Aside. Inter-PCB Look-Aside is a method that enables CAB (or BB) to attempt to find a requested RBA within buffers of other HSSR PCBs.

The INTER control statement must be used to activate this feature. By default, no Inter-PCB Look-Aside is done in CAB. This option has a meaning only when multiple HSSR PCBs refer to the same database.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

INTER YES
```

Position
Description

1
Code the INTER keyword to change the INTER option.

7
Code YES to activate the CAB inter-PCB look-Aside.

NBRDBUF control statement

This optional control statement specifies the number of direct buffers—that is, the number of single blocks or CIs read in direct mode—that should be resident in the buffer for look-aside purposes.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

NBRDBUF n
```

Position
Description

1
Code the NBRDBUF keyword to change the number of direct buffers.

9
Code the numeric value *n* from 3 through 255 to be left-aligned. The default value assigned to NBRDBUF is twice the number assigned to RANSIZE, which is the number of single blocks or CIs resident in a buffer for look-aside purposes.

NBRSRAN control statement

This optional control statement specifies the number of whole ranges to be resident in each buffer for look-aside purposes. NBRSRAN affects both the buffer space and the number of successful look-aside operations. It also affects the number of I/Os if a database is not well organized.

For each data set of each PCB for which CAB is used (except for the prime data set group of HDAM and PHDAM, as described in “OVERFLOW control statement” on page 220), CAB allocates the following number of buffers:

```
    RANSIZE x (NBRSRAN + 1)
```

If the database is well organized, you do not need to change the default value; if the database is disorganized, increasing the NBRSRAN value could increase the benefits achieved through look-aside buffering.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

NBRSRAN n
```

Position
Description

1
Code the NBRSRAN keyword to change the NBRSRAN value.

9
Code the numeric value *n* to be left-aligned with a numeric value from 3 through 9999. It is the number of ranges resident in a buffer for look-aside purposes.

The default value assigned to NBRSRAN is 8.

OCCURRENCE control statement

This optional control statement specifies which HSSR PCBs, in the PSB, are to be buffered by CAB. It is used only when multiple HSSR PCBs refer to the same database. It identifies the HSSR PCB to which the specifications of the control statement group apply.

For a database that uses the OSAM for its database data sets, if no OCCURRENCE control statement is entered, HSSR assumes that the set of control statements applies to all HSSR PCBs referring to the data set or data sets identified by the CABDD statement.

For a database that uses the VSAM ESDS or OSAM LDS for its database data sets, CAB buffering can be used for only one PCB, which, by default, is the first PCB referring to the database. BB buffering is used when HSSR buffer handler accesses the database through the rest of the PCBs. This default can be changed by the OCCURRENCE control statement.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

OCCURRENCE n
```

Position
Description

1
Code the OCCURRENCE keyword to activate the OCCURRENCE option.

12
This entry specifies that the group of control statements applies to the *n*th HSSR PCB referring to the data set identified by the CABDD statement.

OVERFLOW control statement

This optional control statement specifies how the overflow area of the prime data set group of an HDAM database or the overflow area of the prime data set group of each partition of a PHDAM database, should be buffered.

The OVERFLOW control statement affects the buffering efficiency for the prime data set group of an HDAM database, or of the prime data set groups of a PHDAM database; especially, it affects both the size of the buffer space and the elapsed time.

The same OVERFLOW statement must be specified for each prime data set group of partitions of a PHDAM database.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

OVERFLOW CAB
         SHR
         BB
```

Position
Description

1
Code the OVERFLOW keyword to change the OVERFLOW option.

10
Code one of the following three keywords:

CAB
CAB is the default.

CAB specifies that separate CAB buffers are allocated for the root addressable area and the overflow area, respectively. Chained sequential I/O is possible in both areas.

The total number of buffers allocated for the first data set group of HDAM is given by:

```
2 x (RANSIZE x (NBRSRAN + 1) + NBRDBUF)
```

The total number of buffers allocated for the first data set group of a PHDAM database depends on the PARTPROC statement specified for the database.

SHR

SHR instructs CAB to use the same buffer for both the root addressable area and the overflow area.

Chained sequential I/O is possible in both areas.

The number of (shared) buffers allocated for the first data set group of HDAM is:

```
RANSIZE x (NBRSRAN + 1) + NBRDBUF
```

The total number of buffers allocated for the first data set group of a PHDAM database depends on the PARTPROC statement specified for the database.

BB

BB does not allow chained sequential I/O in the HDAM or PHDAM overflow area. If only a small number of I/Os are performed in the overflow area, the OVERFLOW BB option is reasonable.

The OVERFLOW BB option specifies that CAB buffers the root addressable area and that BB buffers the overflow area. No chained sequential I/O takes place in the overflow area.

The OVERFLOW BB option uses less buffer space than the OVERFLOW CAB option. For HDAM, the total number of buffers allocated is the sum of the following:

- For the root addressable area,

  ```
  RANSIZE x (NBRSRAN + 1) + NBRDBUF
  ```

- For the overflow area,

  ```
  NBRDBUF
  ```

The total number of buffers allocated for the root addressable areas and overflow areas of the first data set group of a PHDAM database depends on the PARTPROC control statement specified for the database.

PARTPROC control statement

This optional control statement, valid only for HALDB, specifies the access intent for the database or the databases specified on the statement.

For a HALDB, CAB buffers are shared among the data sets in a data set group. As in the unload utilities such as the FABHURG1 or FABHFSU utility, if only one partition is accessed at a time, you do not need to use this control statement.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

PARTPROC dbd_name
PARTPROC dbd_name S
PARTPROC dbd_name R nnnn
```

Position

Description

1

Code the PARTPROC keyword to change the PARTPROC option for a HALDB or for all HALDB.

10
This required entry identifies the HALDB or HALDBs to which the PARTPROC option applies. The string must be left-aligned.

dbd_name
Indicates that the PARTPROC option applies to the HALDB identified by the DBD *dbd_name*.

***PHD**
Indicates that the PARTPROC option applies to all HALDBs.

19
Code one of the following keywords:

S | blank
The keyword S stands for *sequential access* and tells CAB to prepare a buffer space that can buffer only one partition for each HALDB specified on the column 10.

If this option is specified, the CAB parameters RANSIZE, NBRSRAN, NBRDBUF, OVERFLOW, REFT4, and INTER that are specified for a partition are reset when the processing of the partition starts.

This is the default for all HALDBs.

R
The keyword R stands for *random access* and instructs CAB to prepare the buffer space that can buffer *nnnn* partitions of the HALDB specified on the column 10. If the number *nnnn* on column 21 is omitted, *nnnn* = 2 is assumed.

If this keyword is specified, the CAB buffers each data set on the basis of CAB parameters RANSIZE, NBRSRAN, NBRDBUF, OVERFLOW, REFT4, and INTER that are specified for the partition, as long as no more than *nnnn* partitions are accessed at a time.

21
Specify the maximum number of partitions to be accessed at a time. The default value is 2 when the keyword R is specified in column 19.

This parameter can be specified only when the keyword R is specified in column 19.

The number *nnnn* must be 1- to 4-characters long and must be left-aligned.

Note: If multiple PARTPROC statements are specified for the same HALDB, the last statement is treated as a valid statement for the HALDB.

RANSIZE control statement

This optional control statement denotes the size of a range, which is the number of OSAM blocks, OSAM LDS CIs, or ESDS CIs that are read together in chained mode. RANSIZE affects both the elapsed time and the buffer space.

The number of sequential buffers allocated by CAB for each data set of each PCB to which CAB is used is:

```
    RANSIZE x (NBRSRAN + 1)
```

The default RANSIZE value for each database data set is determined by HSSR buffer handler from the characteristics of the data set, such as the block size or the CI size.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

RANSIZE n
```

Position
Description

1
Code the RANSIZE keyword to change the RANSIZE value.

9
Code the numeric value *n* left-aligned with a numeric value from 2 through 255.

Notes:

- Performance is questionable when CAB buffers OSAM data sets with a RANSIZE smaller than 4.
- A limited number of CIs can be read by the access method within one single-chained I/O operation. The limit depends on the CI size. When HSSR buffer handler detects that the RANSIZE value causes the use of more than a limited number of CIs for an ESDS or OSAM LDS database, the buffer handler changes the value of RANSIZE to the allowable maximum.
- When the RANSIZE default value is overridden, the REFT4 parameter, if it is coded, must be adjusted. (See "REFT4 control statement" on page 223.)

REFT4 control statement

This optional control statement is used as a reference threshold value. The REFT4 threshold value helps determine whether chained sequential I/O or direct I/O should be performed.

When the number of referred-to OSAM blocks, OSAM LDS Control Intervals, or ESDS Control Intervals (CIs) is below REFT4, CAB usually performs direct I/O. When the number of referred-to OSAM blocks, OSAM LDS CIs, or ESDS CIs has reached REFT4, CAB usually performs chained sequential I/O.

The REFT4 setting helps determine how often direct I/O and chained sequential I/O are performed. If the REFT4 setting is too low, CAB may perform chained sequential I/O in cases where direct I/O is superior. Some of the OSAM blocks, OSAM LDS CIs, or ESDS CIs read in chained sequential mode are not referred to at this time. If the REFT4 setting is too high, CAB may perform direct I/O in cases where chained sequential I/O is superior. CAB reads consecutive OSAM blocks, OSAM LDS CIs, or ESDS CIs individually in direct mode, instead of reading them together in chained sequential mode.

Usually, you do not need to code the REFT4 control statement. Consider coding the control statement only when the default value is not satisfactory. The recommended range of values for REFT4 is between 100% and 200% of RANSIZE. For example, when RANSIZE=8, the recommended range of values for REFT4 is from 8 to 16 inclusive.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

REFT4 n
```

Position
Description

1
Code the REFT4 keyword to change the REFT4 threshold.

7
Code the numeric value *n* to be left-aligned.

The default value is equal to the RANSIZE value.

JCL examples for specifying CAB parameters

Use the following examples to specify CAB parameters.

Subtopics:

These examples are for nonpartitioned databases. For examples of how to specify CAB control statements for PHDAM or PHIDAM databases, see Chapter 8, "Methods for processing High Availability Large Databases," on page 97.

These examples are not examples of a real production job. They merely demonstrate how HSSRCABP control statements, as well as HSSROPT control statements, can be specified.

Example 1: CAB control statements for FABHULU jobs

The following is an example of how to specify CAB control statements for an HSSR application program running in the ULU region by using the FABHULU cataloged procedure. The FABHURG1 unload utility is used in this example, but the same explanation applies to any other HSSR application program running in the ULU region.

In the following JCL, 'CABDD *ALL' is coded in HSSRCABP DD, which means that the CAB parameter values are overridden by the value specified in the succeeding control statements.

Assume that the database processed is well organized and not fragmented, and that the set of parameters is selected to reduce the amount of buffer space used by CAB from the default specification and to allocate 19 CAB buffers.

```
// EXEC FABHULU,MBR=FABHURG1,DBD=HIDOSAM
//HSSRCABP DD *
CABDD *ALL
RANSIZE 4
NBRSRAN 3
NBRDBUF 3
OVERFLOW SHR
REFT4 6
/*
//HIDOSAMD DD DSN=TESTDS.HIDOSAMD,DISP=SHR
//HIDOSAMX DD DSN=TESTDS.HIDOSAMX,DISP=SHR
//SYSPRINT DD SYSOUT=A
```

Example 2: OCCURRENCE control statements for a VSAM ESDS database

In the following FABHDLI job, 'CABDD *ALL' is coded in HSSRCABP DD. This means that, for all databases that are read by HSSR Engine using CAB buffering, the CAB parameter values to be overridden by the value specified in the succeeding control statements.

For a database that uses the VSAM ESDS for its database data sets, CAB buffering can be used for only one PCB, which, by default, is the first PCB, in the PSB, that refers to the database; and BB buffering is used when HSSR Engine accesses the database through the rest of the PCBs. This default can be changed by use of the OCCURRENCE control statement.

The OCCURRENCE statement in this example specifies that the second PCB in the PSB is to be buffered with CAB. The succeeding statements allocate 23 CAB buffers for the database when it is accessed through the second PCB.

The HSSROPT data set contains control statements that affect CAB buffering. The 'HSSRPCB *ALL' statement specifies that all DB PCBs in the PSB HRDHDAMG are HSSR PCBs. The BUTR control statement activates a trace of CAB buffer handler activities; the HSSRBUTR DD statement defines the output data set. The NOVSAMOPT prevents CAB from using the default read-ahead threshold value used by VSAM.

The SYSIN data set provides the input to the FABHTEST utility.

```
// EXEC FABHDLI,MBR=FABHTEST,PSB=HRDHDAMG
//SYSIN    DD *
PCB 2
GN  GB
/*
//HSSROPT DD *
HSSRPCB *ALL
BUTR
NOVSAMOPT
/*
//HSSRCABP DD *
CABDD *ALL
OCCURRENCE 2
RANSIZE 4
NBRSRAN 4
NBRDBUF 3
OVERFLOW SHR
REFT4 6
/*
//ESDSDATA DD DSN=TESTDS.ESDSHDAM,DISP=SHR
//SYSPRINT DD SYSOUT=A
//HSSRBUTR DD DSN=HPU.HSSRBUTR,UNIT=TAPE,DISP=(,KEEP)
```

Example 3: Specifying multiple CABDD groups

In this example, assume that the user application program USERAPPL accesses two databases DB1VSAM and DB2OSAM; the HDAM database DB1VSAM over VSAM ESDS has data set groups DB1DSG1 and DB1DSG2; and the HIDAM database DB2OSAM over OSAM has data set groups DB2ROOT, DB2DEP01, and DB2DEP02.

In the following JCL, two CABDD groups, 'DB1DSG%' and 'DB2*', are specified in HSSRCABP DD. The use of the wild cards in DD names 'DB1DSG%' and 'DB2*' causes all data sets for DB1VSAM and for DB2OSAM to be selected.

The CAB control statements succeeding each CABDD statement specify the CAB parameters for the specified CABDD group.

```
// EXEC FABHDLI,MBR=USERAPPL,PSB=PSBAPPL1
//HSSROPT DD *
HSSRPCB *ALL
DBSTATS
/*
//HSSRCABP DD *
CABDD 'DB1DSG%'
RANSIZE 4
NBRSRAN 10
NBRDBUF 10
REFT4 6
OVERFLOW SHR
CABDD 'DB2*'
RANSIZE 4
NBRSRAN 20
REFT4 6
/*
//DB1DSG1  DD DSN=TESTDS.DB1.DSG1,DISP=SHR
//DB1DSG2  DD DSN=TESTDS.DB1.DSG2,DISP=SHR
//DB2ROOT  DD DSN=TESTDS.DB2.DSROOT,DISP=SHR
//DB2DEP01 DD DSN=TESTDS.DB2.DSDEP01,DISP=SHR
//DB2DEP02 DD DSN=TESTDS.DB2.DSDEP02,DISP=SHR
//DB2INDEX DD DSN=TESTDS.DB2.DSINDEX,DISP=SHR
//applout DD SYSOUT=A
```

Chapter 15. Tuning the Basic Buffer handler

The performance of the Basic Buffer handler depends on how well the databases are organized. You can use HSSROPT control statements to specify the number of buffers. Allocating more buffers than the default might improve the performance of your job.

Topics:

Control statements that affect performance

BUF and BUTR are the HSSROPT control statements that affect the performance of the Basic Buffer handler.

The following table shows a brief description of the HSSROPT control statements that you might want to consider using with BB.

Table 32. HSSROPT control statements for Basic Buffer handler

Control statements	Description
BUF	If you want to use BB for ESDSs, OSAM data sets, or OSAM LDSs, you must code this statement for the DBD, and specify the number of BB buffers. This statement is optional. The default number of buffers depends on how the PCB is defined as an HSSR PCB.
BUTR	This optional control statement activates a trace of BB buffering activities. The trace is written to the data set defined by the HSSRBUTR DD statement. This data set is used as input to FABHBSIM to simulate BB buffering in order to tune BB buffers. You cannot use this control statement for PHDAM or PHIDAM databases.

BUF control statement

The optional BUF control statement specifies a database for which the BB buffer handler is to be used, and to override the default number of buffers for BB.

The BB buffer handler allocates a separate buffer pool for each data set group of the database. It also allocates a certain number of buffers to each of these buffer pools—either the default or a specified number.

The default number of buffers depends on how the PCB is defined as an HSSR PCB. If an HSSR PCB is defined by use of either an HSSRPCB or an HSSRDBD control statement, and the value to the KEYLEN keyword for the PCB is less than 200, BB allocates six basic buffers for each data set as the default. If the KEYLEN value for the PCB is greater than 200, the number of basic buffers to be allocated is determined as explained in "Number of Basic Buffers for an HSSR PCB" on page 359.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

BUF dbdname ,nbrbuffers
```

Position

Description

1
Code the BUF keyword to specify a database for which the BB buffer handler is to be used and to override the default number of buffers for BB.

5
Code the 8-byte *dbdname* to specify the database for which the default number of buffers will be overridden (if the database name is not 8 bytes long, include trailing blanks).

13
Add a comma (,) to separate the database name from the number of buffers.

14
nbrbuffers is the number of buffers that you want BB to allocate for a buffer pool.

BUTR control statement

The BUTR control statement directs HSSR Engine to create a file containing a machine-readable trace of internal calls to the buffer handler.

This trace, which is written to the HSSRBUTR data set, can be used as input to FABHBSIM, the HSSR Buffer Handler Simulation utility.

Note: For instructions for using the buffer handler simulation utility, see Chapter 17, "Buffer handler simulation utility (FABHBSIM)," on page 239.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

BUTR
```

Position
Description

1
Code the BUTR keyword to instruct HSSR Engine to create a file that contains a machine-readable trace of internal calls to the buffer handler.

Restriction: No buffer trace is taken for HALDBs.

Determining the appropriate number of BB buffers

Determine the appropriate number of BB buffers to improve the BB performance.

Procedure

You can use the following aids, provided by HSSR Engine, for tuning the BB buffer handler:

- BB I/O and buffer handler statistics are provided in the HSSRSTAT data set (see "Data Set I/O Statistics report" on page 185).
- The FABHBSIM utility enables you to observe the effect of changes to BUF control statements in the HSSROPT data set.

No formula is provided for calculating the optimum number of ESDS, OSAM, or OSAM LDS buffers because each database or application has its own characteristics. However, take the following into account:

- For sequential processing of a recently loaded database, two ESDS, OSAM, or OSAM LDS buffers should be sufficient.
- For sequential processing of an older, not well-organized database, additional ESDS, OSAM, or OSAM LDS buffers might improve performance.
- When a database is processed at random, allocation of more than two buffers might be beneficial. The *n*th random I/O operation might find its data in a buffer already filled during an earlier random operation.

Chapter 16. HSSR call test utility (FABHTEST)

FABHTEST is the HSSR Engine test utility that runs a sequence of HSSR or DL/I calls against an IMS database.

FABHTEST is useful for performance testing and for problem determination of HSSR Engine. It can be used to compare the performance of an application program making HSSR calls with the performance of the same program making DL/I database calls against the same database.

FABHTEST, which runs as an HSSR application program, issues database calls against IMS databases in the sequence that is specified by control statements. It issues HSSR calls or DL/I calls through the appropriate language interface. HSSR calls are made unless the HSSR option DBDL1 forces DL/I calls to be issued. The HSSR or DL/I calls that are acceptable to FABHTEST are a subset of DL/I database calls.

You can use the FABHTEST utility to do the following tasks:

- Run a sequence of database calls against a database
- Help in problem determination
- Compare performance of HSSR calls versus DL/I calls
- Test performance using CAB

Topics:

FABHTEST restrictions

Certain restrictions apply when using the FABHTEST utility.

FABHTEST has the following restrictions:

- No printout of calls issued against a DL/I database PCB can be obtained.
- FABHTEST cannot be run with a DLIBATCH procedure, because it depends on information in the control blocks of HSSR Engine.
- The FABHTEST utility cannot process a HALDB by partition. It processes the HALDB as an entire database.
- For REPL calls, use PROCOPT=R.

  Notes:

  – REPL call is supported for the compatibility with DBT HSSR and PO HSSR.

  – REPL calls are not allowed for PHDAM or PHIDAM databases.

Running FABHTEST to test HSSR calls

You can test HSSR calls by running the FABHTEST utility.

Procedure

Complete the following steps to run the FABHTEST utility.

1. Use one of the FABHDLI, FABHDBB, or FABHULU procedure.

2. Code the SYSIN, HSSROPT, and HSSRCABP control statements.
3. Run FABHTEST as an HSSR application program.
4. Review and analyze output reports.

FABHTEST JCL requirements

FABHTEST runs as an HSSR application program and, therefore, must meet the requirements for the basic JCL (FABHX034 JCL). In addition, FABHTEST JCL requires other DD statements.

Prerequisite: See “Basic JCL requirements” on page 30 for the basic (FABHX034) JCL requirements.

The following table summarizes additional JCL requirements for FABHTEST.

Table 33. FABHTEST DD statements

DDNAME	Use	Format	Need
SYSIN	Input	LRECL=80	Required
SYSPRINT	Output	LRECL=133	Required

EXEC
This statement invokes the procedure FABHDLI, FABHDBB, or FABHULU. The format is as follows:

```
//  EXEC  FABHDLI,MBR=FABHTEST,PSB=psbname
//  EXEC  FABHDBB,MBR=FABHTEST,PSB=psbname
//  EXEC  FABHULU,MBR=FABHTEST,DBD=dbdname
```

The PCB referred to by *psbname* must be declared as an HSSR PCB.

See “HSSR PCB requirements” on page 80.

SYSIN DD
This required data set contains the control statements for FABHTEST.

See “FABHTEST SYSIN input data set” on page 230 for details about the control statements.

SYSPRINT DD
The required output data set contains output created by FABHTEST. The DD statement must be coded as follows:

```
//SYSPRINT DD SYSOUT=A
```

FABHTEST input

The FABHTEST utility uses three data sets as input: the SYSIN data set, the HSSROPT data set, and the HSSRCABP data set.

FABHTEST SYSIN input data set

FABHTEST depends upon the SYSIN data set to provide control statements that specify the sequence of HSSR calls.

FABHTEST processes eight types of control statements. Any syntax error within an FABHTEST control statement leads to an error message and an abend with dump.

Format

This data set contains 80-byte fixed-length records. The control statements can be coded in the input stream or accessed as a member of a partitioned data set.

GN and GHN control statement

These statements instruct FABHTEST to issue a specified number of get-next (GN) or get-hold-next (GHN) calls.

If no segment name is specified, GN or GHN calls without SSA are issued. If a segment name is specified, GN or GHN calls with an unqualified SSA are issued.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

GN  GB          seg_name
GHN GB          seg_name
    n           seg_name
```

Position
Description

1
Code the GN or the GHN keyword to identify this as a get-next control statement or a get-hold-next control statement.

5
Code one of the following optional keywords:

GB
FABHTEST issues GN or GHN calls until the end of the database is reached.

n
The number of GN or GHN calls that FABHTEST issues. Code a number containing up to 10 digits, left-aligned. Leading zeros are not necessary.

Blank
FABHTEST issues 1 GN or GHN call.

16
Code one of the following optional keywords:

seg_name
Code the name of a segment. FABHTEST issues GN or GHN calls with an unqualified SSA.

Blank
FABHTEST issues GN or GHN calls without an SSA.

GNP and GHNP control statement

These statements make FABHTEST issue a specified number of get-next-within-parent (GNP) or get-hold-next-within-parent (GHNP) calls.

If no segment name is specified, GNP or GHNP calls without SSA are issued. If a segment name is specified, GNP or GHNP calls with an unqualified SSA are issued.

Prerequisite: Before specifying the GNP or the GHNP statement, you must establish a valid parentage by specifying a GN, GHN, GNR, GHNR, GU, or GHU statement.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

GNP GE          seg_name
GHNPGE          seg_name
    n           seg_name
```

Position
Description

1
Code the GNP or the GHNP keyword to identify this as a get-next-in-parent control statement or a get-hold-next-in-parent control statement.

5
Code one of the following optional keywords:

GE
FABHTEST issues GNP or GHNP calls until the end of the segment occurrence under the current parent.

n
The number of GNP or GHNP calls that FABHTEST issues. Code a number containing up to 10 digits, left-aligned. Leading zeros are not necessary.

Blank
FABHTEST issues 1 GNP or GHNP call.

16
Code one of the following optional keywords:

seg_name
Code the name of a segment. FABHTEST issues GN or GHN calls with an unqualified SSA.

Blank
FABHTEST issues GN or GHN calls without an SSA.

GNR and GHNR control statement

These statements instruct FABHTEST to issue a specified number of GN or GHN root calls with unqualified SSAs.

```
0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

GNR GB
GHNRGB
    n
```

Position
Description

1
Code the GNR or the GHNR keyword to identify the get-next-root call control statement or the get-hold-next-root call control statement.

5
Code one of the following optional keywords:

GB
FABHTEST issues GNR or GHNR root calls until the end of the database is reached.

n
The number of GNR or GHNR calls that FABHTEST issues. Code a number containing up to 10 digits, left-aligned. Leading zeros are not necessary.

Blank
FABHTEST issues 1 GNR or GHNR call.

GU and GHU control statement

The Get Unique statement instructs FABHTEST to issue GU or GHU calls.

The statement can specify:

- Whether the GU or GHU should be issued with or without an SSA
- The relational operator to be used in the SSA
- The key value to be used in the SSA

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

GU  nbr        = keyvalue.............................................c
GHU nbr        = keyvalue.............................................c
                 ..............
```

Position
Description

1
Code the GU or the GHU keyword to identify the get-unique control statement or the get-hold-unique control statement.

5
This entry lists the number of times the call is to be repeated. Code a number containing up to 10 digits. This value neither requires leading zeros nor has to be aligned. If only one GU or GHU is to be issued, omit this step.

16
Code either a blank or a valid relational operator. SSA relational operators are restricted to =*b*, *b*=, EQ, =>, >=, GE (where *b* represents a single blank).

If this field is blank, FABHTEST issues GU or GHU calls without SSA. Otherwise, it issues GU or GHU calls with SSA qualified on the key field of the root segment and uses the relational operator provided.

18
Code the root key value. If the key value does not fit in this statement, place a continuation character (*c* in this example) in column 72. Then complete the key value in the continuation statement.

If you continue your key value to the next line, leave columns 1 - 4 blank and begin the continuation at column 16. You can continue the statement again if you enter a continuation character in column 72.

72
Enter any nonblank character if a continuation is required.

Leave this space blank if the key value is completed.

PCB control statement

Use this optional statement to select the database PCB for the database call.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

PCB pcbnumber  dbdname
```

Position
Description

1
Code the PCB keyword to identify this statement as a PCB statement.

5
Code the PCB number left-aligned. The first PCB is number 1.

If no PCB control statement is provided, FABHTEST uses the first database PCB.

16
Code the *dbdname* of PCB.

If no *dbdname* is specified on the PCB control statement, FABHTEST uses the PCB number field. If a *dbdname* is specified, the first database PCB referring to that DBD is used.

REPL control statement

The REPL statement instructs FABHTEST to issue a REPL call without SSA. (FABHTEST does not change the content of the segment during a REPL call.)

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

REPL
```

Position
Description

1
Code the REPL keyword to identify the replace control statement.

Notes:

- REPL call is supported for compatibility with DBT HSSR and PO HSSR (see Chapter 30, "Compatibility with DBT V2 HSSR," on page 357 and Chapter 32, "Compatibility with PO HSSR," on page 367).
- REPL calls are not allowed for PHDAM or PHIDAM databases.

FABHTEST HSSROPT input data set

The HSSROPT data set for the FABHTEST utility contains the control statements for HSSR Engine.

Any of the HSSROPT options that are appropriate to your task can be used. The CO, TRHC, and TRDB control statements must always be included, unless a performance test is being conducted. Some HSSROPT control statements that might be useful when used with FABHTEST are as follows:

Table 34. HSSROPT control statements for FABHTEST

HSSROPT control statement	Description
CO	The compare option reissues HSSR calls as DB/I calls. Always include the HSSR CO (compare) control statement and the hardcopy trace control statement in this data set, unless you conduct a performance test.
TRHC	The hardcopy tracing option provides call data, HSSR control block data, buffer handler data, and CAB control block data. To obtain a printout of all database calls issued against an HSSR PCB, include the control statement.
TRDB	Specifies the DBDs against which calls are to be traced.

Note: For a complete description of the HSSROPT control statements, see Chapter 11, "Options for HSSR Engine," on page 155.

FABHTEST HSSRCABP input data set

The HSSRCABP data set for the FABHTEST utility contains the control statements for the buffer handler of HSSR Engine.

Any of the HSSRCABP options that are appropriate to your task can be used. Some HSSRCABP control statements that might be useful when used with FABHTEST are summarized in the following table.

Table 35. HSSRCABP control statements for FABHTEST

HSSRCABP control statement	Description
CABDD	Specifies data sets to which the succeeding CAB control statements apply.

Table 35. HSSRCABP control statements for FABHTEST (continued)

HSSRCABP control statement	Description
NBRDBUF	Specifies the number of single blocks or CIs read in direct mode that are to reside in the buffer for look-aside buffering.
NBRSRAN	Specifies the number of ranges to be buffer-resident.
OVERFLOW	Enables chained sequential I/O in the HDAM overflow area or in the overflow area of each partition of PHDAM database.
PARTPROC	Specifies the processing intent. This statement is valid only for a PHDAM or PHIDAM database.
RANSIZE	Specifies the number of blocks or CIs to be read together in chained mode.
REFT4	Specifies a threshold value used to determine when chained sequential I/O or direct I/O is to be performed.

Note: For a complete description of the HSSRCABP control statements, see "HSSRCABP control statements" on page 212.

FABHTEST output: SYSPRINT output data set

Output from the FABHTEST utility is the FABHTEST Control Statements report that is generated in the SYSPRINT data set.

In addition to the report generated by the FABHTEST utility, reports and statistics produced by HSSR Engine are written in HSSRSTAT and HSSRTRAC data sets. For the reports generated by HSSR Engine, see Chapter 12, "Reports and output from HSSR Engine," on page 181.

Format

The data set contains 133-byte fixed-length records. When the block size is coded in the JCL, the block size must be a multiple of 133.

FABHTEST Control Statements report

This printed report contains a printed copy of the input control statements read by FABHTEST from the SYSIN data set.

The following figure shows an example of this report

```
IMS HIGH PERFORMANCE UNLOAD                          "FABHTEST CONTROL STATEMENTS"                          PAGE:     1
5655-E06                                          DATE: 06/01/2021  TIME:  8.45.00                    FABHB15  - V1.R2

.........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

GHU            GE0000000500
GHNR
GHN
GU
GN  5
GU             EQ0000001500
GNR 4
GHU
REPL
GHN
REPL
GN  GB
```

Figure 52. FABHTEST Control Statements report

FABHTEST JCL examples

Use the following JCL examples to prepare your FABHTEST JCL.

Subtopics:

Example 1: Using FABHTEST for problem determination

To do problem determination with FABHTEST, you can use the JCL shown in the following figure.

```
//TEST   EXEC FABHDLI,MBR=FABHTEST,PSB=USERPSB
//SYSIN    DD *
GU            EQ1045699
PCB 2
GN  GB
/*
//HSSROPT  DD *
HSSRPCB *ALL
CO
TRHC CB,CALL,BUF,BUFCB
TRDB *ALL
/*
//TESTHDAM DD DSN=TESTDB.HDAM,DISP=SHR
//TESTHIIX DD DSN=TESTDB.INDEX,DISP=SHR
//TESTHIDA DD DSN=TESTDB.HIDAM,DISP=SHR
//SYSPRINT DD SYSOUT=A
```

Figure 53. FABHTEST JCL for problem determination

SYSIN control statements request that FABHTEST:

- Issue a GU call. The first database PCB is used with the relational operator EQ and the key value provided.
- Sequentially read the database referred to by the second PCB, until the end of the database is reached.

HSSROPT control statements identify the options to be activated:

- A CO control statement activates the compare option.
- A TRHC and a TRDB control statement request HSSR Engine to trace control blocks, call information, and buffer pool information for calls issued against all HSSR PCBs.

IMS databases are identified by the TESTHDAM, TESTHIIX, and TESTHIDA DD statements. A SYSPRINT DD defines the FABHTEST output data set.

Example 2: Using FABHTEST to test performance

To test performance with FABHTEST, you can use the JCL shown in the following figure.

```
//TEST   EXEC FABHULU,MBR=FABHTEST,DBD=USERDBD
//HSSROPT  DD *
CABSTAT YES
/*
//SYSIN    DD *
GN GB
/*
//TESTDB   DD DSN=TESTDB.HIDAM.OSAM,DISP=SHR
//TESTIDX  DD DSN=TESTDB.HIDAM.INDEX,DISP=SHR
//SYSPRINT DD SYSOUT=A
```

Figure 54. FABHTEST JCL for performance testing

The SYSIN control statement requests FABHTEST to sequentially retrieve the entire database. 'CABSTAT YES' is specified in HSSROPT DD to produce the detailed CAB Statistics report. The TESTDB and TESTIDX DD statements identify an HIDAM database.

Code HSSRCABP DD statement to tune CAB buffering parameters. Statistical reports can be analyzed to validate performance.

Chapter 17. Buffer handler simulation utility (FABHBSIM)

FABHBSIM is the buffer handler simulation utility that is used as an aid in tuning the BB and CAB buffer handlers.

This utility enables you to observe the effect of the changes to parameters of the buffer handlers, without actually performing database I/O operations and segment processing. FABHBSIM might help realize significant productivity gains by determining the optimum numbers and sizes of buffers.

With FABHBSIM, you can simulate a previous run of an IMS High Performance Unload job and obtain standard reports produced by HSSR Engine. You can use the FABHBSIM utility to do the following tasks:

- Analyze buffer handler performance
- Tune buffer handler parameters
- Assist in improving performance of your IMS application programs
- Aid in improving productivity of your IMS database

The following features are provided by FABHBSIM:

- FABHBSIM simulates database I/O and buffer handling. It produces statistical reports that show the results of the simulation.
- FABHBSIM reads the HSSRBUTR buffer handler trace data set that is created during the previous run of an IMS High Performance Unload job. This trace contains a record of all HSSR calls that were issued to an IMS database during the execution of your application program.
- FABHBSIM allows you to use an HSSR buffer handler other than the one used in the original run.
- FABHBSIM reissues all of the database calls. CAB or BB processes the HSSR calls, but no actual database I/O is performed. FABHBSIM produces the statistical reports that are normally generated by HSSR Engine. From these reports, you can analyze the effect of parameter changes on buffer handler performance.

FABHBSIM runs as an HSSR application program. The utility accepts the HSSRBUTR data set as the input and produces output reports. Any of the three cataloged procedures can be used to run FABHBSIM.

Topics:

FABHBSIM restrictions

Certain restrictions apply when using the FABHBSIM utility.

FABHBSIM has the following restrictions:

- The PSB and DBDs used must be identical with those used in the original run that was traced. Do not modify the PSB or the DBD between the traced run and the execution of FABHBSIM. The database itself can be modified with ISRT, DLET, or REPL, and reorganization activities.
- The timing estimates of CAB I/O provided on the CAB Statistics report are not accurate when FABHBSIM is run.
- FABHBSIM does not support the tuning of buffer handlers for a PHDAM or a PHIDAM database.

Running FABHBSIM to simulate the buffer handler

You can simulate the performance of the buffer handler by running the FABHBSIM utility.

Procedure

Complete the following steps to run the FABHBSIM utility.

1. Use one of the FABHDLI, FABHDBB, or FABHULU procedure.
2. Code the HSSROPT and HSSRCABP control statements.
3. Run FABHBSIM as an HSSR application program.
4. Review and analyze output reports to tune buffer handler parameters.
5. Repeat steps "2" on page 240 through "4" on page 240 if further tuning is necessary.

FABHBSIM JCL requirements

FABHBSIM runs as an HSSR application program and, therefore, must meet the requirements for the basic JCL (FABHX034 JCL). In addition, FABHBSIM JCL requires other DD statements.

Prerequisite: See "Basic JCL requirements" on page 30 for the basic (FABHX034) JCL requirements.

The following table summarizes additional JCL requirements for FABHBSIM.

Table 36. FABHBSIM DD statements

DDNAME	Use	Format	Need
SYSUT1	Input	HSSRBUTR data set	Required

EXEC

This statement invokes the procedure FABHDLI, FABHDBB, or FABHULU. The EXEC statement must be in one of the following formats:

```
//  EXEC FABHDLI,MBR=FABHBSIM,PSB=psbname
//  EXEC FABHDBB,MBR=FABHBSIM,PSB=psbname
//  EXEC FABHULU,MBR=FABHBSIM,DBD=dbdname
```

SYSUT1 DD

This required input data set defines the data set for buffer handler trace. You must create the buffer handler trace data set in an earlier run of your IMS High Performance Unload job. It is the data set that was defined by the HSSRBUTR DD statement in the earlier run. Here is an example of the format for this data set:

```
//SYSUT1 DD DSN=HSSRBUTR,DISP=OLD,UNIT=tape,VOL=SER=xxxxxx
```

FABHBSIM input

FABHBSIM uses data sets of control statements and buffer handler trace information as input. Input for the FABHBSIM utility consists of two data sets: the HSSROPT data set and the HSSRCABP data set.

FABHBSIM HSSROPT input data set

The HSSROPT data set for the FABHBSIM utility contains the control statements for HSSR Engine.

The pertinent HSSROPT option to be used with FABHBSIM is the BUF control statement. It modifies the number of BB buffers.

Any of the other HSSROPT options that are appropriate to your task can be used. But the DBDL1 control statement with the *ALL keyword must not be used.

For a complete description of the HSSROPT control statements, see Chapter 11, “Options for HSSR Engine,” on page 155.

FABHBSIM HSSRCABP input data set

The HSSRCABP data set for the FABHBSIM utility contains the control statements for the buffer handler of HSSR Engine.

Restriction: The PARTPROC control statement is not supported, because FABHBSIM does not support PHDAM and PHIDAM databases.

For a complete description of the HSSRCABP control statements, see “HSSRCABP control statements” on page 212.

FABHBSIM output: HSSRSTAT output data set

FABHBSIM produces the standard HSSRSTAT data set.

The reports produced in the HSSRSTAT data set are the primary output from FABHBSIM. These reports are produced by the HSSR Engine. For details about these reports, see Chapter 12, “Reports and output from HSSR Engine,” on page 181.

FABHBSIM JCL example

Use the following JCL examples to prepare your FABHBSIM JCL.

To use FABHBSIM in simulating buffer handlers and tuning buffers, use the JCL shown in the following figure.

```
//BSIM   EXEC FABHULU,MBR=FABHBSIM,DBD=USERDBD
//HSSROPT  DD *
CABSTAT YES
/*
//HSSRCABP DD *
CABDD *HD
NBRSRAN 30
/*
//SYSUT1   DD DSN=TESTDS.HSSRBUTR.DATASET,DISP=OLD,UNIT=tape,VOL=SER=yyyyyy
//HDAM     DD DSN=TESTDS.HDAM,DISP=SHR
```

Figure 55. FABHBSIM JCL for simulating a buffer handler

The CABDD control statement specifies that the succeeding CAB control statements apply to all HD databases. The NBRSRAN control statement specifies that 30 ranges should reside in the buffer for look-aside buffering purposes. (Other HSSRCABP control statements can be inserted as appropriate.) 'CABSTAT YES' in HSSROPT DD requests that HSSR Engine produce detailed CAB Statistics report.

The SYSUT1 DD statement defines an input data set, which is the HSSRBUTR data set produced by an earlier run of an IMS High Performance Unload job.

The HDAM DD statement defines an IMS database.

Statistical reports can be analyzed to tune the CAB buffering parameters.

Chapter 18. System programming interfaces

IMS High Performance Unload provides system programming interfaces for customization or for compatibility with earlier products.

This topic presents product-sensitive programming interface information. See “Programming interface information” on page 518 to understand the restrictions associated with this type of material.

PSPI

The following system programming interfaces are provided:

- Runtime environment exit (FABHRTEX)
- Buffer-handler initialization exit (FABHCEX)
- Return code edit exit (FABHRCEX)
- Record-formatting exit for FABHURG1

PSPI

Topics:

Runtime Environment exit (FABHRTEX)

With IMS High Performance Unload, you can develop a runtime environment exit routine, named FABHRTEX, which enables you to do your own initialization and termination processing for your application programs or your exit routines for unload utilities.

PSPI

The runtime environment exit routine is called during the initialization of the IMS High Performance Unload program controller before the application program is called; the routine is called once more after, control is returned from the application program to the program controller.

Notes:

- For details of system structure, see “IMS High Performance Unload system structure” on page 11.
- The runtime environment exit is not invoked when the language environment option is specified on the EXIT control statement of the FABHURG1 unload utility or the PSB control statement of the FABHFSU unload utility.

You can use an exit routine that has a different name, by specifying the RTEXIT control card in the HSSROPT data set. Use this control statement if you need to set up a special runtime environment for a special application or for an exit routine for FABHURG1 or FABHFSU.

The following information message is issued only if you provide your own runtime environment exit routine:

```
FABH0826I: RUN TIME ENVIRONMENT EXIT ROUTINE IS
           BEING INVOKED, MODULE=xxxxxxxx
```

If the specified exit is not found, abend U4013 occurs and the following message is issued:

```
FABH0850E LOAD FAILED FOR RTEXIT (xxxxxxxx), CC=yyyy, RC=zz
```

PSPI

The following Product-Sensitive Programming Interface explains the interface to runtime environment exit routine.

PSPI

The following table shows what the registers contain on entry to the routine.

Table 37. Register contents upon entry to a user exit

Register	Contents
1	Address of call parameter list
14	Return address to the caller
15	Entry point address of the runtime exit routine

The address of the parameter list is set in register 1; the following table shows the parameters passed to the routine.

Table 38. Parameters passed to the routine

Word	Content
1	Pointer to a 4-byte character field that contains 'INIT' (for initialization call) or 'TERM' (for termination call)
2	Pointer to an 8-byte character field that contains the application program name

When control is returned to the caller, the contents of all registers except register 15 must be restored. Register 15 must contain a return code. The meanings of the return codes are provided in the following table.

Table 39. FABHRTEX return codes

Value	Description
0	Initialization or termination was successful.
Not 0	An error occurred in the runtime environment exit. Error message FABH0827E is issued, and abend U4013 occurs.

Tip: A dummy runtime environment exit routine (FABHRTEX) that returns the return code of 0 at both initialization and termination calls is provided. You can code your own FABHRTEX to meet your particular requirements.

PSPI

Buffer Handler Initialization exit (FABHCEX)

Some users need to reduce the amount of batch processing during peak online periods. You can develop an exit routine named FABHCEX, which can disperse the amount of system resources needed by IMS High Performance Unload jobs over a longer elapsed time.

PSPI

FABHCEX is invoked when the HSSR buffer handler is initialized. It can dynamically allow or disallow the use of CAB. If it detects that the IMS High Performance Unload job step is running during a peak online period, it disallows the use of CAB and enforces the use of BB (regardless of specifications in HSSRCABP data set).

The return codes that FABHCEX can set in Register 15 are listed in the following table.

Table 40. FABHCEX return codes

Code	Description
0	Choose the buffer handler according to specifications of the CAB control statements in HSSRCABP.
Not 0	Use BB.

For example, FABHCEX can be used to check the time of day and whether the IMS online system is running. After making these determinations, FABHCEX issues a return code and selects a buffer handler.

The conventions for linkage between HSSR Engine and FABHCEX are the standard MVS linkage conventions. No parameters are passed to FABHCEX. The FABHCEX routine must be a load module named FABHCEX, and must be in a program library accessible to HSSR Engine.

The program control is transferred to the routine in the addressing mode of the routine.

Tip: IMS High Performance Unload provides a dummy routine (FABHCEX), which always returns a return code of 0. You can code your own FABHCEX exit routine to meet your particular requirements.

PSPI

Return Code Edit exit (FABHRCEX)

You can write a Return Code Edit exit routine (FABHRCEX) to change the return codes of HSSR application programs, including the FABHURG1 and FABHFSU unload utilities.

FABHRCEX is called before control is returned from the IMS High Performance Unload program controller (FABH000) to the IMS region controller (DFSRRC00).

Note: For details of system structure, see "IMS High Performance Unload system structure" on page 11.

If you want to use the exit, you must write an exit routine in Assembler, and then assemble and link-edit it as load module FABHRCEX. The library that contains FABHRCEX must be specified in the STEPLIB DD concatenation of your IMS High Performance Unload job.

The addressing mode of a Return Code Edit exit routine can be either 24 or 31. The residency mode can be either 24 or ANY. The reusability attribute can be either REUS or RENT.

The following informational message is issued only if the IMS High Performance Unload program controller can find FABHRCEX in the libraries specified in the STEPLIB DD statement, and load it:

```
FABH0881I applname ENDED WITH RC=xx, WHICH MIGHT BE CHANGED BY FABHRCEX EXIT
```

If the IMS High Performance Unload program controller fails to load FABHRCEX, abend U4013 occurs and the following error message is issued:

```
FABH0854E LOAD FAILED FOR FABHRCEX EXIT, CC=xxxx, RC=yy
```

Subtopics:

- "Interface to Return Code Edit exit routine" on page 246
- "FABHRCEG sample JCL" on page 246

Interface to Return Code Edit exit routine

The following table shows what the registers contain on entry to the routine.

Table 41. Register contents upon entry to FABHRCEX

Register	Contents
1	Address of call parameter list
14	Return address to the caller
15	Entry point address of the Return Code Edit exit routine

The address of the parameter list is set in register 1; the following table shows the parameters passed to the routine.

Table 42. FABHRCEX parameters

Word	Content
1	Pointer to an 8-byte character field that contains the application program name
2	Pointer to a 4-byte field that contains the return code to be edited

When control is returned to the caller, the contents of all registers except register 15 must be restored.

FABHRCEG sample JCL

FABHRCEG is a sample JCL stream for use in creating a Return Code Edit exit routine. It is provided as a member of the HPS.SHPSSAMP library. FABHRCEG assembles and link-edits the FABHRCEX exit routine in the HPS.SHPSLMD0 load module library.

User record-formatting routine

If you want to perform special processing or editing during the database unload, or if you want to use the database unload format of your own, you can write a user record-formatting routine.

PSPI

The FABHURG1 unload utility ordinarily runs without any user routines. It unloads a database into one of six different formats by invoking one of six standard record-formatting routines.

Some installations might want to perform additional processing or editing during the database unload, or might want to create a database unload format of their own. To do this, they can provide their own user record-formatting routine or optional user exit routine.

These exit routines can be coded in Assembler or COBOL language.

PSPI

Logic of FABHURG1

The database unload processing is performed by the common logic, a user-selectable record-formatting routine, and an optional user exit routine. The user-selectable record-formatting routine can be any one of six standard record-formatting routines or a user record-formatting routine.

Subtopics:

PSPI

Common logic

The common logic performs the following processes:

1. Provides initialization and termination processing.
2. Controls an optional SYSUT2 output data set that contains the database unload output (OPEN, CLOSE, and WRITEs are issued by the common logic).
3. Issues HSSR calls against the PCB you select.
4. Edits a call parameter list for record-formatting routines and optional user exit routines.
5. Calls the selected record-formatting routine. If the record-formatting routine sets a return code other than zero, the next HSSR call is issued.
6. Calls the optional user exit routine. If the routine sets a return code other than zero, the next HSSR call is issued.
7. If a non-DUMMY SYSUT2 DD statement has been provided, issues a PUT to write the record edited by the record-formatting routine and the optional user exit routine.
8. If a non-DUMMY SYSUT3 DD statement has been provided, issues a WRITE macro to write the block edited by the record-formatting routine and optional user exit routine.
9. Issues the next HSSR call.

Record-formatting routine

The record-formatting routine is invoked each time a database segment has been retrieved by the common logic. It is the responsibility of the record-formatting routine to edit output records from the retrieved database segments. FABHURG1 provides six standard record-formatting routines. System programmers can provide their own user record-formatting routines if they want to create a database unload format of their own.

After having reached the database end, the common logic invokes the record-formatting routine one last time so that it can do its own termination processing and cleanup processing. You must check whether the call is the last call by checking the status code in the PCB feedback area of the HSSR PCB (see "Parameter 4: HSSR PCB" on page 251).

Optional user exit routine

The optional user exit routine is invoked (after record-formatting processing) each time a database segment that the record-formatting routine does not skip is retrieved.

The optional user exit routine can modify or edit the record composed by the record-formatting routine. One possible use of user exit routines by system programmers is to build the logical parent's concatenated key.

Both record-formatting routines and user exit routines can optionally perform the following actions:

- Create their own output data sets (in addition to or instead of SYSUT2).

- Issue DL/I calls or HSSR calls against PCBs other than the PCB that is used by the common logic of FABHURG1.
- Indicate that the current database segment or database record should not be processed further and that the skipped database segments should not be written to SYSUT2. They can also indicate that the common logic should resume its retrieval at a root with a specifiable key.

After having reached the database end, the common logic invokes the optional user exit routine one last time so that it can do its own termination processing and cleanup processing. For example, all opened files can be closed when the last call is issued. You must check whether the call is the last call by checking the status code in the PCB feedback area of the HSSR PCB (see "Parameter 4: HSSR PCB" on page 251).

PSPI

Interface to user record-formatting and optional user exit routines

These routines are called, in accordance with standard Assembler and COBOL conventions.

PSPI

On entry, the routines should save the registers; on return, they should restore all registers except Register 15. On entering to the routines, the following registers contain the information provided in the following table.

Table 43. Register contents at entry to routines

Register	Contents
1	Address of call parameter list
13	Address of caller's save area
14	Return address to database unload utility
15	Entry point address into user routine

Upon returning to the common logic, Register 15 must contain a binary return code 0 - 4. The codes are explained in the following table.

Table 44. Exit routine return codes

Code	Description
0	Processes this database segment.
1	Stops processing of this database segment and does not write it to SYSUT2. Retrieves the next data-sensitive segment.
2	Stops processing of this database segment and does not write it to SYSUT2. Retrieves the next database root segment. (All remaining segments of the current database record are skipped.)
3	Stops processing of this database segment, and does not write it to SYSUT2. Continues database retrieval with the root whose key is greater than or equal to the key value specified by the user exit routine in call parameter 9 (the key of the next root). If a Data Conversion exit routine is used for the database, the key of the next root must be specified in the application form.
4	Stops the processing of this database segment, and does not write it to SYSUT2. Does not retrieve any further database segments, and stops processing.

If a routine sets a return code 1, the dependents of the current database segment are not skipped by the common logic. Skipping of these dependent segments is the responsibility of the routine.

PSPI

Call parameters

This reference topic explains the call parameters for user record-formatting routines or optional user exit routines.

Subtopics:

PSPI

Parameter 1: OUTPUT-AREA

The following list describes the OUTPUT-AREA for the user record-formatting routine and optional user exit routines.

OUTPUT-AREA for user record-formatting routine

OUTPUT-AREA contains (at the offset specified by the OFFS utility control statement) the segment data as returned by the HSSR call.

- If a SYSUT2 DD statement has been provided and is not a dummy, OUTPUT-AREA is in the SYSUT2 output buffers. SYSUT2 is a variable-blocked sequential data set. Unless the user record-formatting routine sets a return code that is not zero, the routine should build a variable-length SYSUT2 record in the OUTPUT-AREA, storing the binary record length in the first 2 bytes and binary zeros in the 2 bytes that follow. The SYSUT2 record is written by the common logic.
- If no SYSUT2 DD statement (real or dummy) has been provided, all output operations, including open and close, are done by the routine. The routine can use OUTPUT-AREA to build its output records, or it can use its own area to build its output record. In the former case, the overhead of data movement within virtual storage is reduced. In the latter case, it is the responsibility of the routine to develop a method to pass the address of the output record to the optional user exit routine.

OUTPUT-AREA for optional user exit routine

OUTPUT-AREA contains the output record as it is built by the record-formatting routine. Some user-developed record-formatting routines can build the output record into areas other than the OUTPUT-AREA.

Warning: The header field of the *HD unload format record for HALDB is longer than the one for non-HALDB. The offset of the segment data is different between non-HALDB and HALDB. To refer to the segment data, it is recommended that you use “Parameter 2: Database segment (Segment data)” on page 250 than to use “Parameter 1: OUTPUT-AREA” on page 249.

The header field contains the segment name, but it is not recommended that you use it. Instead, refer to the segment name contained in the HSSR PCB, which is pointed to by "Parameter 4: HSSR PCB" on page 251.

The following figure provides an example of a part of an exit routine that is coded in COBOL.

```
ENVIRONMENT DIVISION.
   :
DATA DIVISION.
   :
WORKING-STORAGE SECTION.
   :
LINKAGE SECTION.
    01  OUTPUT-AREA            PIC X(XXX).
    01  SEGMENT-DATA           PIC X(XX).
    01  SEGMENT-PREFIX         PIC X(XX).
    01  HSSR-PCB.
        02  HPCB-DBDNAME       PIC X(08).
        02  HPCB-SEGLEV        PIC X(02).
        02  HPCB-STATUS        PIC X(02).
        02  HPCB-PROCOPT       PIC X(04).
        02  FILLER             PIC X(04).
        02  HPCB-SEGNAME       PIC X(08).
        02  HPCB-LENKFB        PIC X(04).
        02  HPCB-NUSENS        PIC X(04).
        02  HPCB-KEYFB         PIC X(XX).

PROCEDURE DIVISION USING OUTPUT-AREA,SEGMENT-DATA,SEGMENT-PREFIX,HSSR-PCB
   :
```

Figure 56. A user exit routine for FABHURG1 in COBOL

Parameter 2: Database segment (Segment data)

This parameter contains the pointer to the database segment as returned by the HSSR call. If a standard record-formatting routine is used, the segment data is after the header field that is pointed to by parameter 1. If a user-developed record-formatting routine that can be specified in the FRMT control statement is used, the segment data is within the OUTPUT-AREA at the offset specified by the OFFS control statement. The user-developed record-formatting routines might modify the database segment. In such a case, the optional user exit routine, if it is used, sees this modified data instead of the database segment as returned by the HSSR call. You need to be careful when you modify the database segment because there are other segments in the OUTPUT-AREA.

If the USEGMAX control statement is specified, the work area for segment editing is reserved after the segment data. The total length of the segment data area plus the work area is the length specified by the USEGMAX statement. If the ULEN or OFFS control statement is specified, the work area that is pointed to by Parameter 1 and whose length is equal to the length specified by the OFFS statement is reserved to be used as your record header. The work area whose length is equal to ULEN minus OFFS is reserved after the segment data.

You can use these work areas in editing or expanding the segment data, but if you change the length of the segment data, you must also update the fields in the OUTPUT-AREA that are affected by that change of length. For example, the first two bytes of the OUTPUT-AREA pointed to by Parameter 1 must be updated with the new record length, if the record is written to the SYSUT2 data set. If your segment record header has a segment length field, it must be updated. If the segment itself has fields for segment length or field lengths, they must be updated, too. The exit routine is responsible for updating these fields.

If FABHURG1 is run with the DECN option, the segment data for which a Segment Edit/Compression routine is specified is passed to your exit routine in the compressed format. If you want the segment data to be passed to your exit routine in the decompressed format, run FABHURG1 with the DECY option.

Parameter 3: Segment prefix

This parameter contains the segment prefix as it is stored in the database. The user routine must not modify this parameter.

Note: During a migration unload, this parameter contains the binary zero for a virtual logical child.

Parameter 4: HSSR PCB

This parameter contains the HSSR PCB used by the main logic of FABHURG1 to sequentially retrieve the database. The HSSR PCB contains the segment name, the segment level, the key feedback area, the length of the key feedback area, and the status code.

Note: For information about HSSR PCB, see "HSSR PCB feedback information" on page 81.

The user routine must always test the status code. If the status code is GB, the end of the database has been reached and the user routines should perform their termination processing and cleanup processing, including closing all files opened by the routine.

The user routines must not modify the HSSR PCB.

Parameter 5: HSDB

This parameter contains the HSSR segment descriptor block that describes the currently retrieved segment type.

The user routine must never modify the HSDB. It can, however, use the 4-byte field HSDBUSER.

Parameter 6: Reserved for system use

The user routine must never refer to or modify this parameter.

Parameter 7: RBA of segment prefix

This 4-byte field contains the relative byte address of the segment prefix. The content of this field has a meaning only for HD databases. The user routine must never modify this field.

If your IMS environment supports an OSAM database larger than 4 gigabytes, be careful with the RBA. You must check the flag byte SPRBAFLG when you treat an RBA. The flag SPRBAX4G in this flag byte is on if and only if all of the following conditions are satisfied:

- Your IMS environment supports relative byte addressability for up to 8 gigabytes of data in an OSAM, HDAM, HIDAM, PHDAM, or PHIDAM database data set.
- Your database is an OSAM database.
- Your database has an even block size.
- The RBA of the segment prefix is larger than or equal to 4 gigabytes.

Thus, if your IMS environment satisfies the first condition, you must check whether the flag SPRBAX4G is on or off. If the flag is on, the value in SPRBA is not the real RBA. In this case, you must move bit 1 of SPRBA to bit position 33 to get the real 33-bit RBA. See the sample code provided in the following figure.

```
           LR    R5,R1         GET PARAMETER LIST ADDRESS
           USING PRMPRML,R5
           USING SPRBAD,R7
           L     R7,PRMARBA
           L     R9,SPRBA      GET RBA OF SEGMENT PREFIX
           SR    R8,R8
           IF    (TM,SPRBAFLG,SPRBAX4G,O)
             LA    R8,1
             XR    R9,R8
           ENDIF
           DROP  R7
           STM   R8,R9,WKREAL  GET REAL 33-BIT RBA
                 :
*
WKREAL   DC    2F'0'         REAL RBA OF SEGMENT PREFIX
*
PRMPRML  DSECT
PRMAREC  DC    A(0)          A(START OF OUTPUT RECORD)
PRMASGD  DC    A(0)          A(SEGMENT-DATA)
PRMASGP  DC    A(0)          A(SEGMENT-PREFIX)
PRMAPCB  DC    A(0)          A(HPCB)
PRMAHSD  DC    A(0)          A(HSDB)
PRMATCB  DC    A(0)          A(HTCB)
PRMARBA  DC    A(0)          A(RBA OF SEGMENT PREFIX)
PRMALEN  DC    A(0)          A(SEGMENT DATA LENGTH)
PRMANXK  DC    A(0)          A(KEY OF NEXT ROOT)
         DC    4A(0)
*
SPRBAD   DSECT
SPRBA    DS    F             RBA OF SEGMENT PREFIX
SPRBAFLG DS    XL1           FLAG BYTE
SPRBAX4G EQU   X'80'         RBA >= 4GB
SPHALDB  EQU   X'10'         HALDB
SPMACTV  EQU   X'08'         ODD RBA(SGRBA) ON M-V SIDE
```

Figure 57. RBA of segment prefix

If your IMS environment supports the HALDB Online Reorganization (OLR), you must check whether the SPHALDB flag and the SPMACTV flag, which are shown in the preceding figure, are on. If both of them are on, the value in SPRBA is not the real RBA and the odd RBA indicates that the segment is on the M-V,Y set of the data sets. In this case, you must move bit 1 of SPRBA.

Note: During a migration unload, this parameter contains the binary zero for a virtual logical child.

Parameter 8: Length of segment data

This 2-byte binary field contains the length of the segment data that is returned by the HSSR call. The user routine must never modify this field.

Parameter 9: Key of next root

On entering the user routine, the content of this field is unpredictable. If the user routine sets a return code of 3, the routine must store the key value in this field. The key value is used to retrieve the next root segment. If the secondary index is used to retrieve the root segments, specify the value of the search field of the index segment as the key value. With this key value, the common logic then issues a GU call with the "greater than or equal to" operator.

Note: If you are using a Data Conversion exit (DFSDBUX1) for the database in your FABHURG1 job, you must specify the next root key in the application form. On the other hand, if you are not using DFSDBUX1 in your FABHURG1 job, you must specify the next root key in the stored form.

Parameters 10–13

These parameters are reserved.

Parameters 14–n

PCBs in the same sequence as specified during PSBGEN.

With the exception of the PCB used by the common logic, these PCBs can be used by user routines to issue HSSR and DL/I calls.

PSPI

Special-purpose SYSIN control statements for user exits

You can specify the user record-formatting routine and the optional user exit routine by coding the control statements in the SYSIN data set.

PSPI

The name of a user record-formatting routine is specified by the FRMT control statement. The name of an optional user exit routine is specified by the EXIT control statement. For these user exit routines, the following special-purpose SYSIN control statements are provided:

- OFFS
- ULEN
- USEGMAX

The OFFS and ULEN control statements have meaning only if a user record-formatting routine (specified on the FRMT control statement) is active. These statements specify that the additional space is to be reserved within the area used by FABHURG1 to issue HSSR calls. This additional space can be used by the record-formatting routine to build additional header data in its output records in this area.

The USEGMAX control statement has meaning only if one of the standard record formats, *HD, *CS, *F1, or *F2, is used, and if an optional user exit routine (specified on the EXIT control statement) is active. This statement specifies that the additional space is to be reserved within the I/O area used by FABHURG1 to issue HSSR calls. This additional space can then be used by the user exit routine to edit the segment data. See “Parameter 2: Database segment (Segment data)” on page 250.

PSPI

EXIT control statement

This optional control statement specifies the name of the optional user exit routine.

PSPI

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

EXIT exitname c l
```

Position
Description

1
Code the EXIT keyword to identify the exit routine.

6
The left-aligned load module name of the user exit routine. If no EXIT control statement is provided, no user exit routine is invoked.

15
If a Data Conversion exit routine is used, the user exit routine receives the segment data that has been converted from the stored form to the application form.

The 1-character entry *c* indicates whether the inverse conversion (the conversion from the application form to the stored form) is to be done before the segment data edited in the exit routine is written into the output data set.

Use one of the following codes:

Y
Do the conversion.

The option Y is valid only for *HD unload format.

This option is valid only when the option DATXEXIT YES is specified in the HSSROPT data set.

N | blank
Do not do the conversion. N is the default.

17
Language environment option

L
Indicates that the user exit routine runs in the Language Environment (LE) using the CEEPIPI invocation.

This option is effective when the user exit routine is written with Enterprise COBOL for z/OS. This option is not effective for user exit routines written in assembler language.

This option is mutually exclusive with the RTEXIT control statement. If you specify this option, the runtime environment exit routine specified for the RTEXIT control statement is not invoked.

Restriction: If the EXIT control statement is specified and one or more partitions of PHDAM or PHIDAM are in the HALDB OLR cursor-active status, FABHURG1 ends abnormally.

PSPI

FRMT control statement

This optional control statement specifies the output format of the unloaded database. Use this statement only when specifying a format other than *HD.

PSPI

```
0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

FRMT frmt
```

Position
Description

1
Code the FRMT keyword to identify the format control statement.

6
Code the output format name, which can be either of the following names:

- The name of one of the standard formats provided by the utility: *HD, *CS, *CP, *F1, *F2, or *F3. The default is the *HD unload format, for which you do not need this statement.
- The load module name of a user record-formatting routine (see “User record-formatting routine” on page 246).

Restrictions:

- If you specify a control statement such as MIGRATE or FALLBACK, you cannot specify any format other than *HD.
- If *CS is specified and one or more partitions of PHDAM are in the HALDB OLR cursor-active status, FABHURG1 ends abnormally.

PSPI

OFFS control statement

This optional control statement specifies the offset of the database segment within an area used to retrieve segments with HSSR calls.

PSPI

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

OFFS offs
```

Position
Description

1
Code the OFFS keyword to identify this statement as an OFFS statement.

6
Offset of the database segment within an area used to retrieve segments with HSSR calls. The bytes in front of the database segment within this area are reserved for use by the user record-formatting routine.

The length specified does not need to contain leading zeros, or it does not need to be aligned.

Notes:

- If a standard record-formatting routine is used, OFFS statements are ignored.
- The value specified on the OFFS statement must not be greater than the value specified in the ULEN statement.
- If a SYSUT2 DD statement is provided, OFFS must be at least 4. These 4 bytes are reserved for the OS record descriptor word.
- If no OFFS control statement is provided, the default is zero.
- If you specify a control statement such as MIGRATE or FALLBACK, this statement is ignored.

PSPI

ULEN control statement

This optional control statement specifies the maximum number of bytes that are reserved for the user data within the HSSR call I/O area. These bytes can be used by the user record-formatting routine.

PSPI

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

ULEN ulen
```

Position
Description

1
Code the ULEN keyword to identify this statement as a ULEN statement.

6
Specify the maximum number of bytes reserved for the user data, within the HSSR call I/O area, that can be used by the user record-formatting routine. The length of the HSSR call I/O area is the sum of the length of the longest database segment and the length specified on this control statement. The length specified does not need to contain leading zeros, or it does not need to be aligned.

Notes:

- If a standard record-formatting routine (for the *HD, *CS, *F1, *F2, or *F3 format) is used, this statement is ignored.
- If the SYSUT2 DD statement is present, its block size must be large enough to contain the segment data and the user data:

```
BLKSIZE ≥ 4 + max_segment_length + ulen
```

- If ULEN is larger than necessary, the blocking of SYSUT2 is not optimal.
- If the SYSUT2 DD statement is provided, ULEN must be at least 4. These 4 bytes are reserved for the OS record descriptor word, which contains the binary record length followed by binary zeros.
- If no ULEN control statement is provided, the default is zero.
- If you specify a control statement such as MIGRATE or FALLBACK, this statement is ignored.

USEGMAX control statement

This optional control statement specifies the number of bytes to be reserved for the segment data field within the HSSR call I/O area.


```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

USEGMAX usegmax
```

Position
Description

1
Code the USEGMAX keyword to identify this statement as a USEGMAX statement.

9
Specify the number of bytes to be reserved for the segment data field within the HSSR call I/O area. This reserved space is used by a record-formatting routine for the unload format *HD, *CS, *F1, or *F2. This length must be larger than or equal to the length of the longest database segment. The length of the HSSR call I/O area, which is called the *OUTPUT-AREA*, is the sum of the length specified by this statement and the length of the record header determined by each format. The length specified does not need to contain leading zeros, or it does not need to be aligned.

Notes:

- If you specified a value less than the length of the longest database segment, the *usegmax* value is adjusted to the length of the longest segment. The message FABH0276I is issued.
- If a user record-formatting routine or the standard record format *F3 is specified, the USEGMAX control statement is ignored. The message FABH0276I is issued.
- If no optional user exit routine is specified, this control statement is ignored. The message FABH0276I is issued.
- If the SYSUT2 DD statement is present, its block size must be large enough to contain the record header and the segment data field expanded by the USEGMAX statement:

```
BLKSIZE ≥ 4 + record_header_length + usegmax
```

- If no USEGMAX is specified or if the USEGMAX statement is ignored because of the reason previously stated, the default length (that is, the length of the longest segment in the database) is used as the length of the user data field.
- If you specify a control statement such as MIGRATE or FALLBACK, you cannot specify this statement.

PSPI

Get-by-RBA calls

The Get-by-RBA calls can be used by system programmers to cross logical relationships or secondary index relationships implemented with direct pointers. These calls can also be used to build the logical parent's concatenated key defined as virtual in the SEGM statement of the DBD.

PSPI

Get-by-RBA call is supported for compatibility with DBT HSSR and PO HSSR. The use of this call in IMS High Performance Unload is not recommended.

Restriction: Get-by-RBA call for HALDB is not supported.

Subtopics:

- "Structure" on page 257
- "Finding the RBA required by the Get-by-RBA call" on page 258

Structure

Get-by-RBA calls allow retrieval of segments of HIDAM and HDAM databases by their relative byte address.

The format of this call is:

```
 CALL ASMHSSR,(GU,PCB,IOAREA,SSA),VL     Assembler language
 CALL 'CBLHSSR' USING GU,PCB,IOAREA,SSA. COBOL
 CALL PLIHSSR (FOUR,GU,PCB,IOAREA,SSA);  PL/I
```

The SSA contains the 8-byte segment name, followed by *T, the left parenthesis, the 4-byte RBA, and the right parenthesis, as follows:

```
aaaaaaaa*T(nnnn)
|       |  |
|       |  |
|       |  Four-byte relative byte address
|       |  of segment prefix
|       |
|       Command code
|
Name of segment
```

If the database does not contain a segment prefix of the specified segment type at the specified RBA, HSSR Engine abends.

After completion of the call, HSSR call handler edits the requested segment in the IOAREA exactly as it does for the other types of calls (GN, GU, GHN, GHU, REPL).

The PCB is also edited normally. However, the part of the key feedback area that ordinarily contains the concatenated key of the physical parent now contains binary zeros.

GN calls should be issued with care after a Get-by-RBA call. Starting from the segment retrieval by the RBA call, HSSR Engine proceeds sequentially as far as possible. When it cannot proceed further, it returns a GB status code:

- If the segment retrieved by the RBA call is an HIDAM root segment, HSSR Engine proceeds sequentially to the end of the database.
- If the segment retrieved by the RBA call is an HDAM root segment, HSSR Engine proceeds sequentially to the last database record chained to the same RAP as the root retrieved by the RBA call.

- If the segment retrieved by the RBA call is not a root, the sequential processing stops at a position that depends on pointer options. It is either the last dependent of the last twin (twin pointers) or the last dependent of the last segment on the same hierarchical pointer chain (hierarchical pointers).

Finding the RBA required by the Get-by-RBA call

This information will be more readily useful if you have on hand an assembly listing of the IMS control blocks SDB and PSDB. To get such a listing, use the IMS macro:

```
IDLI  SDBBASE=0,DMBBASE=0
```

After the successful completion of an HSSR call, three control blocks (HJCB, HSDB, and the HDMB) of HSSR Engine contain the following information:

HJCB (Job Control Block of HSSR Engine)

The HJCB is an internal expansion of the HSSR PCB. It has functions similar to the DL/I JCB. It also contains the virtual storage address of the prefix of the segment just retrieved. This address is stored at the label HJCBPFXA.

Note: For variable-length split segments, the segment data is not stored after the prefix.

HSDB (Segment Descriptor Block of HSSR Engine)

The HSDB describes the segment type just retrieved and points to the following IMS control blocks:

- Segment descriptor block (SDB)
- Physical segment descriptor block (PSDB)

Using the HSDB, SDB, and PSDB, you can get an exact and complete description of the segment type just retrieved. This description includes an exact description of the segment prefix, which allows the displacement within the segment prefix of any pointers of interest to be computed.

HDMB (Data Management Block of HSSR Engine)

The HDMB describes the data set group of the segment just retrieved. The HJCB, HSDB, and HDMB can be found as follows:

- The field HPCBJCB of the HSSR PCB points to the HJCB.
- The field HJCBSDBC of the HJCB points to the HSDB of the segment just retrieved.
- The field HSDBHDMB of the HSDB points to the HDMB.

Because the layout of these control blocks can change with IMS High Performance Unload releases, the control blocks should be referred to symbolically by using the macros FABHPCB, FABHJCB, FABHSDB, and FABHDMB. The macros generate DSECTs that describe the control blocks.

The control blocks must never be modified by user routines (except the field HJCBUSER of the HJCB and the field HSDBUSER of the HSDB, which can be freely used).

Note: If you plan to use the Get-by-RBA call in an IMS environment that supports an OSAM database larger than 4 GB, pay attention to the RBA that you get from HSSR Engine. You must check the flag byte HDMBAMDA in the HDMB for the database data set that you are processing when you treat an RBA. The flag HDMBOS8G in this flag byte is on if and only if all of the following conditions are satisfied:

- Your IMS environment supports relative byte addressability for up to 8 GB of data in an OSAM HDAM or HIDAM database data set.
- Your database is an OSAM database.
- Your database has an even block size.

If HDMBOS8G is on, the lowest bit of the internal RBA represents the highest bit of the real 33-bit RBA. For example, the internal RBA X'00000001' represents the real RBA X'100000000'. If HDMBOS8G is off, the internal RBA is equal to the real RBA. The RBA specified on the operand of a Get-by-RBA call must be in the internal format.

PSPI

Considerations for coding and link-editing the routine

Certain considerations apply when you code and link-edit the routine.

PSPI

You must consider the following items when you link-edit the routine:

- The routine must be link-edited with the REUSE option.
- The routine must be link-edited in 31-bit addressing mode (AMODE 31).

  The reason is that database buffer pools are allocated above the 16-MB line and the address of the segment prefix is set into a parameter list as a 31-bit address. The HDMB and HRAN control blocks are also above the 16-MB line.

  Note: If the routine does not refer to the address of any segment prefix, HDMBs, or HRANs, it can be link-edited as AMODE 24. But the 31-bit addressing mode is recommended, to avoid the addressing mode problems.

  If the routine is link-edited in 31-bit addressing mode, but performs functions requiring AMODE 24, you must code the residency mode (RMODE) as needed. Before running the function, you must use the *capping* method to dynamically change the addressing mode to AMODE=24; after the function has run you must return to AMODE 31. The following figure shows a sample code for such capping:

```
*         Change the addressing mode to AMODE 24
          DS    0H
          L     14,VCT001
          BSM   0,14
          CNOP  0,4
VCT001    DC    AL4(*+4)
EXT001    DS    0H
*
          This is the functional part that requires
          24-bit addressing mode.
*
*         Change the addressing mode to AMODE 31
          DS    0H
          L     14,VCT002
          BSM   0,14
          CNOP  0,4
VCT002    DC    AL4(*+4+X'80000000')
EXT002    DS    0H
```

Figure 58. Sample code for capping

PSPI

Product-sensitive macros

IMS High Performance Unload provides product-sensitive macros.

PSPI

The macros described here are provided for use by system programmers in writing programs that use the services of HSSR Engine. Only the macros identified in this topic should be used to request or receive the services of HSSR Engine.

The product-sensitive macros listed in the following table are provided in HPS.SHPSMAC0 macro library.

Table 45. Product-sensitive macros for system program interfaces

Macro name	Description
FABHDMB	Mapping for HSSR Engine Data Management Block (HDMB)
FABHJCB	Mapping for HSSR Engine Job Control Block (HJCB)
FABHPCB	Mapping for HSSR PCB (HPCB)
FABHPTR	Mapping for HSSR Engine Pointer Block (HPTR)
FABHRAN	Mapping for HSSR Engine CAB RBA-range description table (HRAN)
FABHSDB	Mapping for HSSR Engine Segment Descriptor Block (HSDB)
FABHURGR	Mapping for the format *F1, *F2, or *F3 record that can be produced by the FABHURG1 unload utility
FABHFSUR	Mapping for the format HS, VB, or VN record that can be produced by the FABHFSU unload utility

For compatibility with earlier products, see the following topics:

- Chapter 30, “Compatibility with DBT V2 HSSR,” on page 357
- Chapter 31, “Compatibility with DBT V1 HSSR,” on page 365
- Chapter 32, “Compatibility with PO HSSR,” on page 367
- Chapter 33, “Compatibility with FSU II,” on page 373

PSPI

Chapter 19. Site default options

Differences between the default option values of IMS High Performance Unload and those of earlier products might interrupt smooth JCL migration. As a solution, the capability to change the default value for some options is provided. You can do this by replacing the IMS High Performance Unload's default option table (FABHOPT).

The following topics includes product-sensitive programming interface information. See “Programming interface information” on page 518 to understand the restrictions associated with this type of material.

Topics:

How the runtime parameters are determined

IMS High Performance Unload uses the parameter values that are specified in the HSSROPT, HSSRCABP, SYSIN, and CARDIN data sets. If parameter values are not found in these data sets but in the FABHOPT option table, values in the FABHOPT option table are used. If parameter values are not found in the data sets nor in the FABHOPT option table, the system default values are used.

PSPI

The following figure illustrates how the runtime parameters are determined. The sources that determine the runtime parameters are placed in the order of priority; that is, an option in a higher position in the figure overrides one in a lower position.

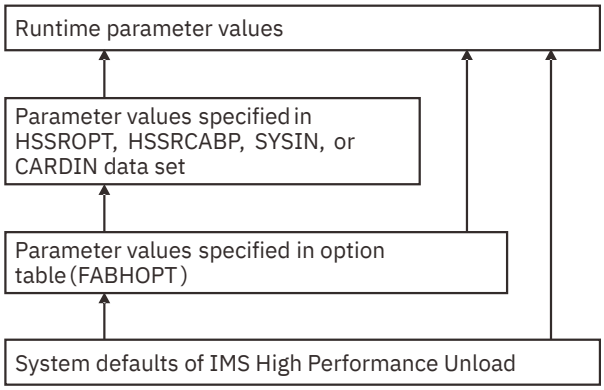

Figure 59. How the runtime parameters are determined

Runtime parameters that can be replaced

Defaults for options listed in the following table can be specified by replacing the default option table.

Table 46. Options for which default values can be specified

Function	Keyword	Description	Related data set
FABHURG1 utility	URG1DEC	DEC option for FABHURG1 (unload utility)	SYSIN (for FABHURG1)
	URG1CHKRC	CHECKREC option for FABHURG1 (unload utility)	SYSIN (for FABHURG1)
	URG1BUFNO	The default number of buffers to be assigned to the DCBs for the output data sets to store the unloaded records.	N/A
FABHFSU utility	FSUDEC	DEC option for FABHFSU (unload utility)	CARDIN (for FABHFSU)
	FSUBUFNO	The number of buffers to be assigned to the DCBs for the output data sets to store the unloaded records other than the HS format records.	N/A
HSSR Engine options	CABBASE_OS	CABBASE option for OSAM sequential buffering	HSSROPT
	CABBASE_OD	CABBASE option for OSAM direct buffering	HSSROPT
	CABBASE_VS	CABBASE option for ESDS or OSAM LDS sequential buffering	HSSROPT
	CABBASE_VD	CABBASE option for ESDS or OSAM LDS direct buffering	HSSROPT
	CABSTAT	CAB statistics report option	HSSROPT
	COMPAUTH	State option for segment compression exit call	HSSROPT
	LSR	LSR option	HSSROPT
	APISET	Level of API set	HSSROPT
	DIAGG	Diagnosis information for status GG	HSSROPT
	PCBLIST	Type of PCB list	HSSROPT
	ZIIPMODE	Specifies whether to offload eligible workloads to zIIP processors.	HSSROPT
Buffer handlers	BUFDEFAULT	Specifies how the default buffer handler is determined	HSSRCABP
	CABDEFAULT	Specifies how the default CAB parameters are determined	HSSRCABP

PSPI

Replacing the HSSR option table (FABHOPT)

You can replace the IBM-supplied default option table (FABHOPT) in HPS.SHPSLMD0 load module library with your own table containing options applicable to your site.

Procedure

PSPI

Complete the following steps to replace the table:

1. Copy the sample JCL FABHOPTG from the sample library.

   FABHOPTG JCL is a sample JCL for use in creating a user-defined default option table. FABHOPTG JCL is provided as a member of the HPS.SHPSSAMP library. FABHOPTG assembles the user-specified FABHTOPT macro statement and replaces the FABHOPT option table module in the HPS.SHPSLMD0 load module library.

2. Code the FABHTOPT macro in the copied JCL.

   For a list of the keywords that can be used in the FABHTOPT macro statements, see "FABHTOPT macro statements" on page 263.

   Use the following examples to code a FABHTOPT macro statement.

   Example 1
   The following FABHTOPT macro statement is used to create a user table that contains the defaults that are compatible with DBT V2 HSSR:

   ```
            FABHTOPT COMPAT=DBT
   ```

   Example 2
   The following FABHTOPT macro statement is used to create a user table that contains the defaults that are compatible with PO HSSR:

   ```
            FABHTOPT COMPAT=5787LAC
   ```

   Example 3
   The following FABHTOPT macro statement is used to create a user table that contains the defaults that are compatible with PO HSSR, except that CAB is used as the default buffer handler for ESDS and OSAM data sets, and that the default CAB buffering parameters are determined from the characteristics of database data sets:

   ```
       FABHTOPT COMPAT=5787LAC,BUFDEFAULT=CAB,CABDEFAULT=HPU
   ```

3. Submit the JCL to replace the IBM-supplied default option table.

PSPI

FABHTOPT macro statements

The rules for coding the FABHTOPT macro are the same as those for coding macro statements in Assembler language.

PSPI

FABHTOPT must be preceded and followed by at least one blank space, and parameters must be separated by commas.

```
(label)      FABHTOPT   COMPAT=[HPU|DBT|5787LAC]
                               [,APISET=1|2|3]
                               [,DIAGG=([CB][,BUF])|NOINT|DIAGONLY]
                               [,CABSTAT=YES|NO]
                               [,LSR=YES|NO]
                               [,URG1DEC=YES|NO]
                               [,URG1BUFNO=nnn]
                               [,URG1CHKRC=YES|NO]
                               [,FSUDEC=YES|NO]
                               [,FSUBUFNO=nnn]
                               [,BUFDEFAULT=CAB|BB]
                               [,CABDEFAULT=HPU|DBT]
                               [,PCBLIST=HSSR|IMS]
                               [,COMPAUTH=YES|NO]
                               [,CABBASE_OS=nnn]
                               [,CABBASE_OD=nnn]
                               [,CABBASE_VS=nnn]
                               [,CABBASE_VD=nnn]
                               [,ZIIPMODE=NEVER|COND]
```

COMPAT=

This optional keyword specifies the basic setting of FABHTOPT. IMS High Performance Unload provides three basic settings: HPU, DBT, and 5787LAC. The default basic setting is HPU. The following table shows the options to be set by the three basic settings:

Table 47. Basic settings of FABHTOPT options

Keyword	COMPAT=HPU	COMPAT=DBT	COMPAT=5787LAC
APISET=	1	1	1
DIAGG=	DIAGONLY	DIAGONLY	(CB,BUF)
CABBASE_OS=	1	1	1
CABBASE_OD=	2	2	2
CABBASE_VS=	*defnum* (see Note 1)	*defnum* (see Note 1)	*defnum* (see Note 1)
CABBASE_VD=	*defnum* (see Note 1)	*defnum* (see Note 1)	*defnum* (see Note 1)
CABSTAT=	NO	YES	NO
COMPAUTH=	NO	NO	NO
LSR=	NO	NO	NO
URG1DEC=	YES	YES	YES
URG1BUFNO=	*defnum* (see Note 2)	*defnum* (see Note 2)	*defnum* (see Note 2)
URG1CHKRC=	NO	NO	NO
FSUDEC=	YES	YES	YES
FSUBUFNO=	*defnum* (see Note 2)	*defnum* (see Note 2)	*defnum* (see Note 2)
BUFDEFAULT=	CAB	BB	BB
CABDEFAULT=	HPU	DBT	DBT
PCBLIST=	HSSR	HSSR	HSSR
ZIIPMODE=	NEVER	NEVER	NEVER

Notes:

1. The number is determined from the CI size of the database data set.
2. The number is determined from the block size of the output data set by each unload utility.

You can use the following keywords to override the values set by the basic settings:

APISET=
This optional keyword specifies the default for the APISET option in HSSROPT.

DIAGG=
This optional keyword specifies the default interpretation for the DIAGG statement with its operand left blank, as follows:

```
//HSSROPT  DD *
DIAGG
/*
```

CABBASE_OS=
This optional keyword specifies the default for the 'CABBASE *nnn* OSAM SEQ' option in HSSROPT.

CABBASE_OD=
This optional keyword specifies the default for the 'CABBASE *nnn* OSAM DIR' option in HSSROPT.

CABBASE_VS=
This optional keyword specifies the default for the 'CABBASE *nnn* VSAM SEQ' option in HSSROPT.

CABBASE_VD=
This optional keyword specifies the default for the 'CABBASE *nnn* VSAM DIR' option in HSSROPT.

CABSTAT=
This optional keyword specifies the default for the CABSTAT option in HSSROPT.

COMPAUTH=
This optional keyword specifies the default for the COMPAUTH option in HSSROPT.

LSR=
This optional keyword specifies the default for the LSR option in HSSROPT.

URG1DEC=
This optional keyword specifies the default for the DEC option in SYSIN for the FABHURG1 utility jobs.

URG1BUFNO=
This optional keyword specifies the default number of buffers to be assigned to the DCBs for the output data sets to store the unloaded records. Specifies a left-aligned decimal number in the range of 1 - 255. If this option is not specified, the default number is determined from the block size. The BUFNO= specification in the JCL DD statement is prior to this default number.

URG1CHKRC=
This optional keyword specifies the default for the CHECKREC option in SYSIN for the FABHURG1 utility jobs.

FSUDEC=
This optional keyword specifies the default for the DEC option in CARDIN for the FABHFSU utility jobs.

FSUBUFNO=
This optional keyword specifies the default number of buffers to be assigned to the DCBs for the output data set to store the unloaded records other than the HS format records. Specifies a left-aligned decimal number in the range of 1 - 255. If this option is not specified, the default number is determined from the block size. The BUFNO= specification in the JCL DD statement is prior to this default number.

BUFDEFAULT=
This optional keyword specifies how the default buffer handler is determined.

If BUFDEFAULT=CAB is specified, CAB is used as the default buffer handler for ESDS, OSAM, and OSAM LDS data sets.

If BUFDEFAULT=BB is specified, BB is used as the default buffer handler for all ESDS, OSAM, and OSAM LDS data sets except data sets of PHDAM and PHIDAM.

For PHDAM and PHIDAM, the default buffer handler for ESDS, OSAM, and OSAM LDS data sets is always CAB.

CABDEFAULT=

This optional keyword specifies how the default CAB buffering parameters that can be specified in HSSRCABP data set are to be determined.

If CABDEFAULT=HPU is specified, the default CAB parameters are determined automatically from the characteristics of the database data sets.

If CABDEFAULT=DBT is specified, the fixed default CAB parameters that are compatible with DBT V2 HSSR are used for ESDS, OSAM, and OSAM LDS data sets.

For PHDAM and PHIDAM, the default CAB parameters are always determined from the characteristics of the database data sets.

PCBLIST=

This optional keyword specifies the type of PCB list that is passed to the application program.

ZIIPMODE=

This optional keyword specifies the default for the ZIIPMODE option in HSSROPT.

Part 4. Using Sequential Subset Randomizer

You can use the Sequential Subset Randomizer to group subsets of database records close together.

Topics:

Chapter 20. Introduction to the Sequential Subset Randomizer

You can use the Sequential Subset Randomizer to create a randomizing module which enables fast sequential processing of database records belonging to a subset.

The Sequential Subset Randomizer utility is used to implement databases that combine:

- Fast random online access provided by HDAM
- Fast sequential processing of all database records belonging to one or multiple subsets of database records

The Sequential Subset Randomizer supports fast sequential processing of database records belonging to a subset by:

- Storing all database records of a subset close together in the HDAM root addressable area
- Allowing the retrieval of the first database record of a subset

With the Sequential Subset Randomizer, all database records of subset *n* are stored before the database records of subset *n+1*. During sequential database processing, all database records of subset *n* are retrieved before database records of subset *n+1*. However, the database records of a subset are neither stored nor retrieved in their logical key sequence.

If a database consists of 20 subsets, the sequential retrieval of all root segments with the Sequential Subset Randomizer will require one 20th, on the average, of the elapsed time which is required with the standard IMS DFSHDC40 randomizer (hereafter, called DFSHDC40).

Topics:

Characteristics of the Sequential Subset Randomizer

The randomizing techniques used by the Sequential Subset Randomizer are an extension of the randomizing techniques used by the standard IMS DFSHDC40 randomizer module.

The Sequential Subset Randomizer maintains (as DFSHDC40 does) the physical RAP sequence of database records through a database reorganization in the following cases:

- The number of CIs/blocks within the Root Addressable Area (RAA) is changed.
- The CI size or block size is changed.
- The number of RAPs per CI/block is changed.
- The relative amount of space within the RAA allocated to each subset is changed.

A change of these characteristics requires a database reorganization.

With the Sequential Subset Randomizer, as with DFSHDC40, if two different databases have database roots with identical keys, the physical RAP sequence of the database roots will be the same in the two databases.

As DFSHDC40, the Sequential Subset Randomizer supports the formatting of the whole root addressable area through the insertion of a root whose key begins with an X'FF'.

Benefits of the Sequential Subset Randomizer

By using the Sequential Subset Randomizer, you can group subsets of database records close together and achieve fast sequential processing of database records that belong to the subsets.

The Sequential Subset Randomizer is designed for the following types of databases:

Customer databases of companies with multiple branch offices.
The key of the customer databases usually consists of a branch office ID and a customer number. In this case, each subset consists of the database records of all customers belonging to the same branch office. With the Sequential Subset Randomizer, a program that has to process all database records of a branch office does not need to scan the whole database. The database scan can be limited to the part of the HDAM database that contains all database records of that branch office.

Databases of corporations that do business in different geographical areas.
For the key of a root segment that contains an ID identifying a geographical area such as county and state, it is possible to define a subset of database records as the group of database records belonging to the same geographical area, county, or state.

Databases of companies that have multiple divisions.
For the key of a root segment that contains the division ID, it is possible to define a subset of database records as the group of database records belonging to the same division.

The Sequential Subset Randomizer performs and benefits the following tasks:

- The Sequential Subset Randomizer is useful for the implementation of new HDAM databases if applications require fast sequential processing of subsets of database records (but do not require retrieval of the database records of a subset in their logical key sequence).
- The Sequential Subset Randomizer is useful when converting the following types of existing databases:
  - HDAM databases randomized with DFSHDC40 (or with other nonsequential randomizers) if applications require fast sequential processing of subsets of database records.
  - HISAM databases, HIDAM databases, and HDAM databases randomized with a sequential randomizer, if the retrieval in the logical key sequence is not required within subsets of database records.

Sequential Subset Randomizer program functions

The Sequential Subset Randomizer can be used to group subsets of database records close together. Therefore, it can speed up the execution of all of the Batch/BMP/MPP/IFP IMS programs and the IMS High Performance Unload utility programs that sequentially scan large HDAM databases in order to process only the database records of subsets.

Subtopics:

Physical clustering of all database records

The Sequential Subset Randomizer allows physical clustering of all database records belonging to the same subset (if the subsets can be defined as groups of database records having root keys within a common key range). With the Sequential Subset Randomizer, the sequential scanning that processes database records of the same subset does not require the scanning of the whole HDAM database. Only

the portion of the HDAM database that contains the database records of the subset to be processed needs to be scanned.

With the Sequential Subset Randomizer, the root addressable area of an HDAM database is conceptually divided into multiple portions. The first portion is used for the database records of the first subset, the second portion is used for the database records of the second subset, and so on.

To use the Sequential Subset Randomizer, all database records belonging to the same subset must have root keys within one common key range. The common key range must be identified by *n* bytes of the root key (where *n* is a number smaller than or equal to the number of bytes within the root key) starting at the fixed position within the root key. These *n* bytes are called the subset ID.

The following figure shows how a root addressable area is divided by the subset IDs. In this example, the branch IDs are the subset IDs.

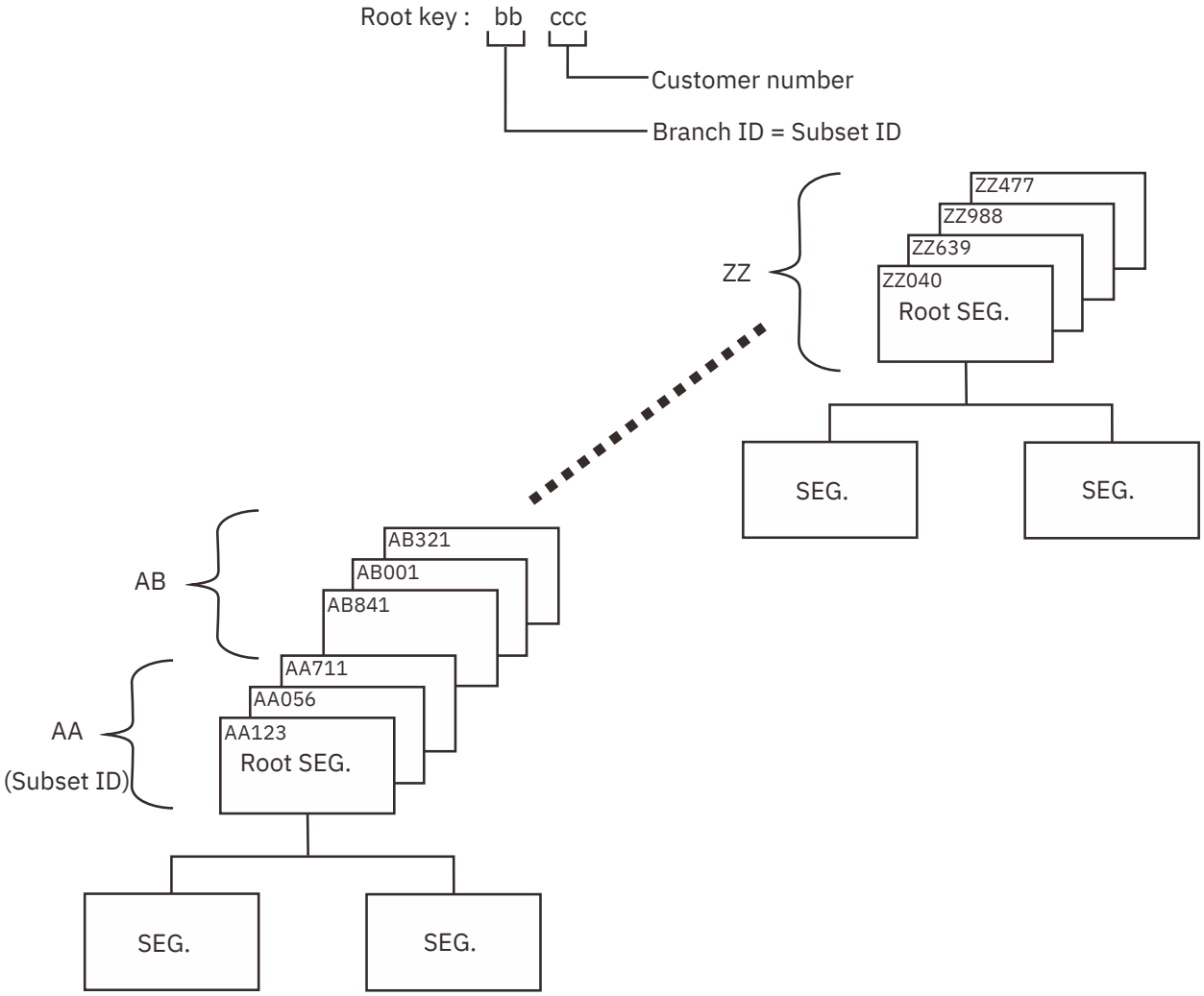

Figure 60. Physical clustering with the Sequential Subset Randomizer

Database retrievals

The Sequential Subset Randomizer allows retrievals through a qualified GU call (with special root key values in the SSA) of the first database root segment belonging to each subset. Application programs which process all database records of a subset can:

- Retrieve the first database record of the subset by using this GU call.
- Retrieve the further database records of the subset by using GN calls.

- Check if it has finished processing all database records of the subset by testing in the PCB key feedback area (or in the segment I/O area) for a root key value which no longer belongs to the key range of the subset.

Generation of a Sequential Subset Randomizer and database

Ordinarily, one Sequential Subset Randomizer is generated for each database to be randomized. However, if multiple databases have the same characteristics (if they have the same subsets, the same relative amount of space to be reserved for each subset, the same offset within the root key of the subset IDs, and so on), one common Sequential Subset Randomizer can be used for these multiple databases.

Splitting the unloaded data set

The database reorganization can be accelerated by splitting an unloaded database data set into multiple data sets with the FABIUNLS utility. The purpose of this splitting is to provide input to the IMS HD Reorganization Reload utility. The split data sets are then sorted in the sequence that allows reloading of large databases in a reasonable amount of time.

Providing statistical information

The SS-STATS (Sequential Subset Statistics) routine (FABISTAT) is an exit routine of the IMS High Performance Unload's database unload utilities FABHURG1 and FABHFSU. It provides database administrators with statistical information that is useful in determining how much relative space in the root addressable area needs to be allocated to each subset.

Differences between the Sequential Subset Randomizer and other sequential randomizers

The Sequential Subset Randomizer provides advantages over other randomizers. However, other randomizers also provide advantages over the Sequential Subset Randomizer. Before using the Sequential Subset Randomizer, learn the differences to determine the use of Sequential Subset Randomizer.

Note: In these topics, the term *sequential randomizer* means a randomizer which randomizes database root segments in the logical sequence of the key value. A sequential randomizer can be one of the IMS HD sequential randomizers or the user's unique randomizer.

Subtopics:

Advantages of the Sequential Subset Randomizer over other sequential randomizers

For databases that do not need to be processed in the logical sequence of the root key values, the Sequential Subset Randomizer provides the following advantages over sequential randomizers:

- The performance of the Sequential Subset Randomizer is less affected by a high volume of insertion and deletion activities than that of sequential randomizers. With a sequential randomizer, the user must often regenerate the sequential randomizer and reorganize the databases after inserting a large number of roots. This is often a long process. The database is unavailable to the applications during this time.

  With the Sequential Subset Randomizer, a regeneration and reorganization are not necessary if the insert activities are evenly distributed across the different subsets, and if the HDAM database has been defined with enough free space to absorb the insert activities.

- The tables of the Sequential Subset Randomizer are smaller than those of the sequential randomizers. With huge HDAM databases, the tables of a sequential randomizer are often huge, and can often create substantial paging activity, which defeats the advantage of HDAM over HIDAM.

Advantages of sequential randomizers over the Sequential Subset Randomizer

Sequential randomizers support storing and retrieving of database root segments in the logical sequence of the root key values. The Sequential Subset Randomizer does not provide such support.

Advantages of the Sequential Subset Randomizer over DFSHDC40

The Sequential Subset Randomizer can be used to group all database records of a subset physically close together, thereby improving the processing time for programs that only process the database records of one or a few subsets. DFSHDC40 does not provide such support.

Advantages of DFSHDC40 over the Sequential Subset Randomizer

With databases for which there is no need to physically group database records into subsets, DFSHDC40 has the following advantages over the Sequential Subset Randomizer:

- DFSHDC40 does not use subsets. Therefore the database administrator does not need to specify the relative amount of space to be reserved for each subset.
- With DFSHDC40, there is no need to periodically monitor (and sometimes adapt) the relative amount of space effectively occupied by each subset. Adapting the relative amount of space allocated to each subset requires a reorganization of the database. Reorganization is often a long process and the database is unavailable during the reorganization.

Sequential Subset Randomizer program structure

The Sequential Subset Randomizer and its tables are generated by means of the SSRGEN process using three macro statements (those are, FABITAB, FABIDEF, and FABIGEN). During this process, the database designer (or the database administrator) identifies the subsets, their key ranges, and the space which must be reserved for each subset in the HDAM root addressable area.

The Sequential Subset Randomizer includes the following components:

- A utility program (FABIUNLS), which is used to speed up the database reorganization while converting from databases randomized with DFSHDC40 to databases randomized with the Sequential Subset Randomizer.
- An exit routine (FABISTAT) of IMS High Performance Unload's database unload utilities (that is, FABHURG1 and FABHFSU), which provides statistics about the number of roots and the total length of database records in each subset. These statistics are called SS-STATS. The SS-STATS can be used to specify the relative amount of space to be reserved within the root addressable area for each subset when you generate a Sequential Subset Randomizer next time.

The Sequential Subset Randomizer complies to the usual IMS conventions for HDAM randomizing routines. Therefore, the Sequential Subset Randomizer can be used with the IMS High Performance Unload database unload utilities and with Batch/BMP/MPP/IFP IMS programs. However, the FABIUNLS utility and the SS-STATS exit routine run exclusively in IMS High Performance Unload jobs.

The Sequential Subset Randomizer can run with multiple IMS versions and releases without re-installing the product as far as the version/release is supported by IMS High Performance Unload. Refer to "Software requirements" on page 17 for information about supported IMS versions and releases.

Sequential Subset Randomizer restrictions

The following restrictions apply to the Sequential Subset Randomizer.

- The Sequential Subset Randomizer supports a maximum of 1000 subsets.

- If the length of a subset ID is not smaller than the length of the root key, the Sequential Subset Randomizer does not provide any special support for retrieval of the first root segment of the subset.
- If a subset ID does not start in the first position of the key of the root segment, the root addressable area must contain at least as many RAPs as the number of subsets.

Chapter 21. Planning for the Sequential Subset Randomizer

Before applying the Sequential Subset Randomizer, you must plan for the use of Sequential Subset Randomizer.

The following topics describe the Sequential Subset Randomizer from the viewpoint of the user tasks. The user tasks include:

- Database design, especially when defining the subset IDs
- Application programming
- Database administration
  - Determining the relative amount of space to each subset
  - Monitoring the database

Topics:

Considerations when defining subset IDs

Database designers need to define the subsets and the subset IDs.

This is often trivial work in the following case:

- The subsets are defined as the database records belonging to a branch office.
- The branch office ID is contained at a fixed location of the database root key.

In this case, the subset ID is defined as the branch office ID. A subset is the group of database records belonging to the same branch office.

The following points should be observed:

- The subset ID must be completely embedded within the root key and must start at a fixed location within the root key. The length of the subset ID is fixed and must be smaller than or equal to the length of the root key.
- The number of bytes within the subset ID should be smaller than the number of bytes within the root key if the application needs the support of the Sequential Subset Randomizer in order to retrieve the first database record of a subset.
- For each subset, the user must define a value for the subset ID during the generation of the Sequential Subset Randomizer. The Sequential Subset Randomizer supports the following two different interpretations of the subset ID values:

  VALTYPE=E (Value type=Exact)
  The Sequential Subset Randomizer assumes that all database records belonging to a subset will have the exact value specified during the generation of Sequential Subset Randomizer as a subset ID.

  VALTYPE=H (Value type=High)
  The Sequential Subset Randomizer assumes that the value specified during the generation of the Sequential Subset Randomizer is a high value. The database records belonging to a subset have subset IDs in the range between two specified subset ID values (between the high value of the preceding subset and the high value of this subset).

- The implementation of the Sequential Subset Randomizer does not support more than 1000 subsets.

Considerations for application programming

The following considerations about the use of the physical clustering of database records apply when using the Sequential Subset Randomizer.

This section describes application programming available to system programmers. It is a product-sensitive programming interface. See “Programming interface information” on page 518 to understand the restrictions associated with this type of material.

PSPI

The Sequential Subset Randomizer is ineffective for application programs that do not use the physical grouping of database records by subsets. These programs are not affected by a conversion from DFSHDC40 randomizing to sequential subset randomizing (if they are not sensitive to the sequence in which the database records are retrieved during sequential database processing).

When an application programmer wants to use the physical clustering of database records, the following should be considered:

Retrieval of the first database record of a subset

The first root segment of a subset can be retrieved by a qualified GU call using the following SSA:

```
ROOTNAME(KEY-NAME=>...SID......)
```

SID
Subset ID. It must reside at the same position as the subset ID in the root key. When VALTYPE=H was specified during the generation of the Sequential Subset Randomizer, the minimum value within the required subset ID must be used for the GU call.

.....
Binary zeros.

Note: When orphan root segments exist in the database, they may also be retrieved by the GU call. In such a case, the succeeding root segments must be processed sequentially until the required subset ID value is encountered.

End of the database records of a subset

An application program which sequentially processes a database can detect that it has processed the last database record of a subset. It can be done by detecting a greater subset ID value than the required subset ID value within the root key field (either in the PCB key feedback area or in the I/O area).

PSPI

Considerations for the relative amount of space to each subset

Database administrators must define how much space should be allocated to each subset.

The amount of space to be allocated to each subset is specified as a relative amount of space. It can be expressed as follows:

- The number of database roots within the subset
- The total length of all database records within the subset
- One of the above two values expressed as the percentage of the database totals

Database administrators can determine the relative amount of space to be allocated to each subset by activating the SS-STATS routine during sequential processing, or unloading of the whole database. The SS-STATS will show for each subset how much space is occupied by all the database segments of

the subset that are stored in the first data set group of the HDAM database. These space statistics are expressed both as the number of bytes and as the percentage of the database totals.

It is recommended that database administrators use the percentage for the length of database records (provided by the SS-STATS routine) for the allocation of space to each subset. The usage of percentage makes it easy to follow how and whether the database growth impacts the relative space allocation for each subset.

Considerations for monitoring the database

After the Sequential Subset Randomizer has been applied to your database, monitor the database like any other HDAM database. This topic provides tips for monitoring the database that has the Sequential Subset Randomizer applied.

An HDAM database randomized with the Sequential Subset Randomizer needs to be monitored like any other HDAM database. Database Tuning Statistics should be used to monitor periodically the quality of the randomizing and the need to reorganize a database. For example, if the Database Tuning Statistics for April show that the random retrieval of a database record requires as an average 1.23 I/Os and the Database Tuning Statistics from June of the same year show that this figure has increased to 1.89 I/Os, it would be time to reorganize.

As a part of this usual database monitoring, database administrators should observe how the space is occupied by the database records of each subset. This is done by activating the SS-STATS routine during the sequential processing or unloading of the whole database. If the Database Tuning Statistics show a need to reorganize a database, and a database administrator decides to do so, the SS-STATS statistics should be considered in deciding whether also to change the relative amount of space allocated to each subset and regenerate the Sequential Subset Randomizer.

Chapter 22. Sequential Subset Randomizer generation

Sequential Subset Randomizer and its tables are generated by means of the process called SSRGEN. A Sequential Subset Randomizer load module is produced by an assembly and link-edit.

Topics:

Generating the Sequential Subset Randomizer load module

To generate a Sequential Subset Randomizer load module, you can use the IBM-supplied cataloged procedure.

About this task

The general data flow for the generation of a Sequential Subset Randomizer module is shown in the following figure.

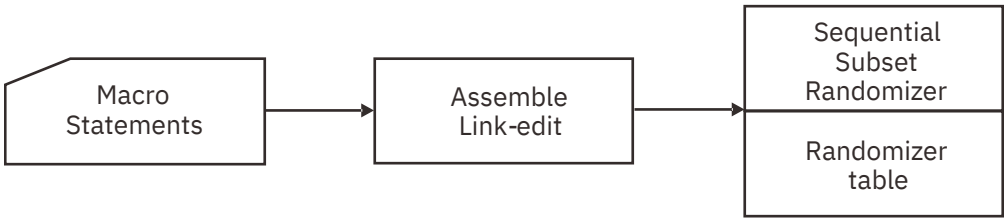

Figure 61. Data flow of the generation of the Sequential Subset Randomizer module (SSRGEN)

Procedure

1. Prepare a JCL job for FABIRGEN.

   To generate a Sequential Subset Randomizer, use the IBM-supplied cataloged procedure FABIRGEN that is located in the HPS.SHPSSAMP sample library.

   Code the EXEC and DD statements for the process. For the JCL requirements, see "FABIRGEN JCL requirements" on page 280. See also Figure 66 on page 284 for a JCL example for generating the Sequential Subset Randomizer. This example assumes that the IBM-supplied cataloged procedure is used.
2. Code the macro statements in the SYSIN data set.

   Database designer, administrators, or both must provide the following information through macro statements:

   - The length of the subset IDs (the default is 1)
   - The relative start position of the subset ID within the root key (the default is 1)
   - Whether the values for the subset IDs are provided as hexadecimal values or as EBCDIC values (the default is EBCDIC)
   - Whether the values for the subset IDs are exact values or high-values (the default is exact value)
   - For each subset:

- The subset ID value
- The relative amount of space to be reserved in the root addressable area

For descriptions of the macro statements, see "FABIRGEN input: SYSIN data set" on page 280.

3. Assemble and link-edit the macro statements to generate the Sequential Subset Randomizer load module and its tables.

   The assembly requires the following macro libraries:

   - SYS1.MACLIB
   - HPS.SHPSMAC0
   - IMSVS.SDFSMAC

FABIRGEN JCL requirements

To generate the Sequential Subset Randomizer, supply an EXEC statement and the appropriate DD statements.

The following table summarizes the DD statements.

Table 48. FABIRGEN DD statements

DDNAME	Use	Format	Need
SYSLIB	Input	LRECL=80	Required
SYSIN	Input	LRECL=80	Required
SYSLMOD	Output	-	Required

EXEC
The EXEC statement must be in the following format:

```
//    EXEC  FABIRGEN,MBR=randname
```

SYSLIB DD
This statement defines the input macro library data set.

SYSIN DD
This statement defines the input data set that contains your control statement.

SYSLMOD DD
This statement defines the output partitioned data set that contains the load module of the generated Sequential Subset Randomizer.

FABIRGEN input: SYSIN data set

All input you must specify to generate the Sequential Subset Randomizer is only a SYSIN data set.

The SYSIN data set contains the following statements:

- FABITAB macro statement
- FABIDEF macro statement
- FABIGEN macro statement
- END statement

Note: Macro statements FKDTAB, FKD, and FKDGEN can be used instead of FABITAB, FABIDEF, and FABIGEN, respectively. These macro statements are provided only for compatibility. The use of macros prefixed by FABI is recommended.

Format

This data set usually resides in the input stream. However, it can be defined either as a sequential data set, or as a member of a partitioned data set. It must contain 80-byte fixed-length records. BLKSIZE, if coded, must be a multiple of 80.

General rules for macro statements

The coding conventions for the macro statements are the same as Assembler coding conventions. The following information pertains to the coding conventions that you must follow in writing macro statements:

- The entries must be written in the following order: label, operation, operand, and remarks.
- The entries must be contained in the begin column (1) through the end column (71) of the first line and, if needed, can continue in column 16 through column 71 of any continuation lines.
- When a macro statement is not completed in the first line, column 72 must be coded with any character.
- The entries must be separated from each other by one or more blanks.
- If used, a label entry must start in the begin column.
- The label and operation entries, each followed by at least one blank, must be contained in the first line of a macro statement.
- The operation entry must begin at least one column to the right of the begin column.

The name, operation, operand, and remark entries are as follows:

Label entry
It is a symbol created by you to identify a macro statement. This entry is optional.

Operation entry
It is the symbolic operation code specifying the macro operation desired. This entry is mandatory.

Operand entries
These entries contain one or more operands that identify and describe data to be acted upon by the macro instruction.

Remarks entries
Remarks are used to describe the current macro instruction.

FABITAB macro statement

The FABITAB macro statement defines the information about subset IDs.

The FABITAB macro statement must be the first statement. The format of the FABITAB macro statement is as follows:

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

Label      FABITAB TABNAME=tabname,                                      C
               START=n,                                                  C
               BYTES=m,                                                  C
               IDTYPE=y,      (C/X)                                      C
               VALTYPE=z      (E/H)MIGRATE
```

Figure 62. FABITAB macro statement

Note: C in column 72 is a continuation character.

Label
Coding of the label is optional.

TABNAME=
This keyword parameter is optional. The value *tabname* of the keyword parameter is a table name; it is used only for documentation purposes. The table name consists of 1- to 8-alphanumeric characters.

START=
This keyword parameter specifies where the subset ID starts within the key field of the database root segment. The start position for the first byte of the root key is 1.

The default is 1.

BYTES=
This keyword parameter specifies the length of the subset ID. The subset ID must be completely located within the key field of the database root segment.

Note: The available values of START=*n* and BYTES=*m* parameters are shown in the following algorithm:

```
n + m ≤ key length
```

The default is 1.

IDTYPE=
This keyword parameter specifies whether the values for the subset IDs on the FABIDEF macro statements are EBCDIC values (C) or hexadecimal values (X).

The default is C.

VALTYPE=
This keyword parameter specifies whether the values of the subset IDs should be interpreted as exact values (E) or high values (H).

The default is E.

E

When E is specified and a database root segment is inserted with a subset ID which has not been defined with an FABIDEF macro statement, the Sequential Subset Randomizer interprets this as an error.

Example:

- FABIDEF macro statements are used to define subsets with subset ID values A, C, and D.
- No FABIDEF macro statement is used to define a subset with a subset ID value B.
- Database root segments are inserted with a subset ID value B.
- Database root segments are inserted with a subset ID value E.

In this example, the Sequential Subset Randomizer considers the insertion of database root segments with subset ID values B and E as an error.

As a result, all database root segments with a subset ID value B will be randomized to the first RAP of the subset with the subset ID value C (the higher and closest subset ID value). If a lot of database root segments are inserted with a subset ID value B, this will create a long HDAM synonym chain in the first RAP of the subset with the subset ID value C.

All database root segments with a subset ID value E will be randomized to the last RAP of the subset with the subset ID value D.

H

When H is specified and a database root segment is inserted with a subset ID which has not been defined with an FABIDEF macro statement, the Sequential Subset Randomizer interprets this as a normal situation.

Example:

- Two consecutive FABIDEF macro statements are used to define subsets with subset ID values of 1000 and 2000.
- No FABIDEF macro statement is used to define a subset with a subset ID value 1500.
- Database root segments are inserted with a subset ID value 1500.
- Database root segments are inserted with a subset ID value 2500.

In this example, the Sequential Subset Randomizer considers the database root segments with a subset ID value 1500 as being normal database roots belonging to that subset which has a high value subset ID 2000. This subset consists of all database records whose subset ID is between 1000 and 2000 (1000 is excluded).

Because the Sequential Subset Randomizer does not store the root segments according to their key sequence within a subset, sequential database processing may retrieve a database root with a subset ID 1500 before a database root with subset ID 1400.

The Sequential Subset Randomizer considers the insertion of database root segments with a subset ID value 2500 as an error. Those segments which have higher subset ID values than the maximum subset ID value (2000) are randomized to the last RAP of the subset with the subset ID value 2000.

FABIDEF macro statement

The FABIDEF macro statement defines the subset ID value and the amount of space to be reserved in the HDAM root addressable area.

One FABIDEF macro statement must be provided for each subset. The FABIDEF macro statements must be provided in the ascending sequence of the subset ID values.

A maximum of 1000 FABIDEF macro statements can be provided.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

Label      FABIDEF   ID=aaaaa,UNITS=nnnn
```

Figure 63. FABIDEF macro statement

Label
Coding of the Label is optional.

ID=
This keyword parameter specifies the subset ID value.

- If IDTYPE=C has been specified on the FABITAB macro statement, the subset ID value must be specified as an alphanumeric value of length *n* which has been defined on the BYTES= keyword of the FABITAB macro statement.
- If IDTYPE=X has been specified on the FABITAB macro statement, the subset ID value must be specified as a hexadecimal value of length 2*n* (where *n* has been defined on the BYTES= keyword of the FABITAB macro statement).

  Example for coding a hexadecimal subset ID value (BYTES=3 on FABITAB):

  ```
  ID=F0F403
  ```

UNITS=
This keyword parameter specifies the relative amount of space to be reserved in the HDAM root addressable area for the database records of the subset to be defined by this FABIDEF macro statement.

The unit can be one of the following numbers:

- The number of roots per subset
- The number of data bytes within all database segments of a subset
- The number of RAPs to be reserved for the database records of a subset.

The keyword parameter value must be an integer number (without decimal points and commas). Valid values are in the range between 0 and 2 GIGA minus 1.

FABIGEN macro statement

The FABIGEN macro statement specifies to generate the Sequential Subset Randomizer module and its tables.

The FABIGEN macro statement must be the last macro statement of the FABIRGEN (SSRGEN) source statements preceding the END statement.

The FABIGEN macro statement must be coded as follows:

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

          FABIGEN
```

Figure 64. FABIGEN macro statement

END statement

This END statement specifies the end of the SYSIN data set.

The END statement must be coded as follows:

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

          END
```

Figure 65. END statement

FABIRGEN JCL examples

Use the following JCL examples to generate the Sequential Subset Randomizer module and its tables.

Example 1: Generating the Sequential Subset Randomizer, case 1

The following figure presents typical example for generating the Sequential Subset Randomizer.

```
//     EXEC FABIRGEN,MBR=SSRRAND
//C.SYSIN   DD  *
        FABITAB START=1,BYTES=1,VALTYPE=E
        FABIDEF ID=A,UNITS=450        BASEL
        FABIDEF ID=B,UNITS=150        BELLINZONA
        FABIDEF ID=C,UNITS=230        BERN
        FABIDEF ID=D,UNITS=170        BIEL
        FABIDEF ID=E,UNITS=110        CHUR
        ..................
        FABIDEF ID=Q,UNITS=800        ZURICH
        FABIGEN
        END
/*
//L.SYSLMOD  DD  DSN=RANDOMIZ.LIB(&MBR),DISP=SHR
```

Figure 66. Example 1: Generating the Sequential Subset Randomizer, case 1

In this example, the FABITAB macro statement specifies that:

- The subset ID starts at the first position of the root key (START=1).
- The length of the subset ID is 1 byte (BYTES=1).
- The values for the subset IDs are exact values (VALTYPE=E).

Each FABIDEF macro statement defines a single subset with the following keywords:

- `ID=` keyword defines the value of the subset ID.
- `UNITS=` keyword defines how much relative space should be allocated to the subset.

Note: The FABIDEF macro statements must be provided in ascending sequence of the subset ID values.

Example 2: Generating the Sequential Subset Randomizer, case 2

The following figure presents another typical example for generating the Sequential Subset Randomizer.

```
//     EXEC FABIRGEN,MBR=SSRRAND
//C.SYSIN   DD  *
        FABITAB START=3,BYTES=2,VALTYPE=H,IDTYPE=X
        FABIDEF ID=0020,UNITS=47         San Francisco
        FABIDEF ID=0030,UNITS=24         San Jose
        FABIDEF ID=005A,UNITS=62         Los Angeles
        FABIDEF ID=0088,UNITS=31         San Diego
        FABIDEF ID=0094,UNITS=3          Los Gatos
        ..................
        FABIDEF ID=01FF,UNITS=6          Bakersville
        FABIGEN
        END
/*
//L.SYSLMOD  DD  DSN=RANDOMIZ.LIB(&MBR),DISP=SHR
```

Figure 67. Example 2: Generating the Sequential Subset Randomizer, case 2

In this example, the FABITAB macro statement specifies that:

- The subset ID starts at the third position of the Root-Key (START=3).
- The length of the subset ID is 2 bytes (BYTES=2).
- The values for the subset IDs are high values (VALTYPE=H).
- The values for the subset ID values are provided as hexadecimal values (IDTYPE=X).

Each FABIDEF macro statement defines a single subset with the following keywords:

- ID= keyword defines the (hexadecimal) values of the subset ID.
- UNITS= keyword defines how much relative space should be allocated to the subset.

The FABIDEF macro statements must be provided in ascending sequence of the hexadecimal subset ID values.

The first FABIDEF macro statement specifies that database roots with subset ID values equal to or lower than X'0020' belong to the first subset. The second FABIDEF macro statement specifies that database roots with subset ID values between X'0020' and X'0030' (X'0020' being excluded) belong to the second subset. The rest of the FABIDEF macro statements define subsets of database roots in the same manner.

Chapter 23. Splitting the unloaded database data set

The FABIUNLS utility is used to speed up the database reorganization while converting databases randomized with DFSHDC40 into databases randomized with the Sequential Subset Randomizer.

FABIUNLS splits an unloaded database data set into multiple data sets as input to the IMS HD Reorganization Reload utility. The split data sets are sorted in the ascending order of subset ID so that a large databases can be reloaded in a reasonable amount of time.

Topics:

Splitting the unloaded database data set with FABIUNLS

FABIUNLS takes an unloaded database data set in the HD unload format as input, and creates as output multiple data sets for use as concatenated input to the IMS HD Reorganization Reload utility program.

About this task

The general data flow for the FABIUNLS utility program is shown in the following figure.

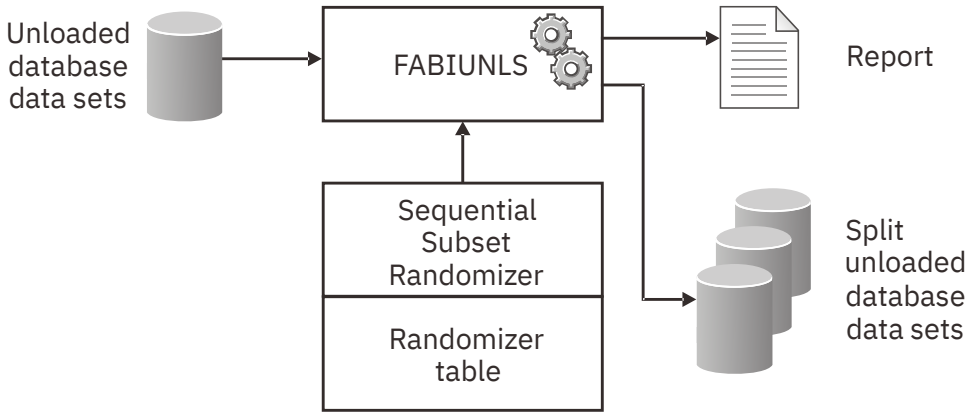

Figure 68. Data flow for FABIUNLS

Procedure

1. Ensure that you have completed the following activities before running FABIUNLS.
   - The database is unloaded in the HD unload format.
   - The Sequential Subset Randomizer is generated for the database.
   - The DBD source statement is changed to specify the name of the generated Sequential Subset Randomizer, and a DBDGEN is performed.
2. Code the EXEC and DD statements for FABIUNLS.

   FABIUNLS must be executed in a ULU region with the FABHULU JCL procedure (or with equivalent JCL).

   For the JCL requirements for FABIUNLS, see "FABIUNLS JCL requirements" on page 288.
3. Run the job.

Related reference

FABIUNLS JCL example
This topic provides a JCL example for generating a database with the FABIUNLS utility program.

FABIUNLS JCL requirements

To execute FABIUNLS, supply an EXEC statement and the appropriate DD statements.

The following table summarizes the DD statements.

Table 49. FABIUNLS DD statements

DDNAME	Use	Format	Need
STEPLIB	Input	-	Required
DFSVSAMP	Input	LRECL=80	Required
SYSPRINT	Output	LRECL=133	Required
SYSUT1	Input	-	Required
HDR	Output	-	Required
FKD*n*	Output	-	Required
TRL	Output	same as HDR	Required

EXEC

This statement invokes the JCL procedure FABHULU and has the following format:

```
//  EXEC  FABHULU,MBR=FABIUNLS,DBD=dbdname
```

The name of the DBD must be provided in place of *dbdname*.

STEPLIB DD

The STEPLIB must provide access to the load library which contains the Sequential Subset Randomizer and IMS load modules.

DFSVSAMP DD

This statement is mandatory and must meet to the usual IMS requirements.

SYSPRINT DD

This DD statement is mandatory and defines a print data set which will contain statistics. The data set can be defined as:

```
//SYSPRINT DD SYSOUT=A
```

SYSUT1 DD

This DD statement is mandatory and defines the input data set which contains the unloaded database in the HD unload format. The data set can be defined as:

```
//SYSUT1   DD DSN=unloaded.db,DISP=OLD
```

HDR DD

This DD statement is mandatory and defines an output data set which will contain:

- The header record of the HD unload format
- The unloaded database records which do not belong to any subset defined during the generation of the Sequential Subset Randomizer

The data set can be defined as:

```
//HDR      DD DSN=unls.hdr,DISP=(,CATLG),
//            UNIT=SYSDA,SPACE=(TRK,(1,1)),
//            DCB=(LRECL=nnnn,BLKSIZE=mmmm,RECFM=VB)
```

Recommended values for the BLKSIZE and LRECL are the same as the BLKSIZE and LRECL of the input data set described by the SYSUT1 DD statement. If it is not possible to meet these recommendations, the following minimum should be observed:

- The minimum LRECL must be the larger of these two items:
  - Largest database segment +39 bytes
  - Length of the HD unload header record.
- The minimum BLKSIZE must be LRECL+4.

FKD*n* DD

There should be one such DD statement for each subset defined during the generation of the Sequential Subset Randomizer. The DD names of these data sets should be: FKD1, FKD2, FKD3, ..., FKD9, FKD10, FKD11, and so on.

These DD statements are mandatory and define output data sets that will contain the unloaded database segments of the subsets. The FKD1 output data set will contain all unloaded database segments of the first subset, the FKD2 output data set will contain the unloaded database segments of the second subset, and so on.

The data sets can be defined as:

```
//FKDn   DD DSN=unls.fkdn,DISP=(,CATLG),
//          UNIT=SYSDA,SPACE=(CYL,(m1,m2)),
//          DCB=*.HDR
```

Enough DASD space must be allocated in order to contain all unloaded database records of the subset. If the Sequential Subset Statistics report has been produced in the previous FABISTAT run, the statistics can help in determining the amount of DASD space required for each FKD*n* output data set.

For a complete description of the Sequential Subset Statistics report and FABISTAT, see “FABIUNLS output” on page 289.

TRL DD

This DD statement is mandatory and defines an output data set that will contain the trailer record of the HD unload format.

The data set can be defined as:

```
//TRL    DD DSN=unls.trl,DISP=(,CATLG),
//          UNIT=SYSDA,SPACE=(TRK,(1,1)),
//          DCB=*.HDR
```

FABIUNLS output

FABIUNLS produces the following four types of output data sets.

SYSPRINT data set

This data set contains statistical reports.

HDR data set

This data set contains the following information:

- A header record of the HD unload format
- Unloaded database records which do not belong to any subset defined during the generation of the Sequential Subset Randomizer

FKD*n* data set
The FKD*n* data set contains the unloaded database segments of the subsets. Where *n* is the order of the generation of the output data sets. That is, the FKD1 data set contains all unloaded database segments of the first subset, the FKD2 data set contains the segments of the second subset, and so on.

TRL data set
This data set contains the trailer record of the HD unload format.

SYSPRINT data set

FABIUNLS SYSPRINT data set contains the Split Unloaded Data Set Statistics report.

Format

The format is 133-byte fixed-length records. When the block size is coded in the JCL, the block size must be a multiple of 133. Code your DD statement as follows:

```
     //SYSPRINT DD SYSOUT=A
```

Split Unloaded Data Set Statistics report

The Split Unloaded Data Set Statistics report contains statistics about each split unloaded data set.

The following two figures show examples of Split Unloaded Data Set Statistics reports.

```
IMS HIGH PERFORMANCE UNLOAD - SSR                    "SPLIT UNLOADED DATA SET STATISTICS"                        PAGE:     1
5655-E06                                                DATE: 06/01/2021  TIME: 11.34.06                      FABIUNLS - V1.R2

DDNAME        NBR OF ROOTS      NBR OF BYTES     SUBSET-ID
------        ------------      ------------     ---------
FKD1                 1,354             2,274     A100
FKD2                 1,561             2,349     A200
FKD3                 2,094             3,225     A300
FKD4                   808             1,250     A400
FKD5                 2,340             3,553     A500
FKD6                   564               957     A600
FKD7                 1,515             2,355     A700
FKD8                   962             1,658     A800
FKD9                 1,250             1,459     A900
FKD10                1,735             2,750     B000



NBR OF ORPHANS=              131

DBDNAME   =SSRHDBD1
RANDOMIZER=SSRANDC1
```

Figure 69. Split Unloaded Data Set Statistics report (IDTYPE=C)

```
IMS HIGH PERFORMANCE UNLOAD - SSR                    "SPLIT UNLOADED DATA SET STATISTICS"                        PAGE:     1
5655-E06                                                DATE: 06/01/2021  TIME: 11.42.10                      FABIUNLS - V1.R2

DDNAME        NBR OF ROOTS      NBR OF BYTES     SUBSET-ID
------        ------------      ------------     ---------
FKD1                  341             5,147     EED100
                                                  22D000
FKD2                  380             6,247     EED200
                                                  22D000
FKD3                  493             7,371     EED300
                                                  22D000
FKD4                  307             4,993     EED400
                                                  22D000
FKD5                  440             6,428     EED500
                                                  22D000
FKD6                  648             9,756     EED600
                                                  22D000



NBR OF ORPHANS=               0

DBDNAME   =SSRHDBD2
RANDOMIZER=SSRANDX1
```

Figure 70. Split Unloaded Data Set Statistics report (IDTYPE=X)

Note: When IDTYPE=X is specified on the FABITAB macro statement as input for the generation of the Sequential Subset Randomizer, the values in the subset ID fields are described as hexadecimal values using two lines. For example, the subset-ID values for FKD1 are described as follows:

```
            EED....
            22D.... .
```

These values are X' E2', X'E2', and X'DD'.

On the SYSPRINT data set, FABIUNLS prints a table containing one entry for each subset with the following information:

- The DD name of the output data set that contains the unloaded database records of the subset (for example, FKD1).
- The number of database roots in the subset.
- The total number of data bytes in all database segments that belong to the subset.

  Notes:

  – This number includes database segments of all data set groups.
  – If segments are compressed, this number reflects the data length of the decompressed segments.
  – The length of segment-prefixes is not included in this figure.
- The first 83 bytes of the subset ID value printed in EBCDIC or in hexadecimal according to the value of IDTYPE= keyword on the FABITAB macro statement.

FABIUNLS also prints how many database roots are orphans (that is, a root segment which does not belong to any subset). If VALTYPE=E is specified on the FABITAB macro statement, a root segment of which subset ID is not equal to any value of ID= keyword on the FABIDEF macro statement becomes an orphan. If VALTYPE=H is specified on the FABITAB macro statement, a root segment of which subset ID is higher than the maximum value of ID= keyword on the FABIDEF macro statement becomes an orphan. The number of orphans should be zero if the SSRGEN specifications (as input for the generation of the Sequential Subset Randomizer) have been done correctly.

FABIUNLS JCL example

This topic provides a JCL example for generating a database with the FABIUNLS utility program.

```
//*****************************************************
//* SPLIT THE UNLOADED-DB INTO MULTIPLE DATA SETS       *
//*****************************************************
//*
//SPLITUNL  EXEC FABHULU,MBR=FABIUNLS,DBD=DPHIN02
//STEPLIB   DD  DSN=HPS.SHPSLMD0,DISP=SHR
//          DD  DSN=IMSVS.SDFSRESL,DISP=SHR
//          DD  DSN=RANDOMIZ.LIB,DISP=SHR
//DFSRESLB  DD  DSN=IMSVS.SDFSRESL,DISP=SHR
//DFSVSAMP  DD  *
4096,8
/*
//SYSPRINT  DD  SYSOUT=A
//SYSUT1    DD  DSN=UNLOADED.DB,DISP=OLD
//HDR       DD  DSN=DPHIN02E.HDR,DISP=(,CATLG),
//              VOL=SER=WRK34B,UNIT=SYSDA,SPACE=(TRK,1),
//              DCB=(RECFM=VB,LRECL=4092,BLKSIZE=4096)
//FKD1      DD  DSN=DPHIN02E.FKD1,DISP=(,CATLG),
//              VOL=SER=WRK34B,UNIT=SYSDA,SPACE=(CYL,(40,10)),
//              DCB=*.HDR
//FKD2      DD  DSN=DPHIN02E.FKD2,DISP=(,CATLG),
//              VOL=SER=WRK34C,UNIT=SYSDA,SPACE=(CYL,(40,10)),
//              DCB=*.HDR
 ........
 ........
 ........
//FKD17     DD  DSN=DPHIN02E.FKD17,DISP=(,CATLG),
//              VOL=SER=WRK35A,UNIT=SYSDA,SPACE=(TRK,(1,10)),
//              DCB=*.HDR
//TRL       DD  DSN=DPHIN02E.TRL,DISP=(,CATLG),
//              VOL=SER=WRK35B,UNIT=SYSDA,SPACE=(TRK,1),
//              DCB=*.HDR
//*
//*****************************************************************
//*  DEFINE THE CLUSTER FOR THE DATABASE                           *
//*****************************************************************
//*
//ALLOCDB   EXEC PGM=IDCAMS
//SYSPRINT  DD   SYSOUT=A
//SYSIN     DD   *
    DEFINE       CLUSTER  (NAME(USER.DB) .....
/*
//*
//*****************************************************************
//* RELOAD THE DATABASE WITH THE SEQUENTIAL SUBSET RANDOMIZER   *
//*****************************************************************
//*
//RELOAD    EXEC PGM=DFSRRC00,PARM='ULU,DFSURGL0,DPHIN02'
//STEPLIB   DD  DSN=IMSVS.SDFSRESL,DISP=SHR
//          DD  DSN=RANDOMIZ.LIB,DISP=SHR
//DFSRESLB  DD  DSN=IMSVS.SDFSRESL,DISP=SHR
//IMS       DD  DSN=IMSVS.DBDLIB,DISP=SHR
//IEFRDER   DD  DUMMY
//DFSVSAMP  DD  DSN=IMSVS.PROCLIB(DFSVSAMP),DISP=SHR
//SYSPRINT  DD  SYSOUT=A
//DFSUINPT  DD  DSN=DPHIN02E.HDR,DISP=OLD
//          DD  DSN=DPHIN02E.FKD1,DISP=OLD
//          DD  DSN=DPHIN02E.FKD2,DISP=OLD
//          DD  ....
//          DD  ....
//          DD  DSN=DPHIN02E.FKD17,DISP=OLD
//          DD  DSN=DPHIN02E.TRL,DISP=OLD
//DB        DD  DSN=USER.DB,DISP=OLD
//SYSUDUMP  DD  SYSOUT=A
```

Figure 71. Example of database generation

Note: If the database is involved in logical relations or secondary indexing, additional DD statements are required by IMS for the reload job step.

Chapter 24. Obtaining statistics from each subset with Sequential Subset Statistics

The Sequential Subset Statistics (SS-STATS) routine (FABISTAT) provides database administrators with statistical information, which is useful in determining how much relative space in the root addressable area needs to be allocated to each subset.

For each subset, SS-STATS provides statistics about the number of database roots and about the total length of database segments within the subset.

The SS-STATS is provided as an exit routine of the IMS High Performance Unload database unload utilities (that is, FABHURG1 and FABHFSU).

Topics:

Obtaining statistics from each subset

By activating the SS-STATS routine (FABISTAT) in a FABHURG1 job or in a FABHFSU job, you can obtain statistics from each subset.

About this task

The general data flow for the SS-STATS routine (FABISTAT) is shown in the following figure.

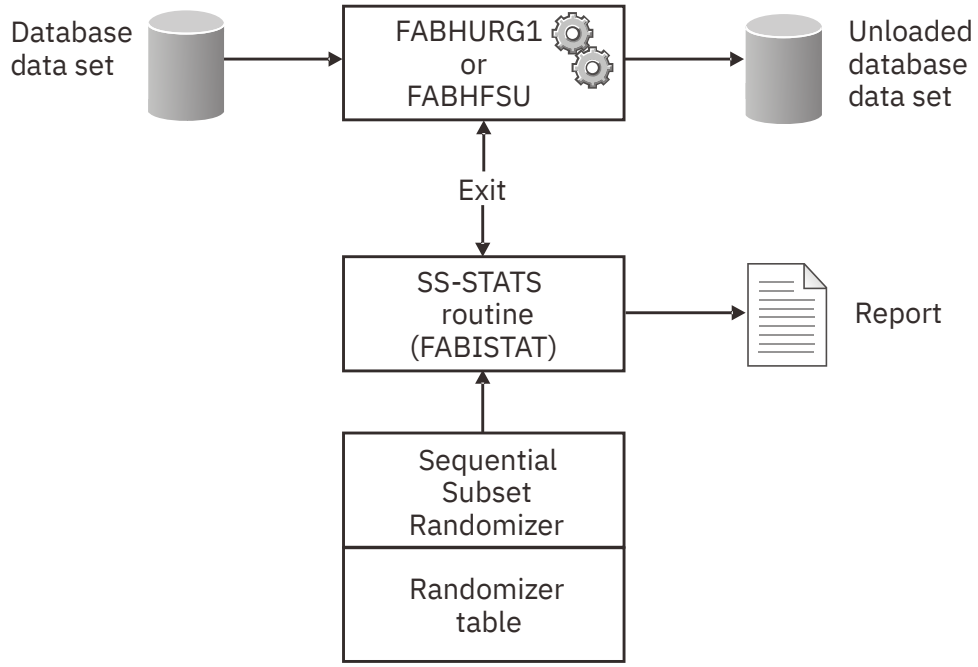

Figure 72. Data flow for the SS-STATS routine

Procedure

1. Generate the Sequential Subset Randomizer.
2. Prepare a FABHURG1 or FABHFSU JCL job and modify the JCL to meet the JCL requirements for using FABISTAT.

   For the FABISTAT JCL requirements, see "JCL requirements when SS-STATS routine is applied" on page 294.
3. In the HSSROPT data set, code the SSSTATS control statement to activate the SS-STATS routine.

   For the format of SSSTATS control statement, see "HSSROPT input data set when SS-STATS routine is applied" on page 294.
4. Run the FABHURG1 or FABHFSU job.

Related reference

JCL example to apply the SS-STATS routine
This topic provides a JCL example for database unload with SS-STATS.

JCL requirements when SS-STATS routine is applied

The SS-STATS routine is used within a FABHURG1 job or a FABHFSU job. Therefore, you must prepare a JCL for either of these unload utilities and modify the JCL to meet the JCL requirements for using the SS-STATS routine.

For the JCL requirements of FABHURG1 and FABHFSU utilities, see Chapter 4, "Basic job control language," on page 29.

The additional DD statements required to use the SS-STATS routine are described in the following table.

Table 50. Additional DD statements for SS-STATS routine

DDNAME	Use	Format	Need
HSSROPT	Input	LRECL=80	Required
HSSRSTAT	Output	LRECL=133	Required

HSSROPT DD
This required input data set contains optional control statements such as the SSSTATS control statement.

Usually, when the SS-STATS routine is activated, the database should already be an HDAM database using the Sequential Subset Randomizer. However, the SS-STATS routine can be activated even if the database is not randomized with the Sequential Subset Randomizer, or if it is not an HDAM database. In this case, the user needs to provide the name of a Sequential Subset Randomizer (which has been generated with an accurate description of the subsets and subset IDs) on the SSSTATS control statement.

HSSRSTAT DD
This required output data set is used for IMS High Performance Unload to write statistical reports. The reports of the SS-STATS routine are also written to the HSSRSTAT data set.

HSSROPT input data set when SS-STATS routine is applied

The SS-STATS routine is activated by the HSSROPT data set.

The HSSROPT data set contains optional control statements of IMS High Performance Unload. However, in this topic and in related topics, only the SSSTATS control statement that activates the SS-STATS routine is described. For the other IMS High Performance Unload input data sets, see the following topics:

- "FABHURG1 input" on page 41
- "FABHFSU input" on page 55

Format

This data set contains 80-byte fixed-length records. The control statements can be coded in the input stream or accessed as a member of a partitioned data set.

SSSTATS control statement

The SS-STATS routine is activated by specifying an SSSTATS control statement in the HSSROPT data set.

The format of the SSSTATS control statement is shown in the following figure.

```
//HSSROPT DD *
SSSTATS randname
/*
```

Figure 73. Activating the SS-STATS routine

The control statement in the HSSROPT data set must be coded as follows:

- Columns 1 - 7 must contain SSSTATS.
- Column 8 must be blank.
- Columns 9 - 16 must contain one of the following parameters:
  - Blank (if the database is an HDAM database using a Sequential Subset Randomizer).
  - If the database is not an HDAM database using a Sequential Subset Randomizer, the field must contain the load module name of a Sequential Subset Randomizer which has been generated with an accurate description of the subsets and subset IDs. The load module library that contains the Sequential Subset Randomizer must be accessible (for example, through a STEPLIB DD statement).

HSSRSTAT output data set when SS-STATS routine is applied

When the SS-STATS routine has been activated, IMS High Performance Unload produces an HSSRSTAT data set that contains printed statistical reports.

This data set contains the Sequential Subset Statistics report when the SS-STATS exit routine has been activated. For information about the other reports that this data set contains, see “HSSRSTAT data set” on page 181.

Format

This data set contains 133-byte fixed-length records. When the block size is coded in the JCL, the block size must be a multiple of 133. Code your DD statement as follows to obtain the statistics output of the SS-STATS routine:

```
//HSSRSTAT  DD  SYSOUT=A
```

Sequential Subset Statistics report

The Sequential Subset Statistics report contains statistics that are useful in determining how much relative space in the root addressable area needs to be allocated to each subset.

The following two figures show examples of Sequential Subset Statistics reports.

```
IMS HIGH PERFORMANCE UNLOAD - SSR                        "SEQUENTIAL SUBSET STATISTICS"                          PAGE:     1
5655-E06                                                DATE: 06/01/2021  TIME: 11.26.15                       FABISTAT - V1.R2

SUBSET NBR    NBR OF ROOTS      NBR OF BYTES     SUBSET-ID
----------    ------------      ------------     ---------
1                    2,026             3,500     10000
2                    1,030             4,280     12000
3                        0                 0     14000
4                        0                 0     16000
5                        0                 0     18000
6                    1,003             1,400     20000
7                        0                 0     22000
8                        0                 0     24000
9                    2,005               620     26000
10                   1,018               310     30000

IMS HIGH PERFORMANCE UNLOAD - SSR                        "SEQUENTIAL SUBSET STATISTICS"                          PAGE:     2
5655-E06                                                DATE: 06/01/2021  TIME: 11.26.15                       FABISTAT - V1.R2

SUBSET NBR      % OF ROOTS        % OF BYTES     SUBSET-ID
----------      ----------        ----------     ---------
1                    28.58             34.61     10000
2                    14.30             42.33     12000
3                     0.00              0.00     14000
4                     0.00              0.00     16000
5                     0.00              0.00     18000
6                    14.26             13.84     20000
7                     0.00              0.00     22000
8                     0.00              0.00     24000
9                    28.56              6.13     26000
10                   14.28              3.06     30000



NBR OF ORPHANS=              121

DBDNAME   =SSRHDBD3
RANDOMIZER=SSRANDC2
```

Figure 74. Sequential Subset Statistics report (IDTYPE=C)

```
IMS HIGH PERFORMANCE UNLOAD - SSR                        "SEQUENTIAL SUBSET STATISTICS"                          PAGE:     1
5655-E06                                                DATE: 06/01/2021  TIME: 10.06.31                       FABISTAT - V1.R2

SUBSET NBR    NBR OF ROOTS      NBR OF BYTES     SUBSET-ID
----------    ------------      ------------     ---------
1                      30              15,645     EED00000FFFF
                                                  22D010000001
2                     121              62,990     EED00000FFFF
                                                  22D020000002
3                      19              15,642     EED00000FFFF
                                                  22D030000003
4                     345             179,760     EED00000FFFF
                                                  22D040000004
5                      83              43,226     EED00000FFFF
                                                  22D050000005

IMS HIGH PERFORMANCE UNLOAD - SSR                        "SEQUENTIAL SUBSET STATISTICS"                          PAGE:     2
5655-E06                                                DATE: 06/01/2021  TIME: 10.06.31                       FABISTAT - V1.R2

SUBSET NBR      % OF ROOTS        % OF BYTES     SUBSET-ID
----------      ----------        ----------     ---------
1                     5.02              4.93     EED00000FFFF
                                                  22D010000001
2                    20.23             19.85     EED00000FFFF
                                                  22D020000002
3                     3.18              4.93     EED00000FFFF
                                                  22D030000003
4                    57.69             56.66     EED00000FFFF
                                                  22D040000004
5                    13.88             13.62     EED00000FFFF
                                                  22D050000005



NBR OF ORPHANS=              154

DBDNAME   =SSRHDBD4
RANDOMIZER=SSRANDX2
```

Figure 75. Sequential Subset Statistics report (IDTYPE=X)

Note: When IDTYPE=X is specified on the FABITAB macro statement as input for the generation of the Sequential Subset Randomizer, the values in the subset ID fields are described as hexadecimal values using two lines. For example, the subset-ID values for FKD1 are described as follows:

```
EED....
22D.... .
```

These values are X' E2', X'E2', and X'DD'.

The SS-STATS routine prints a table containing one table entry for each subset with the following information on the HSSRSTAT data set:

- The relative subset number.
- The number of database roots in the subset.

- The total number of data bytes and prefix bytes in all database segments that belong to the subset.

  Notes:

  – This number includes only database segments of the first data set group.

  – If segments are compressed, this number reflects the data length of the compressed segments (when stored on DASD).

  – The length of segment prefixes is included in this figure.
- The first 83 bytes of the subset ID value printed in EBCDIC or in hexadecimal according to the value of IDTYPE= keyword on the FABITAB macro statement.

The SS-STATS routine prints a second table where the number of root segments and the number of data bytes are expressed as the percentage of all databases.

The SS-STATS routine also prints how many database roots are orphans. An orphan is a root segment which does not belong to any subset. The number of orphans should be zero, if the SSRGEN specifications (as input for the generation of the Sequential Subset Randomizer) have been done correctly.

JCL example to apply the SS-STATS routine

This topic provides a JCL example for database unload with SS-STATS.

```
//UNLOAD  EXEC FABHULU,MBR=FABHURG1,DBD=DPANL0
//STEPLIB   DD DSN=HPS.SHPSLMD0,DISP=SHR
//          DD DSN=IMSVS.SDFSRESL,DISP=SHR
//          DD DSN=RANDOMIZ.LIB,DISP=SHR
//DFSVSAMP  DD DSN=IMSVS.PROCLIB(DFSVSAMP),DISP=SHR
//HSSRCABP  DD  *
CABDD *HD
/*
//HSSROPT   DD  *
SSSTATS randname
/*
//HSSRSTAT  DD SYSOUT=A
//HDAM      DD DSN=HDAM.DB,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//SYSUT2    DD DSN=UNLOAD.HDAM.DB,DISP=(,CATLG),
//             UNIT=TAPE
```

Figure 76. Example of JCL for database unload with SS-STATS

In this example:

- The STEPLIB DD statement provides access to the load library containing the Sequential Subset Randomizer.
- The SSSTATS control statement in the HSSROPT data set requests the activation of the SS-STATS routine, which prints statistics on the HSSRSTAT data set for each subset.
- A randomizer load module name is specified on the SSSTATS control statement because the database to be unloaded is not yet randomized with the Sequential Subset Randomizer. The specified Sequential Subset Randomizer must have been generated with an exact description of all subsets and subset ID values.

Chapter 25. Converting databases to HDAM databases randomized with the Sequential Subset Randomizer

Converting existing databases into the databases randomized with the Sequential Subset Randomizer requires a special database reorganization.

Note: If the database is involved in logical relationships or secondary relationships, IMS might require the execution of additional reorganization utilities (such as the IMS Database Prereorganization utility, the IMS Database Scan utility, the IMS Database Prefix Resolution utility, and so on) during the database reorganization. The execution of these utilities is not mentioned in this guide, since their execution is a standard IMS activity.

Topics:

Converting from a database randomized with DFSHDC40

You can convert from a database that is randomized with DFSHDC40 into an HDAM database randomized with the Sequential Subset Randomizer.

Procedure

Converting an existing DFSHDC40-HDAM database into an HDAM database randomized with the Sequential Subset Randomizer requires the following steps:

1. Generate a temporary Sequential Subset Randomizer.

   For this temporary Sequential Subset Randomizer, the database administrator does not need to define accurately how much space should be allocated to each subset. However, all other specifications need to be accurate. Note that at this time the DBDGEN should not be changed.

   This temporary Sequential Subset Randomizer is required so that the SS-STATS routine produces statistics for each subset. The SS-STATS routine finds the description of all subsets (including the values of the subset IDs) in the tables of the temporary Sequential Subset Randomizer to make statistics.
2. Unload the database using FABHURG1 or FABHFSU with the SSSTATS control statement specified (see Figure 76 on page 297 for an example).

   The database unload will produce the following:

   - An unloaded database which will be used as input for the reload.
   - Statistics that describe how much space should be allocated to each subset. These statistics should be used for a second, definitive SSRGEN which will be used during the reload process.

   It is recommended that you activate the Database Tuning Statistics during this database unload. (See Chapter 26, "Obtaining statistics for database tuning," on page 307.) The Database Tuning Statistics will show whether the database is well organized and whether the current randomizing is efficient. If the current randomizing is inefficient, the database administrator can take corrective actions before reloading the database. For example, if the root addressable area is too crowded, the database administrator can increase the size of the root addressable area. The Database Tuning Statistics can also be compared with the Database Tuning Statistics created at a later time, when the database is randomized with the Sequential Subset Randomizer. This can be used to see whether the Sequential Subset randomizing is as efficient as DFSHDC40 randomizing.

3. Generate the final version of the Sequential Subset Randomizer.

   For this second Sequential Subset Randomizer, the database administrator should provide accurate specifications about the relative amount of space to be allocated to each subset. The relative amount of space to be allocated to each subset can be found in the SS-STATS statistics in the previous database unload.

   If the database administrator knows in advance that some subsets will eventually incur an overproportional increase in data volume, the data administrator can use this knowledge in order to increase the relative amount of space allocated to that subset. This may avoid a later adjustment of Sequential Subset Randomizer tables and an associated database reorganization.
4. Change the DBD source statements in order to specify the name of the generated Sequential Subset Randomizer as the randomizer.

   This is a good time to review the current randomizing parameters and make adjustments with existing Database Tuning Statistics or other statistics if necessary. You can review the number of CIs/blocks in the root addressable area, the number of RAPs per CI/block, the CI/block size, the bytes limit, the scan parameter, and the free space specifications.

   Then the DBDGEN must be performed and the ACBGEN should be performed if necessary.
5. Create sorted input for the database reload.

   For small databases with less than 10,000 roots, this step is not necessary.

   For large databases, the input to the reload should be provided in the correct RAP sequence of the new database in order to complete the reload in a reasonable amount of time. For conversions from DFSHDC40 to Sequential Subset Randomizer, the correct sequence for the reload can be created without lengthy sort by executing the FABIUNLS utility (see Figure 71 on page 292 for an example). The FABIUNLS utility splits the unloaded data set into multiple data sets so that one subset becomes one data set.

   Therefore, the input to the reload will consist of a concatenation of the multiple output data sets of FABIUNLS.
6. Reload the database with the new or modified DBD using the Sequential Subset Randomizer.
7. Obtain Database Tuning Statistics to check whether the quality of the randomizing is appropriate.

   This can be done by unloading a database, activating the Database Tuning Statistics for this unload, and setting the output of the unload to dummy.

Converting from a database randomized with other randomizers

You can convert from a database that is randomized with other randomizer into an HDAM database randomized with the Sequential Subset Randomizer.

Procedure

Converting an existing HDAM database that is randomized with other randomizer into an HDAM database randomized with the Sequential Subset Randomizer requires the following steps:

1. Generate a temporary Sequential Subset Randomizer.

   For this temporary Sequential Subset Randomizer, the database administrator does not need to define accurately how much space should be allocated to each subset. However, all other specifications need to be accurate. Note that at this time the DBDGEN should not be changed.

   This temporary Sequential Subset Randomizer is required so that the SS-STATS routine produces statistics for each subset. The SS-STATS routine finds the description of all subsets (including the values of the subset IDs) in the tables of the temporary Sequential Subset Randomizer to make statistics.
2. Unload the database using FABHURG1 or FABHFSU with the SSSTATS control statement specified (see Figure 76 on page 297 for an example).

The database unload will produce the following:

- An unloaded database which will be used as input for the reload.
- Statistics that describe how much space should be allocated to each subset. These statistics should be used for a second, definitive SSRGEN which will be used during the reload process.

It is recommended that you activate the Database Tuning Statistics during this database unload. (See Chapter 26, “Obtaining statistics for database tuning,” on page 307.) The Database Tuning Statistics will show whether the database is well organized and whether the current randomizing is efficient. If the current randomizing is inefficient, the database administrator can take corrective actions before reloading the database. For example, if the root addressable area is too crowded, the database administrator can increase the size of the root addressable area. The Database Tuning Statistics can also be compared with the Database Tuning Statistics created at a later time, when the database is randomized with the Sequential Subset Randomizer. This can be used to see whether the Sequential Subset randomizing is as efficient as DFSHDC40 randomizing.

3. Generate the final version of the Sequential Subset Randomizer.

   For this second Sequential Subset Randomizer, the database administrator should provide accurate specifications about the relative amount of space to be allocated to each subset. The relative amount of space to be allocated to each subset can be found in the SS-STATS statistics in the previous database unload.

   If the database administrator knows in advance that some subsets will eventually incur an overproportional increase in data volume, the data administrator can use this knowledge in order to increase the relative amount of space allocated to that subset. This may avoid a later adjustment of Sequential Subset Randomizer tables and an associated database reorganization.
4. Change the DBD source statements in order to specify the name of the generated Sequential Subset Randomizer as the randomizer.

   This is a good time to review the current randomizing parameters and make adjustments with existing Database Tuning Statistics or other statistics if necessary. You can review the number of CIs/blocks in the root addressable area, the number of RAPs per CI/block, the CI/block size, the bytes limit, the scan parameter, and the free space specifications.

   Then the DBDGEN must be performed and the ACBGEN should be performed if necessary.
5. Create sorted input for the database reload.

   The input to the reload should be provided in the correct RAP sequence of the new database in order to complete the reload in a reasonable amount of time.

   Conversion from other randomizer to Sequential Subset Randomizer must be done with the Physical Sequence Sort for Reload (PSSR) utility of IMS High Performance Load or by an equivalent program. The DBD created during the DBDGEN of the prior step must be used for PSSR. For details, see the *IMS High Performance Load User's Guide*.

   Remember that sorting a large number of database records can be time-consuming.
6. Reload the database with the new or modified DBD using the Sequential Subset Randomizer.
7. Obtain Database Tuning Statistics to check whether the quality of the randomizing is appropriate.

   This can be done by unloading a database, activating the Database Tuning Statistics for this unload, and setting the output of the unload to dummy.

Converting from a HISAM or HIDAM

You can convert from a HISAM or HIDAM database into an HDAM database randomized with the Sequential Subset Randomizer.

Before you begin

After conversion from an HISAM or HIDAM, the database records can no longer be retrieved any more in the logical sequence of the root key unless the database administrator defines a secondary index for

this purpose. The sequential retrieval of a large number of database records through a secondary index is usually not efficient.

Procedure

Converting an existing HISAM or HIDAM database into an HDAM database randomized with the Sequential Subset Randomizer requires the following steps:

1. Generate a temporary Sequential Subset Randomizer.

   For this temporary Sequential Subset Randomizer, the database administrator does not need to define accurately how much space should be allocated to each subset. However, all other specifications need to be accurate. Note that at this time the DBDGEN should not be changed.

   This temporary Sequential Subset Randomizer is required so that the SS-STATS routine produces statistics for each subset. The SS-STATS routine finds the description of all subsets (including the values of the subset IDs) in the tables of the temporary Sequential Subset Randomizer to make statistics.
2. Unload the database using FABHURG1 or FABHFSU with the SSSTATS control statement specified (see Figure 76 on page 297 for an example).

   The database unload will produce the following:

   - An unloaded database which will be used as input for the reload.
   - Statistics that describe how much space should be allocated to each subset. These statistics should be used for a second, definitive SSRGEN which will be used during the reload process.

   It is recommended that you activate the Database Tuning Statistics during this database unload. (See Chapter 26, “Obtaining statistics for database tuning,” on page 307.) The Database Tuning Statistics will show whether the database is well organized and whether the current randomizing is efficient. If the current randomizing is inefficient, the database administrator can take corrective actions before reloading the database. For example, if the root addressable area is too crowded, the database administrator can increase the size of the root addressable area. The Database Tuning Statistics can also be compared with the Database Tuning Statistics created at a later time, when the database is randomized with the Sequential Subset Randomizer. This can be used to see whether the Sequential Subset randomizing is as efficient as DFSHDC40 randomizing.
3. Generate the final version of the Sequential Subset Randomizer.

   For this second Sequential Subset Randomizer, the database administrator should provide accurate specifications about the relative amount of space to be allocated to each subset. The relative amount of space to be allocated to each subset can be found in the SS-STATS statistics in the previous database unload.

   If the database administrator knows in advance that some subsets will eventually incur an overproportional increase in data volume, the data administrator can use this knowledge in order to increase the relative amount of space allocated to that subset. This may avoid a later adjustment of Sequential Subset Randomizer tables and an associated database reorganization.
4. Change the DBD source statements in order to specify the name of the generated Sequential Subset Randomizer as the randomizer.

   This is a good time to review the current randomizing parameters and make adjustments with existing Database Tuning Statistics or other statistics if necessary. You can review the number of CIs/blocks in the root addressable area, the number of RAPs per CI/block, the CI/block size, the bytes limit, the scan parameter, and the free space specifications.

   Then the DBDGEN must be performed and the ACBGEN should be performed if necessary.
5. Create sorted input for the database reload.

   The input to the reload should be provided in the correct RAP sequence of the new database in order to complete the reload in a reasonable amount of time.

Conversion from a HISAM or HIDAM database into an HDAM must be done with the Physical Sequence Sort for Reload (PSSR) utility of IMS High Performance Load or by an equivalent program. The DBD created during the DBDGEN of the prior step must be used for PSSR. For details, see the *IMS High Performance Load User's Guide*.

Remember that sorting a large number of database records can be time-consuming.

6. Reload the database with the new or modified DBD using the Sequential Subset Randomizer.
7. Obtain Database Tuning Statistics to check whether the quality of the randomizing is appropriate.

   This can be done by unloading a database, activating the Database Tuning Statistics for this unload, and setting the output of the unload to dummy.

Part 5. Tuning databases by using Database Tuning Statistics reports

HSSR Engine can generate database statistics reports which help you tune your databases.

Notes:

- Descriptions in the following topics for an HDAM database also applies to each partition of a PHDAM database.
- Descriptions in the following topics for a HIDAM database also applies to each partition of a PHIDAM database.

Topics:

Chapter 26. Obtaining statistics for database tuning

You can request HSSR Engine to create optional Database Tuning Statistics reports during the sequential processing or unloading of a database. This functionality is provided as the *Database Tuning Statistics* function.

By activating the Database Tuning Statistics function, you can generate the following statistics reports:

- DB Statistics report
- Randomizing Statistics report

You can use these reports for the following purposes:

- Periodically monitor the need for reorganization of HDAM, HIDAM, and HISAM databases.
- Monitor the effectiveness of HDAM randomizing parameters (such as the size of the root addressable area, the number of RAPs, the block/CI size, or the bytes limit).

The Database Tuning Statistics function provides the following additional capabilities to aid your database tuning:

- Creating the optional DB Record Length Distribution report with your own definition of the database record length ranges.
- Creating the HSSRLOUT data set to record the length of each database record.
- Using the FABHLDBR utility to print long database records with the HSSRLOUT data set as an input.

Tip: You can use the FABHEXTR exit routine of the FABHURG1 utility to create a small database extract that can be used to perform database tuning experiments. Creation of a smaller extract database can be useful when the database administrator intends to tune the randomizing parameters of a large HDAM database through an iterative try-and-see process. For information about the FABHEXTR exit routine, see Chapter 29, “Creating a database extract for tuning experiments,” on page 351.

Topics:

Activating the Database Tuning Statistics

You can activate the Database Tuning Statistics function by adding the DBSTATS control statement in the HSSROPT data set. The Database Tuning Statistics function is provided by HSSR Engine.

Procedure

1. Prepare the basic JCL for HSSR Engine.

   For instructions, see “Preparing the basic JCL” on page 29.
2. Specify the additional DD statements that are required for the Database Tuning Statistics function.

   For a list of the additional DD statements that are required for Database Tuning Statistics, see “JCL requirements for the Database Tuning Statistics” on page 308.
3. Specify the DBSTATS control statement in the HSSROPT data set to activate the Database Tuning Statistics function.

   For the format and the parameter of the DBSTATS control statement, see “DBSTATS control statement” on page 309.

4. Optional: If you want to select the records to be written in the HSSRLOUT data set by the length of the database record or by the number of database I/Os that are required to read all database segments of the database record, specify the LOUT control statement in the HSSROPT data set.

   For the format and the parameter of the LOUT control statement, see "LOUT control statement" on page 309.
5. Optional: If you want to change the unit of measure for describing the ranges of database record lengths in the report, specify the HSSRLDEF data set.

   For information about the HSSRLDEF data set, see "HSSRLDEF input data set for Database Tuning Statistics" on page 310.
6. Run the job.

   When the job completes, statistics reports are written in the HSSRSTAT data set and information about each database record is written in the HSSRLOUT data set.

   The HSSRLOUT data set can be used as input for the FABHLDBR utility to obtain information about long database records. For more information about using the FABHLDBR utility, see Chapter 27, "Printing long database records," on page 313.

   These statistics reports can be used for tuning your databases. For a guideline for tuning your databases with these reports, see Chapter 28, "Tuning a database with the Database Tuning Statistics," on page 321.

Example

See "JCL example for Database Tuning Statistics and FABHLDBR" on page 318 for a JCL example to activate the Database Tuning Statistics function and print long database records.

JCL requirements for the Database Tuning Statistics

The Database Tuning Statistics function is provided by HSSR Engine. Therefore, it must meet the requirements for the basic JCL (FABHX034 JCL). In addition, Database Tuning Statistics JCL requires other DD statements.

Prerequisite: See "Basic JCL requirements" on page 30 for the basic (FABHX034) JCL requirements.

The following table summarizes additional JCL requirements for running the Database Tuning Statistics function.

Table 51. DD statements for Database Tuning Statistics

DDNAME	Use	Format	Need
HSSROPT	Input	LRECL=80	Required
HSSRLDEF	Input	LRECL=80	Optional
HSSRSTAT	Output	LRECL=133	Optional
HSSRLOUT	Output	-	Optional

HSSROPT DD
This statement defines the input data set that contains the DBSTATS control statement, which activates the Database Tuning Statistics. The data set also contains the optional LOUT statement, which is used for HSSRLOUT records selection.

HSSRLDEF DD
This optional statement defines the input data set that contains control statements for requesting the DB Record Length Distribution report with your own database record length ranges.

HSSRSTAT DD
This optional statement defines the output data set for the statistical output.

HSSRLOUT DD

This optional statement defines the output sequential data set, in which a small record for each database record is written. Each record of the data set consists of:

- Length of the database record (fullword)
- The number of database I/Os required to read all database segments of the database record (fullword)
- Key of the database root segment

Note: Do not specify the BLKSIZE and LRECL for the HSSRLOUT data set on JCL. These values are determined dynamically by HSSR Engine based on the key length of the root segment. The LRECL is 8 bytes plus the length of the key of the database root segment.

Input for Database Tuning Statistics

The input for the Database Tuning Statistics function consists of the HSSROPT data set and the HSSRLDEF data set.

HSSROPT input data set for Database Tuning Statistics

Database Tuning Statistics uses the control statements in the HSSROPT data set as input.

DBSTATS control statement

The DBSTATS control statement activates the Database Tuning Statistics function.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

DBSTATS nnnnn
```

Position

Description

1

Code the DBSTATS keyword to instruct HSSR Engine to provide the Database Tuning Statistics report.

9

Specify the number of buffers to be simulated with this entry. Enter any number up to five digits in the range of 1 - 32767, left-aligned, and followed by a blank. If this entry is left blank, the default value of 4 is used.

HSSR Engine simulates the LRU algorithms of the IMS OSAM buffer pool and the IMS VSAM buffer pool to provide statistics of the number of I/O operations. If the application program uses multiple HSSR PCBs, HSSR Engine assumes that each PCB has its own dedicated buffer pool containing the user-specified or default number of buffers.

Restrictions:

- If APISET 3 is specified, this statement cannot be specified.
- If one or more partitions of PHDAM or PHIDAM are in the HALDB OLR cursor-active status, this statement is ignored.

LOUT control statement

The LOUT control statement requests that, when the DBSTATS control statement activates the Database Tuning Statistics function, the HSSRLOUT data set contains only the records that satisfy certain conditions.

The records that satisfy one or both of the following conditions are written in the HSSRLOUT data set:

- Database records that are longer than the specified limit.
- Database records that require more than the specified number of database I/Os.

This option is ignored if the DBSTATS control statement is not specified at the same time.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

LOUT LENGTH=llllllll
LOUT IO=nn
```

Position
Description

1
Code the LOUT keyword to request an optional record selection for the HSSRLOUT data set.

6
Code one of the following optional keywords for each LOUT statement:

LENGTH=*llllllll*
Requests that only the records for database records whose length is greater than *llllllll* bytes are written in the HSSRLOUT data set.

IO=*nn*
Requests that only the records for database records that require more than *nn* database I/Os for retrieval are written in the HSSRLOUT data set.

You can specify both LENGTH= and IO= parameters, either on a single LOUT control statement or separately (on multiple LOUT statements). If you specify both parameters on a single LOUT statement, separate them with a comma, as follows:

```
LOUT LENGTH=100,IO=15
```

If you specify both LENGTH= and IO= parameters, both conditions are effective (that is, database records that satisfy both conditions are written in the HSSRLOUT data set.)

If an incorrect parameter keyword or an incorrect parameter value is detected on an LOUT statement, the rest of the parameter specification on that statement is ignored. For example, the IO= parameter specification on the following statement is ignored because the LENGTH= parameter value is incorrect:

```
LOUT LENGTH=1A,IO=10
```

HSSRLDEF input data set for Database Tuning Statistics

The HSSRLDEF input data set requests to generate the DB Record Length Distribution report with your own database record length ranges.

Database Tuning Statistics provides, by default, a report of the distribution of database record lengths. In this default report, the ranges of database record lengths are denoted as 1/16th, 2/16th, 3/16th, and so on, of the CI/block size. However, these default ranges are not always appropriate for your use.

If you want a report with a different unit of measure for describing ranges of database record lengths, you must:

1. Provide an HSSRLDEF DD statement in your JCL.
2. Provide control statements in the HSSRLDEF data set.

   You must provide one control statement for each range of database record length that should appear in the statistics report. Each control statement contains the highest value of the range.

   Each value must start in column 1 of the control statement. The values must be provided in ascending sequence.

For example, if you want to have the DB Record Length Distribution report for the ranges 0 - 50 bytes, 51 - 100 bytes, 101 - 200 bytes, 201 - 300 bytes, 301 - 1000 bytes, and 1001 - 10000 bytes, the following control statements are required:

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

//HSSRLDEF DD *
50
100
200
300
1000
10000
```

Output from the Database Tuning Statistics

The outputs from the Database Tuning Statistics function are generated in the HSSRSTAT data set and the HSSRLOUT data set.

HSSRSTAT output data set for Database Tuning Statistics

The Database Tuning Statistics function generates statistical reports in the HSSRSTAT data set.

The reports include:

- DB Statistics report
- Randomizing Statistics report
- DB Record Length Distribution report (optional)

You can use these reports to tune your database. For a complete description for the reports, see Chapter 28, “Tuning a database with the Database Tuning Statistics,” on page 321.

HSSRLOUT output data set for Database Tuning Statistics

The HSSRLOUT data set is a sequential data set in which a small record for each database record is written.

Each record consists of:

- Length of the database record (fullword)
- The number of database I/Os required to read all database segments of the database record (fullword)
- Key of the database root segment

The HSSRLOUT data set can be:

- Printed as is (for example, with a PRINT command of IDCAMS).
- Printed (for example, with the PRINT command of IDCAMS) after a sort. The records are sorted in descending sequence of the first 4 bytes of each record. This generates a list of records ordered in decreasing sequence of their database record lengths.
- Processed by the FABHLDBR utility (to obtain a printed output of only the longest database records or database records that require the largest number of database I/Os). See Chapter 27, “Printing long database records,” on page 313 for details.

As an option, you can request that the HSSRLOUT data set only contain a record for:

- Database records that are longer than the specified limit
- Database records that require more than the specified number of database I/Os

You can achieve this by providing the LOUT control statement in the HSSROPT data set. See “LOUT control statement” on page 309 for details.

Tip: Because the records of the HSSRLOUT data set do not contain database names, you might find difficulties in interpreting and processing the content of the HSSRLOUT data set that was created during the execution of a program that accesses multiple databases. For this reason, it is recommended that you create the HSSRLOUT data set only during the execution of programs or utilities that access a single database.

Chapter 27. Printing long database records

You can use the FABHLDBR utility to print the longest database records of a database or the database records that require a large number of I/Os for their retrieval.

Prerequisite: The input for the FABHLDBR utility is the HSSRLOUT data set, which is created by the Database Tuning Statistics function during an execution of HSSR Engine. To generate the HSSRLOUT data set, see the following topics:

- For information about the HSSRLOUT data set, see "JCL requirements for the Database Tuning Statistics" on page 308.
- For information about the LOUT control statement to select the records in HSSRLOUT data set, see "LOUT control statement" on page 309.

The HSSRLOUT data set contains one record for each database record with the following data:

- Length of the database record (fullword)
- The number of database I/Os required to read all database segments of the database record (fullword)
- Key of the database root segment

For each record in the HSSRLOUT data set, the FABHLDBR utility issues:

- A GU call in order to retrieve the database root segment
- GN calls to retrieve dependent segments of the database record (optional)

Each retrieved database segment is printed in a snap-like fashion (the printing of the database segments is not done directly by the FABHLDBR utility, but by the hardcopy trace option). The FABHLDBR utility also provides, in the printed output, the length of each database record (the length includes the length of the segment prefixes) and the number of I/Os required to read all database segments of the database record.

Optionally, you can request on the SYSIN data set that FABHLDBR:

- Stops processing after retrieving or printing *nnnn* database records.
- Processes only the HSSRLOUT records for the database records that are longer than or equal to *mmmmmmm* bytes.
- Processes only the HSSRLOUT records for the database records that require *nn* or more I/Os.
- Retrieves or prints only the database root segments.

Topics:

Printing long database records with FABHLDBR

You can print long database records by using the FABHLDBR utility. The FABHLDBR utility runs as an HSSR application.

Procedure

1. Prepare the basic JCL for HSSR Engine.

   For instructions, see "Preparing the basic JCL" on page 29.
2. Specify the additional DD statements that are required for the FABHLDBR utility.

For a list of the additional DD statements that are required for FABHLDBR, see "FABHLDBR JCL requirements" on page 314.

3. Specify the TRDB and the TRHC control statements in the HSSROPT data set to activate the hardcopy trace option.

   For the formats and the parameters of the control statements, see "FABHLDBR HSSROPT input data set" on page 315.

4. Run the job.

   When the job completes, statistics reports are written in the HSSRSTAT data set.

Example

See "JCL example for Database Tuning Statistics and FABHLDBR" on page 318 for a JCL example to run the FABHLDBR utility.

FABHLDBR JCL requirements

The FABHLDBR utility uses HSSR Engine. Therefore, it must meet the requirements for the basic JCL (FABHX034 JCL). In addition, FABHLDBR JCL requires other DD statements.

Prerequisite: See "Basic JCL requirements" on page 30 for the basic (FABHX034) JCL requirements.

The following table summarizes additional JCL requirements for FABHLDBR.

Table 52. FABHLDBR DD statements

DDNAME	Use	Format	Need
HSSROPT	Input	LRECL=80	Required
SYSIN	Input	LRECL=80	Optional
SYSUT1	Input	-	Required
ddname	Input	-	Required
HSSRTRAC	Output	LRECL=133	Required

EXEC
This statement invokes the procedure FABHDLI, FABHDBB, or FABHULU. The EXEC statement must be in one of the following formats:

```
//  EXEC FABHDLI,MBR=FABHLDBR,PSB=psbname
//  EXEC FABHDBB,MBR=FABHLDBR,PSB=psbname
//  EXEC FABHULU,MBR=FABHLDBR,DBD=dbdname
```

HSSROPT DD
This statement is required to define the input data set that contains the TRDB and TRHC control statements, which activate the hardcopy trace option.

SYSIN DD
This optional statement defines the input data set that contains the control statements for controlling the FABHLDBR process.

SYSUT1 DD
This required statement defines the input HSSRLOUT data set.

***ddname* DD**
This required statement defines the input IMS database data set.

HSSRTRAC DD
This required statement defines the output data set in which the snap-like print of long database records is written.

FABHLDBR input

The control statements for the FABHLDBR utility must be specified in the HSSROPT data set and the SYSIN data set.

FABHLDBR HSSROPT input data set

The FABHLDBR utility uses the hardcopy trace option to print database segments. You must specify the TRDB and TRHC control statements in the HSSROPT data set to activate the hardcopy trace option.

TRDB control statement

The TRDB control statement activates the trace of HSSR Engine. Specify this control statement to obtain the snap-like print of database segments.

The TRDB control statement must be used with the TRHC control statement. After the TRHC indicates the type of trace to be run, the TRDB control statement then identifies the specific databases on which these traces are run.

```
0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

TRDB *ALL
     dbdname1,dbdname2,dbdname3,dbdname4
```

Position
Description

1
Code the TRDB keyword to activate the hardcopy tracing for specified database.

6
Enter either *dbdnames* separated by commas or the keyword *ALL.

Each *dbdname* must meet the following requirements:

- It must occupy eight bytes and be left-aligned.
- If the name is less than eight characters, trailing blanks must be specified.
- The last *dbdname* must be followed by a blank.

If multiple TRDB statements are provided, only the last statement is used.

TRHC control statement

The TRHC control statement activates the hardcopy trace option of HSSR Engine. Use this control statement to obtain the snap-like print of database segments.

This control statement is always used in combination with the TRDB control statement. When you activate this option, HSSR Engine traces calls that are issued against the databases named in the TRDB control statement. It writes data about these calls on the HSSRTRAC data set.

```
0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

TRHC CALL,CB,CBX,BUF,BUFCB,START=n,NPF
```

Position
Description

1
Code the TRHC keyword to activate the hardcopy tracing option.

6
Enter one or more of the following keywords, separated by commas, specified in any order. The last keyword must be followed by a blank.

CALL
Traces call information such as call function, PCB, IOAREA, SSA, and segment prefix.

Does a trace for the EXEC DLI command, just the same as for the DL/I call. This is because the command is converted to the format of a DL/I call and made to the HSSR call handler.

CB or CBX
Traces the control blocks of HSSR Engine. If CBX is specified, traces are also done for the extended control blocks of HSSR Engine.

If both CB and CBX are specified, CBX is taken.

BUF
Traces buffer handler information.

BUFCB
Traces the CAB buffer handler control blocks.

START=*n*
Specifies the call number *n*. Trace begins at the *n*-th HSSR call issued by the application program. Enter any number up to nine digits, left-aligned, and followed by a blank.

NPF
Excludes the segment prefix information from the trace.

Only HSSR calls issued against the databases that are specified on the TRDB control statement are traced.

Note: The trace of the control blocks of HSSR Engine, which is called for by specifying the CB, CBX, BUF, or BUFCB option for the TRHC control statement, is not intended to be reviewed by users, but might be needed by the IBM Software Support to analyze a problem.

FABHLDBR SYSIN input data set

The SYSIN data set for FABHLDBR controls the behavior of the FABHLDBR program.

This data set is an optional data set. By specifying control statements in the SYSIN data set, you can request FABHLDBR to take the following specific actions:

- Stop processing after retrieving or printing *nnnnn* database records.
- Process only the HSSRLOUT records for those database records that are longer than or equal to *mmmmmmm* bytes.
- Process only the HSSRLOUT records for those database records that require *nn* or more I/Os.
- Retrieve or print only the database root segments.

NBR control statement

The NBR control statement requests FABHLDBR to stop processing after retrieving or printing the specified number of database records.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

NBR nnnnn
```

Position
Description

1
Code the NBR keyword to activate the NBR option.

5
Specify the number of database records to be processed by FABHLDBR. After the specified number of database records are retrieved or printed, FABHLDBR stops processing. Enter any number up to 5 digits, left-aligned, and followed by a blank.

LENGTH control statement

The LENGTH control statement requests FABHLDBR to process only HSSRLOUT records for those database records that are longer than or equal to the specified length.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

LENGTH mmmmmmm
```

Position
Description

1
Code the LENGTH keyword to activate the LENGTH option.

8
Specify the length of a database record. FABHLDBR processes only the HSSRLOUT records that correspond to the database records that are longer than or equal to the specified length. Enter any number up to 7 digits, left-aligned, and followed by a blank.

IO control statement

The IO control statement requests FABHLDBR to process only HSSRLOUT records that correspond to the database records that require *nn* or more I/Os.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

IO nn
```

Position
Description

1
Code the IO keyword to activate the IO option.

4
Specify the number of I/Os. FABHLDBR processes only HSSRLOUT records for those database records that require more than or equal to the specified number of I/Os. Enter any number up to 2 digits, left-aligned, and followed by a blank.

For an HIDAM database, the number of I/Os required to read the index database is not included in this value. For a HISAM database, HSSR Engine assumes that access to the KSDS requires only one I/O.

ROOT-ONLY control statement

The ROOT-ONLY control statement requests FABHLDBR to retrieve or print only the database root segments.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

ROOT-ONLY
```

Position
Description

1
Code the ROOT-ONLY keyword to activate the ROOT-ONLY option. FABHLDBR retrieves or prints only the database root segments.

FABHLDBR output: HSSRTRAC output data set

This data set contains the snap-like print of long database records.

The snap-like print of long database records are generated by HSSR Engine. The format of the print is the same as the Trace Output reports that are generated by HSSR Engine. For a complete description of the HSSRTRAC data set, see “HSSRTRAC data set” on page 190.

JCL example for Database Tuning Statistics and FABHLDBR

Use the following JCL example to prepare your JCL for the Database Tuning Statistics function and the FABHLDBR utility.

The following JCL example activates the Database Tuning Statistics and runs the FABHLDBR utility.

```
//STEP1     EXEC FABHULU,MBR=FABHURG1,DBD=dbdname
//hdamdb    DD   DSN=user.hdam,DISP=SHR
//HSSRLOUT  DD   DSN=&&LOUT,UNIT=SYSDA,SPACE=(TRK,(3,3)),
//   DISP=(,PASS)
//SYSUT2    DD   DUMMY
//DFSVSAMP  DD   DSN=IMSVS.PROCLIB(DFSVSAMP),DISP=SHR
//HSSROPT   DD   *
DBSTATS
LOUT LENGTH=8000
//*
//*
//*
//SORT      EXEC PGM=SORT,REGION=4096K
//SORTIN    DD   DSN=&&LOUT,DISP=(OLD,PASS)
//SORTOUT   DD   DSN=&&LOUT2,UNIT=SYSDA,SPACE=(TRK,(3,3)),
//   DISP=(,PASS)
//SYSOUT    DD   SYSOUT=*
//SORTWK01  DD   UNIT=SYSDA,SPACE=(TRK,20)
//SORTWK02  DD   UNIT=SYSDA,SPACE=(TRK,20)
//SORTWK03  DD   UNIT=SYSDA,SPACE=(TRK,20)
//SYSPRINT  DD   SYSOUT=*
//SYSIN     DD   *
    SORT   FIELDS=(1,4,CH,D)
    END
//*
//*
//*
//PRINT     EXEC FABHULU,MBR=FABHLDBR,DBD=dbdname
//HSSRTRAC  DD   SYSOUT=*
//hdamdb    DD   DSN=user.hdam,DISP=SHR
//SYSUT1    DD   DSN=&&LOUT2,DISP=(OLD,PASS)
//SYSIN     DD   *
NBR 100
//DFSVSAMP  DD   DSN=IMSVS.PROCLIB(DFSVSAMP),DISP=SHR
//HSSROPT   DD   *
TRDB *ALL
TRHC CALL,NPF
```

Figure 77. JCL example for the FABHLDBR utility

This example consists of three job steps:

1. The first job step creates the HSSRLOUT data set, while the database is scanned by HSSR Engine. The scan can be done, for example, by using the FABHURG1 database unload utility with the FABHULU JCL procedure (instead of FABHURG1, any other HSSR application program that processes the whole database sequentially would also do the job).

   In the first job step:

   - The HSSRLOUT DD statement is required for creating the HSSRLOUT data set.
   - The SYSUT2 data set of FABHURG1 can be set to DUMMY, if you do not need to obtain an unloaded data set.
   - The DBSTATS control statement in the HSSROPT data set is required for creating the HSSRLOUT data set.

- The LOUT control statement in the HSSROPT is optional. In this example, the control statement specifies that the HSSRLOUT data set contain only records for those database records whose lengths are equal to or over 8000 bytes. This reduces the size of the HSSRLOUT data set.

2. The second job step sorts the HSSRLOUT data set in descending sequence of the bytes 1 - 4 of the records (that is, in descending sequence of the database record length).
3. The third job step invokes the FABHLDBR utility with the FABHULU JCL procedure.

   In the third job step:

   - The HSSRTRAC is the output data set that will contain the snap-like print of long database records. In the example, the HSSRTRAC output is written to SYSOUT.
   - The SYSUT1 DD statement defines the HSSRLOUT data set, which has been sorted in the previous job step.
   - The NBR control statement in the SYSIN data set requests that the utility stops after retrieving or printing 100 longest database records.
   - The TRDB control statement in the HSSROPT data set requests that the trace of HSSR Engine be activated. This control statement is required for obtaining the snap-like print of the database segments.
   - The TRHC control statement in the HSSROPT data set requests that the hardcopy trace option be activated. This control statement is required for obtaining the snap-like print of database segments.
   - The CALL parameter of the TRHC control statement requests a trace of the database segments and of their concatenated keys.
   - The NPF parameter of the TRHC control statement requests that the database segment prefix is not traced.

Chapter 28. Tuning a database with the Database Tuning Statistics

You can use the database statistics that are generated by the Database Tuning Statistics function to help you tune your database.

Activation of the optional Database Tuning Statistics during the sequential processing of a database or during a database unload provides statistics that make easy for you to:

- Evaluate and monitor the effectiveness of HDAM randomizing parameters (such as the size of the root addressable area, the number of RAPs, the block or CI size, or the bytes limit)
- Periodically monitor the need for reorganizing HDAM, HIDAM, or HISAM databases.

How is it possible to determine whether the HDAM randomizing parameters are efficient and whether a database needs to be reorganized? The Database Tuning Statistics provide an important indicator to use in answering this question. This key indicator is the *average number of I/Os required to read at random all database segments of one database record*. By looking at this number in the Database Tuning Statistics reports, the database administrator can quickly determine whether a database is well organized and well randomized.

Ideally, this number would be 1.00. If the number is larger than 1.30, then the database might be disorganized or inefficiently randomized.

Other key indicators in the Database Tuning Statistics can be used to determine the cause of the problem, for example, to determine:

- Whether the HDAM root addressable area is overcrowded and whether its size should be increased. The Database Tuning Statistics answer this question by printing the packing density of the HDAM root addressable area (this is the percentage of DASD space occupied by database segments in the HDAM root addressable area). Experience shows that, with efficient randomizer modules (for example, DFSHDC40, Sequential Subset Randomizer), a reasonable goal for the packing density is often in the range of 70% to 80%.
- Whether the number of HDAM RAPs is appropriate. The Database Tuning Statistics show the ratio between the number of RAPs and the number of roots. Experience shows that, with efficient randomizer modules (for example, DFSHDC40, Sequential Subset Randomizer), a reasonable ratio is around 1.5.
- Whether the block or CI size is appropriate for the average database record length and for the distribution of the database record length.
- Whether the HDAM bytes limit is appropriate for the average database record length and the distribution of the database record length.

The following topics contain an example of the Database Tuning Statistics output for an existing HDAM database, together with a short description of the counters in the output. Then, in further topics, you will see how you can use some general *rules of thumb* and the most important key indicators of the Database Tuning Statistics for the tuning of an HDAM, HIDAM, or HISAM database.

Restriction : The general *rules of thumb* described in the following topics are not directly applicable to partitioned databases.

IMPORTANT NOTICE

The settings and target values for some key indicators that are suggested throughout this *IMS High Performance Unload User's Guide* are based on experiments on simulated databases and applications in a controlled laboratory environment. The experiments were run in non-production, test environments.

Any suggested target values are provided as guidance for database administrators who do not have practical experience with tuning of their databases. Database performance may be affected by numerous factors, including, but not limited to, the specific applications run against the database, how the database

is maintained, and factors beyond those described in this guide that may exist. After gaining experience with the tuning of the real-life databases of their installation, database administrators should review and adapt the proposed target values to their specific databases and operational environments. Therefore, the provided target values must be regarded as general rules of thumb that provide a reasonable starting point when concrete tuning experience with the real-life databases of the installation is lacking.

Topics:

Resources for tuning databases

If you activate the Database Tuning Statistics by providing a DBSTATS control statement in the HSSROPT data set, HSSR Engine writes one or more pages of statistics information in the HSSRSTAT data set.

The statistics information written in the HSSRSTAT data set includes:

- Statistics about:
  - The number of I/Os required to read at random the database segments of one database record
  - The length of database records

  This information is provided on multiple pages:
  - Two pages for the database as a whole (for an example, see Figure 78 on page 324)
  - Two pages for the root addressable area of an HDAM database (for an example, see Figure 79 on page 326)
  - Two pages for the overflow area of an HDAM database (for an example, see Figure 80 on page 327)
  - Two pages for each secondary data set group
- (Printed only for HDAM databases) Statistics with key indicators that represent the quality of randomizing (for an example, see Figure 81 on page 328).
- (If you provide control statements in the HSSRLDEF data set) An additional report containing the distribution of database record lengths (see "HSSRLDEF input data set for Database Tuning Statistics" on page 310 for details).

If you provide an HSSRLOUT DD statement, then you also obtain a sequential data set in which the length of each (or of the longest) database record is written.

Figure 78 on page 324 shows an example of the pages with statistical information printed by the Database Tuning Statistics on the HSSRSTAT data set.

Notes:

- For HALDB, the DB Statistics report is produced for each data set of each partition.
- For PHDAM, the Randomizing Statistics report is produced for each partition.

Related concepts

Obtaining statistics for database tuning

You can request HSSR Engine to create optional Database Tuning Statistics reports during the sequential processing or unloading of a database. This functionality is provided as the *Database Tuning Statistics* function.

DB Statistics report

DB Statistics reports that are generated by the Database Tuning Statistics function provide statistics about the database.

Subtopics:

DB Statistics for the entire database

The following figure provides an example of the DB Statistics report for the entire database.

```
IMS HIGH PERFORMANCE UNLOAD                                    "DB STATISTICS"                                    PAGE:     1
5655-E06                                                 DATE: 06/01/2021  TIME: 19.36.20                     FABHC00  - V1.R2


STATISTICS FOR DB=HDAM0010 TOTAL FOR WHOLE DB


 *** DBDGEN SPECIFICATIONS:
           DB-ORG                                  HDAM
           DDNAME                                  DSHDAM01
           BLOCK/CI-SIZE                              4,096
           FREE SPACE
            - FREE BLOCK FREQUENCY FACTOR(FBFF)           0
            - FREE SPACE PERCENTAGE FACTOR(FSPF)         15

 *** KEY INDICATORS FOR QUALITY OF PHYSICAL ORGANIZATION:

           AVG NBR I/O IN THIS DB
            - PER DB-R IN THIS DB                      1.86
           AVG DB-R LENGTH IN THIS DB
            - PER DB-R IN THIS DB                       682
           ACTUAL PCT FREE SPACE                      17.02  ACTUAL PCT FREE SPACE  = (ALLOC'ED DASD - TOTAL DB-R)/ALLOC'ED DASD * 100
           DEFINED PCT FREE SPACE                       N/A

 *** I/O SUMMARY:

           NBR I/O IN THIS DB                         8,353
           NBR DB-RECORDS
            - WITH I/O IN THIS DB                     4,472

 *** DB-RECORD LENGTH SUMMARY:

           TOTAL DB-R LENGTH IN THIS DB
            - LENGTH (BYTES)                      3,052,423
           NBR DB-RECORDS
            - IN THIS DB                              4,472

IMS HIGH PERFORMANCE UNLOAD                                    "DB STATISTICS"                                    PAGE:     2
5655-E06                                                 DATE: 06/01/2021  TIME: 19.36.20                     FABHC00  - V1.R2


STATISTICS FOR DB=HDAM0010 TOTAL FOR WHOLE DB


 NBR I/O TO READ ***AT RANDOM*** THE RETRIEVED                                  DB RECORD LENGTH (INCL PREFIXES OF SEGMENTS)
 SEGMENTS OF ONE DB RECORD WITH   4 PCB-BUFFERS                                 EXPRESSED IN BLOCKSIZE/CI-SIZE UNITS
                                                                                (ONE BLKSIZE/CI-SIZE =  4,096 BYTES)


  NBR IO         NBR DB-R        PCT    CUM PCT                                 LENGTH         NBR DB-R        PCT    CUM PCT
 ---------------------------------------------                                 ---------------------------------------------

                                                                                  <= 1/16             25        .55        .55
                                                                                  <= 2/16            339       7.58       8.13
                                                                                  <= 3/16          3,237      72.38      80.52
                                                                                  <= 4/16            714      15.96      96.48
                                                                                  <= 5/16             94       2.10      98.59
                                                                                  <= 6/16             18        .40      98.99
                                                                                  <= 7/16              5        .11      99.10
                                                                                  <= 8/16             11        .24      99.35
                                                                                  <= 9/16              3        .06      99.41
                                                                                  <=10/16              5        .11      99.53
                                                                                  <=11/16              3        .06      99.59
                                                                                  <=12/16              3        .06      99.66
                                                                                  <=13/16              1        .02      99.68
                                                                                  <=14/16              2        .04      99.73
                                                                                  <=15/16              2        .04      99.77

   <=  1           2,048      45.79      45.79                                   <=  1                2        .04      99.82
   <=  2           1,522      34.03      79.83                                   <=  2                8        .17     100.00
   <=  3             559      12.50      92.33
   <=  4             213       4.76      97.09
   <=  5              85       1.90      98.09

   <=  6              31        .69      99.68
   <=  7               6        .13      99.82
   <=  8               4        .08      99.91
   <=  9               3        .06      99.97
   <= 10               0        .00      99.97

   <= 20               1        .02     100.00
 ---------------------------------------------                                 ---------------------------------------------
```

Figure 78. DB Statistics for the entire database

- At the top of the left side of the figure (page 1), you can find a description of the following DBDGEN/IDCAMS specifications:
  - DBD name
  - DD name
  - KSDS and ESDS record length (for HISAM)
  - Actual OSAM block size, OSAM LDS CI size, or ESDS CI size
  - Percentage of free space left empty in each block or CI during database load and database reload
  - Free block frequency factor
- Further below on the left side of the figure, there are key indicators, which are used to see whether the database is well organized:

  - The average number of I/Os required to read all database segments of a database record (see Notes "1" on page 325 and "2" on page 325).
  - The average database record length. The reported average database record length includes the size of the segment prefixes and the size of the segment data as stored on DASD (see Note "1" on page 325).
  - The actual free space percentage. This field shows the percentage of the DASD space that is neither occupied by the prefix portion nor by the data portion of database segments. (This definition is not very accurate, since DASD space occupied by RAPs, by free space elements, by free space anchor points, and by BIT maps are not included in this percentage. However, in most cases, this inaccuracy is not large and does not really matter.)

    When computing the actual free space percentage, HSSR Engine does not take into account blocks or CIs beyond the current high-used RBA (see Note "1" on page 325).
  - The defined free space percentage. This figure is computed by combining both free space parameters defined during DBDGEN (that is, the free block frequency factor and the free space within each block or CI).
- Then HSSR Engine prints the following totals:
  - The total number of I/Os to read all database segments
  - The total length of all database records in the database (see Note "1" on page 325)
  - The number of database records (see Note "1" on page 325)
- On the left-hand side of the figure (page 2), there is a table showing the distribution of the number of I/Os required to read at random all the database segments of one database record (see Notes "1" on page 325 and "2" on page 325). The table contains the number, the percentage, and the accumulated percentage of database records that can be read with one or more I/O operations.
- On the right-hand part of the figure (page 2), there is a table describing the distribution of the database record length. The reported database record length includes the size of the segment prefix and the size of the segment data as stored on DASD (for compressible segments, the size of the segment after compression). See also Note "1" on page 325. The table contains the number, the percentage, and the accumulated percentage, of database records that are in the following ranges of database record length:
  - Less than or equal to the 1/16th of the block or CI size
  - 1/16th - 2/16th of the block or CI size
  - 2/16th - 3/16th of the block or CI size
  - And so on

The tables showing the distribution of the I/O numbers and the tables showing the distribution of the database record length are printed side by side. This layout makes it easier to recognize whether large numbers of I/Os per database record are due to large database record sizes.

The DB Statistics report also provides, optionally, a table showing the distribution of the database record length with user-provided definitions for the database record length intervals. The use definitions for the database record length intervals are provided on control cards in the HSSRLDEF data set. For example, you can request a table showing the distribution of the database record lengths for following database record length intervals: 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 600, 700, 800, 900, 1000, 1500, and so on.

Notes:

1. While computing database record lengths and the number of I/Os, HSSR Engine takes into account only the database segments that have been retrieved by the application program or utility. The statistic modules of HSSR Engine are not aware of any database records of database segments that have not been retrieved.
2. While computing the number of I/Os, HSSR Engine assumes that the root segment is retrieved through a GU call. HSSR Engine also assumes that at the time that the GU call is issued, the buffer pools do not

contain any block or CI containing a portion of the RAP chain or containing a database segment of the database record.

For an HDAM database, HSSR Engine includes the number of I/Os required to chase the RAP synonym chain.

For a HIDAM database, HSSR Engine does not include in these figures the number of I/Os required to access the primary index of the HIDAM database.

For a HISAM database, HSSR Engine assumes that the access to the KSDS record requires one I/O operation (additional I/Os to access the index component of the KSDS cluster are not included in these figures).

While computing the number of I/Os, HSSR Engine attempts to simulate the behavior of the IMS buffer pools. If, for example, the retrieval of the database segments of a database record requires access to blocks n1, n2, and n3, or n2 and n1, then HSSR Engine will report that (with a buffer pool of three buffers) the number of I/Os is three (not five).

DB Statistics for the HDAM root addressable area

The following figure provides an example of the DB Statistics report for the HDAM root addressable area.

```
IMS HIGH PERFORMANCE UNLOAD                                    "DB STATISTICS"                                    PAGE:     3
5655-E06                                                 DATE: 06/01/2021  TIME: 19.36.20                     FABHC00  - V1.R2


STATISTICS FOR DB=HDAM0010 (ROOT ADDRESSABLE AREA)


 *** DBDGEN SPECIFICATIONS:

           DB-ORG                                HDAM
           DDNAME                                DSHDAM01 (RAA)
           BLOCK/CI-SIZE                            4,096
           FREE SPACE
            - FREE BLOCK FREQUENCY FACTOR(FBFF)         0
            - FREE SPACE PERCENTAGE FACTOR(FSPF)       15

 *** KEY INDICATORS FOR QUALITY OF PHYSICAL ORGANIZATION:

           AVG NBR I/O IN RAA
            - PER DB-R IN RAA                        1.46
            - PER DB-R IN THIS DB                    1.46
           AVG DB-R LENGTH IN RAA
            - PER DB-R IN RAA                         606
            - PER DB-R IN THIS DB                     570
           ACTUAL PCT FREE SPACE                    17.14  ACTUAL PCT FREE SPACE  = (ALLOC'ED DASD - TOTAL DB-R)/ALLOC'ED DASD * 100
           DEFINED PCT FREE SPACE                   15.00  DEFINED PCT FREE SPACE = FSPF + 100/FBFF - FSPF/FBFF

 *** I/O SUMMARY:

           NBR I/O IN RAA                           6,543
           NBR DB-RECORDS
            - WITH I/O IN RAA                       4,472
            - IN THIS DB                            4,472

 *** DB-RECORD LENGTH SUMMARY:

           TOTAL DB-R LENGTH IN RAA
            - LENGTH (BYTES)                    2,549,078
           NBR DB-RECORDS
            - IN RAA                                4,203
            - IN THIS DB                            4,472

IMS HIGH PERFORMANCE UNLOAD                                    "DB STATISTICS"                                    PAGE:     4
5655-E06                                                 DATE: 06/01/2021  TIME: 19.36.20                     FABHC00  - V1.R2


STATISTICS FOR DB=HDAM0010 (ROOT ADDRESSABLE AREA)

 NBR I/O TO READ ***AT RANDOM*** THE RETRIEVED                                  DB RECORD LENGTH (INCL PREFIXES OF SEGMENTS)
 SEGMENTS OF ONE DB RECORD WITH   4 PCB-BUFFERS                                 EXPRESSED IN BLOCKSIZE/CI-SIZE UNITS
                                                                                (ONE BLKSIZE/CI-SIZE =  4,096 BYTES)

  NBR IO         NBR DB-R        PCT    CUM PCT                                 LENGTH         NBR DB-R        PCT    CUM PCT
 ------------------------------------------------                               ------------------------------------------------

                                                                                  <= 1/16             24        .57        .57
                                                                                  <= 2/16            311       7.39       7.97
                                                                                  <= 3/16          3,868      92.02     100.00
   <=  1           3,166      70.79      70.79
   <=  2             822      18.38      89.17
   <=  3             294       6.57      95.75
   <=  4             128       2.86      98.61
   <=  5              45       1.00      99.61

   <=  6               9        .20      99.82
   <=  7               5        .11      99.93
   <=  8               2        .04      99.97
   <=  9               1        .02     100.00
 ------------------------------------------------                               ------------------------------------------------
```

Figure 79. DB Statistics for the HDAM root addressable area

The layout of DB Statistics reports for the HDAM root addressable area is similar to the layout of DB Statistics reports for the entire database. The statistics, however, do not reflect the number of I/Os in the whole database and the total database record size; instead, they reflect the number of I/Os in the root addressable area and the database record size in the root addressable area.

DB Statistics for the HDAM overflow area

The following figure provides an example of the DB Statistics report for the HDAM overflow area.

```
IMS HIGH PERFORMANCE UNLOAD                                    "DB STATISTICS"                                    PAGE:     5
5655-E06                                                 DATE: 06/01/2021  TIME: 19.36.20                     FABHC00  - V1.R2


STATISTICS FOR DB=HDAM0010 (OVERFLOW AREA)


 *** DBDGEN SPECIFICATIONS:

           DB-ORG                                  HDAM
           DDNAME                                  DSHDAM01 (OVFLW AREA)
           BLOCK/CI-SIZE                              4,096
           FREE SPACE
            - FREE BLOCK FREQUENCY FACTOR(FBFF)           0
            - FREE SPACE PERCENTAGE FACTOR(FSPF)         15

 *** KEY INDICATORS FOR QUALITY OF PHYSICAL ORGANIZATION:

           AVG NBR I/O IN OVFLW AREA
            - PER DB-R IN OVFLW AREA                   1.09
            - PER DB-R IN THIS DB                       .40
           AVG DB-R LENGTH IN OVFLW AREA
            - PER DB-R IN OVFLW AREA                    304
            - PER DB-R IN THIS DB                       112
           ACTUAL PCT FREE SPACE                      16.41  ACTUAL PCT FREE SPACE  = (ALLOC'ED DASD - TOTAL DB-R)/ALLOC'ED DASD * 100
           DEFINED PCT FREE SPACE                     15.00  DEFINED PCT FREE SPACE = FSPF + 100/FBFF - FSPF/FBFF

 *** I/O SUMMARY:

           NBR I/O IN OVFLW AREA                      1,811
           NBR DB-RECORDS
            - WITH I/O IN OVFLW AREA                  1,655
            - IN THIS DB                              4,472

 *** DB-RECORD LENGTH SUMMARY:

           TOTAL DB-R LENGTH IN OVFLW AREA
            - LENGTH (BYTES)                        503,345
           NBR DB-RECORDS
            - IN OVFLW AREA                           1,655
            - IN THIS DB                              4,472

IMS HIGH PERFORMANCE UNLOAD                                    "DB STATISTICS"                                    PAGE:     6
5655-E06                                                 DATE: 06/01/2021  TIME: 19.36.20                     FABHC00  - V1.R2


STATISTICS FOR DB=HDAM0010 (OVERFLOW AREA)


 NBR I/O TO READ ***AT RANDOM*** THE RETRIEVED                                    DB RECORD LENGTH (INCL PREFIXES OF SEGMENTS)
 SEGMENTS OF ONE DB RECORD WITH   4 PCB-BUFFERS                                   EXPRESSED IN BLOCKSIZE/CI-SIZE UNITS
                                                                                  (ONE BLKSIZE/CI-SIZE =  4,096 BYTES)

  NBR IO         NBR DB-R        PCT    CUM PCT                                   LENGTH         NBR DB-R        PCT    CUM PCT
 ---------------------------------------------                                    ---------------------------------------------

                                                                                    <= 1/16          1,143      69.06      69.06
                                                                                    <= 2/16            201      12.14      81.20
                                                                                    <= 3/16            183      11.05      92.26
                                                                                    <= 4/16             70       4.22      96.49
                                                                                    <= 5/16             21       1.26      97.76
                                                                                    <= 6/16             10        .60      98.36
                                                                                    <= 7/16              3        .18      98.54
                                                                                    <= 8/16              6        .36      98.91
                                                                                    <= 9/16              2        .12      99.03
                                                                                    <=10/16              1        .06      99.09
                                                                                    <=11/16              1        .06      99.15
                                                                                    <=12/16              4        .24      99.39
                                                                                    <=13/16              2        .12      99.51
                                                                                    <=14/16              0        .00      99.51
                                                                                    <=15/16              1        .06      99.57

   <=  1           1,541      93.11      93.11                                     <=  1                1        .06      99.63
   <=  2              99       5.98      99.09                                     <=  2                6        .36     100.00
   <=  3               9        .54      99.63
   <=  4               3        .18      99.81
   <=  5               1        .06      99.87

   <=  6               0        .00      99.87
   <=  7               0        .00      99.87
   <=  8               1        .06      99.93
   <=  9               0        .00      99.93
   <= 10               0        .00      99.93

   <= 20               1        .06     100.00
 ---------------------------------------------                                    ---------------------------------------------
```

Figure 80. DB Statistics for the HDAM overflow area

The layout of DB Statistics reports for the HDAM overflow area is similar to the layout of DB Statistics reports for the entire database. The statistics, however, do not reflect the number of I/Os in the whole

database and the total database record size; instead, they reflect the number of I/Os in the HDAM overflow area and the database record size in the HDAM overflow area.

Randomizing Statistics report

The Randomizing Statistics report that is generated by the Database Tuning Statistics function provides information that can be used to evaluate and monitor the quality of the HDAM randomizing parameters.

The following figure provides an example of the Randomizing Statistics report.

```
IMS HIGH PERFORMANCE UNLOAD                                "RANDOMIZING STATISTICS"                          PAGE:     1
5655-E06                                                DATE: 06/01/2021  TIME: 19.36.20                   FABHC00  - V1.R2


STATISTICS FOR RETRIEVED ROOTS OF DB=HDAM0010 WITH RANDOMIZER=DFSHDC10


 *** DBDGEN SPECIFICATIONS:

           NBR RAPS PER BLOCK/CI                       4
           HIGHEST BLOCK/CI IN RAA                   750
           BYTES-LIMIT                               700
           BLOCK/CI-SIZE                           4,096
           FREE SPACE:
            - FREE BLOCK FREQUENCY FACTOR(FBFF)        0
            - FREE SPACE PERCENTAGE FACTOR(FSPF)      15

 *** KEY INDICATORS FOR QUALITY OF RANDOMIZING:

           AVG NBR I/O
            - PER DB-R                              1.86
            - ON RAP CHAIN PER ROOT                 1.34
            - ON RAP CHAIN PER RAP CHAIN            1.53
            - IN RAA PER DB-R                       1.46
            - IN OVERFLOW AREA PER DB-R              .40

           NBR OF ACCESSED RAPS WHICH ARE USED
            - NBR                                 2,327
            - PCT OF ACCESSED RAPS                77.56

           NBR OF ACCESSED ROOTS IN OVERFLOW AREA
            - NBR                                   269
            - PCT OF ACCESSED ROOTS                6.01

           PACKING DENSITY OF RAA                 82.86
           NBR OF RAPS PER ROOT                     .66
           AVG POSITION OF ROOT SEGMENTS ON
              RAP SYNONYM CHAINS                  1.73
           AVG NUMBER OF SYNONYMS ON
              RAP SYNONYM CHAINS                  1.92

           AVG DB-R LENGTH                          682

IMS HIGH PERFORMANCE UNLOAD                                "RANDOMIZING STATISTICS"                          PAGE:     2
5655-E06                                                DATE: 06/01/2021  TIME: 19.36.20                   FABHC00  - V1.R2


STATISTICS FOR RETRIEVED ROOTS OF DB=HDAM0010 WITH RANDOMIZER=DFSHDC10


 NBR I/O ON RAP CHAIN TO READ ***AT RANDOM***
 THE ROOT SEGMENTS WITH   4 PCB-BUFFERS

  NBR IO        NBR ROOTS        PCT    CUM PCT
 ------------------------------------------------

   <=  1           3,157      70.59      70.59
   <=  2           1,125      25.15      95.75
   <=  3             170       3.80      99.55
   <=  4              16        .35      99.91
   <=  5               3        .06      99.97
   <=  6               1        .02     100.00




 POSITION OF ROOT SEGMENTS ON RAP CHAINS

 POSITION       NBR ROOTS        PCT    CUM PCT
 ------------------------------------------------

   <=  1           2,327      52.03      52.03
   <=  2           1,344      30.05      82.08
   <=  3             544      12.16      94.25
   <=  4             177       3.95      98.21
   <=  5              64       1.43      99.64
   <=  6              14        .31      99.95
   <=  7               2        .04     100.00
```

Figure 81. Randomizing Statistics

- At the top of the figure, you can find a description of the following DBDGEN/IDCAMS specifications:
  - DBD name
  - Name of the randomizing module
  - Number of RAPs per block or CI
  - Highest block number or CI number in the root addressable area

  - Bytes limit
  - Actual block size or CI size
  - Percentage of free space left empty in each block or CI during database load and database reload
  - Free block frequency factor
- Further below there are some key indicators, which describe the quality of the randomizing. These important indicators can also be used to understand how poor randomizing parameters can be changed to enhance the quality of the randomizing. These key indicators are:
  - The average number of I/Os required to read all database segments of one database record (see Notes “1” on page 329 and “2” on page 329).
  - The average number of I/Os required to read a root segment at random while following the RAP chain (see Notes “1” on page 329 and “2” on page 329).
  - The average number of I/Os in the root addressable area required to read at random all database segments of one database record. This number includes I/Os in the root addressable area that are required to read the root and the dependent database segments and the I/Os in the root addressable area that are required to follow the RAP chain (see Notes “1” on page 329 and “2” on page 329).
  - The average number of I/Os in the overflow area required to read at random all database segments of one database record. This number includes the I/Os in the overflow area that are required to read the root and the dependent database segments and I/Os in the overflow area that are required to follow the RAP chain (see Notes “1” on page 329 and “2” on page 329).
  - The percentage of used RAPs. This number is computed by dividing the total number of RAPs in the root addressable area by the number of accessed RAPs that are nonzero (that is, RAPs that are used).
  - The percentage of the retrieved roots that are located in the overflow area.
  - The packing density of the root addressable area. This is the percentage of the DASD space in the root addressable area space that is occupied by the prefix portion and data portion of database segments (see Note “1” on page 329).
  - The number of RAPs per root. This is a ratio computed by dividing the number of RAPs in the root addressable area by the number of roots that have been read through HSSR GN calls.
  - The average position of the (retrieved) roots on the RAP chains.
  - The average database record length (see Note “1” on page 329).
- The following tables are near the bottom of the report:
  - A table containing the number, percentage and accumulated percentage of database roots that require, during random GU processing 1, 2, 3, 4, and so on, I/O operations while following the RAP synonym chain (see Notes “1” on page 329 and “2” on page 329).
  - A table describing the number, percentage and accumulated percentage of database root segments that are on the first, second, third, and so on, position of a RAP synonym chain (see Note “1” on page 329).

Notes:

1. When computing database record lengths and the number of I/Os, HSSR Engine takes into account only those database segments that have been retrieved by the application program or utility. The statistic modules of HSSR Engine are not aware of any database records or database segments that have not been retrieved.
2. When computing the number of I/Os, HSSR Engine assumes that the root segment is retrieved through a GU call. HSSR Engine also assumes that at the time the GU call is issued, the buffer pools do not contain any block or CI containing a portion of the RAP chain or containing a database segment of the database record.

   When computing the number of I/Os, HSSR Engine attempts to simulate the LRU algorithm of the IMS buffer pools. If, for example, the retrieval of the database segments of one database record requires access to blocks n1, n2, and n3, or n2 and n1, then HSSR Engine will report that (for a buffer pool of three buffers) the number of I/Os is three (not five).

DB Record Length Distribution report

The DB Record Length Distribution report that is generated by the Database Tuning Statistics function provides information about the distribution of database record length.

The following figure provides an example of the DB record length distribution report.

```
IMS HIGH PERFORMANCE UNLOAD                              "DB RECORD LENGTH DISTRIBUTION"                        PAGE:     1
5655-E06                                                 DATE: 06/01/2021  TIME: 20.09.01                    FABHC00  - V1.R2

                                      DATABASE RECORD LENGTH DISTRIBUTION FOR DB=HDAM0010

       LENGTH         NBR DB-R        PCT    CUM PCT
 ------------------------------------------------------

 <=       500              349       7.80       7.80
 <=       550              740      16.54      24.35
 <=       600              264       5.90      30.25
 <=       650            1,355      30.29      60.55
 <=       700              260       5.81      66.36
 <=       750              551      12.32      78.68
 <=       800              176       3.93      82.62
 <=       850              329       7.35      89.98
 <=       900              113       2.52      92.50
 <=       950               82       1.83      94.34
 <=      1000               81       1.81      96.15
 <=      1050               34        .76      96.91
 <=      1100               32        .71      97.62
 <=      1150               12        .26      97.89
 <=      1200               18        .40      98.30
 <=      1250                9        .20      98.50
 <=      1300                7        .15      98.65
 <=      1350                4        .08      98.74
 <=      1400                5        .11      98.85
 <=      1450                3        .06      98.92
 <=      1500                3        .06      98.99
 <=      1600                1        .02      99.01
 <=      1700                1        .02      99.03
 <=      1800                3        .06      99.10
 <=      1900                4        .08      99.19
 <=      2000                6        .13      99.32
 <=      2500                7        .15      99.48
 <=      3000                7        .15      99.64
 <=      3500                3        .06      99.70
 <=      4000                5        .11      99.82
 <=      5000                2        .04      99.86
 <=      6000                2        .04      99.91
 <=      7000                2        .04      99.95
 <=      8000                2        .04     100.00
```

Figure 82. DB Record Length Distribution

The statistics in the report are printed only if you request the generation of a DB Record Length Distribution report with your own definition of the database record length ranges. You can make this request by providing control cards in the HSSRLDEF data set (for more information, see “HSSRLDEF input data set for Database Tuning Statistics” on page 310).

The reported database record length includes the size of the segment prefixes and the size of the segment data as stored on DASD (for compressible segment, the size of the segment data reflects the size of the compressed segment). The table contains the number, the percentage, and the accumulated percentage of database records that are in the user-defined ranges of database record length.

When computing database record lengths, HSSR Engine takes into account only the database segments that have been retrieved by the application program or utility. The statistics modules of HSSR Engine are not aware of any database records or database segments that have not been retrieved.

Tuning the primary data set group of an HDAM database

Lean from these topics the most important indicators in the Database Tuning Statistics output and how these indicators can be used to tune an HDAM database.

Tuning an HDAM database is an iterative process. When changing randomizing parameters, database administrators with little tuning experience should change only one randomizing parameter at a time, and then observe the impact of this change. This can be done by the following sequence of activities:

1. Unloading the database
2. Changing a randomizing parameter and DBDGEN
3. Reloading the database with the modified randomizing parameter
4. Executing FABHURG1 or FABHFSU unload utility to obtain new Database Tuning Statistics and to observe the impact of the modified randomizing parameter

5. Depending on the results of the observations in step "4" on page 330, attempting other changes of the randomizing parameters (step "2" on page 330)

If the database is very large, it is often not practical to do multiple consecutive reloads or unloads in order to observe the effect of multiple consecutive changes. In this case, the database administrator can use the FABHEXTR exit routine in order to create a small database extract consisting of a subset of the real-life database records (see Chapter 29, "Creating a database extract for tuning experiments," on page 351). The database administrators can then perform their tuning experiments with this smaller database extract.

Average number of I/O operations per database record

Obviously, HDAM randomizing can be considered to be efficient if the average number of I/O operations required to read at random all database segments of one database record is low. The average number of I/Os required to read at random all database segments of one database record is one of the most important indicators of the quality of the randomizing.

This average number of I/Os is printed by the Database Tuning Statistics as shown in Figure 81 on page 328, to the right of the phrase "AVG NBR I/O: - PER DB-R".

By looking at this average number in the Database Tuning Statistics, the database administrator can very rapidly see whether a database is efficiently randomized.

For an ideal database consisting of one single data set group, this average number would be 1.0.

In real life, this ideal value of 1.0 can seldom be achieved and the average number of database I/Os per database record will be higher. Note, that due to the "law of the large numbers" it is easier to achieve a good randomizing value when the block size, divided by average database record length, is large.

As a general rule of thumb, the database can be considered as fairly well randomized if the average number of I/Os per database record is below:

- 1.20 (for databases with an average database record length below one tenth of the block or CI size)
- 1.30 (for databases with larger average database record lengths)

For numbers above 1.20/1.30, the database can often be considered to be poorly randomized (unless the database record length is large). In this case, the database administrator can use other numbers provided by the Database Tuning Statistics in order to determine the reason for the poor randomizing. These other numbers in the Database Tuning Statistics can be used to give an answer to the questions in the following checklist and to find, in most cases, the reason for poor randomizing:

1. Is the size of the root addressable area appropriate? (See "Packing density of the root addressable area" on page 332 for details.)
2. Is the number of RAPs appropriate? (See "Number of RAPs per root segment" on page 333 for details.)
3. Is the HDAM bytes limit appropriate for the database record lengths in this database? (See "Bytes limit" on page 334 for details.)
4. Is the block size or CI size appropriate for the database record lengths of this database? (See "CI size and block size" on page 334 for details.)
5. Is the database record length excessive? (See "Databases with long database records" on page 342 for details.)
6. Is the amount of Free Space specified during DBDGEN equal to zero? (See "Free block frequency factor" on page 335 for details.)
7. Is a database reorganization overdue in order to reduce database segment scattering in the overflow area? (See "Periodical database reorganization" on page 342 for details.)
8. For a Sequential Subset Randomizer: Is the relative amount of DASD space allocated to each subset of database records appropriate? (See "Inefficient space suballocation for the Sequential Subset Randomizer" on page 343 for details.)

Notes:

1. Any reference to the Sequential Subset Randomizer in these topics is not intended to state or imply that this product should be used instead of the standard DFSHDC40 randomizing module.
2. HSSR Engine does not know which database record occurrences are the most frequently accessed. For some databases, it is sometimes the longest database records requiring the largest number of I/Os that are the most often accessed database records. In this case, the average number of I/Os per database record reported by HSSR Engine is different from the average number of I/Os per accessed database record.
3. HSSR Engine does not know which segment types are the most frequently accessed. With some databases, it is sometimes only one or two segment types that are often accessed. In this case, the job step used to produce the Database Tuning Statistics can be run with a PSB, which is sensitive only to these two segment types. The Database Tuning Statistics of such a job will probably be more representative than the Database Tuning Statistics of a job that was sensitive to all segment types.

   For detailed statistics about the "probability of I/O" for each segment type, see the Segment and Pointer Statistics report of IMS High Performance Pointer Checker.

Discussion of the example

In the example of Figure 81 on page 328, the average number of I/Os per database record is 1.86. This number is substantially higher than the target values of 1.20 and 1.30.

A quick glance at the average database record length shows that the average database record length is not large (1/6th of the CI size); therefore the reason for the poor average number of I/Os per database record is not large database record lengths. Hence it seems probable that the high average number of I/Os per database record is due to poor randomizing parameters and that the average number of I/Os can be substantially reduced. The next topics show how the Database Tuning Statistics can be used to understand why this database is currently inefficiently randomized, and how the randomizing can be improved.

Note that both the average number of I/Os per database record in the root addressable area and in the overflow area are far from their ideal values. They are 1.46 for the root addressable area (RAA) and 0.40 for the overflow area (the ideal values being 1.00 and 0.00). A high value in the RAA is often an indication that the packing density of the root addressable area is too high (see “Packing density of the root addressable area” on page 332). A high value of I/Os in the overflow area is often an indication that the bytes limit is too low (see “Bytes limit” on page 334 for details).

Packing density of the root addressable area

The packing density of the HDAM root addressable area can be found in the Randomizing Statistics report.

Specifically, you can find the packing density of the HDAM root addressable area in Figure 81 on page 328 to the right of the phrase "PACKING DENSITY OF RAA".

With the standard DFSHDC40 randomizer and with the Sequential Subset Randomizer, a reasonable rule of thumb is a packing density of approximately 75% (for databases with an average database record length smaller than one tenth of the CI/block size) or 70% (for databases with a larger average database record length).

Notes:

1. Aiming for higher packing densities saves DASD space, but often decreases the performance of HDAM database accesses. Each installation will have probably its own idea of what is the ideal trade-off between DASD space and performance and regarding the ideal packing density. For example, some installations may want packing densities of 75% and 80% (instead of the target values of 70% and 75%).
2. For small or medium-sized HDAM databases (when saving DASD space is not important), it is often reasonable to aim for a packing density below 70%.
3. If the total size of all database records will eventually grow, it is then reasonable to oversize the root addressable area at database reload time, in order to provide enough space for the future data growth.

You can change the packing density of the root addressable area by varying the number of blocks or CIs in the root addressable area and by varying the size of a block or CI. Increasing the number of blocks or CIs or increasing the block or CI size will increase the size of the root addressable area and lower its packing density; this will usually increase the performance of random accesses to the database.

Varying the bytes limit may also change the packing density of the root addressable area (reducing the bytes limit will tend to store more information in the overflow area and less information in the root addressable area; this will hence tend to reduce the packing density in the root addressable area).

As a rule of thumb, you should increase the size of the root addressable area, if the Database Tuning Statistics shows that:

- The packing density is higher than the recommended target values of 70% and 75%
- The average number of I/Os in the root addressable area per database record is higher than 1.15.

Discussion of the example

In the example of Figure 81 on page 328, the packing density of the root addressable area is 82.86%. This is higher than the recommended value of 70% for databases with this type of average database record length. This high packing density provides a probable and partial explanation for the following numbers, which look poor:

- The average number of I/O on RAP chain per root, which is 1.34 (usually this number should not be much higher than 1.10).
- The average number of I/O in RAA per database record, which is 1.46 (usually this number should not be much higher than 1.15).
- The percentage of accessed roots in the overflow area, which is 6.01 (usually this number should not be much higher than 1.0).

In this case, you could decrease the packing density of the root addressable area. “Bytes limit” on page 334 shows that the bytes limit is already too low, so a decrease of the packing density should not be attempted through a decrease of the bytes limit. Instead, a decrease of the packing density should be achieved by increasing the number of blocks or CIs in the root addressable area (or by an increase of the block or CI size).

Number of RAPs per root segment

The number of RAPs per root segment can be found in the Randomizing Statistics report.

Specifically, this ratio is found in Figure 81 on page 328, to the right of the phrase "NBR OF RAPS PER ROOT".

Generally, the number of RAPs should be approximately 1.5 times the number of database records. In case of doubt, it is better to specify too many than too few RAPs (since a RAP is only 4 bytes and hence inexpensive in terms of DASD space).

Note: An exception to this recommendation is an Intersystem block-level sharing environment with sequential database processing. In such an environment, the number of RAPs should not be too large, since access to each RAP may create intersystem block-level sharing delays.

Discussion of the example

In the example in Figure 81 on page 328, the number of RAPs per root is 0.66. This is much too low. This low number probably contributes to the high average number of I/Os per root on the RAP chains (which is 1.34).

You could increase the number of RAPs per CI/Block to increase this ratio to 1.5.

CI size and block size

The CI size or the block size can be found in the Randomizing Statistics report.

The current CI size/block size is shown in the top portion of Figure 81 on page 328.

A general rule of thumb is a 4 KB CI/block size for HDAM.

However, if the average database record length is higher than 800 bytes (that is, larger than 1/5th of the block or CI size), then an increase of the block or CI size from 4 KB to 8 KB might improve the performance of HDAM database accesses (especially if the database administrator wants a high packing density (more than 70%) of the root addressable area). In this case, an increase of the CI size/block size from 4 KB to 8 KB can often reduce the number of I/O operations (but will increase slightly the time of an I/O operation).

CI/block sizes larger than 8 KB should be an exception (for example, for very large average database record sizes).

Discussion of the example

In the example in Figure 81 on page 328, the average database record size is 682. This is approximately 1/6th of the block or CI size. 1/6th of the block or CI size is not very large and seems acceptable. However, it could be reasonable to experiment with an 8 KB block or CI size.

A decision to increase the block or CI size from 4 KB to 8 KB for this database is probably a matter of the personal taste of the database administrator.

Bytes limit

The bytes limit is the maximum number of bytes of a database record that can be stored into the root addressable area in a series of insert calls unbroken by a call to another database record.

The current bytes limit is shown in the top portion of Figure 81 on page 328.

The database administrator needs to perform a trade-off when specifying a bytes limit. Specifying a bytes limit that is too small will result in the storing of too many database segments into the overflow area. Access to these many database segments in the overflow area will usually trigger additional I/Os (since these database segments are not stored in the same block or CI as the root segment in the root addressable area); these additional I/Os will adversely affect the performance of access to the HDAM database.

Specifying a bytes limit that is too high (for example, 10 times the average database record length) may create a situation where one individual database record occupies too much space in the root addressable area. Other database records of the same block or CI or of neighboring blocks or CIs will then have an insufficient amount of DASD space available for the storing of their own database segments in the same block or CI as the RAP (See Note). Access to the database segments of these other database records will require additional I/Os. This will adversely affect the performance of access to the HDAM database.

Note: Specification of a bytes limit that is too high often creates a "cascade" effect. If, for example, one database record randomizing to block n is allowed to store as many as five block sizes worth of data in the root addressable area, then the database records that are randomized to neighboring blocks (for example, block n, n+1, n+2, n+3, and n+4) cannot be stored in the same block as their RAPs. The segments of these database records will spill into blocks n+5, n+6, and so forth, and will create problems for the database records that are randomized in block n+5, n+6, and so on. Note that the PSSR utility of IMS High Performance Load assists in preventing such cascading effects.

The Database Tuning Statistics provide the following information, which is useful when evaluating whether the current bytes limit is appropriate:

- The table containing the distribution of the database record length (see Figure 78 on page 324)
- The average number of I/Os per database record in the overflow area (see Figure 81 on page 328).

Discussion of the example

The table printed by the Database Tuning Statistics (shown in the top-right-hand portion of page 2 in Figure 78 on page 324) is useful for a reasonable determination of the bytes limit. This sample report shows that:

- 80.52 percent of the database records have a database record length smaller than 3/16th of a block or CI.
- 96.48 percent of the database records have a database record length smaller than 4/16th of a block or CI.
- 98.59 percent of the database records have a database record length smaller than 5/16th of a block or CI.

Based on this information, the current bytes limit (which is 700 according to Figure 81 on page 328) seems too low. With a bytes limit of 700, at least 19.48% (=100%-80.52%) of the database records will have, after a database reload, some dependent segments in the overflow area. This low bytes limit is one of the major reasons for the high average number (0.40) of I/Os per database record in the overflow area.

You could increase the current bytes limit from 700 bytes to 5/16th of the block or CI size; which is 1280 bytes.

1280 bytes is not excessive, when compared with the average database record length of 862. With a bytes limit of 1280, approximately 98.59% of the database records will have all their database segments in the root addressable area. This will hopefully reduce the average number of I/Os per database record in the overflow area, which is currently 0.40.

Free block frequency factor

This topic discusses about the free block frequency factor.

As explained in *IMS Database Administration*, specifying a free block frequency factor on the DATASET macro of DBDGEN for the first data set group of HDAM is self-defeating.

For data set groups other than the first HDAM data set group, a free block frequency specification is often useful.

Free space within each block/CI

Specification on the DATASET macro of DBDGEN of free space within each block or CI has both advantages and disadvantages.

Free space within each block or CI is often advantageous, if the database will have a large number of insert activities after database load/reload in the overflow area. In this case, free space specifications often reduce the scattering of database segments of the same database record into a large number of different blocks or CIs during subsequent insert activities.

The percentage of specified free space should be taken into account when determining the size of the RAA. For example, when specifying 10% of free space, some database administrators increase the size of the root addressable area by 10%, since the free space is not available at database load time or database reload time.

Free space within each block or CI is disadvantageous if few database segments are inserted after database load/reload or if nearly all database segments can be stored in the root addressable area.

Discussion of the example

For the example described in the previous figures, one should experiment without free space specifications. Why? Because after the recommended increase of the packing density of the root addressable area, enough free space will be available in the RAA; and because after the recommended increase of the bytes limit, only few database segments will be stored in the overflow area.

Examples of other indicators provided by the Database Tuning Statistics

The reports that are generated by the Database Tuning Statistics function provide other indicators that can help you understand more about the database status.

Subtopics:

Percent of roots in the overflow area

This number can be found in Figure 81 on page 328, to the right of the phrase "NBR OF ACCESSED ROOTS IN OVERFLOW AREA: - PCT OF ACCESSED ROOTS".

Normally this percent should be very small (below 1 percent). Higher values will increase the average number of I/Os required to access a root segment and will decrease the performance. The reason for a high value is often an overcrowded root addressable area.

Discussion of the example:

In the example of Figure 81 on page 328, the percentage of roots in the overflow area is too high: 6.01. You could increase the root addressable area (see “Packing density of the root addressable area” on page 332) to take care of this problem.

Number of I/Os on the RAP chain

This number can be found in Figure 81 on page 328 to the right of the phrase "AVG NBR I/O: - ON RAP CHAIN PER ROOT".

Normally, this number should not be substantially larger than 1.10. The reason for a high value is often an overcrowded root addressable area, too few RAPs, a block or CI size that is too small, or a bytes limit that is too high.

Discussion of the example:

In the example of Figure 81 on page 328, the average number of I/Os on the RAP chain is too high: 1.34. The recommended increase of the root addressable area (see “Packing density of the root addressable area” on page 332) and an increase of the number of RAPs (see “Number of RAPs per root segment” on page 333) should take care of this problem.

Number of I/Os in the root addressable area

This number can be found in Figure 81 on page 328 to the right of the phrase "AVG NBR I/O: - IN RAA PER DB-R".

Normally, this number should not be substantially larger than 1.20. The reason of a higher value is often an overcrowded root addressable area, too few RAPs, a block or CI size that is too small, or a bytes limit that is too high.

Discussion of the example:

In the example of Figure 81 on page 328, the average number of I/Os in the RAA is too high: 1.46. The recommended increase of the root addressable area (see “Packing density of the root addressable area” on page 332) and an increase of the number of RAPs (see “Number of RAPs per root segment” on page 333) should take care of this problem.

Number of I/Os in the overflow area

This number can be found in Figure 81 on page 328 to the right of the phrase "AVG NBR I/O: - IN OVERFLOW AREA PER DB-R".

Normally, this number should not be substantially larger than 0.20. The reason of a higher value is often a bytes limit which is too low, a block or CI size which is too small for the database record length of the database, database records that are very long, or a database that needs to be reorganized in order to reduce the fragmentation of database segments of the same database record into multiple blocks or CIs.

Discussion of the example:

In the example of Figure 81 on page 328, the average number of I/Os in the overflow area is too high: 0.40. The recommended increase of the bytes limit (see “Bytes limit” on page 334) should take care of this problem.

Summary of suggested changes for the example database

After applying the suggested changes that were explained in the preceding topics, rerunning the Database Tuning Statistics function to obtain the latest database statistics is an effective approach to confirm improvement in database status.

In the previous discussion of the example of Database Tuning Statistics, the following changes were suggested:

- Increase the size of the root addressable area in order to decrease its packing density (the target packing density being approximately 70%).
- Increase the number of RAPs (the target ratio of RAPs per root being 1.5).
- Increase the bytes limit (to approximately 1280 bytes).
- Get rid of free space specifications.

The following figures show the Database Tuning Statistics of the database after the suggested changes have been made. Notice the considerable improvements of the different indicators. For example, the average number of I/Os per database record has improved from 1.86 to 1.16.

```
IMS HIGH PERFORMANCE UNLOAD                                    "DB STATISTICS"                                    PAGE:     1
5655-E06                                                DATE: 06/01/2021  TIME: 20.09.01                    FABHC00  - V1.R2


STATISTICS FOR DB=HDAM0010 TOTAL FOR WHOLE DB


 *** DBDGEN SPECIFICATIONS:

          DB-ORG                                 HDAM
          DDNAME                                 DSHDAM01
          BLOCK/CI-SIZE                             4,096
          FREE SPACE
           - FREE BLOCK FREQUENCY FACTOR(FBFF)          0
           - FREE SPACE PERCENTAGE FACTOR(FSPF)         0

 *** KEY INDICATORS FOR QUALITY OF PHYSICAL ORGANIZATION:

          AVG NBR I/O IN THIS DB
           - PER DB-R IN THIS DB                     1.16
          AVG DB-R LENGTH IN THIS DB
           - PER DB-R IN THIS DB                      682
          ACTUAL PCT FREE SPACE                     27.16  ACTUAL PCT FREE SPACE  = (ALLOC'ED DASD - TOTAL DB-R)/ALLOC'ED DASD * 100
          DEFINED PCT FREE SPACE                      N/A

 *** I/O SUMMARY:

          NBR I/O IN THIS DB                        5,197
          NBR DB-RECORDS
           - WITH I/O IN THIS DB                    4,472

 *** DB-RECORD LENGTH SUMMARY:

          TOTAL DB-R LENGTH IN THIS DB
           - LENGTH (BYTES)                     3,052,423
          NBR DB-RECORDS
           - IN THIS DB                             4,472

IMS HIGH PERFORMANCE UNLOAD                                    "DB STATISTICS"                                    PAGE:     2
5655-E06                                                DATE: 06/01/2021  TIME: 20.09.01                    FABHC00  - V1.R2


STATISTICS FOR DB=HDAM0010 TOTAL FOR WHOLE DB

 NBR I/O TO READ ***AT RANDOM*** THE RETRIEVED
 SEGMENTS OF ONE DB RECORD WITH   4 PCB-BUFFERS

 NBR IO         NBR DB-R        PCT    CUM PCT
 ------------------------------------------------

   <=  1           3,925      87.76      87.76
   <=  2             413       9.23      97.00
   <=  3             100       2.23      99.23
   <=  4              26        .58      99.82
   <=  5               6        .13      99.95

   <=  6               2        .04     100.00

------------------------------------------------


 DB RECORD LENGTH (INCL PREFIXES OF SEGMENTS)
 EXPRESSED IN BLOCKSIZE/CI-SIZE UNITS
 (ONE BLKSIZE/CI-SIZE =  4,096 BYTES)

 LENGTH         NBR DB-R        PCT    CUM PCT
 ----------------------------------------------

   <= 1/16             25        .55        .55
   <= 2/16            339       7.58       8.13
   <= 3/16          3,237      72.38      80.52
   <= 4/16            714      15.96      96.48
   <= 5/16             94       2.10      98.59
   <= 6/16             18        .40      98.99
   <= 7/16              5        .11      99.10
   <= 8/16             11        .24      99.35
   <= 9/16              3        .06      99.41
   <=10/16              5        .11      99.53
   <=11/16              3        .06      99.59
   <=12/16              3        .06      99.66
   <=13/16              1        .02      99.68
   <=14/16              2        .04      99.73
   <=15/16              2        .04      99.77

   <=  1                2        .04      99.82
   <=  2                8        .17     100.00

----------------------------------------------
```

Figure 83. DB Statistics after tuning (Part 1 of 5)

```
IMS HIGH PERFORMANCE UNLOAD                                    "DB STATISTICS"                                    PAGE:     3
5655-E06                                                 DATE: 06/01/2021  TIME: 20.09.01                      FABHC00  - V1.R2


STATISTICS FOR DB=HDAM0010 (ROOT ADDRESSABLE AREA)


 *** DBDGEN SPECIFICATIONS:

          DB-ORG                                 HDAM
          DDNAME                                 DSHDAM01 (RAA)
          BLOCK/CI-SIZE                             4,096
          FREE SPACE
           - FREE BLOCK FREQUENCY FACTOR(FBFF)          0
           - FREE SPACE PERCENTAGE FACTOR(FSPF)         0

 *** KEY INDICATORS FOR QUALITY OF PHYSICAL ORGANIZATION:

          AVG NBR I/O IN RAA
           - PER DB-R IN RAA                         1.14
           - PER DB-R IN THIS DB                     1.14
          AVG DB-R LENGTH IN RAA
           - PER DB-R IN RAA                          663
           - PER DB-R IN THIS DB                      663
          ACTUAL PCT FREE SPACE                     27.65  ACTUAL PCT FREE SPACE  = (ALLOC'ED DASD - TOTAL DB-R)/ALLOC'ED DASD * 100
          DEFINED PCT FREE SPACE                      .00  DEFINED PCT FREE SPACE = FSPF + 100/FBFF - FSPF/FBFF

 *** I/O SUMMARY:

          NBR I/O IN RAA                            5,113
          NBR DB-RECORDS
           - WITH I/O IN RAA                        4,472
           - IN THIS DB                             4,472

 *** DB-RECORD LENGTH SUMMARY:

          TOTAL DB-R LENGTH IN RAA
           - LENGTH (BYTES)                     2,966,533
          NBR DB-RECORDS
           - IN RAA                                 4,472
           - IN THIS DB                             4,472

IMS HIGH PERFORMANCE UNLOAD                                    "DB STATISTICS"                                    PAGE:     4
5655-E06                                                 DATE: 06/01/2021  TIME: 20.09.01                      FABHC00  - V1.R2


STATISTICS FOR DB=HDAM0010 (ROOT ADDRESSABLE AREA)


 NBR I/O TO READ ***AT RANDOM*** THE RETRIEVED                                  DB RECORD LENGTH (INCL PREFIXES OF SEGMENTS)
 SEGMENTS OF ONE DB RECORD WITH   4 PCB-BUFFERS                                 EXPRESSED IN BLOCKSIZE/CI-SIZE UNITS
                                                                                (ONE BLKSIZE/CI-SIZE =  4,096 BYTES)


  NBR IO         NBR DB-R        PCT    CUM PCT                                 LENGTH         NBR DB-R        PCT    CUM PCT
 ---------------------------------------------                                 ---------------------------------------------

                                                                                  <= 1/16             25        .55        .55
                                                                                  <= 2/16            339       7.58       8.13
                                                                                  <= 3/16          3,237      72.38      80.52
                                                                                  <= 4/16            714      15.96      96.48
                                                                                  <= 5/16            157       3.51     100.00

   <=  1           3,978      88.95      88.95
   <=  2             384       8.58      97.54
   <=  3              81       1.81      99.35
   <=  4              23        .51      99.86
   <=  5               4        .08      99.95

   <=  6               2        .04     100.00
---------------------------------------------                                 ---------------------------------------------
```

Figure 84. DB Statistics after tuning (Part 2 of 5)

```
IMS HIGH PERFORMANCE UNLOAD                                    "DB STATISTICS"                                    PAGE:     5
5655-E06                                                 DATE: 06/01/2021  TIME: 20.09.01                      FABHC00  - V1.R2


STATISTICS FOR DB=HDAM0010 (OVERFLOW AREA)


 *** DBDGEN SPECIFICATIONS:

          DB-ORG                                 HDAM
          DDNAME                                 DSHDAM01 (OVFLW AREA)
          BLOCK/CI-SIZE                             4,096
          FREE SPACE
           - FREE BLOCK FREQUENCY FACTOR(FBFF)          0
           - FREE SPACE PERCENTAGE FACTOR(FSPF)         0

 *** KEY INDICATORS FOR QUALITY OF PHYSICAL ORGANIZATION:

          AVG NBR I/O IN OVFLW AREA
           - PER DB-R IN OVFLW AREA                  1.33
           - PER DB-R IN THIS DB                      .01
          AVG DB-R LENGTH IN OVFLW AREA
           - PER DB-R IN OVFLW AREA                1,363
           - PER DB-R IN THIS DB                      19
          ACTUAL PCT FREE SPACE                     4.69  ACTUAL PCT FREE SPACE  = (ALLOC'ED DASD - TOTAL DB-R)/ALLOC'ED DASD * 100
          DEFINED PCT FREE SPACE                     .00  DEFINED PCT FREE SPACE = FSPF + 100/FBFF - FSPF/FBFF

 *** I/O SUMMARY:

          NBR I/O IN OVFLW AREA                        84
          NBR DB-RECORDS
           - WITH I/O IN OVFLW AREA                    63
           - IN THIS DB                             4,472

 *** DB-RECORD LENGTH SUMMARY:

          TOTAL DB-R LENGTH IN OVFLW AREA
           - LENGTH (BYTES)                        85,890
          NBR DB-RECORDS
           - IN OVFLW AREA                             63
           - IN THIS DB                             4,472

IMS HIGH PERFORMANCE UNLOAD                                    "DB STATISTICS"                                    PAGE:     6
5655-E06                                                 DATE: 06/01/2021  TIME: 20.09.01                      FABHC00  - V1.R2


STATISTICS FOR DB=HDAM0010 (OVERFLOW AREA)


 NBR I/O TO READ ***AT RANDOM*** THE RETRIEVED                                  DB RECORD LENGTH (INCL PREFIXES OF SEGMENTS)
 SEGMENTS OF ONE DB RECORD WITH   4 PCB-BUFFERS                                 EXPRESSED IN BLOCKSIZE/CI-SIZE UNITS
                                                                                (ONE BLKSIZE/CI-SIZE =  4,096 BYTES)


 NBR IO         NBR DB-R        PCT    CUM PCT                                  LENGTH         NBR DB-R        PCT    CUM PCT
-----------------------------------------------                                ----------------------------------------------

                                                                                 <= 1/16             18      28.57      28.57
                                                                                 <= 2/16              4       6.34      34.92
                                                                                 <= 3/16             11      17.46      52.38
                                                                                 <= 4/16              4       6.34      58.73
                                                                                 <= 5/16              5       7.93      66.66
                                                                                 <= 6/16              3       4.76      71.42
                                                                                 <= 7/16              3       4.76      76.19
                                                                                 <= 8/16              1       1.58      77.77
                                                                                 <= 9/16              2       3.17      80.95
                                                                                 <=10/16              2       3.17      84.12
                                                                                 <=11/16              2       3.17      87.30
                                                                                 <=12/16              0        .00      87.30
                                                                                 <=13/16              1       1.58      88.88
                                                                                 <=14/16              1       1.58      90.47
                                                                                 <=15/16              0        .00      90.47

   <=  1              44      69.84      69.84                                   <=  1                0        .00      90.47
   <=  2              17      26.98      96.82                                   <=  2                6       9.52     100.00
   <=  3               2       3.17     100.00
  -----------------------------------------------                               ----------------------------------------------
```

Figure 85. DB Statistics after tuning (Part 3 of 5)

```
IMS HIGH PERFORMANCE UNLOAD                                  "RANDOMIZING STATISTICS"                              PAGE:     1
5655-E06                                                  DATE: 06/01/2021  TIME: 20.09.01                    FABHC00  - V1.R2


STATISTICS FOR RETRIEVED ROOTS OF DB=HDAM0010 WITH RANDOMIZER=DFSHDC10


 *** DBDGEN SPECIFICATIONS:

           NBR RAPS PER BLOCK/CI                          7
           HIGHEST BLOCK/CI IN RAA                    1,000
           BYTES-LIMIT                                1,280
           BLOCK/CI-SIZE                              4,096
           FREE SPACE:
            - FREE BLOCK FREQUENCY FACTOR(FBFF)           0
            - FREE SPACE PERCENTAGE FACTOR(FSPF)          0

 *** KEY INDICATORS FOR QUALITY OF RANDOMIZING:

           AVG NBR I/O
            - PER DB-R                                 1.16
            - ON RAP CHAIN PER ROOT                    1.06
            - ON RAP CHAIN PER RAP CHAIN               1.07
            - IN RAA PER DB-R                          1.14
            - IN OVERFLOW AREA PER DB-R                 .01

           NBR OF ACCESSED RAPS WHICH ARE USED
            - NBR                                      3,347
            - PCT OF ACCESSED RAPS                     47.81

           NBR OF ACCESSED ROOTS IN OVERFLOW AREA
            - NBR                                          0
            - PCT OF ACCESSED ROOTS                      .00

           PACKING DENSITY OF RAA                      72.35
           NBR OF RAPS PER ROOT                         1.56
           AVG POSITION OF ROOT SEGMENTS ON
              RAP SYNONYM CHAINS                        1.30
           AVG NUMBER OF SYNONYMS ON
              RAP SYNONYM CHAINS                        1.33

           AVG DB-R LENGTH                               682

IMS HIGH PERFORMANCE UNLOAD                                  "RANDOMIZING STATISTICS"                              PAGE:     2
5655-E06                                                  DATE: 06/01/2021  TIME: 20.09.01                    FABHC00  - V1.R2


STATISTICS FOR RETRIEVED ROOTS OF DB=HDAM0010 WITH RANDOMIZER=DFSHDC10


 NBR I/O ON RAP CHAIN TO READ ***AT RANDOM***
 THE ROOT SEGMENTS WITH   4 PCB-BUFFERS

  NBR IO        NBR ROOTS        PCT    CUM PCT
 ------------------------------------------------

   <=  1           4,188      93.64      93.64
   <=  2             271       6.04      99.70
   <=  3              12        .26      99.97
   <=  4               1        .02     100.00




 POSITION OF ROOT SEGMENTS ON RAP CHAINS

 POSITION       NBR ROOTS        PCT    CUM PCT
 ------------------------------------------------

   <=  1           3.347      74.84      74.84
   <=  2             917      20.50      95.34
   <=  3             175       3.91      99.26
   <=  4              27        .60      99.86
   <=  5               5        .11      99.97
   <=  6               1        .02     100.00
```

Figure 86. DB Statistics after tuning (Part 4 of 5)

```
IMS HIGH PERFORMANCE UNLOAD                          "DB RECORD LENGTH DISTRIBUTION"                          PAGE:     1
5655-E06                                           DATE: 06/01/2021  TIME: 20.09.01                      FABHC00  - V1.R2

                                  DATABASE RECORD LENGTH DISTRIBUTION FOR DB=HDAM0010

       LENGTH         NBR DB-R        PCT    CUM PCT
 ------------------------------------------------------

 <=       500              349       7.80       7.80
 <=       550              740      16.54      24.35
 <=       600              264       5.90      30.25
 <=       650            1,355      30.29      60.55
 <=       700              260       5.81      66.36
 <=       750              551      12.32      78.68
 <=       800              176       3.93      82.62
 <=       850              329       7.35      89.98
 <=       900              113       2.52      92.50
 <=       950               82       1.83      94.34
 <=      1000               81       1.81      96.15
 <=      1050               34        .76      96.91
 <=      1100               32        .71      97.62
 <=      1150               12        .26      97.89
 <=      1200               18        .40      98.30
 <=      1250                9        .20      98.50
 <=      1300                7        .15      98.65
 <=      1350                4        .08      98.74
 <=      1400                5        .11      98.85
 <=      1450                3        .06      98.92
 <=      1500                3        .06      98.99
 <=      1600                1        .02      99.01
 <=      1700                1        .02      99.03
 <=      1800                3        .06      99.10
 <=      1900                4        .08      99.19
 <=      2000                6        .13      99.32
 <=      2500                7        .15      99.48
 <=      3000                7        .15      99.64
 <=      3500                3        .06      99.70
 <=      4000                5        .11      99.82
 <=      5000                2        .04      99.86
 <=      6000                2        .04      99.91
 <=      7000                2        .04      99.95
 <=      8000                2        .04     100.00
```

Figure 87. DB Statistics after tuning (Part 5 of 5)

Other factors influencing the performance of access to an HDAM database

The following information pertains to other factors that influence the performance in the access to an HDAM database.

Subtopics:

Periodical database reorganization

The overflow area of an HDAM database tends to become physically disorganized, if a large amount of database segments are inserted into the overflow area after a database load or reload.

You can detect the need for reorganization of an HDAM database by observing how the average number of I/Os per database record evolves over time. A substantial increase of the number of I/Os indicates a probable need for database reorganization.

You can also detect the need for reorganization by looking at the Data Set I/O Statistics after a sequential processing of the database by FABHURG1, FABHFSU, or an HSSR application program. After such a processing, look in the Data Set I/O Statistics for the counters that describe the number of DIRECT IO and SEQ IO in the overflow area. (You will find this information only if the CAB buffer handler has been used and if an OVERFLOW CAB has been activated.) Large overflow areas may benefit from a database reorganization if the number of direct I/Os in the overflow area is much higher than the number of sequential I/Os in the overflow area.

Databases with long database records

For HDAM databases with a high percentage of very long database records (for example, 8 KB or more), it is difficult to achieve a low value for the average number of I/O per database record.

Whether the database record length is high can be determined by looking at the Database Tuning Statistics; they provide the average database record length and a table with the distribution of the database record length.

Some of the actions that can be taken, in order to limit the performance problems of long database records, are:

- Define the bytes limit in such a way that:
  - The root segment and a reasonable amount of dependent segments can be stored in the same block of the root addressable area as the root segment.
  - The remaining dependent segments are stored in the overflow area.

  A practice sometimes used is to load the most often referred-to database segments in the root addressable area at database initial load or reload time; the less frequently referred-to database segments are then inserted after the initial load/reload. Sometimes less frequently accessed segment types can be stored in a separate data set group.

  Bytes limits larger than the block or CI size should not be used if the database administrator intends to store database roots in the majority of the blocks or CIs of the root addressable area.
- Use a large block or CI size (for example, 12 KB).
- Evaluate usage of database segment compression in order to reduce the average size of database records.
- Evaluate changes to the database design in order to reduce the average size of database records.
- Evaluate usage of multiple data set groups.

Compressed segments

Using a compression exit routine in order to store the segments on DASD in a compressed format can be useful, especially if the compression can significantly reduce the average database record size in such a way that the average number of database I/Os required to read at random all database segments of a database record can be lowered.

When using compression routines, beware of segment splits created by replace calls that extend the compressed size of a database segment. Some protection against such replace calls can be achieved by avoiding high packing densities in the root addressable area and by defining enough free space for nonprimary data set groups.

Inefficient space suballocation for the Sequential Subset Randomizer

The quality of the randomizing with the Sequential Subset Randomizer depends on a reasonable suballocation of space to each subset of database records.

The output of the FABIGEN and of the SS-STATS statistics of HSSR Engine can be used to compare:

- The relative amount of space that has been allocated during FABIGEN to each subset (see per mill figures in the compilation output of the FABIGEN).
- The relative amount of DASD space occupied by all database segments of one subset (see per mill figures of the SS-STATS).

In case of large discrepancies, the database might need to be reloaded after a change of the space suballocations for the FABIGEN.

Tuning a HIDAM database

The following topics explain how to tune a HIDAM database.

Average number of I/O operations per database record

Obviously, a database can be considered to be efficiently organized, if the average number of I/O operations required to read at random all database segments of one database record is low. The average number of I/Os required to read at random all database segments of one database record is one of the most important indicators for the quality of the physical organization of the database.

This average number of I/O is printed by the Database Tuning Statistics in Figure 78 on page 324, to the right of the phrase "AVG NBR I/O IN THIS DB" (note that HSSR Engine ignores I/Os required to access the index of the HIDAM database record).

By looking at this average number in the Database Tuning Statistics, the database administrator can very rapidly see whether a database is efficiently organized.

For an ideal database consisting of one single data set group, this average number would be 1.0.

In real life, this ideal value of 1.0 can seldom be achieved and the average number of database I/Os per database record will be higher.

As a general rule of thumb, the database can be considered to be fairly well organized if the average number of I/Os per database record is below:

- 1.10 (for databases with an average database record length below one 10th of the block or CI size)
- 1.20 (for databases with larger database record lengths)

For numbers above 1.10 or 1.20, the database can often be considered to be poorly organized. In this case, the following questions should be asked in order to find the reason for a poor organization:

1. Is a database reorganization overdue in order to reduce database segment scattering? (See “Periodical database reorganization” on page 344 for details.)
2. Is the amount of free space specified during DBDGEN sufficient? (See “Free space specifications” on page 345 for details.)
3. Is the block size or CI size appropriate for the database record lengths of this database? (See “CI size and block size” on page 345 for details.)
4. Is the database record length excessive? (See “Databases with long database records” on page 345 for details.)

Periodical database reorganization

An HIDAM database tends to become physically disorganized if a high amount of database segments are inserted after a database load or reload.

You can detect the need for a reorganization of an HIDAM database by observing how the average number of I/Os per database record evolves over time. A substantial increase of the number of I/Os indicates a probable need for database reorganization.

You can also detect the need for a reorganization of the database by looking at the Data Set I/O Statistics after a sequential processing or unloading of the database. After such a processing, look in the Data Set I/O Statistics for those counters that describe the number of DIRECT IO and SEQ IO (you will find this information only if the CAB buffer handler has been used).

Databases might benefit from a database reorganization if the number of direct I/Os is higher than the number of sequential I/Os.

Free space specifications

Specification of enough free space (both in form of free space in each block or CI and in the form of a free block frequency factor) allows the support of more database inserts between two database reorganizations.

The Database Tuning Statistics shows (for an example, see the top portion of Figure 78 on page 324):

- How much free space has been specified in each block or CI
- Whether every *n*th block or CI has been left entirely free.

The Database Tuning Statistics allows also for the comparison of the actual percentage of free space (ACTUAL PCT FREE SPACE) and the defined percentage of free space (DEFINED PCT FREE SPACE). You might wish to observe how the actual percentage of free space is reduced as the database becomes older and older.

CI size and block size

The CI size or the block size can be found in the DB Statistics report.

The current CI size/block size is shown in the top portion of Figure 78 on page 324.

As a general rule of thumb, use a 4 KB CI/block size for the data portion of HIDAM.

However, if the average database record length is more than 1000 bytes (that is, larger than 1/4th of the block or CI size), then an increase of the block or CI size from 4 KB to 8 KB might improve the performance of HIDAM database accesses.

In these cases an increase of the CI size/block size from 4 KB to 8 KB can often reduce the number of I/O operations (but will increase slightly the time of an I/O operation).

CI/block sizes larger than 8 KB should be an exception (for example, for very large average database record sizes).

Databases with long database records

For HIDAM databases with a high percentage of very long database records (for example, 32 KB or more), it is difficult to achieve a low value for the average number of I/O per database record.

Some of the actions that can be taken to limit the performance problems of long database records are:

- Use a large block or CI size (for example, 12 KB)
- Evaluate usage of database segment compression in order to reduce the average size of database records
- Evaluate storage of the most often referred-to database segments in the same block or CI as the root. The less frequently referred-to database segments can be inserted after the initial load/reload or can be stored in another data set group.
- Evaluate changes to the database design in order to reduce the average size of database records.

Tuning a HISAM database

The following topics explain how to tune a HISAM database.

Average number of I/O operations per database record

Obviously, an HISAM database can be considered to be efficiently organized if the average number of I/O operations required to read, at random, all database segments of one database record is low. The average number of I/Os required to read at random all database segments of one database record is one of the most important indicators for the quality of the physical organization of the database.

This average number of I/O is printed by the Database Tuning Statistics, as shown in Figure 78 on page 324, right of the phrase "AVG NBR I/O IN THIS DB".

(HSSR Engine counts the reading of a KDSD record as a single I/O and ignores I/Os required to access the VSAM index CIs.)

By looking at this average number in the Database Tuning Statistics, the database administrator can very rapidly see whether a database is efficiently organized.

For an ideal HISAM database, this average number would be 1.0.

In real life, this ideal value of 1.0 can seldom be achieved and the average number of database I/Os per database record will be higher.

For numbers above 1.30, the HISAM database can often be considered as poorly organized. In this case, the following questions should be asked in order to find the reason for a poor organization:

1. Is the KSDS record length appropriate? (See "KSDS record length (HISAM)" on page 346 for details.)
2. Is a database reorganization overdue in order to reduce database segment scattering? (See "Periodical database reorganization" on page 346 for details.)
3. Is the ESDS CI size appropriate for the database record lengths of this DB? (See "ESDS CI size" on page 347 for details.)

KSDS record length (HISAM)

The database administrator needs to perform trade-offs when determining the KSDS record length for a HISAM database.

Selecting a large KSDS LRECL allows reduction of the number of database records that cannot be stored entirely in the KSDS record. This will reduce the average number of I/Os per database record. On the other hand, selection of a KSDS LRECL that is too large may represent a waste of DASD space, if a high percentage of database record do not fill reasonably the KDSD record.

The table containing the distribution of the database record length (see Figure 82 on page 330) provides assistance in determining a reasonable KSDS LRECL.

Note: By default, HSSR Engine prints only a table with following ranges of database record length: 1/16, 2/16, 3/16, and so on of the ESDS CI size. These default ranges of database record lengths are seldom sufficient to determine a good KSDS LRECL. To get a table with more appropriate ranges of database record length, provide in the HSSRLDEF data set control statements defining database record length ranges of your choice (for example, database record length ranges of 50 or 100 bytes).

Periodical database reorganization

The ESDS portion of an HISAM database tends to become physically disorganized if a large amount of database segments are inserted after a database load or reload.

The need for a reorganization of the ESDS portion of an HISAM database can be detected by observing how the average number of I/Os per database record evolves over time. A substantial increase of the number of I/Os indicates a probable need for database reorganization.

The need to reorganize the ESDS part of a HISAM database can also be detected by looking at the Data Set I/O Statistics after a sequential processing or unloading of the database. After such a processing, look in the Data Set I/O Statistics for the counters that describe the number of DIRECT IO and SEQ IO. (You will find this information only if the CAB buffer handler has been used.)

Databases might benefit from a database reorganization if the number of direct I/Os is higher than the number of sequential I/Os.

Note that the need for a reorganization of the KSDS portion of an HISAM database can be determined by looking at the number of CI splits and CA splits of the KSDS (these numbers can be found in LISTCAT listings of IDCAMS).

ESDS CI size

This topic discusses the ESDS CI size.

The current ESDS CI size is shown in the top portion of Figure 78 on page 324.

As a general rule of thumb, use a 4 KB CI size for the ESDS portion of HISAM database.

However, for long database records, an increase of the ESDS CI size from 4 KB to 8 KB might improve the performance of HISAM database accesses.

How to determine randomizing parameters by using a reasonable first guess method

This topic is provided to assist database administrators who do not have experience with the specifications of efficient HDAM randomizing parameters. This topic contains general rules of thumb for using a *first guess* method to determine the values of the following parameters, which have an impact on the randomizing performance of an HDAM database.

- The block or CI size (See "Determining the block or CI size" on page 347.)
- The bytes limit (See "Determining the bytes limit" on page 347.)
- The number of blocks or CIs in the root addressable area (See "Determining the number of blocks or CIs in the root addressable area" on page 348.)
- The number of RAPs per block or CI (See "Determining the number of RAPs per block/CI" on page 348.)

The following discussion assumes that the values of these parameters are determined in the sequence listed here.

After loading or reloading the database with the randomizing parameters determined by the first-guess method, the database administrator can obtain Database Tuning Statistics in order to verify the efficiency of the first-guess randomizing parameters. Then, if necessary, the randomizing parameters could be adjusted with an iterative, experimental *try and see* process (that is, change the value of a randomizing parameter, reload the database, and observe in the Database Tuning Statistics the effect of the changed parameter).

Determining the block or CI size

A recommended general rule of thumb is a 4 KB block or CI size.

However, if the average database record length reported in the Database Tuning Statistics is larger than 800 bytes, then a larger block or CI size could be tried.

- If the average database record length is in the range of 800–1600 bytes, an 8 KB block or CI size is often a better bet.
- If the average database record length is in the range of 1600–2400 bytes, a 12 KB block or CI size should be investigated.
- For average database record lengths substantially longer than 2400 bytes, the first-guess method described in this topic often does not apply.

Block or CI sizes larger than 12 KB should be an exception.

Block or CI sizes smaller than 4 KB are seldom reasonable.

Determining the bytes limit

As explained in "Bytes limit" on page 334, the database administrator needs to perform trade-offs when specifying a bytes limit.

To define a reasonable bytes limit, the database administrator should check the distribution of the database record lengths in the Database Tuning Statistics. The table with the distribution of the database

length records should be used in order to determine the bytes limit in accordance with the following rules of thumb:

- The bytes limit is large enough, in order to allow for a high percentage of database records to have all their database segments stored in the root addressable area.
- The bytes limit is small enough in order to prevent the cascading effect described in “Bytes limit” on page 334.

  If the average database record length is around 1/5th of the block or CI size, then (as a first guess, which could be revised by experiments by the database administrator) the bytes limit should not be larger than twice the database record length.

  For smaller average database record lengths, the ratio between bytes limit and average database record length can be larger than 2. For larger average database record lengths, the ratio between bytes limit and average database record length should be smaller than 2.
- The bytes limit should not be larger than the block or CI size.

Determining the number of blocks or CIs in the root addressable area

In order to determine the number of blocks or CIs in the root addressable area, the database administrator must have previously determined the block or CI size and the bytes limit (as described in “Determining the block or CI size” on page 347 and “Determining the bytes limit” on page 347).

Then the database administrator should:

1. Select a target packing density for the root addressable area.

   As described in “Packing density of the root addressable area” on page 332, a good bet would be a packing density of 75% (for databases with an average database record length smaller than one tenth of the block or CI size) or of 70% (for databases with a larger average database record length).
2. Estimate the total lengths (expressed in bytes) of all the database segments that should be stored in the root addressable area. This estimate can be done by using:

   a. The total lengths of all database segments (expressed in bytes) reported by the Database Tuning Statistics

   b. The selected bytes limit

   c. The distribution of the database record lengths reported by the Database Tuning Statistics

   Example:

   a. In Figure 78 on page 324, the reported total length of all database segments is 3,052,423.

   b. As suggested at the end of “Bytes limit” on page 334, a bytes limit of 1208 bytes can be selected.

   c. As shown in Figure 78 on page 324, 98.59% of database records have a database record length smaller than or equal to 1280 bytes (5/16 bytes of the CI size).

      Based on the figures shown in this example, you can estimate the total length of all database segments that should be stored in the root addressable areas; approximately 98.6% of 3,052,423 bytes.
3. Divide the total length of database segments to be stored in the root addressable area by the target packing density in order to obtain the size (in bytes) of the root addressable area.
4. Divide the size in bytes of the root addressable area by the block or CI size in order to obtain the number of blocks or CIs in the root addressable area.

Determining the number of RAPs per block/CI

As a general rule of thumb, the total number of RAPs should be around 1.5 times the number of database records. In case of doubt, it is better to specify too many than too few RAPs.

Since the number of database records is printed by the Database Tuning Statistics, the database administrator can multiply this number by 1.5 in order to obtain the desired total number of RAPs.

Then, in order to obtain the number of RAP per block or CI, the total number of RAPs should be divided by the number of blocks or CIs in the root addressable area (and rounded up to the next integer).

Chapter 29. Creating a database extract for tuning experiments

FABHEXTR allows the creation of a small database extract from a large database. The database extract can be used to perform database tuning experiments. Creation of a smaller extract database can be useful when the database administrator intends to tune the randomizing parameters of a large HDAM database through an interactive try-and-see process.

FABHEXTR is a user exit routine of the FABHURG1 database unload utility. Based on control statements, FABHEXTR directs the FABHURG1 utility to include, in its database unload output file, only a subset of the real-life database records. The unloaded database extract is then used as input to the standard IMS HD Reload utility, which will create an extract database.

Assume, for example, you have a large HDAM database with 1 million database records, which takes 20 hours to reorganize. An iterative try-and-see tuning process for such a huge database is not realistic because of the large amount of time required for the database reorganization of each interaction step. However, if the database administrator creates a small database extract consisting of 5000 database records, then the database reorganization for this database extract no longer takes 20 hours, but only 6 minutes. The database administrator can then perform a lot of different experiments with different values for the randomizing parameters with this small database.

The user of FABHEXTR can define, in an EXTR control statement, the number (*n*) of consecutive database records that should be included in the extract database. Usually these will be the first *n* database records of the original database. However, you can specify, in an optional SKIP control statement, that the FABHURG1 utility should skip *m* database records before extracting *n* consecutive database records.

For a HALDB, an alternative way to extract database records from a HALDB partition is provided. The user can define, in a PARTEXTR control statement, the number (*n*) of consecutive database records that should be extracted from each HALDB partition. The first *n* database records of each partition are extracted and put in the same unloaded data set.

Topics:

Considerations when applying the FABHEXTR exit routine

The following considerations apply to using the FABHEXTR exit routine to create a database extract.

- If the extracted database is being used for experimenting with the tuning of HDAM randomizing parameters, the database administrator will need to use a test version of the DBD that is smaller than the root addressable size of the original database. For example, if the original database contains 200,000 blocks in the root addressable area, and if the extract database will contain only one hundredth of the original database records, then the size of the root addressable area of the extract database should be reduced by the same proportion of one hundred and should be set to 2000 blocks.
- If the database is involved in a logical relationship, you must run (as for any other IMS database reorganization) the Database Pre-Reorganization utility (DFSURPR0) before reloading the database extract. The other IMS utilities (Database Scan, Database Prefix Resolution, Database Prefix Update) used to support logical relationships during a database reorganization should not be run.
- Duplicate control statements are not allowed. PARTEXTR control statement must not be specified with either EXTR or SKIP control statement.
- For HALDB, the order of the partitions to be processed is determined by the main logic of FABHURG1. If the partitions to be processed are restricted by the PARTITION control statement specified in SYSIN of

FABHURG1, database records are unloaded from only the selected partitions. If you want to process all of the partitions, you should not specify the PARTITION control statement.

- If the CO control statement is specified in HSSROPT DD and the PARTEXTR control statement is specified in HSSREXTR DD, the message FABH0205W is issued and the CO control statement is ignored.
- If the PARTEXTR control statement is specified and one or more partitions of PHDAM or PHIDAM are in a HALDB OLR cursor-active status, FABHURG1 ends abnormally.

Extracting a subset of database records with FABHEXTR

You can create a small database extract from a large database by applying the FABHEXTR user exit routine in the FABHURG1 job.

Procedure

1. Prepare FABHURG1 utility JCL.

   For instructions, see "Unloading a database with FABHURG1" on page 35.
2. Activate the FABHEXTR exit routine. In the FABHURG1 SYSIN data set, specify an EXIT control statement and the name of the FABHEXTR exit routine as follows:

   ```
   //SYSIN DD   *
   EXIT FABHEXTR
   ```

   This control statement must have the following format:

   - Columns 1 - 4 must contain EXIT.
   - Column 5 must contain a blank.
   - Columns 6 - 13 must contain FABHEXTR.
3. Code an HSSREXTR DD statement for the data set of FABHEXTR control statements.

   In this data set, specify either an EXTR control statement that describes the number of database records to be extracted from the entire database, or a PARTEXTR control statement that describes the number of database records to be extracted from each HALDB partition. If an EXTR control statement is specified, this data set can also contain a SKIP control statement.

   For the formats and the parameters of these control statements, see "HSSREXTR input data set for FABHEXTR" on page 352.
4. Run the job.

Example

See "JCL example for creating a database unload extract" on page 354 for a JCL example to create a database unload extract by using the FABHEXTR exit routine.

HSSREXTR input data set for FABHEXTR

An HSSREXTR DD statement is required by the FABHEXTR exit routine.

This DD statement must point to a sequential data set or to a member of PDS, which contains control statements.

The data set must contain either an EXTR control statement or a PARTEXTR control statement. If it contains an EXTR control statement, it can also contain a SKIP control statement. A PARTEXTR control statement can be specified only for HALDB.

```
   //HSSREXTR DD   *
   EXTR nnnnnnnn
   SKIP mmmmmmmm
or
   //HSSREXTR DD   *
   PARTEXTR nnnnnnnnn
```

Figure 88. HSSREXTR control statements

Subtopics:

EXTR control statement

The EXTR control statement specifies the number of consecutive database records to be extracted.

The EXTR control statement must have the following format:

- Columns 1 - 4 must contain EXTR.
- Column 5 must contain a blank.
- The number of consecutive database records to be extracted must be coded from column 6 onward. The number must be less than or equal to 999999999.

SKIP control statement

The SKIP control statement specifies the number of database records to be skipped.

The optional SKIP control statement that can be specified with the EXTR control statement must have the following format:

- Columns 1 - 4 must contain SKIP.
- Column 5 must contain a blank.
- The number of database records to be skipped at the beginning of the database must be coded from column 6 onward. The number must be less than or equal to 999999999.

PARTEXTR control statement

The PARTEXTR control statement specifies the number of consecutive database records to be extracted from each HALDB partition.

The PARTEXTR control statement must have the following format:

- Columns 1 - 8 must contain PARTEXTR.
- Column 9 must contain a blank.
- The number of consecutive database records to be extracted from each HALDB partition must be coded from column 10 onward. The number must be less than or equal to 999999999. No leading zeros need to be specified, but the number must be left-aligned. This control statement is valid only for HALDB. If the number of database records in a partition is less than the number specified in the PARTEXTR control statement, all records in the partition are unloaded.
- You cannot specify more than one PARTEXTR control statement.
- If the PARTEXTR control statement is specified, you cannot specify both the EXTR and SKIP control statement concurrently.

JCL example for creating a database unload extract

Use the following JCL example to prepare your JCL for creating a database unload extract.

The following figure shows a JCL example for creating an extract database.

```
//UNLOAD   EXEC FABHULU,MBR=FABHURG1,DBD=dbdname
//SYSUT2     DD DISP=(,PASS),DSN=&&UNLOAD,UNIT=SYSDA,SPACE=(CYL,(3,3))
//SYSIN    DD *
EXIT FABHEXTR
//HSSREXTR DD *
EXTR 10000
//DBDD       DD DISP=SHR,DSN=user.db
//*
//*
//*
//RELOAD   EXEC PGM=DFSRRC00,PARM='ULU,DFSURGL0,dbdname'
//STEPLIB    DD ...
//DFSRESLB   DD ..
//IMS        DD DISP=SHR,DSN=IMSVS.TEST.DBDLIB
//DFSUINPT   DD DISP=(OLD,DELETE),DSN=&&UNLOAD
//DBDD       DD DISP=(,CATLG),DSN=extract.db,.......
//... (other DD statements required by Reload Utility)
```

Figure 89. JCL example for creating an extract database

In the first job step, in addition to the usual JCL required to do an unload with the FABHURG1 Unload utility, the following specifications are made:

- A SYSIN data set with an EXIT control statement, which requests the activation of FABHEXTR.
- An HSSREXTR data set with an EXTR control statement, which requests that the (first) 10000 database records be included in the database unload extract.

The second job step is a standard database reload performed with the standard IMS HD Reload utility DFSURGL0. Notice, however, that the IMS DD statement will point to a DBDLIB containing the test version of the DBD, which is used for the tuning experiments.

Part 6. Compatibility with earlier products

IMS High Performance Unload provides compatibility with earlier products.

Topics:

Chapter 30. Compatibility with DBT V2 HSSR

IMS High Performance Unload is compatible with IMS System Utilities/Data Base Tools Version 2, High Speed Sequential Retrieval (program number 5685-093, also referred to as DBT V2 HSSR), with certain exceptions.

The following topics provide information about issues that you must consider when migrating from DBT V2 HSSR.

Topics:

Default buffer handler for ESDS, OSAM, and OSAM LDS

In IMS High Performance Unload, the default buffer handler for ESDS, OSAM, and OSAM LDS is CAB, whereas in DBT V2 HSSR the default is BB.

If your JCL for DBT V2 HSSR does not specify HSSRCABP control statements to use CAB, running the job on IMS High Performance Unload will require more storage.

To avoid the storage problem, the function to change the default buffer handler is provided. For details, see Chapter 19, "Site default options," on page 261.

Default values of CAB buffering parameters

The default values for CAB buffering parameters RANSIZE, NBRSRAN, NBRDBUF, and OVERFLOW in IMS High Performance Unload are different from the default values used in DBT V2 HSSR.

The following table lists the differences of the default values of CAB buffering parameters.

Table 53. Differences of default values of CAB buffering parameters

CAB parameter	Default value in IMS High Performance Unload	Default value in DBT V2 HSSR
RANSIZE	Automatically determined from the characteristics of database data set	8
NBRSRAN	8	4
NBRDBUF	Twice the number assigned to RANSIZE	4

Table 53. Differences of default values of CAB buffering parameters (continued)

CAB parameter	Default value in IMS High Performance Unload	Default value in DBT V2 HSSR
REFT4	Equals to RANSIZE	12
OVERFLOW	CAB	BB

If your JCL for DBT V2 HSSR uses the default values for CAB buffering parameters, running the job on IMS High Performance Unload might require more storage.

To avoid this storage problem, the function to fallback the default values to the ones that are compatible with DBT V2 HSSR is provided. For details, see Chapter 19, “Site default options,” on page 261.

Location of buffer pools and compatibility of exit routines

IMS High Performance Unload always allocates buffer pools above the 16-MB line.

In DBT 2.2 HSSR, buffer pools are allocated above the 16-MB line only for VSAM data sets. In DBT 2.3 HSSR buffer pools are allocated above the 16-MB line for OSAM data sets also only when DFSMS 1.1 or later is used.

Because of these differences, you might need to link-edit your FABHURG1 exit routine or FABHFSU exit routine with the 31-bit addressing mode. For details, see the following topics:

- “User record-formatting routine” on page 246
- Chapter 18, “System programming interfaces,” on page 243 for the requirements for FABHURG1 exit routines
- “FABHFSU user exit routine” on page 69 for the requirements for FABHFSU exit routines

HSSROPT control statements: HDSTATS and NOSAMEOPT

The HDSTATS control statement in DBT V2 HSSR is accepted in IMS High Performance Unload, and is treated as an alias for the DBSTATS control statement. In IMS High Performance Unload, HDSTATS is effective for HISAM databases as well as HD databases.

The NOSAMEOPT control statement in DBT V2 HSSR is accepted in IMS High Performance Unload, but has no effect.

Access method used in Unload utilities to write output records

In FABHURG1 and FABHFSU of IMS High Performance Unload, the access method used to write output records is always QSAM and the default buffer size is determined by the block size of the output data set. In DBT V2 HSSR, both BSAM and QSAM are supported; performance is better with QSAM than with BSAM.

The affected DDs are SYSUT2 and SYSUT3 of FABHURG1, and the output DD specified by a PSB control statement of FABHFSU. In DBT V2 HSSR, the access method used for writing output records to the unloaded data set is specified in the FRMT statement for FABHURG1 or PSB statement for FABHFSU. These parameters are ignored in IMS High Performance Unload; QSAM is always used.

Method for specifying an HSSR PCB through KEYLEN

To maintain compatibility, IMS High Performance Unload supports a method for specifying an HSSR PCB through the KEYLEN keyword.

The following method for specifying an HSSR PCB is supported:

1. Code a PCB that satisfies all requirements stated in “HSSR PCB requirements” on page 80.
2. Assign a value of 200 or greater to the KEYLEN keyword parameter for the PCB. HSSR will then recognizes the PCB as an HSSR PCB.

For a PCB defined this way, the value specified on the KEYLEN keyword parameter is just the indicator of an HSSR PCB and is not related to key length. That value can also be used (if the basic buffer handler (BB) is used to access the database data sets) to change the default number of basic buffers that HSSR allocates for each database data set of the database referenced by the PCB. The default number of BB buffers, when it is used, is determined as follows:

- If the KEYLEN value is 200 or 201, two basic buffers are allocated for each ESDS, OSAM, or OSAM LDS data set of the database referenced by the PCB.
- If the KEYLEN value is greater than or equal to 202, the number of basic buffers is equal to the value minus 200.

Number of Basic Buffers for an HSSR PCB

If an HSSR PCB is defined by the use of either an HSSRPCB or an HSSRDBD control statement, and the value to the KEYLEN keyword for the PCB is less than 200, BB allocates six basic buffers for each data set as a default. If the KEYLEN value for the PCB is greater than or equal to 200, the number of basic buffers to be allocated is the same as that in the preceding description. If some other number of buffers is preferable, use the BUF control statement to allocate that number of buffers.

Support of PROCOPT=R and replace calls

PROCOPT=R is supported for compatibility. If a database is shared at the database level when PROCOPT=GR, DBRC grants access with integrity for updating the IMS subsystem.

Subtopics:

Status code in PCB feedback

The following DL/I status codes, in addition to the blank status code, are returned after a REPL call for an HSSR PCB:

Status code
Meaning

NE
Unable to find the segment.

NI
There is a duplicate segment in the unique secondary index.

VX
VX is used during a REPL call to signal that the application program has not respected the restrictions for HSSR Engine. When VX is displayed, the replace was not performed. Database positioning information is maintained normally.

DFSVSAMP DD

HSSR Engine handles replace calls by issuing IMS DL/I GH and REPL calls internally. IMS uses its own OSAM or VSAM buffer pools to process these DL/I calls.

If your application program issues REPL calls, the usual DFSVSAMP control statements must be coded to tune the processing of these internal DL/I calls.

Restrictions

The following restrictions apply to PROCOPT=R and replace calls:

- PROCOPT=R cannot be specified for HISAM, SHISAM, or secondary index databases.
- PROCOPT=R cannot be specified for PHDAM or PHIDAM databases.
- PROCOPT=R cannot be specified for logical child segments that have the following attributes:
  - Physically paired
  - Logical parent's concatenated key defined as VIRTUAL if the BLDLPCK control statement is not specified in the HSSROPT data set
- PROCOPT=R cannot be specified for secondary index source segments other than root segments.
- The REPL call for PHDAM and PHIDAM databases is not supported.
- The segment length of a variable or compressed segment must not be modified by a replace call.

Support and restriction for replace operations

HSSR Engine provides limited support of replace calls for application programs that replace a small or moderate number of retrieved database segments and modify a small or moderate number of accessed database blocks or CIs. This support includes standard IMS logging of database changes, standard IMS sync point processing, and a standard DBRC interface as provided by the host IMS batch subsystem.

Support of explicit HSSR calls

Explicit *HSSR calls* (ASMHSSR, CBLHSSR, and PLIHSSR) are supported in IMS High Performance Unload for compatibility with DBT HSSR and PO HSSR.

An explicit HSSR call consists of a name, a count parameter, a call function, a PCB, an I/O area, and an SSA (if any). PL/I application programs must include a count parameter. The explicit HSSR call names are ASMHSSR for Assembler application programs, CBLHSSR for COBOL application programs, and PLIHSSR for PL/I application programs.

Note: For a complete description of the call functions, parameters, and layout of the PCB and SSA, see *IMS Application Programming*.

The following examples show the format of the call statement for each type of application program:

Assembler application programs

- Retrieval calls without SSA and REPL calls:

```
CALL ASMHSSR,(function,pcb,ioarea),VL
CALL ASMHSSR,(three,function,pcb,ioarea),VL
```

- Retrieval calls with SSA:

```
CALL ASMHSSR,(function,pcb,ioarea,ssa),VL
CALL ASMHSSR,(four,function,pcb,ioarea,ssa),VL
```

COBOL application programs

- Retrieval calls without SSA and REPL calls:

```
CALL 'CBLHSSR' USING FUNCTION,PCB,IOAREA.
CALL 'CBLHSSR' USING THREE,FUNCTION,PCB,IOAREA
```

- Retrieval calls with SSA:

```
CALL 'CBLHSSR' USING FUNCTION,PCB,IOAREA,SSA.
CALL 'CBLHSSR' USING FOUR,FUNCTION,PCB,IOAREA,SSA
```

PL/I application programs

- Retrieval calls without SSA and REPL calls:

```
CALL PLIHSSR (three,function,pcb,ioarea);
```

- Retrieval calls with SSA:

```
CALL PLIHSSR (four,function,pcb,ioarea,ssa);
```

Considerations for explicit HSSR calls

- HSSR call parameters (count parameter, call function, PCB address, and number of parameters) are validated by HSSR Engine. If you specify HSSR PCB as a DL/I PCB, HSSR Engine does not validate HSSR call parameters, but instead transfers the call to DL/I.
- An application program that uses explicit HSSR calls should be linked with the language interface module (ASMHSSR, CBLHSSR, or PLIHSSR) in the same way that a DL/I program is linked with the DL/I language interface.
- The language interface modules ASMHSSR, CBLHSSR, and PLIHSSR are not reentrant. Therefore, the application program cannot be reentrant after link-editing.
- For the LANG=PLI option on PSBGEN, PLICALLA entry point, and the compatibility with Language Environment for your application program written in PL/I, refer to *IMS Application Programming*; the same restrictions apply to the HSSR application program.
- The module attributes (AMODE, RMODE, and so forth) of the language interface modules (ASMHSSR, CBLHSSR, and PLIHSSR) must not be changed even if the language interface module is dynamically invoked by the application program.
- The language interface modules provided by DBT 2.2 HSSR might cause the system abend 0C1. You can check whether the cause of the problem is in the old language interface module by doing:

  1. Browse or dump the HSSR application program.
  2. Look for the string 'FABHHSSR' followed by the IBM copyright statement.
  3. Check the string 'APAR(*xxxxxxx*)' that follows the copyright statement. The substring '*xxxxxxx*' indicates the maintenance level of the language interface module. If '*xxxxxxx*' is either 'KHK0013' or 'PN61005,' it is definite that the cause of the problem is in the language interface module.

  The solution for the problem is to re-link-edit the application program with the language interface module provided by IMS High Performance Unload.

FABHFSU control statements: CO and CON

The CO and the CON control statements are accepted by FABHFSU for compatibility.

Subtopics:

CO control statement

The CO control statement activates the compare option, which instructs FABHFSU to compare the content of one of its output data sets with the content of an FSU II output data set. (This option should be used only for problem determination.)

Up to three CO control statements can be provided in the CARDIN data set after the PSB control statement.

Restriction: The CO control statement cannot be specified for a PHDAM or PHIDAM database.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

COn         ddname2
```

Position
Description

1
Code the CO keyword to activate the compare option.

3
The required 1-digit entry *n* is used to specify the FABHFSU output data set to be compared. Code values of 1, 2, or 3. *n* specifies that output data set defined by the *n*th PSB control statement is to be compared with an FSU II data set.

12
The required 8-character entry defines the name of the DD statement that defines the FSU II output data set to be compared. *ddname2* is left-aligned with trailing blanks. This output data set must have been created in a prior job or job step. Also, in your FABHFSU JCL, you must code a *ddname2* DD statement that specifies the data set to be compared.

If the data sets are HSAM output data sets, the FSU II and the FABHFSU output data sets must have the same block size.

Notes:

- FABHFSU ends abnormally when it detects a relevant mismatch between itself and the FSU II output data sets. It also issues an error message.
- Do not use IEBCOMPR to compare FABHFSU to FSU II output data sets, since they are not always identical.
- The CO control statement cannot be used when NO is specified for the output format of the unloaded data set.
- When the BLDLPCK statement is specified, the CO control statement for HS-format output records is ignored.

CON control statement

The CON control statement in CARDIN data set for FABHFSU, which activates the concatenate option, allows you to run some FSU II user exit routines under FABHFSU. The exit routines expect to find the segment prefix and the segment data concatenated in contiguous virtual storage.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

CON
```

Position
Description

1
Code the CON keyword to activate the concatenate option.

When you invoke a user exit routine, both FABHFSU and FSU II pass the address of the segment prefix and the segment data to the user exit routine. For FSU II, the segment prefix and the segment data are contiguous in virtual storage. With FABHFSU, the segment prefix and the segment data are not contiguous in virtual storage.

The CON control statement instructs FABHFSU to concatenate the segment prefix and the segment data in contiguous virtual storage, before invoking the user exit routine.

Since concatenation requires additional CPU cycles, use the CON control statement only for user exit routines that access both the segment prefix and the segment data and assume both are contiguous in virtual storage.

Date specification in PSC and CTL control statements

The following information pertains to specifying dates through PSC and CTL control statements.

In IMS High Performance Unload, the year in the expiration date for a scan control data set is specified in four digits for the PSC control statement of FABHFSU and for the CTL control statement for FABHPSFC. This format is the same for DBT 2.3 HSSR.

In DBT 2.2 HSSR, the year is specified in two digits. It is recommended that the year in these two control statements in your new JCLs be coded in four digits, although the two-digit specification is also accepted.

Format of the scan control data set used in Parallel Scan Facility

The format of the scan control data set created in Parallel Scan Facility of FABHFSU is the same as the format used in DBT 2.3 HSSR, but is different from the one for DBT 2.2 HSSR. Therefore, the scan control data sets must be created again if you are migrating from DBT 2.2 HSSR.

Location of control blocks

In IMS High Performance Unload, HDMB and HRAN control blocks are above the 16-MB line.

PSPI

The following table shows where each product-sensitive control block of HSSR Engine is. The minus sign (-) indicates that the block is in private storage; the plus sign (+) indicates that the block is in extended private storage.

Table 54. Location of control blocks (DBT V2)

Control block	Location in IMS High Performance Unload	Location in DBT V2
HDMB	+	-
HJCB	-	-
HPCB	-	-
HPTR	-	-
HRAN	+	-
HSDB	-	-

User exit routines that refer to HDMB or HRAN control block and that have been link-edited with 24-bit addressing mode must be modified so that they can refer to the control blocks above the 16-MB line.

PSPI

Product-sensitive macros

For compatibility with DBT 2.1 HSSR and DBT V1 HSSR, aliases are provided for the mapping macros of control blocks of HSSR Engine.

PSPI

The following table summarizes the aliases for the mapping macros. These macros are in the HPS.SHPSMAC0 macro library.

Table 55. Aliases of mapping macros for HSSR Engine

Control block	Mapping macro	Alias
HDMB	FABHDMB	HDMB
HJCB	FABHJCB	HJCB
HPCB	FABHPCB	HPCB
HPTR	FABHPTR	HPTR
HRAN	FABHRAN	HRAN
HSDB	FABHSDB	HSDB

For the users who have difficulty in changing the name of mapping macros, an alias is provided for each of these macros to support compatibility. These aliases, however, might not be provided in future releases of this product, so you should be careful in deciding to use the aliases.

PSPI

DECN control statement and the unloaded data set

The format of the unloaded data set created by DBT V2 HSSR when the DECN control statement is specified for the FABHURG1 or FABHFSU unload utility was changed by APAR PQ22654. The new format is not compatible with any format that predates that APAR. For details, see the APAR.

The unloaded data set created by IMS High Performance Unload is the same as that created by DBT 2.3 HSSR with APAR PQ22654 applied. If you are migrating from DBT 2.3 HSSR, and APAR PQ22654 has not been applied, you might have a problem with any existing reload job for which the input is an unload data set created by DBT V2 HSSR.

Chapter 31. Compatibility with DBT V1 HSSR

IMS High Performance Unload is compatible with IMS System Utilities/Data Base Tools Version 1, High Speed Sequential Retrieval (program number 5668-856, also referred to as DBT V1 HSSR), with certain exceptions.

IMS High Performance Unload supports compatibility with DBT V1 HSSR in the following ways:

- JCL can be used without modification.
- Input control statements can be used without modification except that the following input control statements might need minor modifications if the same results of the DBT V1 HSSR are required:
  - DIAGG control statement in the HSSROPT data set
  - Position 22 of the DBD control statement in the FABHFSU CARDIN data set

    Note: For complete explanations on these specifications, see the following topics:

    - "DIAGG control statement" on page 167
    - "DBD control statement" on page 58
- An application program that uses a DBT V1 HSSR call can be used without reassembling or re-link-editing.
- A user exit routine of DBT V1 HSSR needs the following modification to be run under IMS High Performance Unload:
  - A user exit routine must be reassembled or re-link-edited or both in the same IMS system environment under which the routine is used, if the IMS macros that depend on the IMS release levels are used in the routine.
  - A user exit routine that refers to the HDMBOKEY field of the HDMB control block must be modified from a 3-byte-long field to a 4-byte-long field.
  - A user exit routine can be run either in AMODE=24 or AMODE=31, but a user exit must be link-edited with AMODE=31. See the following topics for the requirements for exit routines:
    - "User record-formatting routine" on page 246 for the requirements for FABHURG1 exit routines
    - "FABHFSU user exit routine" on page 69 for the requirements for FABHFSU exit routines

Other differences are the same as those for DBT V2 HSSR. See Chapter 30, "Compatibility with DBT V2 HSSR," on page 357.

Chapter 32. Compatibility with PO HSSR

IMS High Performance Unload supports compatibility with Program Offering High Speed Sequential Retrieval Version 2 (IFP 5787-LAC, also referred to as PO HSSR).

The following topics are intended for users who are migrating from PO HSSR to IMS High Performance Unload.

Topics:

Compatibility with Branch Office Randomizer, one of the functions provided by PO HSSR, is supported by Sequential Subset Randomizer. See Chapter 20, "Introduction to the Sequential Subset Randomizer," on page 269.

Program names

To maintain JCL compatibility, IMS High Performance Unload supports aliases for load modules.

The following table summarizes the aliases for load modules.

Module name	Alias
FABHX034	X034000
FABHBSIM	HSSRBSIM
FABHFSU	HSSRFSU
FABHTEST	HSSRTEST
FABHURG1	HSSRURG1
FABHLDBR	HSSRLDBR
FABHEXTR	HSSREXTR

No aliases are provided for the cataloged procedures. If you want to use the names of PO HSSR cataloged procedures, copy the corresponding cataloged procedures of IMS High Performance Unload by the names of those for PO HSSR. See "JCL compatibility" on page 368.

Compatibility of application programs

The following information pertains to the compatibility of application programs.

Reassembling

Reassembling your application program by use of the product-sensitive macros is required only for programs or user exit routines that refer to either the HTCB or the HDMB control block.

Relink

Basically, no relink is required for running the application programs written for PO HSSR on IMS High Performance Unload.

Relinking with the language interface module (ASMHSSR, CBLHSSR, or PLIHSSR) is required only if you want to run an application under the IMS DL/I batch region controller (DFSRRC00). This is not recommended, but PO HSSR supports it. Include the corresponding language interface module (ASMHSSR, CBLHSSR, or PLIHSSR) from IMS High Performance Unload's HPS.SHPSLMD0 library, and link-edit it with the application program; otherwise an unexpected result, such as a system abend, might occur.

Compatibility of exit routines

In IMS High Performance Unload, HDMB and HRAN control blocks are above the 16-MB line.

PSPI

The following table shows where each product-sensitive control block of HSSR Engine is. The minus sign (-) indicates that the block is in private storage; the plus sign (+) indicates that the block is in extended private storage.

Table 56. Location of control blocks (PO HSSR)

Control block	Location in IMS High Performance Unload	Location in PO HSSR
HDMB	+	-
HJCB	-	-
HPCB	-	-
HPTR	-	-
HRAN	+	-
HSDB	-	-

User exit routines that refer to HDMB or HRAN control block and that have been link-edited with 24-bit addressing mode must be modified so that they can refer to the control blocks above the 16-MB line.

See also "Mapping macros for control blocks and output records" on page 371.

PSPI

JCL compatibility

The following information pertains to JCL compatibility.

Procedures

As was explained in "Program names" on page 367, the names of IBM-supplied cataloged procedures for IMS High Performance Unload are different from those for PO HSSR. If you want to continue using the names in PO HSSR, copy the corresponding cataloged procedures by the names in PO HSSR.

The following table lists the cataloged procedures.

Table 57. Changes of procedure names

Region type	Name in IMS High Performance Unload	Name in PO HSSR
DLI Region	FABHDLI	DLIHSSR

Table 57. Changes of procedure names (continued)

Region type	Name in IMS High Performance Unload	Name in PO HSSR
DBB Region	FABHDBB	DBBHSSR
ULU Region	FABHULU	ULUHSSR

Control statements

In IMS High Performance Unload, the year is stated in four digits in each of the following control statements:

- PSC control statement of FABHFSU
- CTL control statement for FABHPSFC

Although the two-digit format is still accepted, consider coding the control statements for your new JCLs in this new format.

Default options

Certain differences between the default option values of PO HSSR and those of IMS High Performance Unload might interfere with smooth JCL migration. As a solution, IMS High Performance Unload provides the capability to change the default values to those of PO HSSR by replacing the default option table (FABHOPT).

This capability also covers the PO HSSR's HSSRGEN function, in which default options are specified.

For details, see Chapter 19, “Site default options,” on page 261.

Return codes and abend codes

The following information pertains to compatibility of return codes and abend codes.

Return codes

- The return codes from FABHURG1, FABHFSU, FABHBSIM, and FABHTEST are the same as those from the corresponding PO HSSR utilities HSSRURG1, HSSRFSU, HSSRBSIM, and HSSRTEST, respectively, except that RC01 is returned for empty databases.
- FABHPSFS returns RC01 if no segment was retrieved in all PSF phases. This return code is different from that returned from the FSUSUMM utility of FSU-II.

Abend codes

IMS High Performance Unload uses only one abend code, U4013. You cannot change the code.

Compatibility of the functions

IMS High Performance Unload provides the functions that were provided in PO HSSR.

Subtopics:

Database Tuning Statistics

Database tuning statistics are also provided by IMS High Performance Unload. The layout of the tuning statistics report is slightly different from that provided by PO HSSR, but the contents are the same.

You can use the same database tuning method you used with the PO HSSR Database Tuning Statistics reports. See Chapter 26, “Obtaining statistics for database tuning,” on page 307.

Parallel Scan Facility

IMS High Performance Unload supports compatibility with FSU II. IMS High Performance Unload supports aliases to keep the JCL compatibility with FSU II. The only change you must make is to specify the name of the IMS High Performance Unload load module library in the STEPLIB DD.

The following table shows the names and aliases of the HSSR modules.

Table 58. Names and aliases of HSSR modules

Name in IMS High Performance Unload	Alias
FABHPSFC	FSUCTRL
FABHPSFS	FSUSUMM
FABHPSFM	FSUMAP

However, the following options specified in the DBD or PSB control statements of the FSUCTRL data set are ignored by IMS High Performance Unload:

- For the DBD control statement:
  - Print line count (position 26)
  - Checkpoint option (position 30)
  - Checkpoint frequency (position 31)
  - Optional scan mode (position 40)
  - FSU service aids area (position 48)
- For the PSB control statement:
  - Format SYNAD option (position 36)

The format of the scan control data set CNTLDD created by FSU II FSUCTRL is different from the one created by IMS High Performance Unload. You cannot use the scan control data set created by FSU II FSUCTRL as an input of FABHFSU or FABHPSFS of IMS High Performance Unload. You must rerun a series of JCLs for a parallel scan by using IMS High Performance Unload.

Db2 DL/I Batch support

The method of running a PO HSSR application program that issues SQL calls is supported by IMS High Performance Unload for maintaining the compatibility with JCL that is written for PO HSSR. However, the method that is used in the IBM-supplied cataloged procedure FABHDB2 is recommended.

Note: For the method of running a PO HSSR application program that issues SQL calls, see the topic "Job Control to Run in An HSSR/IMS-Batch/Db2 Mixed-Mode Environment" in the *PO HSSR Program Descriptions and Operations Manual*.

In the FABHDB2 procedure, you must specify the following parameters:

- In the EXEC statement, specify your HSSR application program name with the MBR= keyword parameter.
- In the EXEC statement, specify the Db2 subsystem ID (*ssid*) with the SSM= keyword parameter.
- In the DDITV02 data set, specify the name FABH000, which is the IMS High Performance Unload's program controller, with the ninth positional parameter.

For more information about the FABHDB2 procedure, see “Considerations for Db2 DL/I Batch interface” on page 88.

FABHLDBR utility

The FABHLDBR utility provides the same function as the HSSRLDBR utility of PO HSSR.

The name HSSRLDBR is available as an alias of FABHLDBR, so you can use this utility without changing your JCL. For a description of the FABHLDBR utility, see Chapter 27, “Printing long database records,” on page 313.

Mapping macros for control blocks and output records

Users who are using, in their exit routine or application programs, the mapping macros provided by PO HSSR must change the name of those macros when they reassemble those programs in the IMS High Performance Unload environment.

PSPI

The relationship between the names of the mapping macros in PO HSSR and those in IMS High Performance Unload is described in the following table.

Table 59. Changes of mapping macro names for HSSR Engine

Macro name in IMS High Performance Unload	Macro name in PO HSSR
FABHDMB	HDMB
FABHJCB	HJCB
FABHPCB	HPCB
FABHPTR	HPTR
FABHRAN	HRAN
FABHSDB	HSDB
FABHURGR	HURGUREC
FABHFSUR	FSUREC

For users who have difficulty in changing the name of mapping macros, an alias is provided for each of these macros to support compatibility. These aliases, however, might not be provided in future releases of this product, so you should be careful in deciding to use the aliases.

PSPI

Chapter 33. Compatibility with FSU II

IMS High Performance Unload supports compatibility with IMS/VS Fast Scan Utility II Version 2 (FDP 5798-DFN, also referred to as FSU II).

FABHFSU provides the following compatibility with FSU II:

- FABHFSU output data sets are compatible with FSU II output data sets.
- Under FABHFSU, FSU II user exit routines can run without changes and without recompilation. The user exit routine must not access or modify internal, undocumented FSU II control block information.
- FABHFSU can often use the same PSBs as FSU II (see “PSBGEN compatibility” on page 373 for details some minor modifications might be required).
- FSU II control statements can usually be used without changes for FABHFSU. They might require minor changes if the different PCBs used to control the content of the output data sets are not contained in the same PSB.
- IMS High Performance Unload provides the aliases shown in the following table for the FSU II modules. These aliases allow the FSU II JCL to run the FABHFSU PSF.

Table 60. Aliases for FSU II modules

Name in IMS High Performance Unload	Name in FSU II
FABHPSFM	FSUMAP
FABHPSFC	FSUCTRL
FABHPSFS	FSUSUMM

PSBGEN compatibility

To ease conversions from FSU II to FABHFSU, some PSBGEN restrictions do not apply to FABHFSU.

For example:

- The PROCOPT statement fields on the PCB and SENSEG statements can have any value acceptable to IMS.
- Field sensitivity can be specified during PSBGEN, but is ignored by FABHFSU.

CON control statement

The CON control statement in CARDIN data set for FABHFSU, which activates the concatenate option, allows you to run some FSU II user exit routines under FABHFSU. The exit routines expect to find the segment prefix and the segment data concatenated in contiguous virtual storage.

```
0.......1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

CON
```

Position
Description

1
Code the CON keyword to activate the concatenate option.

When you invoke a user exit routine, both FABHFSU and FSU II pass the address of the segment prefix and the segment data to the user exit routine. For FSU II, the segment prefix and the segment data are contiguous in virtual storage. With FABHFSU, the segment prefix and the segment data are not contiguous in virtual storage.

The CON control statement instructs FABHFSU to concatenate the segment prefix and the segment data in contiguous virtual storage, before invoking the user exit routine.

Since concatenation requires additional CPU cycles, use the CON control statement only for user exit routines that access both the segment prefix and the segment data and assume both are contiguous in virtual storage.

Part 7. Troubleshooting

Use these topics to diagnose and correct problems that you experience with IMS High Performance Unload.

Topics:

Chapter 34. Troubleshooting IMS High Performance Unload problems

The information in the following topics can be used to help you troubleshoot IMS High Performance Unload problems.

Topics:

HSSR snaps

When you experience an error during initialization of HSSR Engine, you can use the snapshot that is written in the HSSRSNAP data set.

If an unexpected condition is detected while the HSSR PCBs are being initialized, the following actions are taken:

- An error message is issued.
- A snap to the HSSRSNAP data set is written.
- The initialization process is ended and falls back to DL/I. (The application program works with DL/I PCBs, and all calls are processed by DL/I modules.)

If the HSSRSNAP DD statement is not specified, IMS High Performance Unload ends abnormally instead of falling back to DL/I. If you do not need the snap but need the fallback processing, supply the following DD statement:

```
//HSSRSNAP  DD  DUMMY
```

Trapping abends issued by application programs

If your HSSR application program is written in COBOL or PL/I language, you can request that most abends that can happen during the processing of your application program be "trapped" (in other words, you can request that abends be intercepted and hurdled by the high-level language software, and that your application program get back control after an abend).

Abends are usually trapped in one of the following ways:

- By using the runtime option STAE (for OS PL/I V2 programs)
- By using the runtime option TRAP(ON) (for COBOL or PL/I programs running under Language Environment)

If you trap abends for your application programs, you might perhaps want to exclude abends issued by HSSR Engine. The reason is because the purpose of the ABEND macros that HSSR Engine issues is to result in abends; that is, not to return to the application program (in this respect, HSSR Engine is similar to IMS).

OS PL/I V2 programs

If you want to exclude abends in HSSR Engine from trapping for OS PL/I V2 programs, update the IBMBXITA Assembler user exit to include the user abend code of IMS High Performance Unload in the list of user-abend codes that should be percolated. For detailed descriptions of IBMBXITA, see *OS PL/I V2 Programming Guide*.

COBOL or PL/I programs running under Language Environment

If you want to exclude abends in HSSR Engine from the trapping for COBOL or PL/I programs running under Language Environment (LE), update the LE's CEEDOPT Assembler module that establishes installation defaults. With the ABPERC= keyword, include the user abend code of IMS High Performance Unload in the list of abend codes that should be percolated. For detailed descriptions of CEEDOPT and of the ABPERC runtime option, see *z/OS Language Environment Programming Guide*. As the Programming Guide explains, you can also request percolation of the abend code of HSSR Engine in:

- The CEEUOPT Assembler module
- The Assembler user exit CEEBXITA

FABHTEST utility for problem determination

You can use the FABHTEST utility to diagnose software errors in HSSR Engine. The FABHTEST utility is the HSSR Engine test utility that runs a sequence of HSSR or DL/I calls against an IMS database.

Activate the compare and hardcopy trace options, and run the failing call sequence with the FABHTEST utility.

- Running the failing call sequence with FABHTEST (instead of the application program) eliminates the possibility that the application program will destroy code or control blocks of IMS High Performance Unload.
- The compare option will check whether HSSR Engine returns the same PCB feedback and I/O area as DL/I would have returned.
- The hardcopy trace option provides a trace of HSSR Engine activities and control blocks. This trace can be used by the system programmer to find out why HSSR Engine made the error.
- The trace option should be activated by issuing the START keyword at a point before the error occurred.
- The TRHC control statement requests that call information, buffer handler information, control blocks (CB), and buffer control blocks (BUFCB) be traced. A TRDB control statement must be included.

Related concepts

HSSR call test utility (FABHTEST)
FABHTEST is the HSSR Engine test utility that runs a sequence of HSSR or DL/I calls against an IMS database.

Chapter 35. Messages and codes

This reference section provides detailed information about the abend codes, return codes, and messages that might be issued during the execution of IMS High Performance Unload.

Topics:

Abend code U4013

IMS High Performance Unload uses only one abend code, U4013. A message identifying the problem is always written before this abend code is issued.

Return codes

The following topics explain the return codes that are issued by FABHURG1, FABHFSU, FABHPSFS, FABHBSIM, and FABHTEST.

Tip: You can change the return codes, except for FABHPSFS, by using FABHRCEX, the Return Code Edit exit routine. For details, see "Return Code Edit exit (FABHRCEX)" on page 245.

FABHURG1 return codes

This reference topic explains the return codes of the FABHURG1 utility.

The FABHURG1 utility issues a return code that is the logical sum of the following return codes:

Code
Meaning

00
Successful completion.

01
One or both of the following conditions were detected:

- Some segment types are insensitive.
- No segment was retrieved from the database.

02
Some segments have been skipped in accordance with the logic of a user's exit routine.

04
A sensitive logical child segment type has a logical parent's concatenated key defined as VIRTUAL, but the BLDLPCK control statement is not specified.

08
A database error was detected, and the SKERROR option might have skipped the unloading of one or more database segments.

FABHURG1 return codes when IMS HD Reorganization Unload JCL is used

If FABHURG1 is run by using IMS HD Reorganization Unload JCL (DFSURGU0 JCL), the FABHURG1 utility issues one of the following return codes:

Code
Meaning

00
Successful completion.

04
Any of the following conditions were detected:

0001
No segment was retrieved from the database.

0002
Some segments have been skipped in accordance with the logic of a user's exit routine.

0004
A sensitive logical child segment type has a logical parent's concatenated key defined as VIRTUAL, but the BLDLPCK control statement is not specified.

0008
A database error was detected, and the SKERROR option might have skipped the unloading of one or more database segments.

HSSR Engine ends abnormally with the user completion code of 4013 for errors to which IMS HD Reorganization Unload utility would return the return code of 8.

Related concepts

IMS HD Reorganization Unload JCL for running FABHURG1
You can use a JCL that is written for IMS HD Reorganization Unload (DFSURGU0) to run FABHURG1, by adding the IMS High Performance Unload's SHPSLMD0 library ahead of the IMS RESLIB in the STEPLIB concatenation, and by using the DFSISVI0 exit of IMS.

FABHFSU return codes

This reference topic explains the return codes of FABHFSU.

FABHFSU issues a return code that is the logical sum of the following return codes:

Code
Meaning

00
Successful completion.

01
No segment was retrieved from the database. When you are running in the PSF mode, no segment was retrieved in the current PSF phase.

04
A sequence error was detected in the standard mode.

In the PSF mode, this return code means that all the PSF phases were completed, but one or more sequence errors were detected.

08
In the PSF mode, the current PSF phase was completed normally, but other phases are incomplete.

FABHPSFS return codes

This reference topic explains the return codes of FABHPSFS.

FABHPSFS issues a return code that is the logical sum of the following return codes:

Code
Meaning

00
Successful completion. No errors were detected except the STATUS option.

01
No segment was retrieved in all PSF phases.

04
Successful completion of a STATUS option run.

08
Any of the following conditions were detected:

a
One or more scan phases were not completed.

b
Preceding FABHPSFS has been completed successfully, but the RERUN option was not specified.

c
The FORCE option was specified, but no complete phases were found.

d
Inconsistent DEC options were specified in the scan phases.

FABHBSIM and FABHTEST return codes

This reference topic explains the return codes of FABHBSIM and FABHTEST utilities.

FABHBSIM and FABHTEST issue the following code:

Code
Meaning

00
Successful completion.

Messages

Use the information in these messages to help you diagnose and solve IMS High Performance Unload problems.

Message format

IMS High Performance Unload messages adhere to the following format:

```
FABHnnnnx
FABInnnnx
```

Where:

FABH
Indicates that the message was issued by IMS High Performance Unload.

FABI
Indicates that the message was issued by the Sequential Subset Randomizer utility of IMS High Performance Unload.

nnnn
Indicates the message identification number.

x
Indicates the severity of the message:

E
Indicates that an error occurred, which might or might not require operator intervention.

I
Indicates that the message is informational only.

W
Indicates that the message is a warning to alert you to a possible error condition.

Each message also includes the following information:

Explanation:
The Explanation section explains what the message text means, why it occurred, and what its variables represent.

System action:
The System action section explains what the system will do in response to the event that triggered this message.

User response:
The User response section describes whether a response is necessary, what the appropriate response is, and how the response will affect the system or program.

Message variables

In the message text, you will see lowercase variable names (such as *xxx...*). The variable names take on values when the message appears and represent such things as:

- The name of a data set
- A DD name
- A DBD name
- A status code
- A command keyword
- A name or value provided by the user
- Text generated by HSSR Engine when an I/O error has occurred

The keyword *HSSR* in message texts and in the explanation, system action, and user response represents the HSSR Engine unless otherwise stated.

FABH messages

Use the information in these messages to help you diagnose and solve IMS High Performance Unload problems.

FABH0001E INVALID CARD TYPE

Explanation

A CAB control statement with an incorrect statement type was detected in the HSSRCABP data set.

System action

HSSR Engine ends abnormally.

User response

Ensure that the CAB control statements begin in column 1 with a keyword defining the statement type.

FABH0002E INVALID NUMERIC FIELD

Explanation

A CAB control statement contains a non-numeric byte in a field that should be numeric.

System action

HSSR Engine ends abnormally.

User response

Correct the control statement.

FABH0003E CARD TYPE NOT FOLLOWED BY VALUE

Explanation

The statement type on a CAB control statement is not immediately followed by a single blank and the required parameter value.

System action

HSSR Engine ends abnormally.

User response

Correct the incorrect control statement.

FABH0004E PARAMETER IS TOO LONG

Explanation

A CAB control statement contains a parameter field that is too long.

System action

HSSR Engine ends abnormally.

User response

Correct the control statement.

FABH0005E FIRST CARD SHOULD BE A "CABDD" CARD

Explanation

The first statement in the HSSRCABP data set is not a CABDD control statement.

System action

HSSR Engine ends abnormally.

User response

Set the statements of the HSSRCABP data set into the correct sequence. (Each group of CAB control statements must begin with a CABDD control statement.)

FABH0011E RANSIZE IS TOO LOW; MINIMUM IS 2 CABDD=*xxxxxxxx*

Explanation

The value of a RANSIZE control statement is lower than the acceptable minimum. *xxxxxxxx* represents the keyword that was specified on the CABDD control statement (that is, *ALL, *HD, *HS, or *ddname*).

System action

HSSR Engine ends abnormally.

User response

Correct the control statement.

FABH0012E RANSIZE IS TOO HIGH; MAXIMUM IS 255 CABDD=*xxxxxxxx*

Explanation

The value of a RANSIZE control statement is higher than the acceptable maximum. *xxxxxxxx* represents the keyword that was specified on the CABDD control statement (that is, *ALL, *HD, *HS, or *ddname*).

System action

HSSR Engine ends abnormally.

User response

Correct the control statement.

FABH0013E NBRSRAN IS TOO LOW; MINIMUM IS 3 CABDD=*xxxxxxxx*

Explanation

The value of an NBRSRAN control statement is lower than the acceptable minimum. *xxxxxxxx* represents the keyword that was specified on the CABDD control statement (that is, *ALL, *HD, *HS, or *ddname*).

System action

HSSR Engine ends abnormally.

User response

Correct the control statement.

FABH0015E NBRDRAN IS TOO LOW; MINIMUM IS 2 CABDD=*xxxxxxxx*

Explanation

The value of an NBRDRAN control statement is lower than the acceptable minimum. *xxxxxxxx* represents the keyword that was specified on the CABDD control statement (that is, *ALL, *HD, *HS, or *ddname*).

System action

HSSR Engine ends abnormally.

User response

Correct the control statement.

FABH0016E NBRDBUF IS TOO LOW; MINIMUM IS 2 CABDD=*xxxxxxxx*

Explanation

The value of an NBRDBUF control statement is lower than the acceptable minimum. *xxxxxxxx* represents the keyword that was specified on the CABDD control statement (that is, *ALL, *HD, *HS, or *ddname*).

System action

HSSR Engine ends abnormally.

User response

Correct the control statement.

FABH0017W DDNAME NOT FOUND CABDD=*ddname*

Explanation

The *ddname* specified on a CABDD statement was not found within any DBD referred to by HSSR PCBs.

System action

HSSR Engine ignores the CAB control statements that begin with this CABDD statement.

User response

Correct the CABDD statement.

FABH0018E SPECIFY EITHER "YES" OR "NO"

Explanation

HSSR Engine detected an incorrect CAB control statement. The CAB control statement does not allow a specification other than YES or NO.

System action

HSSR Engine ends abnormally.

User response

Correct the CAB control statement.

FABH0019E INVALID OVERFLOW SPECIFICATION

Explanation

HSSR Engine detected an incorrect OVERFLOW CAB control statement.

System action

HSSR Engine ends abnormally.

User response

Correct the CAB control statement.

FABH0020E OPEN OF HSSRCABP HAS FAILED

Explanation

HSSR Engine could not open the HSSRCABP data set.

System action

HSSR Engine ends abnormally.

User response

Check whether MVS issued an additional message describing the type of error with more details. Correct the error.

FABH0021E NOGO SET BECAUSE OF ERRORS ON CAB CONTROL STATEMENTS

Explanation

CAB control statements in the HSSRCABP data set contain errors. The errors are described in detail in other error messages issued by HSSR Engine.

System action

HSSR Engine ends abnormally.

User response

Correct the errors.

FABH0022E NBRSRAN IS TOO HIGH; MAXIMUM IS 9999 CABDD=*xxxxxxxx*

Explanation

The value of an NBRSRAN control statement exceeds the acceptable maximum. *xxxxxxxx* represents the keyword that was specified on the CABDD control statement (that is, *ALL, *HD, *HS, or *ddname*).

System action

HSSR Engine ends abnormally.

User response

Correct the control statement.

FABH0023E NBRDBUF IS TOO HIGH; MAXIMUM IS 255 CABDD=*xxxxxxxx*

Explanation

The value of an NBRDBUF control statement exceeds the acceptable maximum. *xxxxxxxx* represents the keyword that was specified on the CABDD control statement (that is, *ALL, *HD, *HS, or *ddname*).

System action

HSSR Engine ends abnormally.

User response

Correct the control statement.

FABH0024E REQUESTED BUFFER TOO LARGE; DDNAME=*ddname*

Explanation

The requested size of the buffer for BB or CAB sequential I/O exceeds the acceptable maximum of X'7FFFFFFF' (2,147,483,647) bytes. *ddname* represents the DD name of the data set for which the storage for buffer was requested at the time of error.

System action

HSSR Engine ends abnormally.

User response

For CAB, check the RANSIZE and NBRSRAN control statements in the HSSRCABP data set; for BB, check the BUF control statement in the HSSROPT data set. Correct these control statements so that the total amount of buffers requested does not exceed the maximum value of X'7FFFFFFF'. Note that HSSR buffer handler requests the system to acquire the following size of bytes as the read buffer for *ddname*:

Buffering type	Storage size to be requested
BB	(Block size of *ddname*) / (CI size) x (RANSIZE x (NBRSRAN + 1))
CAB	(Block size of *ddname*) / (CI size) x BUF

FABH0025E THE SECOND OPERAND OF THE PARTPROC STATEMENT MUST BE EITHER 'S' OR 'R' (DBD: *dbdname*)

Explanation

The second operand of a PARTPROC statement must be either 'S' or 'R'. The string *dbdname* shows the first operand of the PARTPROC statement in error.

System action

HSSR Engine ends abnormally.

User response

Correct the control statement.

FABH0026E THE LENGTH OF THE THIRD OPERAND OF THE PARTPROC STATEMENT MUST BE LESS THAN FIVE (DBD: *dbdname*)

Explanation

The length of the third operand of the PARTPROC control statement in HSSRCABP DD is five bytes or more. The length must be less than five bytes.

System action

HSSR Engine ends abnormally.

User response

Correct the error and rerun the job.

FABH0027W THE DBD SPECIFIED ON THE PARTPROC STATEMENT IS NOT FOUND OR IS NOT FOR HALDB (DBD: *dbdname*)

Explanation

The DBD *dbdname* specified in a PARTPROC control statement is neither a DBD that is referred to from an HSSR PCB, nor a DBD for a HALDB.

System action

The PARTPROC statement is ignored, and the processing continues.

User response

Ensure that the correct DBD name is specified. If necessary, specify the correct DBD name or remove the statement.

FABH0028E INCONSISTENT BUFFER HANDLERS ARE SPECIFIED FOR DSGROUP *n* OF HALDB *dbdname*

Explanation

Different buffer handlers are specified for partition data sets that belong to the same data set group *n*.

System action

HSSR Engine ends abnormally.

User response

Correct the CABDD control statements so that either CAB or BB can be used for all data sets that belong to the same data set group.

FABH0029E SYNTAX ERROR IN A PARTPROC STATEMENT: *reason*

Explanation

A syntax error is found in a PARTPROC statement that is specified in the HSSRCABP data set. *reason* shows the reason for the error.

System action

HSSR Engine ends abnormally.

User response

Correct the control statement.

FABH0030E THE THIRD OPERAND OF A PARTPROC STATEMENT IS NOT A NUMERIC (DBD: *dbdname*)

Explanation

The value specified as the third operand for a PARTPROC control statement is not a numeric. *dbdname* shows the first operand of the PARTPROC statement in error.

System action

HSSR Engine ends abnormally.

User response

Correct the control statement.

FABH0031E DDNAME=*ddname*; DATA REQUEST OUTSIDE OF DATA SET LIMITS

Explanation

The CAB buffer handler has received a data request for an OSAM block number or ESDS CI number that is not within the extents of the data set (indicated by *ddname*).

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378).
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.
- If an OSAM block above the 4 GB boundary is requested by an odd RBA in the HALDB, ensure that the HALDB is defined as OSAM8G in the RECON data sets.

FABH0032E BUFFER HANDLER LOGIC ERROR

Explanation

The CAB buffer handler detected an unexpected error.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378). If necessary, contact IBM Software Support.

FABH0033E BUFFER HANDLER LOGIC ERROR

Explanation

The CAB buffer handler detected an unexpected error.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378). If necessary, contact IBM Software Support.

FABH0034E REQUESTED STORAGE FOR DBD=*dbdname* NOT AVAILABLE WITHIN THE REGION

Explanation

The total size of the requested buffers exceeds the acceptable maximum that can be acquired in the region. *dbdname* indicates the name of the DBD of the database for which the buffer was requested at the time of error.

System action

HSSR Engine ends abnormally.

User response

Check the BUF control statement in the HSSROPT data set and the RANSIZE, NBRSRAN, and NBRDBUF control statements in the HSSRCABP data set that are specified for the DBD *dbdname*. Correct these control statements so that the total storage size for the buffers requested for *dbdname* does not exceed the allowable maximum that can be acquired in the region. Otherwise, change the REGION parameter in the JCL for the job so that the requested storage for the buffer can be acquired. The total size of the buffers for *dbdname* is calculated by summing up the required size for each DD that refers to *dbdname*, using the following formula:

Buffering type used for the DBD	Storage size to be requested
BB	(Block or CI size of the DD) x BUF
CAB	(Block or CI size of the DD) x (RANSIZE x (NBRSRAN + 1) + NBRDBUF)

FABH0035E UNSUPPORTED LEVEL OF IMS IS BEING USED: *xxx* IMS LEVEL OF THIS RUN

Explanation

You are running an IMS batch region controller (DFSRRC00) that is not supported by IMS High Performance Unload. *xxx* represents the IMS level.

System action

HSSR Engine ends abnormally.

User response

Run the job with the correct version of IMS.

FABH0036E THIRD OPERAND FOR PARTPROC STATEMENT IS NOT ALLOWED IF THE SECOND OPERAND IS 'S' (DBD: *dbdname*)

Explanation

The third operand of a PARTPROC control statement for the database *dbdname* is specified although the second operand is 'S'.

System action

HSSR Engine ends abnormally.

User response

Remove the third operand.

FABH0037E INCONSISTENT OVERFLOW BUFFERING OPTIONS FOR PHDAM *dbdname*

Explanation

Inconsistent overflow buffering options are specified for the overflow area of each partition of the PHDAM database *dbdname*.

System action

HSSR Engine ends abnormally.

User response

Correct the OVERFLOW control statements so that the same overflow buffering options (CAB, SHR, or BB) can be used for all overflow areas.

FABH0038E INCONSISTENT PAGEFIX OPTIONS FOR DATA SETS OF DSGROUP *n* OF HALDB *dbdname*

Explanation

Different pagefix options are specified for partition data sets that belong to the same data set group *n*.

System action

HSSR Engine ends abnormally.

User response

Specify the same keyword, either YES or NO, for the pagefix option of BUFFERS control statement so that the same pagefix options can be specified for all partitions.

FABH0039I UNSUPPORTED LEVEL OF IMS IS BEING USED: *xxx*

Explanation

You are running an IMS batch region controller (DFSRRC00) that is not supported by IMS High Performance Unload. *xxx* represents the IMS level.

System action

IMS continues the processing and DFSURGU0 will be invoked.

User response

None. This message is informational.

FABH0041E VSAM POINT ERROR DURING SEQUENTIAL ESDS I/O

Explanation

VSAM returned an unexpected code to the CAB buffer handler during the execution of a POINT macro.

System action

HSSR Engine ends abnormally.

User response

Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred. If the cause is not clear, collect the dump and contact IBM Software Support.

FABH0042E VSAM GET ERROR DURING SEQUENTIAL ESDS I/O

Explanation

VSAM returned an unexpected code to the CAB buffer handler during the execution of a GET macro.

System action

HSSR Engine ends abnormally.

User response

Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred. If the cause is not clear, collect the dump and contact IBM Software Support.

FABH0044W WARNING: NUMBER OF VSAM BUFFERS NOT O.K.

Explanation

Normally, the CAB buffer handler specifies how many buffers should be used by VSAM. However, CAB detected that its specifications could not be honored by VSAM.

System action

HSSR Engine continues processing. However, the performance will suffer.

User response

Possible reasons are specifications of VSAM buffering options on JCL (BUFND, BUFSP, STRNO) or through IDCAMS (buffer space).

Remove any BUFND, BUFSP, or STRNO specifications from the JCL. Remove through IDCAMS BUFFERSPACE specifications for the ESDS from the catalog.

FABH0045E VSAM LOGICAL ERROR AFTER POINT MACRO

Explanation

VSAM returned an unexpected logical error code to the CAB buffer handler during the execution of a POINT macro.

System action

HSSR Engine ends abnormally.

User response

Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred. If the cause is not clear, collect the dump and contact IBM Software Support.

FABH0046E VSAM PHYSICAL ERROR

Explanation

VSAM returned an unexpected physical error code to the CAB buffer handler.

System action

HSSR Engine ends abnormally.

User response

Locate I/O error messages that were issued by VSAM and resolve the physical I/O error.

FABH0047E SHOULD NOT OCCUR ERROR AFTER VSAM CHECK MACRO

Explanation

VSAM returned an unexpected error code to the CAB buffer handler during the execution of a CHECK macro.

System action

HSSR Engine ends abnormally.

User response

Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred. If the cause is not clear, collect the dump and contact IBM Software Support.

FABH0048E SHOULD NOT OCCUR ERROR AFTER VSAM CHECK MACRO

Explanation

VSAM returned an unexpected error code to the CAB buffer handler during the execution of a CHECK macro.

System action

HSSR Engine ends abnormally.

User response

Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred. If the cause is not clear, collect the dump and contact IBM Software Support.

FABH0049E VSAM MACRO GENCB BLOCK=RPL FAILED

Explanation

VSAM returned an unexpected code to the CAB buffer handler during the execution of a GENCB BLOCK=RPL macro.

System action

HSSR Engine ends abnormally.

User response

Ensure that enough virtual storage is available within the region (MVS).

As a temporary bypass, the Basic Buffer handler can be used instead of CAB.

FABH0050E VSAM MACRO MODCB RPL=... FAILED

Explanation

VSAM returned an unexpected code to the CAB buffer handler during the execution of a MODCB RPL macro.

System action

HSSR Engine ends abnormally.

User response

Ensure that enough virtual storage is available within the region (MVS).

As a temporary bypass, the Basic Buffer handler can be used instead of CAB.

FABH0053E RANSIZE SHOULD BE THE SAME FOR ALL PCB'S REFERENCING THE SAME ESDS

Explanation

Multiple HSSR PCBs that refer to the same database were specified to be buffered by CAB. This situation is not supported by CAB.

System action

HSSR Engine ends abnormally.

User response

Use the OCCURRENCE control statement to make sure that no more than one HSSR PCB per database is buffered by CAB.

FABH0054E NUMBER OF VSAM BUFFERS NOT OK., PLHBFRNO=*xxx*

Explanation

The indicated number of CIs to be read in one sequential read could not be followed by VSAM. The number was specified by the RANSIZE control statement.

System action

HSSR Engine ends abnormally.

User response

Remove any BUFND, BUFSP, or STRNO specifications from the DD statement for the database. If BUFFERSPACE is specified in the definition of the data set, remove it for the ESDS. Disable the use of VSAM optimization tool if any is used for the job.

If the VSAM ESDS attributes cannot be changed, specify the value *xxx*, shown in the message text, for the RANSIZE statement. However, you might not obtain the maximum buffering performance if the recommended value from PLHBFRNO is much greater or much lower than the user's calculated optimum value, if HSSR buffer handler assumes that BUFND, BUFSP, and STRNO are not specified in the JCL, or if BUFFERSPACE is not used when the data set is defined.

FABH0055W **SPECIFIED NUMBER OF BUFFERS IN A SEQ_BUF FOR DDNAME=*ddname* NOT OPTIMAL FOR ESDS**

Explanation

The number of CIs to be read by one sequential read for the data set that is identified by ddname is too large. This message is accompanied by a FABH0056W message.

System action

See the system action section of FABH0056W message.

User response

See the user response section of FABH0056W message.

FABH0056W **NUMBER OF BUFFERS IN A SEQ_BUF *xxx* CHANGE TO *yyy***

Explanation

The number (*xxx*) of CIs to be read by one sequential read for the data set identified by *ddname* is too large. The number *xxx* is the one that was specified by the RANSIZE control statement or the CABBASE control statement. The ESDS cannot chain CCWs for that number of CIs in a single start I/O. The maximum number of CIs (*yyy*) that can be read depends on the CI size and is displayed in the message.

System action

The processing continues with the maximum number (*yyy*) of CIs.

User response

Specify the value *yyy* indicated in the message for RANSIZE.

FABH0057W **NUMBER OF VSAM BUFFERS IS OVERRIDDEN: PLHBFRNO=*xxx***

Explanation

Normally, the CAB buffer handler specifies the number of VSAM buffers, however, CAB detected that its specifications could not be used by VSAM.

System action

HSSR Engine continues processing. However, the performance will decrease.

User response

If you want to use the CAB buffering optimization, remove any BUFND, BUFSP, or STRNO specifications from the JCL, and remove the IDCAMS BUFFERSPACE specifications for the ESDS from the catalog. If the system-managed buffering is activated in the SMS definition, deactivate it.

FABH0060E **DEVTYPE MACRO FAILED ON DDNAME: *ddname* RC=*rc***

Explanation

An error was returned by the DEVTYPE macro to get information about the device associated with the DD name *ddname*. The value *rc* shows the return code.

System action

HSSR Engine issues a user abend.

User response

Ensure that the DD statement for the specified DD name points to the correct data set. Correct the error, and rerun the job.

FABH0061E **OPEN FAILED FOR DDNAME: *ddname***

Explanation

An attempt to open the data set identified by *ddname* failed.

System action

HSSR Engine issues a user abend.

User response

Ensure that the DD statement associated with *ddname* is the correct data set. Correct the error, and rerun the job.

FABH0062E **BLDL MACRO FAILED FOR MEMBER *mbrname* IN LIBRARY *ddname* (RC=*rc*,RSN=*rsn*)**

Explanation

The BLDL macro failed for member *mbrname* in the library that has the DD name *ddname*. The return code from the macro call was *rc*, and the reason code was *rsn*.

System action

HSSR Engine issues a user abend.

User response

This error is likely an internal system error. Contact IBM Software Support.

FABH0063E BSAM CLOSE HAS FAILED

Explanation

The CLOSE macro issued by the CAB buffer handler failed.

System action

HSSR Engine ends abnormally.

User response

Ensure that enough virtual storage is available in the address space. If the cause is not clear, collect the dump and contact IBM Software Support.

FABH0064E HDCB NOT FOUND

Explanation

HSSR Engine detected an unexpected error.

System action

HSSR Engine ends abnormally.

User response

This error is likely an internal system error. Collect the dump and contact IBM Software Support.

FABH0065E DDNAME SPECIFIED IN DBD (*dbdname*) DOES NOT HAVE A CORRESPONDING MATCH WITH A DD STATEMENT

Explanation

During the allocation of a database data set, neither an MDA member corresponding to the specified DBD nor a DD statement for the DD name specified in the DBD could be found.

System action

HSSR Engine issues a user abend.

User response

Either add a DD statement for the database data set or specify (on the IMSDALIB or STEPLIB/JOBLIB) a PDS data set that contains an MDA member for the DBD.

FABH0066E INCORRECT DFSMDA MEMBER: *xxxxxxxx*

Explanation

The module *xxxxxxxx* was loaded as a DFSMDA member, but it does not have a correct DFSMDA format. The eye-catcher MDA is not found in the module.

System action

HSSR Engine issues a user abend.

User response

This problem can occur if a library that contains a member with the name *xxxxxxxx* is specified in the higher order of the STEPLIB or JOBLIB concatenation.

Ensure that the correct DFSMDA members exist in the IMSDALIB concatenation or STEPLIB/JOBLIB concatenation. Correct the error, and rerun the job.

FABH0067E DD: *ddname* INFORMATION OF DB: *dbdname* IS NOT FOUND IN DFSMDA MEMBER

Explanation

The DD information associated with the DD *ddname* is not found in the DFSMDA member. HSSR Engine cannot perform the dynamic allocation for the *ddname* of the indicated database (*dbdname*).

System action

HSSR Engine issues a user abend.

User response

Ensure that the DFSMDA member *dbdname* contains the appropriate information about the database *dbdname*. Correct the error, and rerun the job.

FABH0068E DYNAMIC ALLOCATION FAILURE OCCURRED FOR DB: *dbdname* DD: *ddname* RC=*xx* RSN=*yyyy*

Explanation

An attempt at dynamic allocation for *ddname* of *dbdname* failed. *xx* is the return code in register 15

and *yyyy* is the associated hexadecimal reason code from the SVC99 routine.

System action

HSSR Engine issues a user abend.

User response

See the *MVS Programming: Authorized Assembler Services Guide* for the return code and reason code from the SVC99 routine. Correct the error, and rerun the job.

FABH0070E UNEXPECTED DEB LAYOUT

Explanation

The MVS DEB control block used by OSAM has an unexpected layout.

System action

HSSR Engine ends abnormally.

User response

This error is likely an internal system error. Collect the dump and contact IBM Software Support.

FABH0071E UNEXPECTED DEB LAYOUT

Explanation

The MVS DEB control block used by OSAM has an unexpected layout.

System action

HSSR Engine ends abnormally.

User response

This error is likely an internal system error. Collect the dump and contact IBM Software Support.

FABH0072E UNEXPECTED DEB LAYOUT

Explanation

The MVS DEB control block used by OSAM has an unexpected layout.

System action

HSSR Engine ends abnormally.

User response

This error is likely an internal system error. Collect the dump and contact IBM Software Support.

FABH0073E UNSUCCESSFUL BSAM OPEN BY CAB BUFFER HANDLER

Explanation

The CAB buffer handler could not OPEN an OSAM data set with BSAM.

System action

HSSR Engine ends abnormally.

User response

Check for an error message issued by MVS or by access method service.

If you have specified DBALLABOVE in DFSVSAMP and if message IEC133I is written in the job log, remove the specification of DBALLABOVE from DFSVSAMP and rerun the job.

If the cause is not clear, collect the dump and contact IBM Software Support.

FABH0074E INVALID ENTRY INTO BSAM EOV EXIT ROUTINE

Explanation

The CAB buffer handler detected an unexpected entry in its own EOV exit routine, used in processing an OSAM data set with BSAM. The most likely cause is that a multivolume OSAM data set has been reused without scratching and reallocating the space before reloading to the database. In that case, an invalid end-of-file mark can be left and an embedded EOF mark might be caused somewhere in the middle of the data set.

System action

HSSR Engine ends abnormally.

User response

Check whether the multivolume OSAM data set was scratched and reallocated correctly as described in *IMS Database Administration*. If the cause is not clear, collect the dump and contact IBM Software Support.

FABH0075E INVALID ENTRY INTO BSAM EODAD ROUTINE

Explanation

The CAB buffer handler detected an unexpected entry into its own EODAD exit routine, used in processing an OSAM data set with BSAM. The most likely cause is that a multivolume OSAM data set has been reused without scratching and reallocating the space before reloading to the database. In that case, an invalid end-of-file mark can be left and an embedded EOF mark might be caused somewhere in the middle of the data set.

System action

HSSR Engine ends abnormally.

User response

Check whether the multivolume OSAM data set was scratched and reallocated correctly as described in *IMS Database Administration*. If the cause is not clear, collect the dump and contact IBM Software Support.

FABH0076E ***text***

Explanation

HSSR buffer handler encountered an I/O problem when using BSAM to read an OSAM data set. This message *text* contains bytes 51-127 of the SYNADAF message.

System action

HSSR Engine ends abnormally.

User response

See *DFSMS/MVS Macro Instructions for Data Sets*, and identify the cause of the error. Complete one or more of the following tasks depending on the cause of the error:

- Activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378).
- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

FABH0081I I/O WILL BE PERFORMED

Explanation

A FABHBSIM control statement requested that the buffer handler should not bypass database I/O operations.

System action

The buffer handlers perform database I/O operations.

User response

None. This message is informational.

FABH0082I BSIM NORMAL END OF PROCESSING

Explanation

Program FABHBSIM reached normal end of processing.

System action

The processing continues.

User response

None. This message is informational.

FABH0083E HSSR ENVIRONMENT IS NOT ESTABLISHED

Explanation

The HSSR environment has not been properly established. The reason is probably that program FABHBSIM was not invoked by FABHDLI, FABHDBB, or FABHULU procedures (or by equivalent JCL).

System action

FABHBSIM ends abnormally.

User response

Use the FABHDLI, FABHDBB, or FABHULU procedures, or correct the JCL.

FABH0084E SYSUT1 COULD NOT BE OPENED

Explanation

The SYSUT1 data set that contains the buffer handler trace could not be opened.

System action

Program FABHBSIM ends abnormally.

User response

Correct the JCL.

FABH0085E INVALID RECORD TYPE IN SYSUT1

Explanation

Program FABHBSIM found an incorrect record type in the buffer handler trace data set.

System action

FABHBSIM ends abnormally.

User response

Ensure that the correct data set is defined on the SYSUT1 DD statement.

FABH0086E PCB-NBR STORED IN SYSUT1 RECORD DOES NOT EXIST IN PSB

Explanation

Each record of the SYSUT1 data set contains the number of the PCB that was being used during the traced buffer handler call. The recorded PCB-NBR does not exist in the PSB used for the FABHBSIM run.

System action

Program FABHBSIM ends abnormally.

User response

Ensure that the same PSB was used for the traced run and for the FABHBSIM run. Ensure that the PSB has not changed.

FABH0087E PCB-NBR STORED IN SYSUT1 RECORD IS NOT THE NUMBER OF A HSSR PCB

Explanation

Each record of the SYSUT1 data set contains the number of the PCB that was being used during the traced buffer handler call. The recorded PCB-NBR is not the number of an HSSR PCB.

System action

Program FABHBSIM ends abnormally.

User response

Ensure that the same PSB was used for the traced run and for the FABHBSIM run. Ensure that the PSB has not changed.

FABH0088E BUFFER HANDLER CALL TYPE STORED IN SYSUT1 RECORD IS INVALID

Explanation

Program FABHBSIM found an incorrect buffer handler call type in the buffer handler trace data set.

System action

FABHBSIM ends abnormally.

User response

Ensure that the correct data set is defined on the SYSUT1 DD statement.

FABH0089E NOT NUMERIC FIELD ON SYSIN CARD

Explanation

A FABHBSIM control statement on the SYSIN data set does not contain a numeric value.

System action

Program FABHBSIM ends abnormally.

User response

Correct the control statement.

FABH0090E INVALID HSSRPCB OR HSSRDBD STATEMENT IS SPECIFIED : *reason*

Explanation

The specified HSSRPCB or HSSRDBD control statement is not correct. The text *reason* indicates the cause of the error:

reason
Description

NOT NUMERIC
A non-numeric character was specified for the operand of an HSSRPCB statement.

TOO MANY PARAMETERS
More than 500 *pcbnum* or *dbdname* were specified.

TOO LARGE PCBNUM
pcbnum larger than 500 was specified.

MIXED SPECIFICATION
Both HSSRPCB and HSSRDBD statements were specified.

System action

HSSR Engine ends abnormally.

User response

Correct the error in the HSSRPCB or HSSRDBD control statement and rerun the job.

FABH0091W UNABLE TO PROCESS THE FOLLOWING PCB AS AN HSSR PCB (REASON: *rrrrrrrr*)

Explanation

This message is issued with another FABH0091W message. See the explanation in "FABH0091W" on page 395.

System action

See the system action section of the other FABH0091W message.

User response

See the user response section of the other FABH0091W message.

FABH0091W DBD=*xxxxxxxx*, PCB#=*yyyy*, PCBNAME=*zzzzzzzz*

Explanation

The PCB that is specified as an HSSR PCB cannot be processed as an HSSR PCB. The string *xxxxxxxx* shows the name of the DBD that the PCB refers to; the number *yyyy* shows the PCB number in the PSB; and the string *zzzzzzzz* shows the label, if there is one, of the PCB assigned at PSBGEN. The string *rrrrrrrr* shows the reason for the error as follows:

Reason (*rrrrrrrr*)
Description

IOPCB
The specified PCB is an I/O PCB.

TPPCB
The specified PCB is an alternative PCB.

GSAMPCB
The specified PCB is a GSAM PCB.

PROCOPT
The PROCOPT parameter on the PCB statement specifies an incorrect code or combination of codes. The codes you can use are G, O, N, T, R, A, P, and E.

DBDL1
The specified PCB refers to the DBD that is specified on a DBDL1 control statement.

USREXIT
The specified PSB is generated with PROCOPT=R, and the DBD referred by the PCB is generated with the data capture exit routine.

NOTFOUND
The specified PCB does not exist.

OTHER
The specified PCB cannot be processed, for some reason other than the preceding. See the accompanying FABH*xxxxx* messages issued by HSSR Engine.

System action

The processing continues. If the PCB can be processed as a DL/I PCB, it is treated as a DL/I PCB, and all HSSR calls to the PCB are transferred to the DL/I call interface modules.

User response

If the database calls need to be processed by HSSR Engine, ensure that the specified PCB is correct (check the HSSRPCB and HSSRDBD control statements or the KEYLEN value for the PCB). Otherwise, ignore this message and continue processing by using the DL/I module.

FABH0092W PROCOPT=R IS SPECIFIED ON THE FOLLOWING HSSR PCB:

Explanation

This message is issued with another FABH0092W message. See the explanation in "FABH0092W" on page 395.

System action

See the system action section of the other FABH0092W message.

User response

See the user response section of the other FABH0092W message.

FABH0092W DBD=*xxxxxxxx*, PCB#=*yyyy*, PCBNAME=*zzzzzzzz*

Explanation

PROCOPT=R is specified on the PCB specified by the PCB control statement or the PCBNAME control statement in the HPSIN data set of IPR Unload utility. The string *xxxxxxxx* shows the name of the DBD that the PCB refers to; the number *yyyy* shows the PCB

number in the PSB; and the string *zzzzzzzz* shows the label, if there is one, of the PCB assigned at PSBGEN.

System action

HSSR Engine continues processing. If an HSSR REPL call is issued for the PCB, HSSR Engine ends abnormally.

User response

You can ignore the message if you do not issue any HSSR REPL call, but it is recommended that you change the PROCOPT.

FABH0101E **//CARDIN FILE CONTAINS AN INVALID FSU CONTROL STATEMENT**

Explanation

Program FABHFSU detected a control statement with an incorrect control statement ID.

System action

FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0102E **OPEN-ERROR FOR //CARDIN OR // PRNTOUT**

Explanation

Program FABHFSU cannot open the CARDIN or PRNTOUT data set.

System action

FABHFSU ends abnormally.

User response

Check for additional error messages issued by the access method. Correct the error.

FABH0103E **'DBD' OR 'PSC' CONTROL CARD MISSING IN //CARDIN**

Explanation

The CARDIN data set must contain either a DBD or a PSC control statement.

System action

Program FABHFSU ends abnormally.

User response

Provide the required control statement.

FABH0104E **INVALID NBR OF 'PSB' CONTROL-STATEMENTS IN //CARDIN**

Explanation

The CARDIN data set must contain one, two, or three PSB control statements.

System action

Program FABHFSU ends abnormally.

User response

Provide a correct number of PSB control statements.

FABH0105E **NOGO-SWITCH SET BECAUSE OF PREVIOUS ERROR IN //CARDIN CONTROL STATEMENTS**

Explanation

Program FABHFSU detected an error, which is described in a previous error message.

System action

FABHFSU ends abnormally.

User response

Correct the error.

FABH0106E **INVALID 'SEQUENCE CHECK OPTION' ON 'DBD' CARD IN // CARDIN**

Explanation

A DBD control statement contains an error. The error is described in the error message.

System action

Program FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0107E **INVALID 'SEQUENCE ERROR OPTION' ON 'DBD' CARD IN // CARDIN**

Explanation

A DBD control statement contains an error. The error is described in the error message.

System action

Program FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0108E **INVALID 'SEQUENCE ERROR PRINT OPTION' ON 'DBD' CARD IN //CARDIN**

Explanation

A DBD control statement contains an error. The error is described in the error message.

System action

Program FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0109E **'SEQUENCE ERROR THRESHOLD' FIELD ON 'DBD' CARD IS NOT NUMERIC**

Explanation

A DBD control statement contains an error. The error is described in the error message.

System action

Program FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0110E **MULTIPLE 'DBD' CONTROL STATEMENTS NOT ALLOWED IN // CARDIN**

Explanation

More than one DBD control statement was detected. Only one DBD control statement is allowed in the CARDIN data set.

System action

Program FABHFSU ends abnormally.

User response

Ensure that the CARDIN data set contains a single DBD control statement.

FABH0111E **INVALID 'POINTER BYPASS OPTION' ON 'DBD' CARD**

Explanation

A DBD control statement contains an error. The error is described in the error message.

System action

Program FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0112E **FABHFSU CANNOT RUN WITH ACB'S**

Explanation

Program FABHFSU can run either in a DL/I or in a ULU region, but cannot in a DBB region.

System action

FABHFSU ends abnormally.

User response

Modify the JCL in order to run FABHFSU in a DL/I region with PSBs and DBDs (instead of running FABHFSU in a DBB region with ACBs).

FABH0113E **MULTIPLE 'BLM' CONTROL CARDS ARE NOT SUPPORTED**

Explanation

More than one BLM control statement was specified. Program FABHFSU supports only one BLM control statement.

System action

FABHFSU ends abnormally.

User response

Ensure that the CARDIN data set contains only one BLM control statement.

FABH0114E MULTIPLE 'ELM' CONTROL CARDS ARE NOT SUPPORTED

Explanation

More than one ELM control statement was specified. Program FABHFSU supports only one ELM control statement.

System action

FABHFSU ends abnormally.

User response

Ensure that the CARDIN data set contains only one ELM control statement.

FABH0115E INVALID 'LIMIT VALUE TYPE' ON 'BLM' OR 'ELM' CONTROL CARD

Explanation

Either a BLM or an ELM control statement contains an error. The error is described in the error message.

System action

Program FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0116E 'RBN LIMIT VALUE' NOT NUMERIC OR ZERO ON 'BLM' OR 'ELM' CONTROL CARD

Explanation

Either a BLM or an ELM control statement contains an error. The error is described in the error message.

System action

Program FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0117E 'LL' FIELD NOT NUMERIC OR ZERO ON 'BLM' OR 'ELM' CONTROL CARD

Explanation

Either a BLM or an ELM control statement contains an error. The error is described in the error message.

System action

Program FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0118E 'LL' FIELD GREATER THAN 74 ON 'BLM' OR 'ELM' CONTROL CARD

Explanation

Either a BLM or an ELM control statement contains an error. The error is described in the error message.

System action

Program FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0119E HEXADECIMAL STRING FOR LIMIT VALUE DOES NOT CONTAIN EVEN NUMBER OF HEXA-CHARACTERS

Explanation

Either a BLM or an ELM control statement contains an error. The error is described in the error message.

System action

Program FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0120E 'PCB NR' FIELD ON THE 'PSB' CONTROL CARD IS NOT NUMERIC

Explanation

A PSB control statement contains an error. The error is described in the error message.

System action

Program FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0121E INVALID FORMAT-NAME ON THE 'PSB' CONTROL CARD

Explanation

A PSB control statement contains an error. The error is described in the error message.

System action

Program FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0122E INVALID 'SEGMENT MODIFICATION OPTION' ON PSB CARD

Explanation

A PSB control statement contains an error. The error is described in the error message.

System action

Program FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0123E INVALID 'CONCATENATED KEY OPTION' ON PSB CARD

Explanation

A PSB control statement contains an error. The error is described in the error message.

System action

Program FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0124E INVALID 'EXIT CONTROL OPTION' ON PSB CARD

Explanation

A PSB control statement contains an error. The error is described in the error message.

System action

Program FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0125E INVALID 'DBR SKIP OPTION' ON PSB CARD

Explanation

A PSB control statement contains an error. The error is described in the error message.

System action

Program FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0126E INVALID 'DATA CONVERSION' OPTION ON PSB CARD

Explanation

A PSB control statement contains an error. The error is described in the error message.

System action

Program FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0127E 'PSB NAME' FIELD MUST BE SPECIFIED AS '*' WHEN RUNNING IN ULU REGION TYPE

Explanation

When running in ULU Region, the name of a PSB must not be specified in the 'PSB NAME' field. When running in ULU region, specify an asterisk (*) in the 'PSB NAME' field of the PSB control statement.

System action

Program FABHFSU ends abnormally.

User response

Either specify an asterisk (*) in the 'PSB NAME' field of the PSB control statement or run FABHFSU in DL/I region.

FABH0128E 'PSB NAME' FIELD MUST SPECIFY THE PSBNAME OF THE //EXEC PARM-FIELD

Explanation

Program FABHFSU requires that the name of the PSB coded in the 'PSB NAME' field of the PSB control statement is the same as the name of the PSB specified in the PARM field of the //EXEC JCL statement.

System action

FABHFSU ends abnormally.

User response

Either specify the PSB name in the 'PSB NAME' field of the PSB control statement or run FABHFSU with the 'PSB*' control statement. See the description of the PSB control statement in “PSB control statement” on page 61.

FABH0129E PSB HAS NO DB-PCB WITH THE REQUESTED DBDNAME AND/OR REQUESTED RELATIVE NUMBER

Explanation

The PSB specified on the PSB control statement in the CARDIN data set has no database PCBs.

System action

Program FABHFSU ends abnormally.

User response

Correct the PSB control statement.

FABH0130E PCB WITH SPECIFIED 'RELATIVE PCB NUMBER' DOES NOT REFER TO DBD NAMED ON DBD CARD

Explanation

The specified PCB on the PSB control statement refers to a dbdname that does not match the dbdname on the DBD control statement in the CARDIN data set.

System action

Program FABHFSU ends abnormally.

User response

Correct the PSB control statement.

FABH0131E INVALID 'DEC' OPTION

Explanation

A DEC control statement contains an incorrect option.

System action

Program FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0132E THERE IS NO JCL DD-STATEMENT FOR THE SPECIFIED 'DDNAME' ON THE 'PSB' CONTROL CARD

Explanation

Program FABHFSU detected an incorrect ddname on the PSB control statement in the CARDIN data set or on the JCL DD statement.

System action

FABHFSU ends abnormally.

User response

Specify correct ddname either on the PSB control statement or on the JCL DD statements.

FABH0133E 'CO*n*' CONTROL CARD INVALID: NO MATCHING OUTPUT FORMAT

Explanation

You have provided a CO*n* (CO1, CO2, or CO3) control statement in the CARDIN data set. Program FABHFSU tried to match these control statements with the first, second, or third PSB control statements. This matching was unsuccessful because the PSB statement was not provided in the CARDIN data set (before the CO control statement).

System action

FABHFSU ends abnormally.

User response

Correct the CARDIN control statement.

FABH0134E THERE IS NO JCL DD-STATEMENT FOR THE SPECIFIED 'DDNAME' ON THE CONTROL CARD

Explanation

Program FABHFSU detected an incorrect ddname on the CO control statement of the CARDIN data set or a missing JCL DD statement.

System action

FABHFSU ends abnormally.

User response

Correct the CO control statement or the JCL DD statement.

FABH0135E MULTIPLE 'PSC' CONTROL STATEMENTS NOT ALLOWED

Explanation

More than one PSC control statement was specified. Program FABHFSU supports only one PSC control statement.

System action

FABHFSU ends abnormally.

User response

Ensure that the CARDIN data set contains only one PSC control statement.

FABH0136E INVALID //CARDIN CONTROL STATEMENT FOR A PSC OPERATION

Explanation

A control statement that is not supported for PSF was provided in the CARDIN data set.

System action

Program FABHFSU ends abnormally.

User response

Remove the unsupported control statement.

FABH0137E 'PSC' AND 'DBD' CONTROL STATEMENTS ARE MUTUALLY EXCLUSIVE

Explanation

The PSC and DBD control statements must not be specified in the same FABHFSU run.

System action

Program FABHFSU ends abnormally.

User response

Run FABHFSU in correct mode.

FABH0138E OPEN ERROR FOR //CNTLDD

Explanation

Program FABHFSU could not open the CNTLDD data set.

System action

FABHFSU ends abnormally.

User response

See the additional error messages that were issued by the access method. Correct the error.

FABH0139E INCORRECT 'PARALLEL SCAN NAME'

Explanation

The parallel scan name that is specified on the PSC control statement is different from the parallel scan name that is specified on the CTL statement that was used to create the scan control data set.

System action

Program FABHFSU ends abnormally.

User response

Correct the PSC control statement.

FABH0140E INCORRECT 'DBD NAME'

Explanation

The DBD name that is specified on the PSC control statement is different from the DBD name that is specified on the DBD control statement that was used to create the scan control data set.

System action

Program FABHFSU ends abnormally.

User response

Correct the PSC control statement.

FABH0141E 'EXPECTED NUMBER OF SCANS' ON 'PSC' CONTROL CARD IS INVALID

Explanation

Program FABHFSU detected an incorrect number of scans specified on a PSC control statement.

System action

FABHFSU ends abnormally.

User response

Correct the PSC control statement.

FABH0142E 'PHASE NUMBER' ON 'PSC' CONTROL CARD INVALID

Explanation

Program FABHFSU detected an incorrect phase number specified on a PSC control statement

System action

FABHFSU ends abnormally.

User response

Correct the PSC control statement.

FABH0143E 'POINTER BYPASS OVERRIDE' INVALID

Explanation

Program FABHFSU detected an incorrect pointer bypass override specified on a PSC control statement.

System action

FABHFSU ends abnormally.

User response

Correct the PSC control statement.

FABH0144E 'RERUN INDICATOR' INVALID

Explanation

Program FABHFSU detected an incorrect rerun indicator specified on a PSC control statement.

System action

FABHFSU ends abnormally.

User response

Correct the PSC control statement.

FABH0145E 'EXPIRATION DATE OVERRIDE' NOT NUMERIC

Explanation

Program FABHFSU detected an incorrect expiration date override specified on a PSC control statement.

System action

FABHFSU ends abnormally.

User response

Correct the PSC control statement.

FABH0146E 'PHASE NUMBER' HIGHER THAN 'EXPECTED NUMBER OF SCANS'

Explanation

Program FABHFSU detected an incorrect phase number specified on a PSC control statement.

System action

FABHFSU ends abnormally.

User response

Correct the PSC control statement.

FABH0147E UNEXPECTED LAYOUT OF // CNTLDD CONTROL RECORD

Explanation

Program FABHFSU detected an unexpected layout of the CNTLDD data set of the Parallel Scan Facility. The reason can be:

- A FABHFSU error or a FABHPSFC error
- An incompatibility between IMS High Performance Unload and FSU II
- A user error

System action

FABHFSU ends abnormally.

User response

Ensure that the control statements are correct.

FABH0148E UNEXPECTED LAYOUT OF // CNTLDD CONTROL RECORD

Explanation

Program FABHFSU detected an unexpected layout of the CNTLDD data set of the Parallel Scan Facility. The reason can be:

- A FABHFSU error or a FABHPSFC error
- An incompatibility between IMS High Performance Unload and FSU II
- A user error

System action

FABHFSU ends abnormally.

User response

Ensure that the control statements are correct.

FABH0149E UNEXPECTED LAYOUT OF // CNTLDD CONTROL RECORD

Explanation

Program FABHFSU detected an unexpected layout of the CNTLDD data set of the Parallel Scan Facility. The reason can be:

- A FABHFSU error or a FABHPSFC error
- An incompatibility between IMS High Performance Unload and FSU II
- A user error

System action

FABHFSU ends abnormally.

User response

Ensure that the control statements are correct.

FABH0150E INCORRECT 'EXPECTED NUMBER OF SCANS'

Explanation

The expected number of scans specified on the PSC control statement is different from the number of parallel scans specified on the CTL control statement for FABHPSFC, which was used to create the scan control data set.

System action

Program FABHFSU ends abnormally.

User response

Correct the PSC control statement or the CTL control statement.

FABH0151E COMPARE OPTION IS NOT ALLOWED

Explanation

Program FABHFSU detected one of the following errors:

- "NO" was specified for the output format of the unloaded data set in the PSB control statement for which the CO (compare) control statement was specified.
- A CO control statement is specified for HALDB.

System action

FABHFSU ends abnormally.

User response

Remove the CO control statement or modify the PSB control statement.

FABH0152I COMPARE OPTION DEACTIVATED BY SEGMENT EXTENSION OPTION

Explanation

Both of the following requests were made:

- The use of the segment extension option of FABHFSU (by coding "E" in column 32 of the PSB control statement in the CARDIN data set).
- The use of the compare option (by providing a CO control statement in the HSSROPT data set).

These two options are mutually exclusive.

System action

Program FABHFSU ignores the requested CO option and continues processing.

User response

None. This message is informational.

FABH0153E **SEGMENT EXTENSION OPTION ONLY VALID WITH HDAM OR HIDAM**

Explanation

You requested the use of the segment extension option of FABHFSU by coding "E" in column 32 of the PSB control statement of the FABHFSU unload utility. This option is valid only for an HDAM or HIDAM database, including PHDAM and PHIDAM.

System action

Program FABHFSU ends abnormally.

User response

Remove the E option because you cannot expand segments in your database with an exit routine.

FABH0154E **'LENGTH OF EXTENDED AREA' FIELD ON 'PSB' CARD IS NOT NUMERIC**

Explanation

The value specified in column 41 of the PSB control statement is not numeric.

System action

Program FABHFSU ends abnormally.

User response

Specify a correct numeric value.

FABH0155E **'LENGTH OF EXTENDED AREA' FIELD ON 'PSB' CARD IS TOO HIGH: MAXIMUM IS 32767**

Explanation

The value specified in column 41 of the PSB control statement exceeds the acceptable maximum.

System action

Program FABHFSU ends abnormally.

User response

Specify the smallest length that can hold each of the extended segments.

FABH0156E **'DATA CONVERSION' OPTION FOR PSB CONTROL STATEMENT IS NOT ALLOWED**

Explanation

The 'data conversion' option 'Y' is specified in column 36 of a PSB control statement. However, the Data Conversion exit routine is not called for the database, for one of the following reasons:

- 'DATXEXIT NO' option is specified in HSSROPT data set.
- DATXEXIT=NO is specified for the database at DBDGEN, and the module DFSDBUX1 was not found in any STEPLIB library.
- DATXEXIT=NO is specified for the database at DBDGEN, and the Data Conversion exit set SRCHFLAG to X'FF'.

System action

Program FABHFSU ends abnormally.

User response

If you need to call the Data Conversion exit routine, you must prepare it in a STEPLIB library and then specify DATXEXIT YES in HSSROPT data set.

FABH0157W **COMPRESSED SEGMENTS MUST BE DECOMPRESSED BEFORE CONVERSION; DECN IS IGNORED**

Explanation

You specified the DECN option in the SYSIN data set for your FABHURG1 job. But there is a compressed segment that also needs to be converted by use of the Data Conversion exit routine (DFSDBUX1). You cannot use the DECN option for a database that contains such a segment.

System action

HSSR Engine continues its processing with the DECY option.

User response

You need to be careful when you use the output data set for reloading because the output data set contains decompressed segments.

FABH0158I **COMPARE OPTION DEACTIVATED BY BLDLPCK CONTROL STATEMENT**

Explanation

The CO control statement (specified in the CARDIN data set) for HS-format output records is ignored when the BLDLPCK statement is specified.

System action

Program FABHFSU ignores the requested CO option and continues processing.

User response

Remove the CO control statement or the BLDLPCK control statement from the CARDIN data set.

FABH0159W DECY FORCED SINCE KEY COMPRESSION OPTION IS SPECIFIED FOR THE ROOT SEGMENT OF HIDAM DB

Explanation

DECN was specified, although the key compression option is specified for the root segment of the HIDAM database to be unloaded in CS format. DECY is assumed.

System action

Program FABHURG1 continues processing.

User response

If you want to unload an HIDAM database without decompressing segments, use an output record format other than the CS format. Otherwise, you can ignore the message.

FABH0161E FABHFSU DOES NOT SUPPORT SECONDARY INDEX PROCESSING

Explanation

The name of an index that is not the name of the HIDAM primary index was specified on a DBD control statement.

System action

Program FABHFSU ends abnormally.

User response

Ensure that the correct DBDs, PSBs, and ACBs were being used. Correct the DBD control statement.

FABH0162E 'LIMIT VALUE TYPE' ON BLM/ELM CONTROL STATEMENT INVALID FOR TYPE OF DB-ORG

Explanation

A value type that is not valid for the database organization was specified on a BLM/ELM control statement (or on an NPT control statement).

System action

Program FABHFSU ends abnormally.

User response

Ensure that the correct DBDs, PSBs, and ACBs were being used. Also ensure that the limit value type is specified correctly on the BLM or ELM control statement.

FABH0164E HIGH-KEY IS SPECIFIED ON BLM/ELM CONTROL STATEMENT

Explanation

A HIGHKEY value for the partitioned HIDAM database is specified on the BLM or ELM control statement. The HIGHKEY values cannot be specified as the limit value for the BLM or ELM control statement.

System action

Program FABHFSU ends abnormally.

User response

Modify the BLM or ELM control statement not to use the HIGHKEY value, and rerun the job.

FABH0165E BLM OR ELM CONTROL STATEMENT IS SPECIFIED FOR A HALDB

Explanation

A BLM or ELM control statement is specified for a PHDAM or PHIDAM database. BLM and ELM statements are not allowed for HALDB.

System action

Program FABHFSU ends abnormally.

User response

Remove the BLM and ELM statements.

FABH0170E HSSR ENGINE IS NOT INITIALIZED

Explanation

Program FABHFSU was not run in the correct environment. It must be run with the FABHDLI or FABHULU procedure (or with equivalent JCL).

System action

FABHFSU ends abnormally.

User response

Run FABHFSU using FABHDLI or FABHULU procedure.

FABH0171E FABHFSU ENVIRONMENT NOT PROPERLY INITIALIZED

Explanation

Program FABHFSU was not run in the correct environment. FABHFSU was renamed, but HSSR Engine does not allow FABHFSU to be renamed.

System action

HSSR Engine ends abnormally.

User response

Run FABHFSU using FABHDLI or FABHULU procedure.

FABH0172E 'LIMIT VALUE TYPE' = 'R' IS ONLY VALID FOR HDAM

Explanation

A Limit Value Type or Node Point Value Type 'R' was specified on a BLM control statement or on a NPT control statement. A Value Type 'R' is only supported for HDAM.

System action

Program FABHFSU ends abnormally.

User response

Correct the control statement.

FABH0175E UNEXPECTED PCB STATUSCODE

Explanation

An HSSR call that was issued internally by program FABHFSU returned an unexpected status code.

System action

FABHFSU ends abnormally.

User response

As a temporary bypass, specify DBDL1 control statement in the HSSROPT data set to force DL/I calls to be made by your program.

FABH0176E DDN=*ddname* BLKSIZE OR LRECL IS TOO SMALL

Explanation

The block size or record size for the output *ddname* data set is too small.

System action

Program FABHFSU ends abnormally.

User response

Specify a block size or record size (LRECL) that is large enough. Record size can be specified only for VB and VN format. If the record size (LRECL) is coded on the JCL statement, it must be less than or equal to the block size minus 4.

FABH0177E DDN=*ddname* OPEN HAS FAILED

Explanation

Program FABHFSU could not open the output *ddname* data set.

System action

FABHFSU ends abnormally.

User response

Check for further error messages issued by the access method and correct the error.

FABH0178E MIGRATION UNLOAD IS NOT SUPPORTED FOR THIS DATABASE TYPE

Explanation

The MI (migration unload format) option on the PSB control statement cannot be used for this database. For more information about the control statement, see “Unload output format supported by FABHFSU” on page 52.

System action

Program FABHFSU ends abnormally.

User response

Ensure that the correct DBD is specified.

FABH0179E TWO OR MORE PSB CANNOT BE SPECIFIED WITH THE MI OPTION

Explanation

When the MI option is specified on the PSB control statement, two or more PSB control statements cannot be specified.

System action

Program FABHFSU ends abnormally.

User response

Specify only one PSB control statement and rerun the job.

FABH0180E INVALID SEGMENT LENGTH RETURNED FROM USER EXIT ROUTINE

Explanation

A FABHFSU user exit routine changed the length of a database segment. The modified length is larger than the allowed maximum segment length.

System action

Program FABHFSU ends abnormally.

User response

Complete the following tasks to resolve the error:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Check the segment length that is returned by the user exit routine, and if necessary, correct the user exit routine.

FABH0181E USER PROGRAMMING ERROR: LENGTH OF CURRENT OUTPUT RECORD IS TOO BIG

Explanation

A FABHFSU user exit routine changed the length of the current output record. The modified length is larger than the allowed maximum record length.

System action

Program FABHFSU ends abnormally.

User response

Correct the user exit routine.

FABH0182E INVALID RETURN CODE FROM USER ROUTINE

Explanation

A FABHFSU user exit routine has returned an incorrect return code.

System action

Program FABHFSU ends abnormally.

User response

Correct the user exit routine or specify the DBR skip option on the PSB control statement.

FABH0183E DDN=*ddname*: OPEN HAS FAILED

Explanation

Program FABHFSU could not open the input *ddname* data set. A *ddname* data set created by FSU II was specified, whereas it should be compared with the FABHFSU output data set.

System action

FABHFSU ends abnormally.

User response

Check for further error messages issued by the access method, and correct the error.

FABH0184E DDN=*ddname* OUTPUT AND INPUT STATISTIC RECORDS ARE NOT EQUAL

Explanation

A FABHFSU output data set was compared with an input *ddname* data set created by FSU II. FABHFSU detected that the statistic records contained in both data sets are not equal.

System action

Program FABHFSU ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that FABHFSU has been provided with the correct FSU II output data set.
- Ensure that FABHFSU and FSU II have been run under identical conditions (same segment sensitivity, same user exit routines, no database updates between the FSU II execution and the FABHFSU execution).
- Determine whether it is an error of FABHFSU or an error of FSU II.
- Ensure that the correct DBDs, PSBs, and ACBs were being used.

FABH0185E DDN=*ddname* OUTPUT AND INPUT RECORDS NOT EQUAL

Explanation

A FABHFSU output data set was compared with an input *ddname* data set created by FSU II. FABHFSU detected at least one record that is not identical in both data sets.

System action

Program FABHFSU ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that FABHFSU has been provided with the correct FSU II output data set.
- Ensure that FABHFSU and FSU II have been executed under identical conditions (same segment sensitivity, same user exit routines, no database updates between the FSU II execution and the FABHFSU execution).
- Determine whether it is an error of FABHFSU or an error of FSU II.
- Ensure that the correct DBDs, PSBs, and ACBs were being used.

FABH0186E DDN=*ddname* CONTAINS TOO MANY RECORDS

Explanation

A FABHFSU output data set was compared with an output *ddname* data set created by FSU II. Program FABHFSU detected that the FSU II output *ddname* data set does not contain the same number of records as the FABHFSU output data set.

System action

FABHFSU ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that FABHFSU has been provided with the correct FSU II output data set.
- Ensure that FABHFSU and FSU II have been executed under identical conditions (same segment sensitivity, same user exit routines, no database updates between the FSU II execution and the FABHFSU execution).
- Determine whether it is an error of FABHFSU or an error of FSU II.
- Ensure that the correct DBDs, PSBs, and ACBs were being used.

FABH0187E DDN=*ddname* CONTAINS TOO FEW RECORDS

Explanation

A FABHFSU output data set was compared with an output *ddname* data set created by FSU II. Program FABHFSU detected that the FSU II output *ddname* data set does not contain the same number of records as the FABHFSU output data set.

System action

FABHFSU ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that FABHFSU has been provided with the correct FSU II output data set.
- Ensure that FABHFSU and FSU II have been executed under identical conditions (same segment sensitivity, same user exit routines, no database updates between the FSU II execution and the FABHFSU execution).
- Determine whether it is an error of FABHFSU or an error of FSU II.
- Ensure that the correct DBDs, PSBs, and ACBs were being used.

FABH0190E DDN=*ddname* LENGTH OF INPUT AND OUTPUT RECORDS DIFFER

Explanation

A FABHFSU output data set was compared with an input *ddname* data set created by FSU II. FABHFSU detected at least one record that is not identical in both data sets.

System action

Program FABHFSU ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that FABHFSU has been provided with the correct FSU II output data set.
- Ensure that FABHFSU and FSU II have been executed under identical conditions (same segment sensitivity, same user exit routines, no database updates between the FSU II execution and the FABHFSU execution).
- Determine whether it is an error of FABHFSU or an error of FSU II.
- Ensure that the correct DBDs, PSBs, and ACBs were being used.

FABH0191E DB ERROR ENCOUNTERED AND "POINTER BYPASS OPTION" NOT ACTIVATED

Explanation

HSSR Engine encountered a database error. By default, FABHFSU issues this message and ends abnormally. However, the pointer bypass option can be activated to bypass the abend.

System action

HSSR Engine ends abnormally.

User response

Check why the database error was encountered. If appropriate, activate the pointer bypass option on the DBD or PSC control statement.

If the problem persists, complete the following tasks to identify the cause of the error:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

FABH0192E SEQUENCE ERROR THRESHOLD EXHAUSTED

Explanation

Program FABHFSU performed sequence-checking of the segment key fields, and detected more sequence errors than specified (or defaulted) on the DBD control statement.

System action

HSSR Engine ends abnormally.

User response

Check why the database has so many sequence errors. If appropriate, increase the sequence error threshold value on the DBD control statement.

If the problem persists, complete the following tasks to identify the cause of the error:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

FABH0193E ROOT-SEGMENT IS NOT SEQUENCED

Explanation

Program FABHFSU was requested to retrieve a root segment by key (either on a BLM or NPT control statement or through a return code from a user exit routine). However, the root segment of the database has no defined sequence field.

System action

HSSR Engine ends abnormally.

User response

Correct the error. If the problem persists, ensure that the correct DBDs, PSBs, and ACBs were being used.

FABH0194E DDN=*ddname*: DSNAME IS NOT VALID

Explanation

The data set name used for the FABHFSU output *ddname* data set was requested to be checked on a CTL control statement. The data set name that was used did not follow the FABHFSU standard naming convention.

System action

Program FABHFSU ends abnormally.

User response

Correct the data set name of the output data set DD statement.

FABH0195E RETURN CODE 4 FROM HDAM RANDOMIZER, RMOD=*xxxxxxxx*

Explanation

Program FABHFSU received a status code FM because of the return code 4 from the HDAM randomizing module. *xxxxxxxx* is the name of the HDAM randomizing module. Register 9 contains the address of the key used by the HSSR call.

System action

FABHFSU ends abnormally.

User response

The randomizer is different from the one for the real database. Use the correct randomizer and rerun the job, if necessary.

FABH0201E OPEN ERROR FOR //IMS DD

Explanation

Program FABHFSU could not open the IMS DD data set to read the IMS DBD control blocks.

System action

FABHFSU ends abnormally.

User response

Provide correct DD statements. Check for additional error messages issued by the access method.

FABH0202E FABHFSU ENCOUNTERED AN UNEXPECTED ERROR

Explanation

Program FABHFSU detected an unexpected error. A possible reason could be a change of IMS DBD control blocks.

System action

FABHFSU ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the FABHFSUI module corresponds to the correct IMS version.
- Ensure that the correct DBDs, PSBs, and ACBs were being used.

FABH0203W WARNING: SOME SEGMENTS WILL NOT BE DECOMPRESSED

Explanation

You requested, by specifying the DECN option on the DEC control statement, that program FABHURG1 or FABHFSU does not decompress compressed segments. The output data sets of FABHURG1 or FABHFSU will contain segments in the compressed format, unless other decompression program (for example, a FABHURG1 or FABHFSU user exit routine) is activated.

System action

HSSR Engine continues processing.

User response

If you really do not want to decompress the compressed segments, ignore this message.

FABH0204W COMPARE OPTION CANNOT BE USED FOR THIS FABHFSU EXECUTION

Explanation

HSSR Engine encountered a condition that precludes use of the compare option.

System action

HSSR Engine deactivates the compare option.

User response

None.

FABH0205W COMPARE OPTION CANNOT BE USED FOR THIS FABHURG1 EXECUTION

Explanation

HSSR Engine encountered a condition that precludes use of the compare option.

System action

HSSR Engine deactivates the compare option.

User response

None.

FABH0206I SOME SEGMENTS WILL NOT BE DECOMPRESSED

Explanation

A segment compression exit routine is defined in the database. HSSR Engine sets the compressed segment data in the I/O area without decompressing it because the DECN control statement is specified in the HSSROPT data set. If a key field is compressed, the key data is not set in the PCB key feedback area.

System action

HSSR Engine continues processing.

User response

None. This message is informational.

FABH0211E ENQ CONFLICT: OTHER JOB CURRENTLY ACTIVE FOR THE SAME PSC PHASE

Explanation

The concurrent multiple jobs were started for the same phase of a parallel scan.

System action

Program FABHFSU ends abnormally.

User response

Correct the error.

FABH0212E DATE FOR PSC RUN HAS ALREADY EXPIRED

Explanation

Program FABHFSU was started after the expiration date specified on the CTL or on a PSC control statement.

System action

FABHFSU ends abnormally.

User response

Correct the CTL or PSC control statement.

FABH0213E PSC PHASE HAS ALREADY BEEN STARTED

Explanation

Program FABHFSU was started multiple times for the same phase.

System action

FABHFSU ends abnormally.

User response

If appropriate, specify a Rerun Indicator on the PSC control statement.

FABH0214E DDN=*ddname*: BLOCKSIZE SHOULD NOT CHANGE

Explanation

Program FABHFSU was run multiple times with different block sizes for the output data set. The block sizes must be the same.

System action

FABHFSU ends abnormally.

User response

Specify identical block sizes for all executions.

FABH0215I ALL SCAN PHASES STARTED

Explanation

All the FABHFSU parallel scan phases were in starting condition.

System action

Program FABHFSU continues processing.

User response

None. This message is informational.

FABH0216E PARTITION DB IS NOT SUPPORTED IN PSF MODE

Explanation

The partition database cannot be unloaded in PSF mode.

System action

Program FABHFSU ends abnormally.

User response

If you want to unload the partitioned database, run FABHFSU in standard mode or use the FABHURG1 unload utility.

FABH0221E UNEXPECTED LAYOUT OF PSC CONTROL-FILE RECORDS

Explanation

Program FABHFSU read from the PSC control data set a record with an unexpected record format.

System action

FABHFSU ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the PSC control data set has been created correctly by FABHPSFC, and that it has been created with the correct DBD version.
- Ensure that the correct DBDs, PSBs, and ACBs were being used.

FABH0222E UNEXPECTED CONDITION ON PSC CONTROL FILE: PHASE ALREADY COMPLETED

Explanation

At job-step termination, program FABHFSU detected that its phase is already marked as "completed" in the PSC control data set. Such an incorrect situation can occur in an uncontrolled multi-system environment.

System action

FABHFSU ends abnormally.

User response

Do not run the PSF option of FABHFSU in an uncontrolled multi-system environment. Consider using Global Resource Serialization.

If the problem persists, ensure that the correct DBDs, PSBs, and ACBs were being used.

FABH0223I ALL SCAN PHASES COMPLETED

Explanation

All the FABHFSU parallel scan phases were completed.

System action

Program FABHFSU continues processing.

User response

None. This message is informational.

FABH0224W NO SEGMENT WAS RETRIEVED

Explanation

Program FABHFSU attempted to unload the database, but no segment was retrieved from the database. The database might be empty.

System action

FABHFSU normally ends with a return code of 01.

User response

Determine whether it is acceptable that the input database has no valid segments. If a wrong database data set, or wrong DBD or PSB was specified, specify the appropriate ones, and rerun the job.

FABH0225W NO SEGMENT WAS RETRIEVED IN THE CURRENT PSF PHASE

Explanation

Program FABHFSU attempted to unload a portion of a database in this PSF phase, but no segment was retrieved in the phase.

System action

FABHFSU normally ends with a return code of 01.

User response

Determine whether it is acceptable that the portion has no valid segments. If a wrong database data set, or wrong DBD or PSB was specified, specify the appropriate ones, and rerun the job.

FABH0231E PCB'S CANNOT BE FOUND

Explanation

On entry, FABHTEST could not find the PCBs. This was probably caused by FABHTEST not being invoked by the FABHDLI or FABHDBB procedure (or by equivalent JCLs).

System action

Program FABHTEST ends abnormally.

User response

Use the FABHDLI or FABHDBB procedure.

FABH0232E PSB HAS NO DB-PCB'S

Explanation

The provided PSB has no database PCBs. Therefore, the run of FABHTEST is meaningless.

System action

Program FABHTEST ends abnormally.

User response

Provide a PSB that refers to a database.

FABH0233E SYSIN CANNOT BE OPENED

Explanation

The SYSIN data set containing the FABHTEST control statements could not be opened.

System action

Program FABHTEST ends abnormally. System messages are issued to describe the problem in more detail.

User response

Add a SYSIN DD statement to the appropriate control statements.

FABH0234E INVALID TYPE OF CONTROL STATEMENT

Explanation

The last control statement read has an incorrect statement type in columns 1–4. Only the following control statement types can be specified: PCB, GU, GN, or GNR.

System action

Program FABHTEST ends abnormally.

User response

If the abend is unintentional, correct the control statement.

FABH0235E SPECIFIED PCB IS NEITHER A HSSR-PCB NOR A DL/I-DB-PCB

Explanation

The PCB specified on the PCB control statement is neither an HSSR PCB nor a DL/I database PCB.

System action

Program FABHTEST ends abnormally.

User response

Correct the PCB control statement or modify the PSB.

FABH0236E SPECIFIED PCB-NUMBER NOT WITHIN PSB

Explanation

The PCB number specified on the PCB control statement does not exist within the provided PSB.

System action

Program FABHTEST ends abnormally.

User response

Correct the PCB control statement or modify the PSB.

FABH0237E GU WITH SSA NOT POSSIBLE BECAUSE ROOT HAS NO KEY-FIELD

Explanation

The last printed GU control statement contains a relational operator in columns 16–17. This means that program FABHTEST should issue GU calls with an SSA qualified on the key field of the root segment. However, the root segment has no key fields.

System action

FABHTEST ends abnormally.

User response

Remove the GU control statement.

FABH0238E THE 'NUMBER FIELD' IS NOT NUMERIC

Explanation

Positions 5–14 of the last printed control statement contains a character that is neither numeric nor blank.

System action

Program FABHTEST ends abnormally.

User response

Correct the control statement.

FABH0239E REPL STATEMENT IS NOT SUPPORTED FOR PARTITION DB

Explanation

A REPL statement is specified in the SYSIN data set for the FABHTEST utility. REPL statement is not allowed for the FABHTEST utility.

System action

Program FABHTEST ends abnormally.

User response

Modify the SYSIN statement not to use a REPL statement.

FABH0240E UNEXPECTED CONTINUATION STATEMENT

Explanation

The last printed statement is a continuation control statement that is not preceded by a control statement containing the continuation indicator in column 72.

System action

Program FABHTEST ends abnormally.

User response

Remove the offending continuation control statement, or insert a continuation indicator in the preceding control statement.

FABH0241E ILLEGAL CONTINUATION; KEY FIELD SHOULD END ON THIS CONTROL STATEMENT

Explanation

The last printed control statement contains a continuation indicator in column 72, but the FIELD statement of DBDGEN indicates that the key field of the root is short enough to end on the last printed control statement.

System action

Program FABHTEST ends abnormally.

User response

Check for the length of the key field of the root segment, and correct the FABHTEST control statements.

FABH0242E CONTINUATION CARD IS MISSING

Explanation

Program FABHTEST expected a continuation control statement, but received the last printed control statement.

System action

FABHTEST ends abnormally.

User response

Insert the missing continuation control statement.

FABH0243E SYSPRINT CANNOT BE OPENED

Explanation

The SYSPRINT data set containing a listing of the FABHTEST control statements could not be opened.

System action

Program FABHTEST ends abnormally. System messages will be issued to describe the problem in more detail.

User response

Correct the SYSPRINT DD statement.

FABH0248W INCORRECT ROOT KEY IN GU-CALL (STATUS CODE FM)

Explanation

Program FABHTEST received a status code FM from HSSR call handler because of a return code 4 from the HDAM randomizing module.

System action

FABHTEST program continues processing.

User response

Determine whether the preceding is acceptable.

FABH0249E PCB WITH SPECIFIED DBDNAME NOT FOUND

Explanation

The last printed PCB control statement contains a DBDNAME that is not referred to by any PCB of the PSB.

System action

Program FABHTEST ends abnormally.

User response

Correct the PCB control statement or provide another PSB.

FABH0250E FABH001 HAS NOT BEEN PROPERLY INITIALIZED

Explanation

The HSSR control block HTCB located within the load module FABH001 was not properly initialized by HSSR Engine.

System action

HSSR Engine ends abnormally.

User response

Possible reason for this error is that FABHURG1 was not run with JCL equivalent to the procedures FABHDLI, FABHDBB, or FABHULU but with JCL similar to the procedures DLIBATCH or DBBBATCH.

Execute FABHURG1 with correct JCL.

FABH0251E SYSPRINT COULD NOT BE OPENED

Explanation

Program FABHURG1 could not open the SYSPRINT data set.

System action

FABHURG1 ends abnormally.

User response

Provide or correct a SYSPRINT DD statement.

FABH0252E INVALID STATEMENT WAS DETECTED

Explanation

An incorrect control statement type was specified in the FABHURG1 SYSIN data set.

System action

Program FABHURG1 ends abnormally.

User response

Correct the utility control statement.

FABH0253E PCB NUMBER IS NOT POSITIVE

Explanation

A FABHURG1 SYSIN data set PCB control statement specified a PCB number that was not positive.

System action

Program FABHURG1 ends abnormally.

User response

Correct the PCB control statement.

FABH0254E SPECIFIED HSSR PCB IS NOT FOUND

Explanation

A FABHURG1 SYSIN data set PCB control statement specified an HSSR PCB that could not be found. Note that the PCB control statement should refer to HSSR PCBs, never to DL/I PCBs.

System action

Program FABHURG1 ends abnormally.

User response

Correct the utility control statement or the PSB.

FABH0255E NUMERIC FIELD IS NOT NUMERIC

Explanation

A FABHURG1 SYSIN data set control statement, in a field defined as numeric, contains non-numeric information.

System action

Program FABHURG1 ends abnormally.

User response

Correct the control statement.

FABH0256E BLKSIZE OR LRECL OF *ddname* IS TOO SMALL

Explanation

The block size or record size of the *ddname* data set is too small. For *HD output, the block size is always the maximum device capacity. For other output, the block size or record size is the maximum device capacity unless a block size or record size is coded on the JCL statement.

System action

Program FABHURG1 ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- If a user record-formatting routine is used, check whether ULEN can be reduced.
- If the utility uses the maximum device capacity, check whether *ddname* can be allocated on a device with more track capacity.
- If the utility uses the block size or record size specified on the JCL statement, check whether the block size or record size can be increased or use the default maximum device capacity.
- Record size can be specified only for *F1, *F2 and *F3 format. If the record size (LRECL) is coded on the JCL statement, it must be less than or equal to the (block size) minus 4.
- If an optional exit routine is used for the FABHURG1 utility, check whether the length specified on the USEGMAX control statement can be reduced.

FABH0257E OPEN OF *ddname* HAS FAILED

Explanation

Program FABHURG1 could not open *ddname*.

System action

FABHURG1 ends abnormally.

User response

Correct the *ddname* DD statement.

FABH0258E THERE ARE NO HSSR PCBS

Explanation

The PSB does not contain an HSSR PCB.

System action

Program FABHURG1 ends abnormally.

User response

Correct the PSB or remove the HSSROPT DBDL1 control statement.

FABH0259E USER PROGRAMMING ERROR: LENGTH OF CURRENT *ddname* RECORD IS TOO LARGE

Explanation

A user record-formatting routine or a user exit routine created a *ddname* record that is larger than allowed.

System action

Program FABHURG1 ends abnormally.

User response

Correct the user routine.

FABH0260E INVALID RETURN CODE FROM USER ROUTINE

Explanation

A user routine set an incorrect return code. Only the following return codes are valid: 0, 1, 2, 3.

System action

Program FABHURG1 ends abnormally.

User response

Correct the user routine.

FABH0261E OPEN OF SYSUT1 HAS FAILED

Explanation

Program FABHURG1 could not open SYSUT1.

System action

FABHURG1 ends abnormally.

User response

Correct the SYSUT1 DD statement.

FABH0262E RECORDS OF *ddname* ARE NOT EQUAL

Explanation

Program FABHURG1 compared its *ddname* output with the output from the IMS HD Reorganization Unload utility (which was defined by the SYSUT1 DD

statement). FABHURG1 detected that these data sets contain records that are not equal.

System action

FABHURG1 ends abnormally. In the dump, Register 7 will point to the SYSUT1 record and Register 8 to the *ddname* record.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.
- Ensure that the database and DBD have not been updated between the execution of IMS HD Reorganization Unload and the execution of FABHURG1.
- Ensure that SYSUT1 is created with the IMS HD Reorganization Unload utility without any user exit routine.
- Check whether the execution of the IMS HD Reorganization Unload utility is successful.

If the problem persists, run FABHURG1 with the HSSROPT compare option and the HSSROPT hardcopy trace option. If necessary, contact IBM Software Support.

FABH0263E SYSUT1 CONTAINS MORE DATA RECORDS THAN *ddname*

Explanation

Program FABHURG1 compared its *ddname* output with the output of the IMS HD Reorganization Unload utility (which was defined by the SYSUT1 DD statement). FABHURG1 detected that these data sets did not contain the same number of database segments.

System action

FABHURG1 ends abnormally.

User response

See message FABH0262E, and take an appropriate action.

FABH0264E SYSUT1 CONTAINS LESS DATA RECORDS THAN *ddname*

Explanation

Program FABHURG1 compared its *ddname* output with the output of the IMS HD Reorganization Unload utility (which was defined by the SYSUT1 DD statement). FABHURG1 detected that these data sets did not contain the same number of database segments.

System action

FABHURG1 ends abnormally.

User response

See message FABH0262E, and take an appropriate action.

FABH0265W SIZE OF STATISTIC TABLES ARE NOT THE SAME IN SYSUT1 AND *ddname*

Explanation

Program FABHURG1 compared its *ddname* output with the output of the IMS HD Reorganization Unload utility (which was defined by the SYSUT1 DD statement). FABHURG1 detected that the statistic tables stored in the first and last record do not have the same length.

System action

FABHURG1 ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the database and DBD have not been updated between the execution of IMS HD Reorganization Unload and the execution of FABHURG1.
- Ensure that SYSUT1 is created with the IMS HD Reorganization Unload utility without any user exit routine.
- Check whether the execution of the IMS HD Reorganization Unload utility is successful.

If the problem persists, run FABHURG1 with the HSSROPT compare option and the HSSROPT hardcopy trace option. If necessary, contact IBM Software Support.

FABH0266W COMPARE OF SYSUT1 AND *ddname* STOPPED

Explanation

Program FABHURG1 compared its *ddname* output with the output of the IMS HD Reorganization Unload utility. FABHURG1 stopped the compare for one of the following reasons:

- Some segments are not sensitive.
- A user routine returned a nonzero return code.

System action

Because the comparison is no longer possible, FABHURG1 continues its processing without performing the comparison of SYSUT1 and *ddname*.

User response

None.

FABH0267E GG STATUS CODE ENCOUNTERED AND SKERROR OPTION WAS NOT ACTIVATED

Explanation

HSSR Engine detected a database error and returned a GG status code to program FABHURG1. FABHURG1 can handle this status code only if the SKERROR option has been activated.

System action

FABHURG1 ends abnormally.

User response

If you want to unload a corrupted database, activate the SKERROR option. In other cases, complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

FABH0268W WARNING: NOT ALL SEGMENTS WERE SENSITIVE

Explanation

Program FABHURG1 detected that some database segment types were not sensitive.

System action

FABHURG1 continues processing.

User response

Check whether FABHURG1 uses a PSB with some segment types not defined as sensitive.

FABH0269W WARNING: SOME SEGMENTS HAVE BEEN SKIPPED BY USER ROUTINES

Explanation

Program FABHURG1 detected that a user routine has set a nonzero return code, indicating that some segment occurrences should be skipped and should not be written to any unload data set.

System action

FABHURG1 continues processing.

User response

Check whether the user routines should really set a nonzero return code.

FABH0270W WARNING: SOME SEGMENTS HAVE A VIRTUAL LOGICAL PARENT KEY

Explanation

Program FABHURG1 detected that a logical child segment has a virtual logical-parent-concatenated-key (LPCK), but the BLDLPCK control statement is not specified. Hence the output will contain blanks instead of the LPCK (unless user routines have performed additional processing).

System action

FABHURG1 continues processing.

User response

Check whether this condition is acceptable. If it is not, specify the BLDLPCK control statement.

FABH0271W WARNING: SEGMENTS MAY BE MISSING ON UNLOADED OUTPUT BECAUSE OF GG STATUS CODE

Explanation

Program FABHURG1 is running with the SKERROR option in order to unload a database that contained database errors. Some segments might be missing on the unloaded output.

System action

FABHURG1 continues processing.

User response

Determine whether the unloaded database version is acceptable to the applications. If not, complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

FABH0272I DB UNLOAD COMPLETE

Explanation

Program FABHURG1 reached the end of the database and has finished its processing.

System action

FABHURG1 continues processing.

User response

None. This message is informational.

FABH0273I *nnn*,*nnn*,*nnn* DB RECORDS RETRIEVED - DBD=*dbdname* PART=*ddname*

Explanation

nnn,*nnn*,*nnn* database records have been retrieved from database *dbdname*. PART=*ddname* is displayed only for partitioned databases, and in that case, the database records are counted per partition.

System action

Program FABHURG1 continues processing.

User response

None. This message is informational.

FABH0274E RETURN CODE 4 FROM HDAM RANDOMIZER, RMOD=*xxxxxxxx*

Explanation

Program FABHURG1 received a status code FM because of the return code 4 from HDAM randomizing module. *xxxxxxxx* is the name of HDAM randomizing module. Register 9 contains the address of key used by the HSSR call.

System action

FABHURG1 ends abnormally.

User response

The randomizer is different from the one for the real database. Use the correct randomizer and rerun the job, if necessary.

FABH0275W WARNING: NO SEGMENT WAS RETRIEVED

Explanation

Program FABHURG1 attempted to unload a database, but no segment was retrieved from the database. The database might be empty.

System action

FABHURG1 normally ends with a return code of 01.

User response

Check whether it is acceptable that the input database has no valid segments. If a wrong database data set, or wrong DBD or PSB was specified, specify the appropriate ones, and rerun the job.

FABH0276I USEGMAX STATEMENT IS IGNORED

Explanation

This message is issued in one of the following cases:

1. The length specified on the USEGMAX control statement is less than the length of the longest database segment.
2. No user exit routine is specified.
3. The USEGMAX statement is specified for the *F3 format or for the user record-formatting routine.

System action

For each case, the following system action is taken:

1. The length specified on the USEGMAX statement is adjusted to the length of the longest segment.
2. The USEGMAX statement is ignored.
3. The USEGMAX statement is ignored.

In any of these cases, program FABHURG1 continues its processing and ends with a return code of 0.

User response

If the system action is acceptable, remove the USEGMAX statement or modify the value for the USEGMAX control statement for the next run. If the action is not acceptable, check your control statements or the DBD that you specified. Then correct the error, and rerun the job.

FABH0278I NOT ALL PARTITIONS ARE PROCESSED

Explanation

This message is for information only. This message notifies the user that all partitions are not processed by the job. Only the selected partitions are processed.

System action

HSSR Engine continues processing.

User response

None. This message is informational.

FABH0279E 'DATA CONVERSION' OPTION FOR EXIT CONTROL STATEMENT IS NOT ALLOWED

Explanation

The data conversion option Y is specified in column 15 of an EXIT control statement although the Data Conversion exit routine is not called for the database for one of the following reasons:

- DATXEXIT NO option is specified in the HSSROPT data set.
- DATXEXIT=NO is specified for the database at DBDGEN, and the module DFSDBUX1 was not found in any STEPLIB library.
- DATXEXIT=NO is specified for the database at DBDGEN, and the Data Conversion exit set SRCHFLAG to X'FF'.

System action

Program FABHURG1 ends abnormally.

User response

If you want to call a Data Conversion exit routine, you must prepare it in a STEPLIB library and specify 'DATXEXIT YES' in the HSSROPT data set.

FABH0280E INVALID 'DATA CONVERSION' OPTION ON EXIT CONTROL STATEMENT

Explanation

Program FABHURG1 detected an error in the 'data conversion' option field of the EXIT control statement.

System action

FABHURG1 ends abnormally.

User response

Correct the error.

FABH0281E OUTPUT FORMAT *CS IS NOT SUPPORTED FOR THIS DATABASE: DBD=*dbdname*

Explanation

This message is issued if the *CS format is specified for a SHISAM database or for a secondary index.

System action

Program FABHURG1 ends abnormally.

User response

Use an unload format other than *CS.

FABH0282E [MIGRATION|FALLBACK] UNLOAD IS NOT SUPPORTED IN THIS REGION TYPE

Explanation

The migration unload and the fallback unload are not supported in the DLI or the DBB region. You must run these jobs in the ULU region.

System action

Program FABHURG1 or FABHFSU ends abnormally.

User response

Run the job in the ULU region.

FABH0283E INVALID PART STATEMENT IS SPECIFIED: *reason*

Explanation

An incorrect PART statement is specified in the SYSIN data set. *reason* indicates the reason of the error, as follows:

reason
Description

NON-PARTITIONED DB
The PART control statement is specified for a nonpartitioned database.

TOO MANY PARAMETERS
More than 32 partitions are specified on the PART control statements.

DUPLICATION (*ddname*)
The partition DD name *ddname* is specified more than once.

NOT FOUND (*ddname*)
The partition DD name *ddname* specified on a PART statement was not found in the DBD.

System action

Program FABHURG1 ends abnormally.

User response

Correct the error and rerun the job.

FABH0284E INCORRECT CHECKREC OPTION

Explanation

An incorrect CHECKREC option is specified in the SYSIN data set. It must be either YES or NO.

System action

Program FABHURG1 ends abnormally.

User response

Correct the error.

FABH0285E OUTPUT FORMAT *CP IS NOT SUPPORTED FOR THIS DATABASE: DBD=*dbdname*

Explanation

This message is issued if the *CP format is specified for a database that is not a HALDB.

System action

Program FABHURG1 ends abnormally.

User response

Use an unload format other than *CP.

FABH0286E SPECIFIED DBVER IS NOT SUPPORTED

Explanation

IMS database versioning is enabled, but the specified database is not the current version of the database. When IMS database versioning is enabled, only the current version of the database can be used.

System action

FABHURG1 or FABHFSU ends abnormally.

User response

Specify the current version of the database.

FABH0287E LOAD FAILED FOR MODULE: *module*

Explanation

Load failed for the indicated module.

System action

Program FABHURG1 or FABHFSU ends abnormally.

User response

Check the contents of the load module library. Correct the error and rerun the job.

FABH0288E SINGLE PARTITION PROCESSING IS NOT SUPPORTED

Explanation

When the entire database processing is requested with no PARTITION control statement of program FABHURG1 or FABHFSU, the DFSHALDB data set for HALDB Single Partition Processing cannot be used.

System action

FABHURG1 or FABHFSU ends abnormally.

User response

Remove the HALDB control statement on the DFSHALDB DD statement. If you want to unload a partition, specify the PARTITION control statement in the SYSIN data set of FABHURG1 or in the CARDIN data set of FABHFSU.

FABH0289E CEEPIPI ERROR OCCURRED. FUNC=*function*, RC=*return_code*

Explanation

IMS HP Unload invoked the Language Environment CEEPIPI service, but the CEEPIPI service returned a non-zero return code.

System action

IMS HP Unload ends abnormally.

User response

When *function* shows INIT, the function of CEEPIPI is *init_sub* or *identify_entry*. When *function* shows CALLSUB, the function of CEEPIPI is *call_sub*. For the return codes, see the topic about preinitialization services in the *z/OS Language Environment Programming Guide*.

Correct the error and rerun the job.

FABH0290E **MUTUALLY EXCLUSIVE CONTROL STATEMENTS ARE SPECIFIED: *xxxxxxxx* AND *yyyyyyyy***

Explanation

The control statements *xxxxxxxx* and *yyyyyyyy* are mutually exclusive.

System action

Program FABHURG1 ends abnormally.

User response

Select only one of these statements, and remove the rest.

FABH0291E **AN ERROR IS FOUND IN A 'PARTITION' CONTROL STATEMENT: *reason***

Explanation

A PARTITION statement is coded incorrectly. The string *reason* indicates the *reason* for the error:

reason
Description

NON-HALDB
The PARTITION statement is specified for a database that is not a HALDB.

NAME TOO LONG
The value specified as the first operand of the PARTITION statement has more than seven bytes.

NUMBER TOO LONG
The numeric value specified as the second operand of the PARTITION statement has more than five digits.

NUMBER TOO LARGE
The numeric value specified as the second operand of the PARTITION statement is greater than 1001.

NUMBER IS ZERO
The numeric value specified as the second operand of the PARTITION statement is 0.

System action

Program FABHURG1 ends abnormally.

User response

Correct the error in the PARTITION statement.

FABH0292E **MIGRATE CONTROL STATEMENT IS SPECIFIED FOR AN UNSUPPORTED DB TYPE**

Explanation

The MIGRATE control statement cannot be used for this database. For details about the control statement, see "MIGRATE control statement" on page 43.

System action

Program FABHURG1 ends abnormally.

User response

Ensure that the correct DBD is specified.

FABH0293E **FALLBACK CONTROL STATEMENT IS SPECIFIED FOR NON-HALDB**

Explanation

The FALLBACK control statement is specified for a database that is not a HALDB.

System action

Program FABHURG1 ends abnormally.

User response

Ensure that the correct DBD is specified.

FABH0294E **INVALID UNLOAD FORMAT IS SPECIFIED FOR [MIGRATION|FALLBACK] UNLOAD**

Explanation

A format other than *HD is specified on the FRMT control statement for the FABHURG1 job, although the migration or fallback unload is designated.

System action

Program FABHURG1 ends abnormally.

User response

Specify the *HD format, and rerun the job.

FABH0295E USEGMAX CONTROL STATEMENT IS SPECIFIED FOR [MIGRATION| FALLBACK] UNLOAD

Explanation

The USEGMAX control statement is specified for the FABHURG1 job, although the migration or fallback unload is designated. The USEGMAX control statement is not allowed in migration and fallback unload.

System action

Program FABHURG1 ends abnormally.

User response

Remove the USEGMAX control statement, and rerun the job.

FABH0296E LE OPTION IS NOT ALLOWED WHEN THE RUN TIME ENVIRONMENT EXIT ROUTINE IS BEING INVOKED

Explanation

If the RTEXIT control statement is specified for a user runtime environment exit routine, the language environment option cannot be specified to activate the Language Environment.

System action

IMS HP Unload ends abnormally.

User response

Remove the RTEXIT control statement and rerun the job.

FABH0299I UNLOAD FUNCTION ENDED, HIGHEST RETURN CODE IS *xxx* (REASON CODE: *yyyy*)

Explanation

Program FABHURG1 that had been invoked by using a JCL that is compatible with IMS HD Reorganization Unload utility ended with the decimal return code *xxx*. For the meaning of hexadecimal reason code *yyyy* for the return code of 4, see “FABHURG1 return codes when IMS HD Reorganization Unload JCL is used” on page 379.

System action

Program FABHURG1 ends its processing.

User response

Check the return code. If the return code is 4, check also the reason code. Check the accompanying FABH*xxxxx* messages.

FABH0300E ERROR IN OPENING SYSIN

Explanation

The runtime initializer of IMS High Performance Unload called from the DFSISVI0 exit failed in opening the SYSIN data set.

System action

IMS High Performance Unload's runtime initializer ends abnormally.

User response

Find the cause of the open error.

FABH0301E //EXEC PARM-FIELD SHOULD BEGIN WITH NAME OF REGION CONTROLLER FOLLOWED BY '/'

Explanation

The PARM field of the EXEC statement does not begin with the name of the region controller followed by a slash.

System action

The runtime initializer ends abnormally.

User response

Correct the EXEC statement.

FABH0302E NAME OF REGION CONTROLLER IN //EXEC PARM-FIELD IS LONGER THAN 8 BYTES

Explanation

The PARM field of the EXEC statement does not begin with the name of the region controller followed by a slash. Either the slash is omitted, or the region controller name is more than 8 bytes long.

System action

The runtime initializer ends abnormally.

User response

Correct the EXEC statement.

FABH0303E PGM-NAME HAS NOT BEEN PROVIDED ON //EXEC PARM-FIELD

Explanation

The PARM field of the EXEC statement does not contain the name of the application program.

System action

The runtime initializer ends abnormally.

User response

Correct the EXEC statement.

FABH0304E PGM-NAME IN //EXEC PARM-FIELD IS LONGER THAN 8 BYTES

Explanation

The PARM field of the EXEC statement contains an application program name that is longer than 8 bytes.

System action

The runtime initializer ends abnormally.

User response

Correct the EXEC statement.

FABH0305E DFSURGU0 ALREADY IN VIRTUAL STORAGE

Explanation

In a ULU region type, the runtime initializer identifies the entry point DFSURGU0 as the entry into FABH000. This requires that the module DFSURGU0 is not already in virtual storage at initialization time. This condition was not met during this execution.

System action

The runtime initializer ends abnormally.

User response

Contact IBM Software Support.

FABH0306E UNEXPECTED RETURN CODE FROM IDENTIFY

Explanation

In a ULU region type, the runtime initializer issues an IDENTIFY macro. The IDENTIFY failed. This should not occur.

System action

The runtime initializer ends abnormally.

User response

Contact IBM Software Support.

FABH0307E LOAD MODULE FABH000 IS NOT REENTRANT

Explanation

FABH000 must be link-edited with the link-editor attribute RENT. This was not the case.

System action

The runtime initializer ends abnormally.

User response

Contact IBM Software Support.

FABH0308E MEMBER NAME DFSURGU0 IS INVALID FOR A JOB RUNNING IN ULU REGION

Explanation

The IMS utility DFSURGU0 cannot be executed in a ULU region.

System action

The runtime initializer ends abnormally.

User response

Instead of executing DFSURGU0 in the ULU region, execute it with normal IMS job control statements.

FABH0309E FABH001 IS NOT REUSABLE

Explanation

The load module FABH001 must have the link-editor attribute REUS. This was not the case.

System action

The runtime initializer ends abnormally.

User response

Find out why FABH001 is not reusable (it might be a submodule which is not reusable). If necessary, contact IBM Software Support.

FABH0310E MODULE FABH001 IS NOT REUSABLE; PLEASE INFORM YOUR IMS–SPECIALISTS

Explanation

IMS High Performance Unload's program controller detected an error. One likely cause of the error is that the load module FABH001 does not have the linkage-editor attribute REUS.

System action

The job ends abnormally with a dump.

User response

Check in the dump if FABH001 is serially reusable. If necessary, relink FABH001 as serially reusable.

FABH0311E DYNAMIC ALLOCATION FAILED FOR DD *xxxxxxxx*: RC=*xx* RSN=*yyyy*

Explanation

The dynamic allocation for the DD *xxxxxxxx* failed. The values *xx* and *yyyy* are the return code and reason code that are returned from the SVC99 routine.

System action

HSSR Engine ends abnormally.

User response

If the DD name *xxxxxxxx* is DFSVSAMP, this might be an internal system error. If so, collect the dump and contact IBM Software Support.

FABH0312E DFSVSAMP OPEN FAILURE OCCURRED

Explanation

The runtime initializer failed to open the DFSVSAMP data set that the initializer dynamically allocated.

System action

The runtime initializer ends abnormally.

User response

This error is likely an internal system error. Collect the dump and contact IBM Software Support.

FABH0313E SYNTAX ERROR IN SYSIN CONTROL STATEMENTS

Explanation

Both a control statement for IMS HD Reorganization Unload utility (DFSURGU0) and a control statement for program FABHURG1 are specified in the SYSIN data set. You must specify control statements for one, but not both, of these utilities.

System action

IMS High Performance Unload's runtime initializer ends abnormally.

User response

Correct the SYSIN control statements.

FABH0314E HSSR INITIALIZATION FAILED

Explanation

HSSR engine detected an error, which is described in the previous error message.

System action

HSSR Engine ends abnormally.

User response

Remove the cause of the error by referring to the message issued before this message. If the cause is not clear, collect the dump and contact IBM Software Support.

FABH0315E [GU/GHU/GN/GHN/GNP/GHNP] CALL SSA-*n* IS UNSUPPORTED: *reason*

Explanation

The application program issued a call with one or more SSAs. The *n*-th SSA is not supported by the HSSR call handler. The reason is one of:

SEGMENT NAME
The segment name is not defined in the HSSR PCB.

SEGMENT LEVEL
The SSA for the segment level is not supported.

COMMAND CODE
The command code is not supported.

NON-SEQ FIELD
The field is not a sequence key.

NON-UNIQUE KEY
The key field is not unique.

RELATIONAL OPERATOR
The relational operator is not supported.

MULTI QUALIFICATION STATEMENTS
The multi-qualification statement is not supported.

QUALIFIED SSA
The qualified SSA for root segment is not supported because the database is HDAM or PHDAM.

CALL FUNCTION
The call type is not supported.

UNKNOWN
Unknown reason.

The unsupported call is printed in the Trace Output report in the HSSRTRAC data set.

System action

If APISET is 2, HSSR Engine ends abnormally. If APISET is 3, the call and all the succeeding calls to the HSSR PCB are passed to the IMS DL/I call handler to continue the processing instead of ending it abnormally.

User response

If APISET is 2, specify 'APISET 3'. If APISET is 3, ignore this message. For details about the call types and command types supported by the HSSR call handler, see "DL/I calls and EXEC DLI command for HSSR PCB" on page 84.

FABH0316E ERROR RETURN FROM IGWASYS (RC=*xx*, RSN=*yy*)

Explanation

HSSR Engine called IGWASYS, but an error return code was returned from IGWASYS. Here, *xx* is the return code and *yy* is the reason code.

System action

HSSR Engine ends abnormally.

User response

See *DFSMSdfp Advanced Services* for the meaning of the return code and reason code.

FABH0319E GN WITH QUALIFIED SSA CALL IS ISSUED TO THE COMPRESSED ROOT KEY

Explanation

The GN call with an SSA qualified on the root key field is issued, however, HSSR Engine cannot compare the compressed key with the provided value.

System action

HSSR Engine ends abnormally.

User response

For HIDAM or PHIDAM database, specify the BYINDEX control statement in HSSROPT, then the root key stored in the index database is used. Otherwise, check whether the correct APISET control statement is specified. For details about the call types and command types that APISET supports, see "DL/I calls and EXEC DLI command for HSSR PCB" on page 84.

FABH0320E GB ON PCF OR TWIN-NOT-ROOT

Explanation

In HD organizations, HSSR call handler follows the HD chains to find out which is the next segment to be processed. During this process, HSSR call handler "lost itself."

System action

HSSR Engine ends abnormally.

User response

To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378).

FABH0321E RBA OUT OF DB-RANGE

Explanation

In an HD organization, HSSR call handler found a pointer that points beyond the data set end.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378).
- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

FABH0322E S-C DOES NOT MATCH

Explanation

In an HD organization, HSSR call handler checks the segment code of a retrieved segment against the expected segment code. This check was not successful.

System action

HSSR Engine ends abnormally.

User response

See message FABH0321E, and take an appropriate action.

FABH0324E BAD RETURN-CODE FROM BUFFER HANDLER

Explanation

HSSR buffer handler encountered an I/O problem. More information can be found in the accompanying SYNADAF message buffer.

System action

HSSR Engine ends abnormally.

User response

See message FABH0321E, and take an appropriate action.

FABH0325E INVALID SEGMENT-CODE IN HISAM

Explanation

HSSR call handler checked the segment code of an HISAM database. The segment code is not within the range of defined segment codes.

System action

HSSR Engine ends abnormally.

User response

See message FABH0321E, and take an appropriate action.

FABH0326E BAD RETURN-CODE FROM BUFFER-HANDLER (KSDS)

Explanation

While reading a KSDS record, HSSR buffer handler encountered an I/O problem. More information will be found in the accompanying SYNADAF message buffer.

System action

HSSR Engine ends abnormally.

User response

See message FABH0321E, and take an appropriate action.

FABH0327E BAD RETURN-CODE FROM BUFFER-HANDLER (ESDS/OSAM)

Explanation

While reading an OSAM block or ESDS CI, HSSR buffer handler encountered an I/O problem. More information will be found in the accompanying SYNADAF message buffer.

System action

HSSR Engine ends abnormally.

User response

See message FABH0321E, and take an appropriate action.

FABH0329E REPLACE CALL WITHOUT PREVIOUS SUCCESSFUL GH CALL

Explanation

The application program issued an HSSR Replace call, but it had not previously issued a successful Get Hold call.

System action

HSSR Engine ends abnormally.

User response

Correct the program error.

FABH0330E INVALID CALL-FUNCTION

Explanation

The application program issued an HSSR call with a call function that was neither GU, GN, GNP, GHU, GHN, GHNP, REPL, or RBA.

System action

HSSR Engine ends abnormally.

User response

Correct the error.

FABH0331E INVALID PARM-NUMBER IN GN-CALL

Explanation

The application program issued an HSSR GN call with an incorrect number of parameters (too few or too many).

System action

HSSR Engine ends abnormally.

User response

Ensure that your application program issues HSSR GN calls with the correct number of parameters.

Assembler and COBOL programs should issue GN calls with the following parameters:

- Function
- PCB
- IOAREA
- SSA (optional)

PL/I programs should issue GN calls with the following parameters:

- Parameter Count
- Function
- PCB
- IOAREA
- SSA (optional)

FABH0332E SEGMENT-NAME IN SSA IS NOT ROOT-NAME

Explanation

The application program issued an HSSR GN call with an SSA; the segment name in the SSA is not the name of the root segment.

System action

HSSR Engine ends abnormally.

User response

Specify 'APISET 2' or 'APISET 3'. For details about the call types and command types that APISET supports, see “DL/I calls and EXEC DLI command for HSSR PCB” on page 84.

FABH0333E SEGMENT-NAME IN SSA NOT FOLLOWED BY A BLANK

Explanation

The application program issued an HSSR GN call with an SSA; the 8-byte segment name in the SSA is not followed by a blank.

System action

HSSR Engine ends abnormally.

User response

Specify 'APISET 2' or 'APISET 3' in the control statement. For details about the call types and command types that APISET supports, see “DL/I calls and EXEC DLI command for HSSR PCB” on page 84.

FABH0334E ROOT-SEGMENT IS NOT (DATA-) SENSITIVE

Explanation

The application program issued an HSSR GN call to retrieve the next database root segment; however, in the PSBGEN, the root segment is not defined as data-sensitive.

System action

HSSR Engine ends abnormally.

User response

Modify the PSB or modify the program.

FABH0335E INTERNAL ERROR OCCURRED IN HSSR CALL HANDLER

Explanation

HSSR call handler detected an unexpected error.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

FABH0336E NO ERROR TEXT AVAILABLE

Explanation

HSSR Engine detected an unexpected error.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

FABH0337E ADDRESS-LIST OF HSSR CALL-PARAMETERS IS INVALID

Explanation

The format of the call-parameter address list used in the HSSR call is incorrect.

System action

HSSR Engine ends abnormally.

User response

This problem can happen in the following cases:

- An Assembler application program provides an incorrect parameter address.
- The member of the call parameters is incorrect.

Ensure that Register 6 points to the address list of the call parameters. Correct the application program to provide appropriate parameter addresses.

FABH0338E INCORRECT SEGMENT NAME IN SSA

Explanation

The application program issued an HSSR GN call with an incorrect segment name in the SSA. The segment name must be the name of a sensitive segment.

System action

HSSR Engine ends abnormally.

User response

Correct the program error.

FABH0339E A DEPENDENT SEGMENT IS NOT (DATA-) SENSITIVE

Explanation

The application program issued an HSSR GN call to retrieve the next dependent segment; in the PSBGEN, however, the dependent segment is not defined as data-sensitive.

System action

HSSR Engine ends abnormally.

User response

Modify the PSB or modify the program.

FABH0340E EXECUTION IN AN ONLINE REGION IS NOT SUPPORTED

Explanation

An attempt was made to run IMS High Performance Unload in an online region. This is not supported.

System action

All HSSR calls fall back to DL/I calls. All HSSR PCBs fall back to DL/I PCBs and all calls issued against these PCBs are processed by DL/I action modules.

User response

Correct the region type in the PARM field of the EXEC statement.

FABH0341E PST LAYOUT HAS CHANGED

Explanation

During the initialization of HSSR Engine, an incorrect DL/I PST layout was detected.

System action

All HSSR calls fall back to DL/I calls. All HSSR PCBs fall back to DL/I PCBs, and all calls issued against these PCBs are processed by DL/I action modules.

User response

Check whether IMS High Performance Unload supports the active IMS release.

FABH0342E LANGUAGE OF PSB IS NOT SUPPORTED

Explanation

During the initialization of HSSR Engine, a PSB language flag defining a language other than Assembler, COBOL, or PL/I was detected.

System action

All HSSR calls fall back to DL/I calls. All HSSR PCBs and all calls are processed by DL/I action modules.

User response

Correct the PSB language statement and rerun the job, if necessary.

FABH0343E HSSR INITIALIZATION HAS INTERCEPTED A PGM CHECK

Explanation

During the initialization of HSSR Engine, an ESPIE EXIT routine intercepted a program check; the possible cause is an incorrect DL/I control block layout.

System action

All HSSR calls fall back to DL/I calls. All HSSR PCBs fall back to DL/I PCBs, and all calls issued against these PCBs are processed by DL/I action modules.

User response

If necessary, contact IBM Software Support.

FABH0344E LAYOUT OF PCPARMS AND/OR RCPARMS HAS CHANGED

Explanation

During the initialization of HSSR Engine, HSSR detected an unexpected layout of DL/I PCPARMS or RCPARMS control blocks.

System action

All HSSR calls fall back to DL/I calls. All HSSR PCBs fall back to DL/I PCBs, and all calls issued against these PCBs are processed by DL/I action modules.

User response

Check whether IMS High Performance Unload supports the active IMS release.

FABH0346E DB-ACCESS NOT AUTHORIZED BY DBRC

Explanation

This message is issued when IMS High Performance Unload is running in ULU region type and when DBRC does not authorize access to the database. Notice that in ULU region types, the requested authorization is for database level sharing with read-integrity.

System action

HSSR Engine ends abnormally.

User response

Resubmit the job at a time when DBRC can authorize access to the database. If the function of the job step is not an HD unload of the database, make sure that the program owner really wants to have the program executed in a ULU Region type.

FABH0347E INVALID PSB OR DBD CONTROL-BLOCKS FOR FABHFSU

Explanation

FABH010 detected a problem with a PSB or DBD control block during initialization.

System action

HSSR Engine ends abnormally.

User response

Ensure that the correct PSBs and DBDs are used.

FABH0348E INTERNAL ERROR OCCURRED IN HSSR CALL ANALYZER

Explanation

HSSR call analyzer detected an unexpected error.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

FABH0349E PSB HAS NO PCB'S

Explanation

The PSB provided has no PCBs. Therefore, the use of HSSR Engine is meaningless.

System action

HSSR Engine ends abnormally.

User response

Correct the utility control statement or the PSB.

FABH0350W HSSR CALLS FALL BACK TO DL/I FOR [NAMED DBD|ALL DBD'S] BECAUSE OF FOLLOWING PROBLEM IN DBD=*dbdname*

Explanation

During the building of control blocks for the *dbdname*, HSSR Engine detected an abnormal situation. This situation is described by the next message.

System action

All HSSR calls fall back to DL/I calls, either for the named DBD or for all DBDs. The corresponding PCBs fall back to DL/I PCBs. All calls issued against these PCBs are processed by DL/I action modules.

User response

If the fallback is done for all DBDs, the problem could be an unexpected layout of DL/I control blocks. See the accompanying messages issued by HSSR Engine.

FABH0351W SYSTEM PERFORMANCE WILL SLOW DOWN, PLEASE INFORM YOUR IMS SPECIALISTS

Explanation

This message is a warning message to the operator. Because of an abnormal situation described in a preceding message, HSSR calls fall back to DL/I. Calls issued by application programs are processed by DL/I action modules instead of HSSR Engine. This might lead to degraded system performance.

System action

The processing continues.

User response

Inform the IMS specialists.

FABH0352E THE FALL BACK IS ABORTED BECAUSE HSSRSNAP DD IS NOT SPECIFIED

Explanation

HSSR Engine detected an abnormal situation described in preceding messages, and attempted to issue a SNAP. However, the HSSRSNAP data set could not be opened, and the abnormal situation could not be documented with a SNAP.

System action

Instead of issuing a SNAP, HSSR Engine ends abnormally with dump.

User response

To prevent an abend and allow the fallback to DL/I, the HSSRSNAP DD statement must be included in the JCL. If the SNAP output is not required, this data set can be defined as dummy.

FABH0353E GET-BY-RBA CALL IS NOT SUPPORTED FOR THE PARTITIONED DB: DBD=*xxxxxxxx*

Explanation

A Get-by-RBA call is issued for a partitioned database. The Get-by-RBA call is not supported for the partitioned database.

System action

HSSR Engine ends abnormally.

User response

Modify your application program not to use the Get-by-RBA call, and rerun the job. See "Get-by-RBA calls" on page 257.

FABH0354E **REPL CALL IS NOT SUPPORTED FOR THE PARTITIONED DB: DBD=*xxxxxxxx***

Explanation

An HSSR REPL call is issued for a partitioned database. The HSSR REPL call is not supported for the partitioned database.

System action

HSSR Engine ends abnormally.

User response

Modify your application program not to use the HSSR REPL call, or replace the HSSR REPL calls with the DL/I REPL calls.

FABH0355E **DBRC REQUIRED FOR THIS EXECUTION**

Explanation

DBRC=N is specified for the job that runs in the ULU region and processes a HALDB.

System action

HSSR Engine ends abnormally.

User response

Specify DBRC=Y and rerun the job. If you want to process an IMS catalog database without using DBRC, see "Considerations for unloading an IMS catalog" on page 27.

FABH0356E **COMMAND CODE IS NOT SUPPORTED FOR HALDB DB: DBD=*dbdname***

Explanation

A command code was specified in a get call for the HALDB database *dbdname*. Command codes cannot be used in a get call for HALDB databases.

System action

HSSR Engine ends abnormally.

User response

Modify your application program so that no HSSR get call with a command code is to be used.

FABH0358E **REPL CALL IS ISSUED FOR THE FOLLOWING HSSR PCB:**

Explanation

This message is issued with another FABH0358E message. See the explanation in "FABH0358E" on page 432.

System action

See the system action section of the other FABH0358E message.

User response

See the user response section of the other FABH0358E message.

FABH0358E **[DBD=*xxxxxxxx* | DBD=*xxxxxxxx*(PARTITIONED)], PCB#=*yyyy*, PCBNAME=*zzzzzzzz***

Explanation

The application program issues an HSSR REPL call for the HSSR PCB shown in the message. The HSSR REPL call is not supported in the IPR Unload's API function. The string *xxxxxxxx* shows the name of the DBD that the PCB refers to; the number *yyyy* shows the PCB number in the PSB; and the string *zzzzzzzz* shows the label, if there is one, of the PCB assigned at PSBGEN. If the database is partitioned, the string (PARTITIONED) is printed after the DBD name.

System action

HSSR Engine ends abnormally.

User response

If the database is a partitioned database, you cannot issue any HSSR REPL calls for it. If it is not a partitioned database and you want to issue HSSR REPL

calls for it, consider running the application program by using the API of IMS High Performance Unload.

FABH0360E INVALID STATEMENT IN HSSROPT DD HAS BEEN IGNORED

Explanation

An HSSROPT control statement with an incorrect statement type was found. (HSSROPT control statements must begin in column 1 with a keyword defining the control statement.)

System action

The incorrect control statement is ignored.

User response

Correct the control statement.

FABH0361E INVALID KEYWORD ON TRHC STATEMENT WAS DETECTED

Explanation

An undefined keyword is specified in the HSSROPT 'TRHC' control statement. Only the following keywords are allowed: CALL, CB, CBX, BUF, BUFCB, and START=*nnn*.

System action

The incorrect keyword and the rest of the control statement are ignored.

User response

Correct the TRHC control statement.

FABH0362E START KEYWORD ON TRHC STATEMENT SHOULD BE FOLLOWED BY A COMMA OR BLANK

Explanation

An incorrect TRHC control statement is specified in the HSSROPT data set. The START=*nnn* keyword must be followed by either a comma or a blank.

System action

The START=*nnn* keyword and the rest of the control statement are ignored.

User response

Correct the TRHC control statement.

FABH0363E INVALID START KEYWORD ON TRHC STATEMENT WAS DETECTED

Explanation

An incorrect TRHC control statement is specified in the HSSROPT data set. The counter in the keyword START=*nnn* was either longer than 16 bytes or shorter than 1 byte.

System action

The START=*nnn* keyword and the rest of the control statement are ignored.

User response

Correct the TRHC control statement.

FABH0364E INVALID TRXC STATEMENT HAS BEEN IGNORED

Explanation

An incorrect TRXC control statement is specified in the HSSROPT data set. The following errors are possible:

- The number of core trace entries is not followed by a blank.
- The length of the field containing the number of entries is longer than 16 bytes.

System action

The TRXC control statement is ignored.

User response

Correct the TRXC control statement.

FABH0365E INVALID BUF STATEMENT HAS BEEN IGNORED

Explanation

An incorrect BUF control statement is specified in the HSSROPT data set. The following errors are possible:

- The 8-byte DBD name was not followed by a blank or a comma.
- The number of buffers was not followed by a blank.
- The length of the field containing the buffer number was either longer than 16 bytes or shorter than 1 byte.

System action

The BUF control statement is ignored.

User response

Correct the BUF control statement.

FABH0366E INVALID DBSTATS STATEMENT HAS BEEN IGNORED

Explanation

An incorrect DBSTATS control statement is specified in the HSSROPT data set.

System action

The incorrect DBSTATS control statement is ignored.

User response

Correct the DBSTATS control statement.

FABH0368E INVALID RETRY STATEMENT HAS BEEN IGNORED

Explanation

An incorrect RETRY control statement is specified in the HSSROPT data set. Bytes 7–10 must contain KSDS.

System action

The incorrect RETRY control statement is ignored.

User response

Correct the RETRY control statement.

FABH0369E INVALID KEYWORD ON GOTRETRY STATEMENT WAS DETECTED

Explanation

An undefined keyword in the GOTRETRY control statement is specified in the HSSROPT data set. Only the following keywords are permitted:

- NBR=
- WAIT=

System action

The incorrect keyword and the rest of the control statement are ignored.

User response

Correct the GOTRETRY control statement.

FABH0370E RETRY NUMBER EXCEEDS MAXIMUM VALUE

Explanation

An incorrect GOTRETRY control statement is specified in the HSSROPT data set. The number of times that HSSR Engine re-accesses the database is too high; it must not exceed 999.

System action

The incorrect NBR= keyword and the rest of the control statement are ignored.

User response

Correct the GOTRETRY control statement.

FABH0371E WAIT TIME EXCEEDS MAXIMUM VALUE

Explanation

An incorrect GOTRETRY control statement is specified in the HSSROPT data set. The number of seconds that HSSR Engine should wait before re-accessing the database is too high; it must not exceed 999 seconds.

System action

The incorrect WAIT= keyword and the rest of the control statement are ignored.

User response

Correct the GOTRETRY control statement.

FABH0372E NUMERIC FIELD SHOULD BE FOLLOWED BY A COMMA OR BLANK

Explanation

An incorrect control statement is specified in the HSSROPT data set.

System action

The incorrect numeric field and the rest of the control statement are ignored.

User response

Correct the control statement in the HSSROPT data set.

FABH0373E NUMERIC FIELD IS TOO LONG

Explanation

An incorrect control statement is specified in the HSSROPT data set.

System action

The incorrect numeric field and the rest of the control statement are ignored.

User response

Correct the control statement in the HSSROPT data set.

FABH0374E INVALID SKERROR STATEMENT HAS BEEN IGNORED

Explanation

An incorrect SKERROR control statement is specified in the HSSROPT data set.

System action

The incorrect statement is ignored.

User response

Correct the SKERROR statement.

FABH0375E INVALID KEYCHECK STATEMENT HAS BEEN IGNORED

Explanation

An incorrect KEYCHECK control statement is specified in the HSSROPT data set.

System action

The incorrect statement is ignored.

User response

Correct the KEYCHECK statement.

FABH0376E INVALID KEYWORD ON DIAGG STATEMENT WAS DETECTED

Explanation

An undefined keyword on the DIAGG control statement is specified in the HSSROPT data set.

System action

The incorrect keyword and the rest of the control statement are ignored.

User response

Correct the DIAGG statement.

FABH0377E INVALID CABSTAT CONTROL STATEMENT IS SPECIFIED

Explanation

An incorrect operand on the CABSTAT control statement is specified in the HSSROPT data set. The operand must be either YES or NO.

System action

The incorrect statement is ignored.

User response

Correct the CABSTAT statement.

FABH0378W DATXEXIT YES IS IGNORED BECAUSE OF IMS LEVEL

Explanation

DATXEXIT YES is specified in the HSSROPT data set, but HSSR Engine is running in an IMS environment, where the Data Conversion exit is not supported.

System action

The DATXEXIT YES option is ignored, and processing continues with DATXEXIT NO.

User response

Ensure that an appropriate IMS load module library is specified for your job. If you do not need to use a Data Conversion exit routine, remove the DATXEXIT YES statement.

FABH0379E AN INVALID OPERAND IS SPECIFIED FOR THE PARTINFO CONTROL STATEMENT

Explanation

Either an invalid parameter is specified or no parameter is specified for the PARTINFO statement in HSSROPT DD. For details of the PARTINFO control statement, refer to “PARTINFO control statement” on page 173.

System action

The statement is ignored and the processing continues.

User response

Correct the error and, if necessary, rerun the job.

FABH0380W BUFFER TRACE WILL NOT BE TAKEN FOR HALDB: *dbdname*

Explanation

The buffer trace for the HALDB whose DBD name is indicated in the message will not be taken.

System action

HSSR Engine continues processing.

User response

If you want to suppress the message, remove the BUTR control statement from the HSSROPT data set.

FABH0381E DBD IS NOT A PHYSICAL DBD

Explanation

Because of PSBGEN specifications, HSSR Engine tried to build an HSSR PCB referring to a logical DBD. Logical databases are not supported by HSSR Engine.

System action

HSSR Engine issues a SNAP for this PCB and falls back to DL/I. The PCB is defined as a DL/I PCB, and all calls issued against it are processed by DL/I action modules.

User response

Change the PCB as follows: specify the name of the physical DBD, or do not specify the PCB as an HSSR PCB.

FABH0382E SEGMENT CODE IN SDB'S ARE NOT ASCENDING

Explanation

One of the following two problems occurred:

1. During PSBGEN, the SENSEG statements were not coded in the same hierarchical sequence as in the DBD.
2. HSSR Engine detected an incorrect or unexpected DL/I control block layout.

System action

HSSR Engine issues a SNAP and falls back to DL/I for all PCBs. All PCBs are defined as DL/I PCBs and all calls are processed by DL/I action modules.

User response

Verify that the sequence of the SENSEG statements in the PSBGEN is the same as in the DBD; if not, correct the sequence of the SENSEG statements and perform the PSBGEN again.

FABH0383E SEGMENTS ARE NOT ALL IN SAME DATABASE

Explanation

Because of PSBGEN specifications, HSSR Engine tried to build an HSSR PCB. However, the sensitive segments are not all in the same physical database. HSSR Engine does not support sensitive segments in multiple databases.

System action

HSSR Engine issues a SNAP and falls back to DL/I for all PCBs. All PCBs are defined as DL/I PCBs and all calls are processed by DL/I action modules.

User response

Change PSBGEN as follows: remove the problematic SENSEG statements causing the problem, or do not specify the PCB as an HSSR PCB.

FABH0384I PROCSEQ=*indexdbd* IS SPECIFIED IN PCB#=*nnn*

Explanation

The secondary index *indexdbd* is specified by the PROCSEQ= parameter in the PCB. *nnn* shows the PCB number. HSSR Engine uses the secondary index to retrieve the root segments.

System action

HSSR Engine continues the processing.

User response

None. This message is informational.

FABH0386E MORE THAN 10 DATA SET GROUPS NOT SUPPORTED

Explanation

Neither IMS nor HSSR Engine supports HD databases with more than 10 data set groups.

System action

HSSR Engine issues a SNAP for this PCB and falls back to DL/I. The PCB is defined as a DL/I PCB, and all calls issued against it are processed by DL/I action modules.

User response

Change the PCB as follows: specify the name of the physical DBD, or do not specify the PCB as an HSSR PCB.

FABH0387E DMB POINTERS TO PSDB'S OR AMP NOT OK

Explanation

One of the following two problems occurred:

1. During PSBGEN, the SENSEG statements were not coded in the same hierarchical sequence as in the DBD.
2. HSSR Engine detected an incorrect or unexpected DL/I control block layout.

System action

HSSR Engine issues a SNAP and falls back to DL/I for all PCBs. All PCBs are defined as DL/I PCBs and all calls are processed by DL/I action modules.

User response

Verify that the sequence of the SENSEG statements in the PSBGEN is the same as in the DBD; if not, correct the sequence of the SENSEG statements and perform the PSBGEN again.

FABH0388E UNSUPPORTED DBD: IMS/ESA PARTITION SUPPORT PRODUCT

Explanation

Because of PSBGEN specifications, an HSSR PCB was tried to be built. However, the IMS/ESA Partition Support Product is not supported by HSSR Engine.

System action

HSSR Engine issues a SNAP for this PCB and falls back to DL/I. The PCB is defined as a DL/I PCB, and all calls issued against it are processed by DL/I action modules.

User response

Modify the database organization, or do not specify the PCB as an HSSR PCB.

FABH0389E MORE SDB'S THAN PSDB'S

Explanation

One of the following two problems occurred:

1. During PSBGEN, the SENSEG statements were not coded in the same hierarchical sequence as in the DBD.
2. HSSR Engine detected an incorrect or unexpected DL/I control block layout.

System action

HSSR Engine issues a SNAP and falls back to DL/I for all PCBs. All PCBs are defined as DL/I PCBs and all calls are processed by DL/I action modules.

User response

Verify that the sequence of the SENSEG statements in the PSBGEN is the same as in the DBD; if not, correct the sequence of the SENSEG statements and perform the PSBGEN again.

FABH0390E LENGTH OF PSDB'S NOT OK

Explanation

One of the following two problems occurred:

1. During PSBGEN, the SENSEG statements were not coded in the same hierarchical sequence as in the DBD.
2. HSSR Engine detected an incorrect or unexpected DL/I control block layout.

System action

HSSR Engine issues a SNAP and falls back to DL/I for all PCBs. All PCBs are defined as DL/I PCBs and all calls are processed by DL/I action modules.

User response

Verify that the sequence of the SENSEG statements in the PSBGEN is the same as in the DBD; if not, correct the sequence of the SENSEG statements and perform the PSBGEN again.

FABH0391E SEGMENT CODE OF PSDB'S NOT ASCENDING

Explanation

One of the following two problems occurred:

1. During PSBGEN, the SENSEG statements were not coded in the same hierarchical sequence as in the DBD.
2. HSSR Engine detected an incorrect or unexpected DL/I control block layout.

System action

HSSR Engine issues a SNAP and falls back to DL/I for all PCBs. All PCBs are defined as DL/I PCBs and all calls are processed by DL/I action modules.

User response

Verify that the sequence of the SENSEG statements in the PSBGEN is the same as in the DBD; if not, correct the sequence of the SENSEG statements, and perform the PSBGEN again.

FABH0393E SDBPSDB DOES NOT POINT TO PSDB

Explanation

One of the following two problems occurred:

1. During PSBGEN, the SENSEG statements were not coded in the same hierarchical sequence as in the DBD.
2. HSSR Engine detected an incorrect or unexpected DL/I control block layout.

System action

HSSR Engine issues a SNAP and falls back to DL/I for all PCBs. All PCBs are defined as DL/I PCBs and all calls are processed by DL/I action modules.

User response

Verify that the sequence of the SENSEG statements in the PSBGEN is the same as in the DBD; if not, correct the sequence of the SENSEG statements, and perform the PSBGEN again.

FABH0394E NUMBER OF DSG'S CALCULATED FROM MASTER DMB IS UNEQUAL TO THE NUMBER CALCULATED FROM DMB FOR PARTITION *ppppppp*

Explanation

The number of DSGROUPs calculated from the master DMB of the HALDB for which the partition *ppppppp* is defined is not equal to the number of DGGROUPs calculated from the partition DMB of the partition.

System action

HSSR Engine issues a SNAP for this PCB and falls back to DL/I. The PCB is defined as a DL/I PCB, and all calls issued against it are processed by DL/I action modules.

User response

To identify the cause, ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

FABH0395E PREFIX SIZE IN PSDB NOT OK

Explanation

One of the following two problems occurred:

1. During PSBGEN, the SENSEG statements were not coded in the same hierarchical sequence as in the DBD.
2. HSSR Engine detected an incorrect or unexpected DL/I control block layout.

System action

HSSR Engine issues a SNAP and falls back to DL/I for all PCBs. All PCBs are defined as DL/I PCBs and all calls are processed by DL/I action modules.

User response

Verify that the sequence of the SENSEG statements in the PSBGEN is the same as in the DBD; if not, correct the sequence of the SENSEG statements, and perform the PSBGEN again.

FABH0396E SENSITIVE VIRTUAL LOGICAL CHILD (*segmname*) IS FOUND IN PCB#=*nnnn*

Explanation

A sensitive virtual logical child segment *segmname* is found in the HSSR PCB. *nnnn* shows the PCB number. HSSR Engine does not support sensitive virtual logical child segment.

System action

HSSR Engine issues a SNAP and falls back to DL/I for all PCBs. All PCBs are defined as DL/I PCBs and all calls are processed by DL/I action modules.

User response

If you do not need to retrieve the virtual logical child segment, specify the SKIPVLC YES option in the HSSROPT data set.

FABH0397E UNSUPPORTED DATABASE ORG

Explanation

Because of PSBGEN specifications, an HSSR PCB was tried to be built. However, the organization of the referred-to DBD is not supported by HSSR Engine.

System action

HSSR Engine issues a SNAP for this PCB and falls back to DL/I. The PCB is defined as a DL/I PCB, and all calls issued against it are processed by DL/I action modules.

User response

Modify the database organization, or do not specify the PCB as an HSSR PCB.

FABH0398E INDEX DMB NOT FOUND

Explanation

Because of PSBGEN specifications, an HSSR PCB referring to a logical DBD was tried to be built. Logical databases are not supported by HSSR Engine.

System action

HSSR Engine issues a SNAP for this PCB and falls back to DL/I. The PCB is defined as a DL/I PCB, and all calls issued against it are processed by DL/I action modules.

User response

Change the PCB as follows: specify the name of the physical DBD, or do not specify the PCB as an HSSR PCB.

FABH0399I HALDB *dbdname* IS DEFINED AS OSAM8G

Explanation

The indicated HALDB is defined as OSAM8G in the RECON data set and the maximum capacity of OSAM data sets is 8 GB of data.

System action

HSSR Engine continues processing.

User response

None. This message is informational.

FABH0400I DB HAS VIRTUAL LOGICAL CHILD SEGMENT (*segmname*)

Explanation

HSSR Engine ignores the virtual logical child segment *segmname* that is specified in the HSSR PCB. In the DLI or the DBB region, the 'SKIPVLC YES' option is used. In the ULU region, it is always ignored.

However, in the case of the migration unload to HALDB it is not applicable. The virtual logical child segment will be retrieved.

System action

HSSR Engine continues the processing.

User response

If you need the virtual logical child to be retrieved in a user application program, specify the SKIPVLC NO control statement. The PCB will be passed to IMS DL/I and the virtual logical child segment will be retrieved.

FABH0403E SEC LIST FOR INDEXED-SEGM NOT FOUND

Explanation

One of the following two problems occurred:

1. During PSBGEN, the SENSEG statements were not coded in the same hierarchical sequence as in the DBD.
2. HSSR Engine detected an incorrect or unexpected DL/I control block layout.

System action

HSSR Engine issues a SNAP and falls back to DL/I for all PCBs. All PCBs are defined as DL/I PCBs and all calls are processed by DL/I action modules.

User response

Verify that the sequence of the SENSEG statements in the PSBGEN is the same as in the DBD; if not, correct the sequence of the SENSEG statements and perform the PSBGEN again.

FABH0404E PCB WITH FIELD SENSITIVITY NOT SUPPORTED

Explanation

Because of PSBGEN specifications, an HSSR PCB for which field sensitivity has been specified was tried to be built. Field sensitivity is not supported by HSSR Engine.

System action

HSSR Engine issues a SNAP for this PCB and falls back to DL/I. The PCB is defined as a DL/I PCB, and all calls issued against it are processed by DL/I action modules.

User response

Either remove field sensitivity specifications for this PCB, or do not specify the PCB as an HSSR PCB.

FABH0405E INDEX DMB *dbdname* IS NOT FOUND

Explanation

The DMB for the primary or secondary index DBD *dbdname* cannot be found in the DMB directories.

System action

HSSR Engine ends abnormally.

User response

Check whether the specified index DBD has the index relationships with the database to be unloaded.

FABH0406E INDEX DBD *dbdname* IS NOT SUPPORTED

Explanation

The specified secondary index *dbdname* is not supported due to one of the following reasons:

- The DB organization is not INDEX.
- The target DB organization is not HIDAM or HDAM.
- The target segment is not the root segment.
- The key field is not defined as unique.
- The secondary index uses symbolic pointing.

System action

HSSR Engine ends abnormally.

User response

Change or remove the index DBD name.

FABH0407E DMB *dbdname* IS NOT CORRECT

Explanation

HSSR Engine detected an incorrect or unexpected DL/I control block layout in the dbdname DMB.

System action

HSSR Engine ends abnormally.

User response

Contact IBM Software Support.

FABH0411E SEGMENT CODE IN SDB AND HSDB DISAGREE

Explanation

One of the following two problems occurred:

1. During PSBGEN, the SENSEG statements were not coded in the same hierarchical sequence as in the DBD.
2. HSSR Engine detected an incorrect or unexpected DL/I control block layout.

System action

HSSR Engine issues a SNAP and falls back to DL/I for all PCBs. All PCBs are defined as DL/I PCBs and all calls are processed by DL/I action modules.

User response

Verify that the sequence of the SENSEG statements in the PSBGEN is in the same as the DBD; if not, correct the sequence of the SENSEG statements, and perform the PSBGEN again.

FABH0412E VIRTUAL PAIRED SEGMENTS NOT SUPPORTED

Explanation

Because of PSBGEN specifications, an HSSR PCB containing a sensitive logical child that is virtually paired. Virtually paired segments are not supported by HSSR Engine.

System action

HSSR Engine issues a SNAP for this PCB and falls back to DL/I. The PCB is defined as a DL/I PCB, and all calls issued against it are processed by DL/I action modules.

User response

Modify the virtually paired segment to, for example, a physically paired segment, or remove the SENSEG statement of the virtually paired segment from the PCB, or do not specify the PCB as an HSSR PCB.

FABH0413E INVALID SEGMENT SENSITIVITY FOR AN HSSR PCB

Explanation

Because of PSBGEN specifications, an HSSR PCB was tried to be built. This PCB contains a SENSEG statement whose PROCOPT is neither G nor K.

System action

HSSR Engine issues a SNAP for this PCB and falls back to DL/I. The PCB is defined as a DL/I PCB, and all calls issued against it are processed by DL/I action modules.

User response

Modify the PROCOPT field of the SENSEG statement, or do not specify the PCB as an HSSR PCB.

FABH0415E SOMETHING WRONG WITH NUMBER OF HPTRS

Explanation

HSSR Engine has problems with its own control blocks.

System action

HSSR Engine issues a SNAP and falls back to DL/I for all PCBs. All PCBs are defined as DL/I PCBs and all calls are processed by DL/I action modules.

User response

Verify that the sequence of the SENSEG statements in the PSBGEN is the same as in the DBD; if not, correct the sequence of the SENSEG statements, and perform the PSBGEN again.

FABH0416E SOMETHING WRONG WITH NUMBER OR LENGTH OF HSDB

Explanation

HSSR Engine has problems with its own control blocks.

System action

HSSR Engine issues a SNAP and falls back to DL/I for all PCBs. All PCBs are defined as DL/I PCBs and all calls are processed by DL/I action modules.

User response

Verify that the sequence of the SENSEG statements in the PSBGEN is the same as in the DBD; if not, correct the sequence of the SENSEG statements and perform the PSBGEN again.

FABH0420W WARNING: SEGMENT *segname* HAS A VIRTUAL LOGICAL PARENT KEY

Explanation

The logical parent's concatenated key (LPCK) of the segment *segname* is defined as virtual, but the BLDLPCK control statement is not specified.

Note: During retrieval operations, the part of the I/O area that should contain the logical parent key will contain blanks.

System action

HSSR Engine continues processing.

User response

If the control statements for the pre-reorganization utility specified DBIL for the logical child database, and the database is unloaded by FABHURG1 or FABHFSU, you must specify the BLDLPCK control statement. In other cases, you do not need to specify BLDLPCK. If you need LPCKs to be built, specify the BLDLPCK control statement and rerun the job.

FABH0421E REPLACE PROCOPT NOT SUPPORTED FOR DB-ORGANIZATION OF DB=*dbdname*

Explanation

HSSR Engine supports the replace processing option only for HIDAM and HDAM databases. The database (named *dbdname*) has another organization.

System action

HSSR Engine issues a SNAP for this PCB and falls back to DL/I. The PCB is defined as a DL/I PCB, and all calls issued against it are processed by DL/I action modules.

User response

Modify the PSB and do not define the PCB as an HSSR PCB.

FABH0422E REPLACE PROCOPT NOT SUPPORTED FOR BLOCK-LEVEL SHARED DB=*dbdname*

Explanation

HSSR Engine does not support the replace processing option for databases (named *dbdname*) that are shared at the block level.

System action

HSSR Engine issues a SNAP for this PCB and falls back to DL/I. The PCB is defined as a DL/I PCB, and all calls issued against it are processed by DL/I action modules.

User response

Investigate to specify IRLM=N on the JCL procedure in order to run in a database-level sharing environment instead of a block-level sharing environment. Otherwise, modify the PSB, and do not define the PCB as an HSSR PCB.

FABH0423E REPLACE PROCOPT NOT SUPPORTED FOR PHYSICAL PAIRED SEGMENT=*segname*

Explanation

HSSR Engine does not support the replace processing option for segment types (named *segname*) that are physically paired logical children.

System action

HSSR Engine issues a SNAP for this PCB and falls back to DL/I. The PCB is defined as a DL/I PCB, and all calls issued against it are processed by DL/I action modules.

User response

Modify the PSB. Either specify PROCOPT=G on the SENSEG of the named segment type or do not define the PCB as an HSSR PCB.

FABH0424E REPLACE PROCOPT NOT SUPPORTED FOR INDEX-SOURCE SEGMENT-TYPE=*segname*

Explanation

HSSR Engine does not support the replace processing option for segment types (named *segname*) that are index-source segments and not root segments.

System action

HSSR Engine issues a SNAP for this PCB and falls back to DL/I. The PCB is defined as a DL/I PCB, and all calls issued against it are processed by DL/I action modules.

User response

Verify that the sequence of the SENSEG statements in the PSBGEN is the same as in the DBD; if not, correct the sequence of the SENSEG statements and perform the PSBGEN again.

FABH0425E REPLACE PROCOPT NOT SUPPORTED WITH VIRTUAL LP-KEY, SEGMENT-TYPE=*segname*

Explanation

Because the BLDLPCK control statement is not specified, HSSR Engine cannot perform the replace processing for the segment *segname*, which is a logical child whose logical parent's concatenated key (LPCK) is defined as virtual.

System action

HSSR Engine issues a SNAP for this PCB and falls back to DL/I. The PCB is defined as a DL/I PCB, and all calls issued against it are processed by DL/I action modules.

User response

If you need REPLACE processing for the segment type, specify the BLDLPCK control statement. If you do not need REPLACE processing, modify the PSB; either specify PROCOPT=G on the SENSEG of the named segment type, or do not define the PCB as an HSSR PCB.

FABH0426E OTHER PCB'S MAY NOT HAVE AN UPDATE PROCESSING OPTION, SEGMENT-TYPE=*segname*

Explanation

The segment type (named *segname*) has a replace processing option in an HSSR PCB. This allows update processing options in another HSSR PCB or DL/I PCB of the PSB.

System action

HSSR Engine issues a SNAP for this PCB and falls back to DL/I. The PCB is defined as a DL/I PCB, and all calls issued against it are processed by DL/I action modules.

User response

Modify the PSB in order to observe the restrictions.

FABH0430W DB WILL BE READ BY HSSR WITHOUT READ-INTEGRITY

Explanation

The application program is reading a database that is shared at the block level. The PSBGEN processing option specifies an access intent (read or update access intent) requesting full read-integrity. However, HSSR Engine does not have an interface to the IRLM to provide full read-integrity.

System action

HSSR Engine continues processing and reads the database without read-integrity. The database is processed with read-only processing intent. Unpredictable results might occur.

User response

If the application program requires read-integrity, the database must be shared at the database level (instead of the block level). This will prevent concurrent execution with an updating IMS subsystem.

FABH0433W RBN VALUE IS LARGER THAN THE SIZE OF THE PARTITION DATA SET (DDNAME: *ddname*)

Explanation

HSSR Engine detected that the RBN value defined in the DBD for the partition data set was larger than the real data set size. *ddname* indicates the DD name of the partition data set.

System action

HSSR Engine continues processing.

User response

Ensure that the correct DBD is specified. Specify the correct DBD, and rerun the job.

FABH0441E STATISTIC PRINTER GOT INVALID CALL

Explanation

HSSR Engine detected an unexpected error.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

FABH0451E SKIP-COUNT EXHAUSTED

Explanation

HSSR Engine detected more GG status code situations than the number that was specified on the HSSROPT SKERROR control statement.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

FABH0452W HSSR CALL HANDLER RETURNS FIRST GG STATUS CODE

Explanation

HSSR call handler returned a GG status code to the calling application program or utility program. This is a warning.

System action

The processing continues.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

FABH0453E UNEXPECTED ENTRY INTO FABH080

Explanation

HSSR Engine detected an unexpected error.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

FABH0457E HSSR DETECTED AN UNEXPECTED ERROR SITUATION

Explanation

HSSR Engine detected an unexpected error.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

FABH0461E COMPARE OPTION HAS DETECTED AN INEQUALITY BETWEEN HSSR AND DL1

Explanation

This message is issued when the CO control statement is specified in the HSSROPT data set, and the results of an HSSR call and the corresponding DL/I call are not the same.

System action

HSSR Engine ends abnormally.

User response

If PROCOPT=R is specified for a VSAM database, ensure that VSAM SHAREOPTIONS are defined (2,3) or (3,3).

Complete the following tasks to identify the cause, and take appropriate actions:

- Activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378).
- Ensure that the correct DBDs, PSBs, and ACBs were being used.

FABH0462E THE PARTITIONS SELECTED BY HSSR AND DL/I ARE UNEQUAL; HSSR-PID=*xxxx* (NAME: *xxxxxxx*), DL/I-PID=*xxxx* (NAME: *xxxxxxx*)

Explanation

This message is issued when the CO control statement is specified in the HSSROPT data set, and HSSR call handler detected a difference between the partition IDs selected by HSSR call handler and DL/I (IMS) in an HSSR call.

System action

HSSR Engine ends abnormally.

User response

Check the HSSR Engine trace report to see the difference; and contact IBM Software Support.

FABH0463E HSSR REACHED THE END OF THE PARTITION *ppppppp* EARLIER THAN DL/I

Explanation

This message is issued when the CO control statement is specified in the HSSROPT data set, and HSSR call handler reaches the end of the partition *ppppppp* earlier than DL/I detects it.

System action

HSSR Engine ends abnormally.

User response

Contact IBM Software Support.

FABH0465E AN UNSUPPORTED HSSR CALL WAS ISSUED FOR NON-HD DATABASE *xxxxxxxx*

Explanation

An unsupported HSSR call was issued for the non-HD database *xxxxxxxx*. Only the call types defined by APISET 1 are supported for non-HD databases. Any API set other than APISET 1 can be specified in HSSROPT or in the site default option table, but it is ignored for non-HD databases.

System action

HSSR Engine ends abnormally.

User response

If you want to use a call that is not supported by HSSR Engine, modify the application program, or use the DBDL1 control statement.

FABH0471E OPEN OF DDNAME=HSSRBUTR HAS FAILED

Explanation

An HSSROPT BUTR control statement instructed HSSR Engine to activate the machine-readable trace of buffer handler activities. This trace is written on the HSSRBUTR data set. HSSR Engine cannot open this file.

System action

HSSR Engine ends abnormally.

User response

Check whether MVS or SAM issued other error messages. Correct the problem.

FABH0473E INTERNAL ERROR OCCURRED IN HSSR CALL HANDLER

Explanation

HSSR call handler detected an unexpected error.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause of the database error:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

If necessary, contact IBM Software Support.

FABH0475E INTERNAL ERROR OCCURRED IN HSSR CALL HANDLER

Explanation

HSSR call handler detected an unexpected error.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause of the database error:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

If necessary, contact IBM Software Support.

FABH0477E INTERNAL ERROR OCCURRED IN HSSR CALL HANDLER

Explanation

HSSR call handler detected an unexpected error.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause of the database error:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

If necessary, contact IBM Software Support.

FABH0479E INTERNAL ERROR OCCURRED IN HSSR BUFFER HANDLER

Explanation

HSSR buffer handler detected an unexpected error.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause of the database error:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

If necessary, contact IBM Software Support.

FABH0480E INVALID DATABASE ORGANIZATION

Explanation

HSSR Engine detected a request for a GU call for a database whose organization is not supported by HSSR.

System action

HSSR Engine ends abnormally.

User response

See "Database organizations supported" on page 10 for a list of supported database organizations. Correct the application program and DBD.

FABH0481E INVALID PARAMETER NUMBER IN CALL

Explanation

The application program issued an HSSR call with an incorrect number of parameters (too few or too many). The HSSR call handler does not support three or more SSAs. The unsupported call is printed in the Trace Output report in the HSSRTRAC data set.

System action

If APISET is 2, HSSR Engine ends abnormally. If APISET is 3, the call and all the succeeding calls to the HSSR PCB are passed to the IMS DL/I call handler to continue the processing instead of ending it abnormally.

User response

If APISET is 2, specify 'APISET 3' in the control statement. If APISET is 3, ignore this message.

The following parameters are required for Assembler and COBOL programs:

- Call function
- PCB
- I/O AREA
- SSA (optional)

The following parameters are required for PL/I programs:

- PARM count
- Call function
- PCB
- I/O AREA
- SSA (optional)

FABH0482E INCORRECT CALL FUNCTION

Explanation

The application program issued an HSSR call with an incorrect call function. Only GU and GN are supported.

System action

HSSR Engine ends abnormally.

User response

Correct the program error.

FABH0483E INCORRECT SEGMENT NAME IN SSA

Explanation

The application program issued an HSSR GU call with an incorrect segment name in the SSA. The segment name must be the name of the root segment. (For RBA calls, the segment name must be the name of a sensitive segment.)

System action

HSSR Engine ends abnormally.

User response

Correct the program error. If there is no error, check whether the correct APISET statement is specified. For details about the call types and command types that APISET supports, see "DL/I calls and EXEC DLI command for HSSR PCB" on page 84.

FABH0484E INCORRECT FIELD NAME IN SSA

Explanation

The application program issued an HSSR GU call with an incorrect field name in the SSA. The field name must be the name of the key field of the root.

System action

HSSR Engine ends abnormally.

User response

Correct the program error. If there is no error, check whether the correct APISET statement is specified. For details about the call types and command types that APISET supports, see "DL/I calls and EXEC DLI command for HSSR PCB" on page 84.

FABH0485E LEFT PARENTHESIS IN SSA IS MISSING OR AT WRONG POSITION

Explanation

The application program issued an HSSR GU call with an incorrect SSA. The 8-byte segment name must be immediately followed by a (or by ⋆T). GUs with unqualified SSAs are not supported by HSSR call handler; the only supported command code is T.

System action

HSSR Engine ends abnormally.

User response

Correct the program error. If there is no error, check whether the correct APISET statement is specified. For details about the call types and command types that APISET supports, see "DL/I calls and EXEC DLI command for HSSR PCB" on page 84.

FABH0486E RELATIONAL OPERATOR IN SSA IS INCORRECT

Explanation

The application program issued an HSSR GU call with an incorrect relational operator in the SSA. Only the following relational operators are supported:

- *b*=
- =*b*
- EQ
- =>
- >=
- GE

where *b* represents a required blank.

System action

HSSR Engine ends abnormally.

User response

Correct the program error. If there is no error, check whether the correct APISET statement is specified. For details about the call types and command types that APISET supports, see "DL/I calls and EXEC DLI command for HSSR PCB" on page 84.

FABH0487E RIGHT PARENTHESIS IN SSA IS MISSING OR AT WRONG POSITION

Explanation

The application program issued an HSSR GU call with an incorrect SSA. The key value field or the RBA must be immediately followed by the right parenthesis. (Boolean operators are not supported by HSSR Engine.)

System action

HSSR Engine ends abnormally.

User response

Correct the error. If there is no error, check whether the correct APISET statement is specified. For details about the call types and command types that APISET supports, see "DL/I calls and EXEC DLI command for HSSR PCB" on page 84.

FABH0488E DB ORG IS NOT HIDAM OR HDAM

Explanation

The application program issued an RBA call to a database that is neither HIDAM nor HDAM. RBA calls can be issued only against HIDAM or HDAM databases.

System action

HSSR Engine ends abnormally.

User response

Correct the program error.

FABH0489E UNSUPPORTED COMMAND CODE

Explanation

The application program issued a call with a command code that is not supported by HSSR call handler. HSSR call handler supports only the "T" and "NULL" command codes.

System action

HSSR Engine ends abnormally.

User response

Specify 'APISET 3'. For details about the call types and command types that APISET supports, see "DL/I calls and EXEC DLI command for HSSR PCB" on page 84.

FABH0490E SEGMENT CODE OF RETRIEVED RBA IS NOT THE SEGMENT CODE OF REQUESTED SEGMENT

Explanation

The application program issued an RBA call. The SSA specified a segment name and an RBA. The database does not contain a segment of the specified segment type at the specified RBA.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause (this error might be a program error), and take appropriate actions:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.
- Activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378).

FABH0491E REQUESTED RBA NOT IN DATA SET

Explanation

The application program issued an RBA call. The SSA specified an RBA. However, the specified RBA is not within the extents of the data set.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause (this error might be a program error), and take appropriate actions:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.
- Activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378).

FABH0492E ROOT HAS NO SEQUENCE FIELD

Explanation

The application program issued an HSSR GU call with a qualified SSA for the root segment. HSSR call handler supports only qualification on the sequence field and assumes that the SSA is qualified on the Root Sequence Field. However, during DBDGEN, the root is defined without sequence field.

System action

HSSR Engine ends abnormally.

User response

Either remove the GU call from the program, or have the database administrator define a sequence field for the root segment. If there is no error, check whether the correct APISET statement is specified. For details about the call types and command types that APISET supports, see "DL/I calls and EXEC DLI command for HSSR PCB" on page 84.

FABH0499E INTERNAL ERROR OCCURRED IN HSSR CALL HANDLER

Explanation

HSSR call handler detected an unexpected error.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

If necessary, contact IBM Software Support.

FABH0501E DATA SET GROUP NUMBER NOT POSITIVE

Explanation

The buffer handler was invoked with an incorrect data set group number. The data set group number must have been positive.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, ensure that the correct DBDs, PSBs, and ACBs were being used. If necessary, contact IBM Software Support.

FABH0502E DATA SET GROUP NUMBER DOES NOT EXIST

Explanation

The buffer handler was invoked with an incorrect data set group number. The specified data set group number does not exist.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, ensure that the correct DBDs, PSBs, and ACBs were being used. If necessary, contact IBM Software Support.

FABH0503E PCB IS NOT SENSITIVE TO ANY SEGMENT OF SPECIFIED DATA SET GROUP

Explanation

The buffer handler is invoked with an incorrect data set group number. No segment of the specified data set group is sensitive.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, ensure that the correct DBDs, PSBs, and ACBs were being used. If necessary, contact IBM Software Support.

FABH0504E NO OSAM/ESDS IN DATABASE

Explanation

The buffer handler was invoked to handle a data set that was not OSAM or ESDS.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, ensure that the correct DBDs, PSBs, and ACBs were being used. If necessary, contact IBM Software Support.

FABH0505E RBA NOT WITHIN SPECIFIED DATA SET

Explanation

The buffer handler was invoked with an incorrect RBA. The RBA is not within the extents of the data set.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause of the database error:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

If necessary, contact IBM Software Support.

FABH0506E BUFFER HANDLER IO-ERROR

Explanation

The buffer handler has encountered an I/O error. Other error messages will describe the problem in detail.

System action

HSSR Engine ends abnormally.

User response

See the system error messages.

FABH0507E INVALID SUBTYPE OF *Z COMMAND CODE

Explanation

The internal HSSR call with the *Z command code has an incorrect format. HSSR Engine generates this undocumented call to start retrieval from an HDAM database at a specific relative block number. For example, the FABHFSU BLM control statement might specify retrieval by RBN causing HSSR Engine to generate this command.

System action

HSSR Engine ends abnormally.

User response

Contact IBM Software Support.

FABH0511E POINTER IN INDEX TO ROOT-SEGMENT IS ZERO

Explanation

During a call against an HIDAM database, HSSR Engine must retrieve a primary index record in order to locate the root segment. The index pointer record contained zero instead of a root pointer.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see “FABHTEST utility for problem determination” on page 378). If necessary, contact IBM Software Support.

FABH0512E INVALID RETURN-CODE FROM BUFFER-HDLR

Explanation

HSSR Engine encountered an I/O problem. For more information, see the accompanying SYNADAF message buffer.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see “FABHTEST utility for problem determination” on page 378). If necessary, contact IBM Software Support.

FABH0513E SEGMENT-CODE NOT 01

Explanation

HSSR call handler retrieved a root segment whose segment code was not 01.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see “FABHTEST utility for problem determination” on page 378). If necessary, contact IBM Software Support.

FABH0515E CANNOT POSITION THE FIRST SEGMENT IN THE DB

Explanation

During unloading an HIDAM or PHIDAM DB, HSSR Engine retrieved a primary index record in order to locate the first database position. In this process, HSSR Engine detected an incorrect situation where the delete byte in the index record was not X' 5A' nor X'EA' nor X'F2' although the delete byte in the corresponding database record was X'5A', X'EA', or X' F2'.

System action

HSSR Engine ends abnormally.

User response

Check the delete byte of the first root segment and correct it.

FABH0516E LAST SEGM IN PRIMARY INDEX DOES NOT POINT TO SEGMENT WITH A KEY OF ALL X'FF'S

Explanation

The last pointer segment in the primary index for HIDAM or PHIDAM points to a root segment with a key that is not all X'FF's. The database can be corrupted.

System action

HSSR Engine ends abnormally.

User response

Check whether the database is corrupted. If it is not, check whether the correct pair of primary index data set and the primary data set was used.

FABH0517E HSSR CALL HANDLER DETECTED UNEXPECTED ERROR SITUATION

Explanation

HSSR call handler detected an unexpected error.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

If necessary, contact IBM Software Support.

FABH0521E POINTER DESCRIBED BY HSDB1-HPTR1 IS NOT A ROOT-TWIN-POINTER

Explanation

HSSR control blocks contain incorrect information.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378). If necessary, contact IBM Software Support.

FABH0523E SEGMENT-CODE OF LOCATED SEGMENT NOT 01

Explanation

HSSR call handler retrieved a root segment whose segment code is not 01.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378). If necessary, contact IBM Software Support.

FABH0524E BLM BLOCK-NUMBER NOT WITHIN RAA

Explanation

An HDAM segment beyond the root addressable area was tried to be retrieved.

System action

HSSR Engine ends abnormally.

User response

Correct BLM control statement on the FABHFSU CARDIN data set.

FABH0525E RETURN CODE 8 FROM HDAM RANDOMIZER, RMOD=*xxxxxxxx*

Explanation

HSSR Engine received a return code 8 from HDAM randomizing module. *xxxxxxxx* is the name of HDAM randomizing module.

System action

HSSR Engine ends abnormally.

User response

Correct the application program or HDAM randomizing module, and rerun the job, if necessary.

FABH0526E RETURN CODE 8 FROM RANDOMIZER: DBF*xxxxx*

Explanation

HSSR Engine received the return code of 8 from the randomizing module DBF*xxxxx*. DBF*xxxxx* is a randomizing module that has the DEDB randomizer interface.

System action

HSSR Engine ends abnormally.

User response

Correct the application program or the randomizing module, and rerun the job.

FABH0527E INVALID PARTITION NUMBER '*xxxxxxxx*' RETURNED FROM THE RANDOMIZER: DBF*yyyyy*

Explanation

HSSR Engine received an incorrect partition number '*xxxxxxxx*' from the randomizing module DBF*yyyyy*. '*xxxxxxxx*' indicates the value contained in Register 1 in hexadecimal format. DBF*yyyyy* is a randomizing module that has the DEDB randomizer interface.

System action

HSSR Engine ends abnormally.

User response

Correct the application program or the randomizing module, and rerun the job.

FABH0528E INVALID RELATIVE RAP NUMBER '*xxxxxxxx*' FOR PART#:*nnn* RETURNED FROM RANDOMIZER: DBF*yyyyy*

Explanation

HSSR Engine received an incorrect relative RAP number '*xxxxxxxx*' from the randomizing module DBF*yyyyy*. '*xxxxxxxx*' indicates the value contained in Register 0 in hexadecimal format. DBF*yyyyy* is a randomizing module that has the DEDB randomizer interface.

System action

HSSR Engine ends abnormally.

User response

Correct the application program or the randomizing module, and rerun the job.

FABH0541E SEGMENT CODE OF ROOT NOT 01

Explanation

HSSR Engine retrieved a root segment whose segment code was not 01.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378). If necessary, contact IBM Software Support.

FABH0547E INVALID RETURN CODE FROM BUFFER HANDLER

Explanation

HSSR Engine encountered an I/O problem. For more information, see the accompanying SYNADAF message buffer.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378). If necessary, contact IBM Software Support.

FABH0551E INVALID S-C OR D-F IN SPLIT DATA SEGM

Explanation

During the processing of a variable length split segment, HSSR Engine encountered an incorrect segment code, or an incorrect delete flag.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378). If necessary, contact IBM Software Support.

FABH0552E INVALID RC FROM BUF HDLR

Explanation

HSSR Engine encountered an I/O problem in getting the split up segment data.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378). If necessary, contact IBM Software Support.

FABH0553E USER COMPRESSION ROUTINE HAS RETURNED A TOO LONG SEGMENT

Explanation

A user segment compression/decompression exit routine has edited a segment longer than the maximum length defined during DBDGEN for that segment type.

System action

HSSR Engine ends abnormally.

User response

Correct either the DBD or the user compression/decompression exit routine.

FABH0554E SEGMENT *ssssssss* IN DATA BASE *dddddddd* HAS INVALID VALUE FOR THE LENGTH FIELD

Explanation

During the processing of a segment, HSSR Engine encountered a segment for which the length field contained a value that was incorrect.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378). If necessary, contact IBM Software Support.

FABH0555E KEYCHECK OPTION HAS DETECTED A SEQUENCE ERROR; ABEND FOLLOWS

Explanation

HSSR Engine detected a sequence error in the segment key fields and issued an abend as specified by the KEYCHECK option.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378). If necessary, contact IBM Software Support.

FABH0556E AN OCCURRENCE OF SEGMENT *segmname* IN DBD *dddddddd* OVER THE BLOCK BOUNDARY

Explanation

HSSR Engine detects an incorrect occurrence of the segment *segmname*, which is stored over the block boundary. If it is a fixed-length segment, the segment length which is defined in the DBD *dddddddd* can be inconsistent with the actual segment.

System action

HSSR Engine ends abnormally.

User response

Ensure that the specified DBD is same as the one used for inserting the segment. Determine the segment length from the content of the following registers:

- Register 6 contains the address of the segment data.
- Register 5 contains the segment length which is defined in DBD.
- Register 7 contains the address of the block boundary.

FABH0557W HSSR CALL HANDLER RETURNS FIRST GX STATUS CODE

Explanation

HSSR call handler returned a GX status code to the calling application program or utility program because a sequence error was detected. The KEYCHECK option is active. The segment with the incorrect key was returned to the calling programs.

System action

The processing continues.

User response

None.

FABH0559E INTERNAL ERROR OCCURRED IN HSSR CALL HANDLER

Explanation

HSSR call handler detected an unexpected error.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

If necessary, contact IBM Software Support.

FABH0560E STATUSCODE=*xx* ENCOUNTERED DURING INTERNAL ASMTDLI CALL

Explanation

In order to build a logical parent's concatenated key (LPCK) that is defined as virtual, HSSR internally issued an IMS GU call. But the unexpected status code of *xx* was returned. This is probably due to either an HSSR Engine or an IMS software error.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

If necessary, contact IBM Software Support.

FABH0561E REPL CALL WITHOUT REPLACE PROCESSING OPTION

Explanation

The application program issued an HSSR REPL call. However, during PSBGEN, the PCB or SENSEG was not defined with a PROCOPT=R.

System action

HSSR Engine ends abnormally.

User response

Correct either the program or the PSBGEN.

FABH0562E REPL CALL WITHOUT IOAREA

Explanation

The application program issued an HSSR REPL call without providing an IOAREA as a call parameter.

System action

HSSR Engine ends abnormally.

User response

Correct the program.

FABH0563E REPL CALL WITH SSA

Explanation

This message is issued when either of the following occurs:

- The application program issued an HSSR REPL call with an SSA as call parameter. HSSR Engine supports only REPL calls without an SSA.
- The application program issued an HSSR REPL call as an EXEC DLI command. HSSR Engine does not support the REPL command.

System action

HSSR Engine ends abnormally.

User response

Correct the program. If there is no error, check whether the correct APISET statement is specified. For details about the call types and command types that APISET supports, see "DL/I calls and EXEC DLI command for HSSR PCB" on page 84

FABH0564E STATUSCODE=*xx* ENCOUNTERED DURING INTERNAL GHU ASMTDLI CALL

Explanation

During the processing of a GHU call, HSSR Engine issues internally an IMS GHU call; this internal IMS GHU call has returned an unexpected *xx* status code that is displayed in the message.

System action

HSSR Engine ends abnormally.

User response

If the status code is AI, check for an IMS error message that describes the problem in more detail. If PROCOPT=R for a VSAM database, ensure that VSAM SHAREOPTIONS are defined (2,3) or (3,3). If the status code is not AI, the problem might be either an HSSR or IMS software error.

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

FABH0565E STATUSCODE=*xx* ENCOUNTERED DURING INTERNAL REPL ASMTDLI CALL

Explanation

During the processing of a replace call, HSSR Engine issues internally an IMS REPL call; this internal IMS REPL call has returned an unexpected *xx* status code, which is displayed in the message. This is probably either an HSSR Engine or IMS software error.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

If necessary, contact IBM Software Support.

FABH0566E INTERNAL HSSR OR DL1 ERROR DURING PROCESSING OF REPL CALL

Explanation

HSSR Engine detected an unexpected error.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

If necessary, contact IBM Software Support.

FABH0567E PCB-FEEDBACK: DBDNAME=*dbdname* SEGNAME = *segname*

Explanation

This message lists the *dbdname* and the segment name contained in the PCB feedback area at the moment of an error, which is described by another FABH*xxxxx* message.

System action

HSSR Engine ends abnormally with dump.

User response

See other FABH*xxxxx* messages.

FABH0568I FIRST REPL CALL IS BEING ISSUED FOR PCB=*xxxxxxxx*

Explanation

This message informs you that the database (PCB=*xxxxxxxx*) is being modified by an HSSR REPL call.

System action

The processing continues.

User response

None. This message is informational.

FABH0569E REPL CALL WITHOUT PCB

Explanation

The application program issued an HSSR REPL call without providing a PCB name as a call parameter.

System action

HSSR Engine ends abnormally.

User response

Correct the program.

FABH0570E *text*

Explanation

HSSR Engine encountered an I/O problem on a VSAM KSDS. This message *text* contains bytes 27 - 127 of the MSGAREA of RPL as described in *DFSMS/MVS Macro Instructions for Data Sets*.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378).
- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

If necessary, contact IBM Software Support.

FABH0572E INTERNAL ERROR --- MODCB FOR KSDS NOT SUCCESSFUL

Explanation

During KSDS processing, HSSR buffer handler unsuccessfully issued a VSAM MODCB macro.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378). If necessary, contact IBM Software Support.

FABH0573E INTERNAL ERROR --- RPL FOR KSDS NOT INACTIVE

Explanation

During KSDS processing, HSSR buffer handler tried to use an RPL that was active for another VSAM request.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378). If necessary, contact IBM Software Support.

FABH0574E VSAM LOGICAL ERROR --- RPL-FDBK-CODE IS IN R3

Explanation

VSAM signaled to HSSR buffer handler an unexpected logical error after a KSDS GET macro. Use the RPL feedback code in Register 3 to analyze the error in details.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378). If necessary, contact IBM Software Support.

FABH0575E INTERNAL ERROR --- ENDREQ FAILED

Explanation

During KSDS processing, HSSR buffer handler issued an ENDREQ macro, which failed.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378). If necessary, contact IBM Software Support.

FABH0576E DDNAME=*ddname*; DATA REQUEST OUTSIDE OF THE DATASET LIMITS

Explanation

The HSSR buffer handler received a request for a block or CI that is outside the currently known extents of the data set (*ddname*). This problem might occur if the database is physically damaged or if an updating IMS program is concurrently updating the database.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378).
- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the real data set name on the DD statement for *ddname* was specified.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

If necessary, contact IBM Software Support.

FABH0577E RETRY OF KSDS I/O OPERATIONS NOT ENABLED

Explanation

The HSSR buffer handler detected an error during the reading of a VSAM KSDS. This error might result either from a physically damaged KSDS or from the concurrent execution of an updating IMS program.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378).
- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

If the problem results from concurrent execution of an updating IMS program, try to resolve the problem by adding a RETRY KSDS control statement to the HSSROPT data set.

FABH0578E UNEXPECTED RETURN CODE FROM ATTACH

Explanation

FABH350 attempted to ATTACH module FABH351. The attempt was not successful, and the return code was not zero.

System action

HSSR Engine ends abnormally.

User response

Examine the dump to determine the contents of Register 4. Register 4 contains the return code from ATTACH that was returned in Register 15.

FABH0579E UNEXPECTED RETURN CODE FROM DETACH

Explanation

FABH350 attempted to DETACH module FABH351. The attempt was not successful, and the return code was not zero.

System action

HSSR Engine ends abnormally.

User response

Examine the dump to determine the contents of Register 4. Register 4 contains the return code from DETACH that was returned in Register 15.

FABH0581E OPEN OF DBD=*dbdname* DDN=*ddname* --- "*yyyy*" "*text*"

Explanation

The database named *dbdname* type *yyyy* (KSDS, ESDS, OSAM, or KEYD) could not be opened. *ddname* indicates the DD name of the failing data set. The possible combinations of *yyyy* and *text*, and their explanations, are described in detail in the user response section.

System action

HSSR Engine ends abnormally.

User response

The following list provides the subtext (*yyyy* and *text*), explanation, and the user response for each subtext:

"ESDS" "GENCB BLOCK=RPL FAILED"
A GENCB for an RPL failed. A possible reason is that insufficient virtual storage is available in the address space.

See the general steps for troubleshooting the FABH0581E error, and take appropriate actions.

"KEYD" "KEYL IN DATASET AND DBD DIFFERS"
The key length of the data set differs from the key length of the DBD, for example, KSDS. This condition might be caused by a user error such as DD statements referring to the wrong data set, or using the wrong version of a DBD.

Correct the DBD or the DD statement. If the problem persists, run the FABHTEST utility by referring to the general steps for troubleshooting the FABH0581E error, and take appropriate actions. For KSDS, print a LISTCAT of the cluster and its components.

"KEYD" "KEY-POSITION IN DATA SET AND DBD DIFFERS"
The key position of the data set differs from the key position of the DBD. This condition might be caused by a user error such as DD statements referring to the wrong data set, or using the wrong version of a DBD.

Correct the DBD or the DD statement. If the problem persists, run the FABHTEST utility by referring to the general steps for troubleshooting the FABH0581E error, and take appropriate actions. For KSDS, print a LISTCAT of the cluster and its components.

"KSDS" "OPEN FAILED"
The OPEN macro issued against a KSDS is not successful, perhaps for the following reasons:

- The KSDS has the wrong VSAM SHAREOPTIONS, or the data set is allocated by this region with DISP=OLD in the JCL statements.
- There might not be enough virtual storage in the address space.

Ensure that the KSDS definition is correct. If the problem persists, run the FABHTEST utility by referring to the general steps for troubleshooting the FABH0581E error, and take appropriate actions. The error message that is issued by VSAM contains additional information. Refer to the VSAM message for additional assistance.

"KSDS" "SHOWCB FIELD=DDNAME FAILED"
VSAM was asked to retrieve the DD name of a KSDS, but was unsuccessful.

See the general steps for troubleshooting the FABH0581E error, and take appropriate actions.

"KSDS" "SHOWCB FIELD=ERROR FAILED"
VSAM was asked to retrieve the ACB error field after an OPEN, but was unsuccessful.

The error message issued by VSAM contains additional information. Refer to the VSAM message. To identify the cause, see the general steps for troubleshooting the FABH0581E error, and take appropriate actions.

"KSDS" "SHOWCB FIELD=(CINV,...) FAILED"
VSAM was asked to retrieve the length of the control interval, record, and key, and the relative key position of the KSDS.

See the general steps for troubleshooting the FABH0581E error, and take appropriate actions.

"KSDS" "GENCB BLOCK=ACB FAILED"
VSAM was unable to generate an ACB. This was probably caused by insufficient virtual storage in the address space.

See the general steps for troubleshooting the FABH0581E error, and take appropriate actions.

"KSDS" "GENCB BLOCK=RPL FAILED"
VSAM was not able to generate an RPL. This was probably caused by insufficient virtual storage in the address space.

See the general steps for troubleshooting the FABH0581E error, and take appropriate actions.

"KSDS" "MODCB RPL, OPTCD=CNV FAILED"
VSAM was asked to modify an RPL to allow control interval processing in order to allow a VERIFY, but was unsuccessful.

The error message issued by VSAM contains additional information. Refer to the VSAM message. To identify the cause, see the general steps for troubleshooting the FABH0581E error, and take appropriate actions.

"KSDS" "VERIFY FAILED"
HSSR buffer handler issued a VERIFY macro to the KSDS indicated by the message, but it failed.

The error message issued by VSAM contains additional information. Refer to the VSAM message. To identify the cause, run the FABHTEST utility by referring to the general steps for troubleshooting the FABH0581E error, and take appropriate actions.

"KSDS" "ENDREQ FAILED"
HSSR buffer handler issued an unsuccessful ENDREQ macro.

Register 2 at the time of the abend contains the address of the RPL. Inspect the feedback field of the RPL to find out what exactly happened. To identify the cause, run the FABHTEST utility by referring to the general steps for troubleshooting the FABH0581E error, and take appropriate actions.

"KSDS" "MODCB RPL, OPTCD=(KEY, ...) FAILED"
HSSR buffer handler issued an unsuccessful MODCB macro to switch back from CI processing to key processing. If the utility uses the maximum device capacity, check whether *ddname* can be allocated on a device with more track capacity. If the utility uses the block size specified on the JCL statement, check whether the block size or the record size can be increased or use the default maximum device capacity.

See the general steps for troubleshooting the FABH0581E error, and take appropriate actions.

"KSDS" "MODCB RPL, RECLEN..... FAILED"
HSSR buffer handler issued an unsuccessful MODCB macro to attempt to change the size of the input area.

See the general steps for troubleshooting the FABH0581E error, and take appropriate actions.

"KSDS" "EXTENDED ADDRESSABILITY IS NOT SUPPORTED"
The KSDS is an SMS data set with the extended addressability attribute. IMS does not support such a data set.

"OSAM" "OPEN FAILED"
A (BSAM) OPEN macro issued against an OSAM data set is not successful.

Check other error messages, and make sure that the OSAM data set does not have more than 16 extents. To identify the cause, run the FABHTEST utility by referring to the general steps for troubleshooting the FABH0581E error, and take appropriate actions.

"OSAM" "BLOCKSIZE IN DCB NOT EQ BLOCKSIZE IN DBD"
The block size of the data set differs from the block size of the DBD. This condition was probably caused by a user error, such as DD statements referring to the wrong data set, or the wrong version of a DBD has been used.

Correct the DBD or the DD statement. If the problem persists, run the FABHTEST utility by referring to the general steps for troubleshooting the FABH0581E error, and take appropriate actions. A LISTVTOC of the data set should also be produced.

"OSAM" "BLOCKSIZE IS NOT A MULTIPLE OF LRECL"
The block size of the data set is not a multiple of the record length.

Correct the block size. To identify the cause, run the FABHTEST utility by referring to the general steps for troubleshooting the FABH0581E error, and take appropriate actions. A LISTVTOC of the data set should also be produced.

"OSAM" "IMS OPEN PROBLEM"
HSSR Engine issued an internal DL/I call in order to force IMS to open the database. This DL/I call was not successful.

Refer to additional error messages issued by IMS or the access method.

To identify the cause, take either or both of the following general steps for troubleshooting the FABH0581E error.

1. At the time of the problem, register 0 contained a VSAM reason code, which is described in *DFSMS/MVS Macro Instructions for Data Sets*. The contents of register 0 are found in the dump, within the module FABH001, in the diagnosis area. This area has the following layout:
   - A 16-byte section header: 'DIAGNOSIS AREA'
   - A double word PSW
   - 16 full words, showing the content of Registers 0 - 15.
2. Activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378).

If necessary, contact IBM Software Support.

FABH0591E OPEN OF DBD=*dbdname* ---- "*yyyy*" "*text*"

Explanation

The database named *dbdname* type *yyyy* (ESDS, KSDS, or OSAM) could not be opened. *text* describes the problem in detail. The possible combinations of *yyyy* and *text*, and their explanations are described in detail in the user response section.

System action

HSSR Engine ends abnormally.

User response

"ESDS2" "LOGICAL RECORD LENGTH IN ACB IS ZERO"
After OPEN, VSAM finds that the logical record length is zero.

Correct the logical record length. If the problem persists, run the FABHTEST utility by referring to the general steps for troubleshooting the FABH0591E error, and take appropriate actions. A LISTCAT of the ESDS should also be produced.

"ESDS" "CONTROL INTERVAL SIZE IN ACB IS ZERO"
After OPEN, VSAM finds that the control interval size is zero.

Correct the control interval size. To identify the cause, run the FABHTEST utility by referring to the general steps for troubleshooting the FABH0591E error, and take appropriate actions. A LISTCAT of the ESDS should also be produced.

"ESDS" "GENCB BLOCK=ACB FAILED"
VSAM could not generate an ACB. This condition was probably caused by insufficient virtual storage in the address space.

Ensure that sufficient virtual storage is available. To identify the cause, see the general steps for troubleshooting the FABH0591E error, and take appropriate actions.

"ESDS" "GENCB BLOCK=RPL FAILED"
VSAM could not generate an RPL. This condition was probably caused by insufficient virtual storage in the address space.

Ensure that sufficient virtual storage is available. To identify the cause, see the general steps for troubleshooting the FABH0591E error, and take appropriate actions.

"ESDS" "GENCB BLOCK=EXLST FAILED"
VSAM could not generate an exit list. This condition was probably caused by insufficient virtual storage in the address space.

Ensure that sufficient virtual storage is available. To identify the cause, see the general steps for troubleshooting the FABH0591E error, and take appropriate actions.

"ESDS" "OPEN FAILED"
The OPEN macro issued against a KSDS was not successful. This condition was probably caused by:

- The ESDS had the wrong VSAM SHAREOPTIONS, or the data set was allocated by this region with DISP=OLD on the JCL statements.
- There is not enough virtual storage in the address space.

Ensure that the ESDS is correctly defined. The error message issued by VSAM contains additional information. Refer to the VSAM message. To identify the cause, run the FABHTEST utility by referring to the general steps for troubleshooting the FABH0591E error, and take appropriate actions.

"ESDS" "VERIFY FAILED"
HSSR buffer handler issued a VERIFY macro to the KSDS indicated by the message, but it failed.

Ensure that the ESDS is correctly defined. The error message issued by VSAM contains additional information. Refer to the VSAM message. To identify the cause, run the FABHTEST utility by referring to the general steps for troubleshooting the FABH0591E error, and take appropriate actions.

"ESDS" "ENDREQ FAILED"
HSSR buffer handler issued an unsuccessful ENDREQ macro.

Register 2 at the time of abend contains the address of the RPL. Inspect the feedback field of the RPL to find out what exactly happened. To identify the cause, run the FABHTEST utility by referring to the general steps for troubleshooting the FABH0591E error, and take appropriate actions.

"ESDS" "SHOWCB FIELD=CINV FAILED"
VSAM could not retrieve the control interval size of the ESDS.

See the general steps for troubleshooting the FABH05891E error, and take appropriate actions.

"ESDS" "SHOWCB FIELD=ERROR FAILED"
VSAM could not retrieve the error code.

See the general steps for troubleshooting the FABH05891E error, and take appropriate actions.

"ESDS" "HSSR-CAB CANNOT READ THE SAME ESDS-DB THROUGH MULTIPLE PCBs"
HSSR buffer handler detected that multiple HSSR PCBs that are referring to the same database should be buffered by CAB. This situation is not supported by CAB.

Modify the CAB control statements in such a way that no more than one HSSR PCB per database is buffered by CAB. The OCCURRENCE CAB control statement can be used to achieve this modification.

"ESDS" "ENDRBA NOT FOUND"
This error might occur if the VSAM control blocks are changing.

Rerun the program.

To identify the cause, take either or both of the following general steps for troubleshooting the FABH0591E error.

1. At the time of the problem, register 0 contained a VSAM reason code, which is described in *DFSMS/MVS Macro Instructions for Data Sets*. The contents of register 0 are found in the dump, within the module FABH001, in the diagnosis area. This area has the following layout:
   - A 16-byte section header: 'DIAGNOSIS AREA'
   - A double word PSW
   - 16 full words, showing the content of Registers 0 - 15.
2. Activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378).

If necessary, contact IBM Software Support.

FABH0592E UNACCEPTABLE VSAM LOGICAL ERROR (RPLREQ=*xx*, RPLFDBWD=*xxxxxxxx*)

Explanation

HSSR buffer handler has encountered an unexpected VSAM logical error while it was processing VSAM ESDS data sets. RPLREQ shows the VSAM macro function that was invoked, and RPLFDBWD shows VSAM feedback information related to VSAM failure. These values are shown in hexadecimal.

System action

HSSR Engine ends abnormally.

User response

This error is likely an internal system error. Contact IBM Software Support.

FABH0595E STATUSCODE=*XX* ENCOUNTERED DURING INTERNAL ASMTDLI CALL (*segmname*)

Explanation

Program FABHURG1, run with the MIGRATE control statement specified, issued a DL/I GU call internally for the virtual logical child segment *segmname*, but the status code of the call was not blank.

System action

HSSR Engine ends abnormally.

User response

Check the status code and detect the cause of the error. If the status code is AI, the most likely cause is the absence of the DD statement for the logically related database. If the cause is not clear, collect the dump and contact IBM Software Support.

FABH0596E STATUSCODE=*XX* ENCOUNTERED DURING INTERNAL ASMTDLI CALL (*segmname*)

Explanation

Program FABHURG1, run with the MIGRATE control statement specified, issued a DL/I GU call internally for an occurrence of the physically paired logical child segment *segmname*, but the status code of the call was not blank.

System action

FABHURG1 ends abnormally.

User response

Check the status code and detect the cause of the error. If the status code is AI, the most likely cause is the absence of the DD statement for the logically related database. If the cause is not clear, collect the dump and contact IBM Software Support.

FABH0597E **STATUSCODE=*XX* ENCOUNTERED DURING INTERNAL ASMTDLI CALL (*segmname*)**

Explanation

Program FABHURG1, run with the MIGRATE control statement specified, issued a DL/I GU call internally for an occurrence of the physically paired logical child segment *segmname*, but the status code of the call was not blank.

System action

FABHURG1 ends abnormally.

User response

Check the status code and detect the cause of the error. If the cause is not clear, collect the dump and contact IBM Software Support.

FABH0598E **STATUSCODE=*XX* ENCOUNTERED DURING INTERNAL ASMTDLI CALL (*segmname*)**

Explanation

Program FABHURG1, run with the MIGRATE control statement specified, issued a DL/I GU call internally for an occurrence of the unidirectional logical child segment *segmname*, but the status code of the call was not blank.

System action

FABHURG1 ends abnormally.

User response

Check the status code and identify the cause of the error. If the status code is AI, the most likely cause is the absence of the DD statement for the logically related database. If the cause is not clear, collect the dump and contact IBM Software Support.

FABH0602E **BAD RETURN CODE FROM OSAM MACRO ... PLEASE REFER TO PRECEDING DFS730I MESSAGE**

Explanation

HSSR buffer handler tried to open an OSAM database data set with the OSAM open functions. The open was not successful.

System action

HSSR Engine ends abnormally.

User response

Refer to the message DFS730I and read its explanation in *IMS Messages and Codes*.

FABH0603E ***ddname* DD-STATEMENT IS MISSING; UNABLE TO OPEN DATA SET**

Explanation

HSSR buffer handler tried to open the OSAM database data set with the named *ddname*. The DD statement is missing or misspelled.

System action

HSSR Engine ends abnormally.

User response

Correct the DD statement.

FABH0604E **DATA SET IS NOT INITIALIZED AND IS EMPTY: DB=*dbdname*, DD=*ddname***

Explanation

The OSAM data set that is specified in the *ddname* DD statement and that is being used as input for IMS High Performance Unload has not been initialized as a DL/I database and contains no data block. Only the header record is written in output unloaded data sets.

System action

HSSR Engine ends abnormally.

User response

Ensure that the data set specified in the DD statement is correct.

FABH0605E **RDJFCB MACRO FAILED ON DDNAME: *ddname* (RC=*xx*)**

Explanation

The RDJFCB macro failed with a return code *xx* for the data set that was specified for the DDNAME *ddname*.

System action

HSSR Engine ends abnormally.

User response

Check the job log messages and other printed output for an indication that some system service has failed.

If you specified DBALLABOVE in DFSVSAMP, make the following change and rerun the job:

- For z/OS 1.12 or later, specify NON_VSAM_XTIOT=YES in the DEVSUP*xx* PARMLIB member.
- For z/OS 1.11 or earlier, remove the specification of DBALLABOVE from DFSVSAMP.

If none can be found, it is likely that IMS High Performance Unload has had an internal logic error. Contact IBM Software Support.

FABH0606E **OBTAIN MACRO FAILED ON DDNAME: *ddname* (RC=*xx*)**

Explanation

The OBTAIN macro failed with a return code *xx* for the data set that was specified for the DDNAME *ddname*.

System action

HSSR Engine ends abnormally.

User response

Check the job log messages and other printed output for an indication that some system service has failed. If none can be found, it is likely that IMS High Performance Unload has had an internal logic error. Contact IBM Software Support.

FABH0607I **DD=*dd_name* BLOCK SIZE IS NOT SIZE DEFINED IN RECON**

Explanation

The actual block size of the indicated OSAM data set is not the same as the block size that is defined in the RECON data sets.

System action

The processing continues.

User response

It is recommended that you correct the block size in the RECON data sets.

FABH0608E **RANSIZE IS TOO LOW: MINIMUM IS *nnn***

Explanation

The value of a RANSIZE control statement is less than the allowable minimum value.

System action

HSSR Engine ends abnormally.

User response

Increase the value of the RANSIZE control statement and rerun the job.

FABH0611E **OPEN OF DBD=*dbdname* DDN=*ddname* --- ESDS GEN BLOCK= RPL FAILED**

Explanation

The ESDS database named *dbdname* could not be opened. A GENCB for an RPL failed. A possible reason is that insufficient virtual storage is available in the address space. *ddname* indicates the DD name of the failing data set.

System action

HSSR Engine ends abnormally.

User response

See message FABH0591E, and take an appropriate action. If necessary, contact IBM Software Support.

FABH0621E **PAGE-FIXING HAS FAILED**

Explanation

HSSR buffer handler tried to fix its own OSAM or ESDS buffers by issuing the IMSAUTH FUNC=PGFIX macro. The IMSAUTH macro set a return code indicating that it could not perform page-fixing.

System action

HSSR Engine ends abnormally.

User response

Examine the contents of register 4 and take an appropriate action. Register 4 contains the error code returned by IMSAUTH in register 15. If necessary, contact IBM Software Support.

FABH0622E HSSR FAILED IN PARTITION SELECTION; REQUEST=*xxxxxx*, RC=*yy*, RSN=*zzzzz*

Explanation

An error occurred in a partition selection request when the IMS's DFSPSEL macro was used. The string *xxxxxxx* indicates the type of the request; FIRST, NEXT, SELECT, PSET, or PRSET. The value *yy* shows the return code, and the value *zzzzz* shows the reason code from the DFSPSEL service.

RSN=4020 means a DB authorization error occurred, because the partition has been authorized to another IMS subsystem or a data set name that is specified in the JCL for the partition is inconsistent with the registration in RECON. For other reason codes, see the explanation of message DFS0832I in *IMS Messages and Codes Volume 2*.

System action

HSSR Engine ends abnormally.

User response

If RSN=4020, issue the /DBD or the /DBR command to the partition before the IMS High Performance Unload job step, or remove the DD statements for the partition from the JCL.

FABH0623E SELECTED PARTITION *ppppppp* OF DATABASE *dddddddd* IS NOT AVAILABLE

Explanation

The partition *ppppppp* selected by an HSSR call that was issued for the HALDB *dddddddd* had not been specified as a partition to be processed. This message is issued when a user exit routine for FABHURG1 or FABHFSU returns a root sequence key of the segment to be processed next, but the segment is in an inaccessible partition.

System action

HSSR Engine ends abnormally.

User response

Check whether the partition *ppppppp* belongs to the set of partitions that are specified by a PARTITION control statement in the SYSIN data set of FABHURG1 or in the CARDIN data set of FABHFSU. If not, correct the PARTITION control statement so that the partition *ppppppp* can be included. If a HALDB control statement in the DFSHALDB DD statement is defined, remove it.

FABH0624E TARGET PARTITION *ppppppp* IS NOT FOUND

Explanation

IMS DFSPSEL macro returned return code 8 and reason code X'8010' for a PSET request. The string *ppppppp* shows the name of the partition that is not found.

System action

HSSR Engine ends abnormally.

User response

Check whether the partition *ppppppp* belongs to the set of partitions that are specified by a PARTITION control statement in the SYSIN data set of FABHURG1 or in the CARDIN data set of FABHFSU; if not, correct the PARTITION control statement so that the partition *ppppppp* can be included.

FABH0625E USER PARTITION SELECTION FAILED (SELECTION TYPE=*aaaaaa*)

Explanation

The partition selection exit got the return code 4 from the routine written for the exit, and the IMS DFSPSEL macro returned return code 8 and reason code X'8051'. The string *aaaaaa* shows the type of the partition selection request (the value specified for the PART parameter of DFSPSEL) that was issued at the time of the error.

System action

HSSR Engine ends abnormally.

User response

Determine the reason why the partition selection exit routine returned return code 4.

FABH0626E PARTITION SELECTION EXIT ROUTINE REQUESTED A

PSEUDO ABEND (SELECTION TYPE=*aaaaaa*, RC=*xx*)

Explanation

The user partition selection exit routine returned the return code *xx* and the IMS DFSPSEL macro returned return code 16 and the reason code X'10001'. The string *aaaaaa* shows the type of the partition selection request (the *v* specified for the PART parameter of DFSPSEL) that was in effect at the time of the error.

System action

HSSR Engine ends abnormally.

User response

For the explanation of the IMS user abend code of 3499, see *IMS Messages and Codes*. The programmer response described in *IMS Messages and Codes* applies also to HSSR Engine.

FABH0627E **EXIT ROUTINE *xxxxxxxx* RETURNED RC=03 ALTHOUGH PARTITION AND CO STATEMENTS ARE SPECIFIED**

Explanation

The user exit routine *xxxxxxxx* for FABHURG1 returned return code 3 when both the PARTITION and CO control statements are specified. Exit routines are not allowed to return a return code of 3 when both of these statements are specified.

System action

HSSR Engine ends abnormally.

User response

Remove the CO or PARTITION statement from the HSSROPT data set.

FABH0628E **EXIT ROUTINE *xxxxxxxx* RETURNED RC=16 ALTHOUGH PARTITION AND CO STATEMENTS ARE SPECIFIED**

Explanation

The user exit routine *xxxxxxxx* for FABHFSU returned return code 16 when both the PARTITION and CO control statements are specified. Exit routines are not allowed to return return code 16 when both of these statements are specified.

System action

HSSR Engine ends abnormally.

User response

Remove the CO or PARTITION statement from the HSSROPT data set.

FABH0629W **PARTITION *ppppppp* IS SKIPPED BECAUSE OF AN ERROR IN PARTITION SELECTION**

Explanation

The processing of the HALDB partition *ppppppp* was skipped because an error occurred when a partition selection request was processed.

System action

HSSR Engine continues processing.

User response

See the Trace Output report with Diagnostics for the reason of the error.

FABH0630E **CANNOT PROCESS THE PARTITION NEXT TO *ppppppp* BECAUSE OF STATUS CODE 'GG' FROM THE PRIOR DL/I CALL**

Explanation

This message is issued if a CO statement is specified in the HSSROPT data set and the status code 'GG' was returned at the DL/I call issued earlier for the comparison with an HSSR call. The string *ppppppp* shows the name of the partition that had been processed before the DL/I call in question was issued. HSSR Engine cannot continue the processing of the database in this case even if PROCOPT=GON or GOT is specified for the PCB.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, remove the CO statement and specify a DIAGG statement in HSSROPT data set; then rerun the job to see the diagnostics information for the status code 'GG'. If you are running a FABHURG1 or FABHFSU job, it is recommended that you specify the PARTITION statement with *ppppppp* as the first operand and 2 as the second operand.

FABH0631E PROBLEMS WITH DL/I-STAT CALL

Explanation

HSSR buffer handler tried to retrieve the VSAM Shared Resource Pool Statistics by issuing an internal DL/I STAT call. DL/I returned an unexpected status code.

System action

HSSR Engine ends abnormally.

User response

To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378). If necessary, contact IBM Software Support.

FABH0632E CLOSE FAILED

Explanation

After program termination, HSSR Engine is asked to close the DCB/ACB of the databases. The CLOSE macro is unsuccessful.

System action

HSSR Engine ends abnormally.

User response

The access method error message contains additional information. Refer to the message. To identify the cause, activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378). If necessary, contact IBM Software Support.

FABH0633E TARGET PARTITION IS NOT FOUND FOR THE SPECIFIED KEY

Explanation

IMS DFSPSEL macro returned return code 8 and the reason code X'8010' for a SELECT request, but no partition corresponding to the key specified in the request was found.

System action

HSSR Engine ends abnormally.

User response

Check whether the root key specified in the SSA is correct.

DMTI

The key specified in the SSA is at offset X'8AC' from the address pointed to by general register 10.

DMTI

FABH0634E ONLINE REORG RUNNING FOR PARTITION=*pppppppp* (DBD=*dddddddd*)

Explanation

HALDB Online Reorganization (OLR) is currently processing partition *pppppppp*. HSSR Engine cannot process the partition.

System action

HSSR Engine ends abnormally.

User response

Rerun after OLR for all partitions are completed.

FABH0635W ONLINE REORG ACTIVE FOR PARTITION=*pppppppp* (DBD=*dddddddd*)

Explanation

HALDB Online Reorganization (OLR) processing for partition *pppppppp* was stopped prior to the completion of the partition, and the OLR cursor is still active. HSSR Engine will process the partition which is comprised of both the A-J and X data sets as well as the M-V and Y data sets.

System action

HSSR Engine continues processing. If one of the following options is specified, it is ignored:

- BYINDEX, CO, DBSTATS, KEYCHECK, or SKERROR.

User response

Refer to the additional messages.

FABH0636E UNSUPPORTED OPTION FOR OLR ACTIVE PARTITIONS: *option*

Explanation

The option (*option*) is specified, but this is one of the following options with which HSSR Engine cannot process the HALDB OLR active partition:

- DECN
- PARTEXTR
- User exit routine
- *CS format for PHDAM

System action

FABHURG1 or FABHFSU ends abnormally.

User response

Remove this option or rerun after OLR for all partitions are completed.

FABH0637W HSSR CALLS FALL BACK TO DL/I FROM PARTITION=*pppppppp* (DBD=*dddddddd*)

Explanation

HSSR Engine will pass all of the following DL/I calls to IMS's DL/I call handler from partition *pppppppp*, because HALDB online reorganization (OLR) is active for this partition.

System action

HSSR Engine continues processing. The performance, however, decreases.

User response

If you want to use the ignored option, rerun after HALDB OLR for all partitions are completed.

FABH0638W SPECIFIED DBVER IS NOT SUPPORTED. HSSR CALLS FALL BACK TO DL/I

Explanation

IMS database versioning is enabled, but the specified database is not the current version of the database. HSSR Engine supports only the current version of the database when IMS database versioning is enabled. HSSR Engine passes subsequent DL/I calls to the DL/I call handler of IMS.

System action

HSSR Engine ignores BYINDEX, CO, DBSTATS, KEYCHECK, and SKERROR control statements specified in the HSSROPT data set, and activates the BLDLPCK option. HSSR Engine continues processing, however, the performance decreases.

User response

Specify the current version of the database.

FABH0639E IMS TOOLS CATALOG INTERFACE ERROR: FUNCTION=*func* RC=*rc* RSN=*rsn*

Explanation

HSSR Engine received an error return code from IMS Tools Catalog Interface.

System action

HSSR Engine ends abnormally.

User response

Contact IBM Software Support.

FABH0640E DBD=*dbdname* IS NOT FOUND IN THE IMS DIRECTORY DATA SETS

Explanation

HSSR Engine could not find the indicated DBD in the IMS directory data sets.

System action

HSSR Engine ends abnormally.

User response

Ensure that the correct DBD name is specified on the EXEC statement and rerun the job.

FABH0641E VSAM PHYSICAL I/O ERROR

Explanation

HSSR buffer handler encountered an I/O problem on an ESDS. The return code and reason code are described in *DFSMS/MVS Macro Instructions for Data Sets*.

System action

HSSR Engine ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378).
- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

If necessary, contact IBM Software Support.

FABH0642E VSAM LOGICAL ERROR; RPL-FDBK-CODE IS IN R3

Explanation

VSAM signaled an unexpected logical error to HSSR buffer handler after an ESDS GET macro.

System action

HSSR Engine ends abnormally.

User response

At the time of the dump, register 3 contains the RPL feedback code. Identify and resolve the cause of the error from the RPL feedback code.

If the problem persists, complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.
- Activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378).

If necessary, contact IBM Software Support.

FABH0643E SHOWCB OF RPL-FDBK-CODE FAILED

Explanation

The VSAM SHOWCB macro issued by HSSR buffer handler failed.

System action

HSSR Engine ends abnormally.

User response

If necessary, contact IBM Software Support.

FABH0645I ZIIP TIME (HH:MM:SS.THMIJU) WAS: *hh:mm:ss.thmiju*

Explanation

This informational message shows the CPU time consumed by zIIP processors for the IMS HP Unload job.

System action

Processing continues.

User response

None. This message is informational.

FABH0646E ERROR RETURNED FROM GEXAPI00: FUNC=*xxxxxxx*, SERVICE CODE=*xxxx*, RC=*xxxx*, RSN=*xxxx*

Explanation

HSSR Engine called the GEXAPI00 module, but an error return code was returned from GEXAPI00.

System action

If ZIIPMODE=FORCE is specified, HSSR Engine ends abnormally. If ZIIPMODE=COND is specified and the error occurred during initialization (FUNC=INIT), processing continues.

User response

If the return code is 04 and the reason code is 04, ensure that your environment is correctly set up:

- You are using one of the currently supported versions of z/OS and the PTF for using zIIP has been applied.
- The zIIP processor is correctly set up.

If the problem persists, contact IBM Software Support.

FABH0653E OSAM I/O ERROR, DDNAME=*ddname* DECB-STATUS=*xxx...20*

Explanation

HSSR buffer handler encountered an I/O problem when using BSAM to read an OSAM block. *ddname* is the ddname. *xxx...20* is the field DECBSTAT from the DECB used in the I/O operation followed by a description of the error.

System action

HSSR Engine ends abnormally.

User response

See the description of message DFS0451I in *IMS Messages and Codes*.

Complete the following tasks to identify the cause, and take appropriate actions:

- Activate the compare and hardcopy trace options, and execute the failing call sequence with the FABHTEST utility (see "FABHTEST utility for problem determination" on page 378).
- Ensure that the correct DBDs, PSBs, and ACBs were being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

If necessary, contact IBM Software Support.

FABH0654E MEDIA MANAGER *function* ERROR, RC=*rc*

Explanation

When HSSR Engine issued MMGRSRV to process *function* (CONNECT, DISCONNECT, or CATREAD) to the data set, an unexpected Media Manager MMGRSRV error occurred. The *rc* is the return code of Media Manager. For details, see *DFSMSdfp Diagnosis Reference*.

System action

HSSR Engine ends abnormally.

User response

Collect the dump and contact IBM Software Support.

FABH0655E MEDIA MANAGER I/O ERROR, RC=*rc*

Explanation

When HSSR Engine issued MMGRCALL to get access to the data set, an unexpected Media Manager MMGRCALL error occurred. The *rc* is the return code of Media Manager.

System action

HSSR Engine ends abnormally.

User response

Collect the dump and contact IBM Software Support.

FABH0656I MEDIA MANAGER IS USED TO ACCESS DB: *dbdname*

Explanation

Media Manager is used to read VSAM data sets of database *dbdname* because all concatenations of the JOBLIB/STEPLIB are APF-authorized.

System action

HSSR Engine continues processing.

User response

None. This message is informational.

FABH0657E STEPLIB IS NOT APF-AUTHORIZED

Explanation

APF-authorization is required for this type of run of the HSSR Engine.

System action

HSSR Engine ends abnormally.

User response

APF-authorize all libraries in STEPLIB DD concatenation, and rerun the job.

FABH0658W "HPIO YES" IS IGNORED DUE TO IMS HD UNLOAD COMPATIBILITY MODE

Explanation

HPIO YES is specified in the HSSROPT DD statement. However, the HPIO YES specification is ignored because HPIO statements are not supported for IMS High Performance Unload jobs that are started by IMS HD Reorganization Unload (DFSURGU0) JCL.

System action

HSSR Engine continues processing.

User response

If you want to use Media Manager to process VSAM ESDSs, change the JCL to FABHURG1 JCL. Otherwise, remove HPIO YES from the HSSROPT DD statement.

FABH0659E SPECIFIED DBVER IS NOT SUPPORTED

Explanation

IMS database versioning is enabled in the ULU region, but the specified database is not the current version of the database. When IMS database versioning is enabled, only the current version of the database can be used.

System action

HSSR Engine ends abnormally.

User response

Check whether DBVERSION=(YES,DBLEVEL=BASE) statement is coded in the DFSDF*xxx* member of the IMS.PROCLIB data set. If the statement exists, remove it or change the DBLEVEL value so that the current version of the database is used.

FABH0660W COMPAUTH YES IS IGNORED IN UNAUTHORIZED PROGRAM

Explanation

Although one or more segments are compressed, the COMPAUTH YES option in the HSSROPT data set is ignored because of one of the following reasons:

- One or more STEPLIB libraries are not APF-authorized.
- Authorization code 1 (AC=1) is not assigned to the first module. This condition occurs when, for example, EXEC PGM=DFSRRC00 is specified in the JCL.

System action

HSSR Engine ignores the COMPAUTH YES option and continues processing with the COMPAUTH NO option.

User response

To apply the COMPAUTH YES option, APF-authorize all the STEPLIB libraries and specify the program name of FABHX034 on the PGM parameter of the EXEC statement.

FABH0661E RBA REQUEST BEYOND ORIGINAL DATA SET LIMITS DDNAME=*ddname*

Explanation

The HSSR buffer handler received a request to read a block or CI that is beyond the extents of the data set (*ddname*). The reason might be that concurrent database updates have extended the data set.

System action

HSSR buffer handler tries to get new up-to-date information about the data set extents. If the requested block/CI is no longer beyond the known data set limits, HSSR buffer handler proceeds normally; otherwise HSSR Engine ends abnormally.

User response

None.

FABH0662E OPEN OF DBD=*dbdname* DDN=*ddname* --- ESDS "*text*"

Explanation

HSSR buffer handler tried to access a CI that is beyond the currently known extents of the data set (*dbdname*). To get up-to-date extent information, HSSR buffer handler is issuing a new VSAM OPEN. The VSAM OPEN is unsuccessful. *ddname* indicates the DD name of the failing data set.

System action

HSSR Engine ends abnormally.

User response

See message FABH0591E, and take an appropriate action.

FABH0663E RETRYING KSDS I/O OPERATIONS

Explanation

The HSSR buffer handler detected an unexpected VSAM logical error code. This might be explained by concurrent database updates.

System action

HSSR buffer handler refreshes the KSDS buffers and retries the I/O operation. If the retry is not successful, an abend is issued. If the retry is successful, HSSR Engine proceeds normally. Note that in this case, some database records might have been skipped or retrieved twice.

User response

None.

FABH0664E OPEN OF DBD=*dbdname* DDN=*ddname* --- KSDS "*text*"

Explanation

HSSR buffer handler has encountered an unexpected VSAM logical error and decided to re-OPEN the VSAM KSDS (*dbdname*) and to retry the I/O operation. However, the re-OPEN is not successful. *ddname* indicates the DD name of the failing data set.

System action

HSSR Engine ends abnormally.

User response

See message FABH0581E, and take an appropriate action.

FABH0670W DB-ACCESS NOT AUTHORIZED BY DBRC: DB=*dbdname*

Explanation

The DBRC authorization request for database access was not sent. When two or more database PCBs (DBPCBs) are defined in the PSB and the library of DFSMDA members for dynamic allocation of the database data sets is specified on the IMSDALIB DD statement, HSSR Engine does not send the DBRC authorization request.

System action

HSSR Engine continues processing without obtaining DBRC authorization.

User response

Specify the library of DFSMDA members on the STEPLIB DD statement.

FABH0671W DB (*xxxxxxxx*) PCB IS NOT A VALID HSSR PCB, RC=*yyyyyyy*

Explanation

When the first HSSR call is issued to the database, the database (*xxxxxxxx*) PCB is incorrect as HSSR PCB. If no database PCBs are in the PSB, then *xxxxxxxx* is blank. The reason code (RC=*yyyyyyy*) shows an error in the PCG. The program control is transferred to DL/I modules instead of HSSR Engine. It means that the database does not take advantage of HSSR Engine.

RC
Meaning

PROCOPT
The PROCOPT parameter on the PCB statement specifies the incorrect code or combination of codes. The codes you can use are G, O, N, T, R, A, P, and E.

KEYLEN
A value of the KEYLEN parameter on the PCB statement is less than 200. This reason code is returned also when the specified PCB refers to the DBD that is listed on the DBDL1 control statement.

USREXIT
An application program issued an HSSR call for the following PCB:

- PCB is generated with PROCOPT=R.
- And DBD referred to by the PCB is generated with the data capture exit routine.

The following table summarizes the reason codes of FABH0671W for the combination of errors.

Key length error	PROCOPT error	User exit error	Reason code of FABH0671W
Yes	Yes	Yes	RC=PROCOPT
Yes	Yes	No	RC=PROCOPT
Yes	No	No	RC=KEYLEN
Yes	No	Yes	RC=USREXIT
No	No	Yes	RC=USREXIT
No	No	No	N/A
No	Yes	Yes	RC=PROCOPT
No	Yes	No	RC=PROCOPT

System action

HSSR Engine continues processing.

User response

If you want to process the database with HSSR Engine, check and correct the PCB that defines the database. Otherwise, ignore this message and continue processing by using DL/I modules.

To identify the cause, check the PROCOPT or KEYLEN parameter in the PCB. If the reason code is "KEYLEN" and the specified PCB does not refer to any DBDs that are listed on the DBDL1 control statement, the PCB can be specified as an HSSR PCB using the HSSRPCB control statement or through the HSSRDBD control statement.

FABH0672E INVALID NUMBER OF PARAMETERS FOR PLIHSSR CALL

Explanation

The HSSR PL/I language interface detected an incorrect number of parameters for PLIHSSR call.

System action

HSSR Engine ends abnormally with dump.

User response

Check and correct the PLIHSSR call statement in your PL/I application program and rerun. If there is no error, check whether the correct APISET statement is specified. For details about the call types and command types that APISET supports, see "DL/I calls and EXEC DLI command for HSSR PCB" on page 84.

FABH0673E INCORRECT PCB ADDRESS WAS PASSED BY APPLICATION PROGRAM

Explanation

HSSR Engine detected an incorrect PCB address internally. This message was issued because of the incorrect PCB address passed by application program or an internal error in HSSR Engine.

System action

HSSR Engine ends abnormally.

User response

Check the PCB parameter or other parameters. If there is no error in DL/I call statement, contact IBM Software Support.

FABH0674E ERROR FOUND IN HSSR CALL STATEMENT, INVALID *xxxxxx*

Explanation

The HSSR language interface detected an error in the HSSR call statement described in *xxxxxx*. *xxxxxx* shows one of the following texts:

PARAMETER COUNT

- If APISET is 1, the parameter count must be 3 or 4.
- If APISET is 2, the count must be from 3 to 5.
- If APISET is 3, the count must be from 3 to 18, inclusive.

PCB ADDRESS
PCB address is incorrect.

NUMBER OF PARAMETERS

- If APISET is 1, the number of parameters must be 3 or 4.
- If APISET is 2, the number must be from 3 to 5.
- If APISET is 3, the number must be from 3 to 18, inclusive.

System action

HSSR Engine ends abnormally.

User response

Specify 'APISET 2' or 'APISET 3' in the control statement.

FABH0675E ROOT SEGMENT POSITION IS NOT ESTABLISHED

Explanation

Two or more SSAs are specified and the first is a qualified SSA, but the root segment position has not been established yet by the preceding call. This call is not supported by the HSSR call handler. The unsupported call is printed in the Trace Output report in the HSSRTRAC data set.

System action

If APISET is 2, HSSR Engine ends abnormally. If APISET is 3, the call and all the succeeding calls to the HSSR PCB are passed to the IMS DL/I call handler to continue the processing instead of ending it abnormally.

User response

If APISET is 2, specify 'APISET 3' in the control statement. If APISET is 3, ignore this message.

FABH0676E INCORRECT COMPAUTH CONTROL STATEMENT IS SPECIFIED

Explanation

In the HSSROPT data set, an incorrect operand is specified for the COMPAUTH control statement. The operand must be YES or NO.

System action

HSSR Engine ignores the COMPAUTH control statement and continues processing.

User response

Correct the COMPAUTH control statement.

FABH0677E INVALID SKIPAUTH CONTROL STATEMENT IS SPECIFIED

Explanation

An incorrect operand is specified on the SKIPAUTH control statement in the HSSROPT data set. The operand must be either YES or NO.

System action

The incorrect statement is ignored.

User response

Correct the SKIPAUTH statement and then rerun the job.

FABH0678E INCORRECT HPIO CONTROL STATEMENT IS SPECIFIED

Explanation

An incorrect operand on the HPIO control statement is specified in the HSSROPT data set. The operand must be either YES or NO.

System action

The incorrect statement is ignored.

User response

Correct the HPIO statement.

FABH0679E INCORRECT SKIPVLC CONTROL STATEMENT IS SPECIFIED

Explanation

An incorrect operand on the SKIPVLC control statement is specified in the HSSROPT data set. The operand must be either YES or NO.

System action

The incorrect statement is ignored.

User response

Correct the SKIPVLC statement.

FABH0680E FABHKEYX CANNOT PROCESS THE COMPRESSED ROOT KEY

Explanation

The FABHKEYX exit routine cannot process the compressed root key. It must be decompressed.

System action

Program FABHURG1 ends abnormally.

User response

If you want to use the FABHKEYX exit for this database, specify the DECY control statement.

FABH0681E INCORRECT APISET CONTROL STATEMENT IS SPECIFIED

Explanation

An incorrect operand or no operand is specified for the APISET control statement in the HSSROPT data set. The operand must be 1, 2, or 3.

System action

The statement is ignored and the system or site default is used.

User response

Correct the APISET control statement.

FABH0682E *xxxxxxxx* CANNOT BE SPECIFIED WHEN APISET 3 IS SELECTED

Explanation

The control statement *xxxxxxxx* is not supported for APISET 3.

System action

HSSR Engine ends abnormally.

User response

Remove the control statement *xxxxxxxx* or specify APISET 1 or 2.

FABH0683E INCORRECT PCBLIST CONTROL STATEMENT IS SPECIFIED

Explanation

An incorrect operand or no operand is specified for the PCBLIST control statement in the HSSROPT data set. The operand must be HSSR or IMS.

System action

The statement is ignored and the system or site default is used.

User response

Correct the PCBLIST control statement.

FABH0684I **DD=*ddname01* HIGHKEY=*KeyString* SEGMENT=*segname1* *n*,*nnn*,*nnn*,*nnn* OCCURRENCES *TOTAL *n*,*nnn*,*nnn*,*nnn* OCCURRENCES**

Explanation

The FABHKEYX exit routine used the specified DD name *ddname01* and the high key value *KeyString*. The indicated number of segment records were written to this unload file.

System action

Program FABHURG1 continues processing.

User response

None. This message is informational.

FABH0685E **INCORRECT STATEMENT IS FOUND IN FABHKEYX DATA SET**

Explanation

The FABHKEYX exit routine detected an incorrect statement that is specified in the FABHKEYX data set.

System action

Program FABHURG1 ends abnormally.

User response

Correct the statement.

FABH0686E **HIGH KEY VALUES ARE NOT SPECIFIED IN ASCENDING ORDER**

Explanation

The FABHKEYX exit routine detected the high key values that are specified in the FABHKEYX data set are not in ascending order.

System action

Program FABHURG1 ends abnormally.

User response

Correct the statements.

FABH0687E **OPEN FAILED FOR DDNAME: *ddname***

Explanation

An attempt to open the data set that is identified by *ddname* failed.

System action

Program FABHURG1 ends abnormally.

User response

Ensure that the DD statement associated with *ddname* is the correct data set.

FABH0688E **BLKSIZE OR LRECL OF *ddname* IS TOO SMALL**

Explanation

The block size or record size of the *ddname* data set is too small. The block size is always the maximum device capacity.

System action

Program FABHURG1 ends abnormally.

User response

If the utility uses the maximum device capacity, check whether *ddname* can be allocated on a device with more track capacity. If the utility uses the block size specified on the JCL statement, check whether the block size or the record size can be increased or use the default maximum device capacity.

Record size cannot be specified.

FABH0689E **MIGRATE CONTROL STATEMENT IS NOT SPECIFIED**

Explanation

For a non-HALDB, the FABHKEYX exit routine requires to specify the MIGRATE control statement together.

System action

Program FABHURG1 ends abnormally.

User response

Add the MIGRATE control statement.

FABH0690E **ROOT SEGMENT HAS NO SEQUENCE-FIELD**

Explanation

The FABHKEYX exit routine cannot process this HDAM database because no sequence field is defined in the root segment of the HDAM database.

System action

Program FABHURG1 ends abnormally.

User response

Remove the FABHKEYX exit routine from the EXIT control statement.

FABH0691E STATUS CODE=*xx* RETURNED ON AN INTERNAL DL/I CALL

Explanation

If APISET 3 is specified in HSSROPT data set, HSSR Engine can issue a DL/I GU call internally. This message indicates that the status code *xx* is returned for one such DL/I call.

System action

HSSR Engine ends abnormally.

User response

Check the status code and detect the cause of the error. If the cause is not clear, collect the dump and contact IBM Software Support.

FABH0692E ERROR RETURN CODE *xx* IN RESPONSE TO A DL/I REQUEST ON DDNAME *ddname* FOR FUNCTION *yy*

Explanation

HSSR Engine issued an internal DL/I call to read database data set (*ddname*). Code *yy* is the content of PSTFNCTN, and code *xx* is the content of PSTRTCDE; both values are in hexadecimal.

System action

HSSR Engine ends abnormally.

User response

Check the return code and detect the cause of the error. If the cause is not clear, collect the dump and contact IBM Software Support.

FABH0693E STATUS CODE=*xx* RETURNED ON AN INTERNAL DL/I CALL

Explanation

If APISET 3 is specified in HSSROPT data set, HSSR Engine can issue a DL/I call internally. This message indicates that the status code *xx* is returned for one such DL/I call.

System action

HSSR Engine ends abnormally.

User response

Check the status code and detect the cause of the error. If the cause is not clear, collect the dump and contact IBM Software Support.

FABH0695I DL/I CALLS TO HSSR PCB (PCB#=*nnnn*) ARE PASSED TO IMS DL/I

Explanation

HSSR application program issued a DL/I call that is not fully supported by HSSR Engine. This call and the succeeding calls to the HSSR PCB are passed to the IMS DL/I call handler, because APISET 3 is specified.

System action

The processing continues. The performance, however, decreases.

User response

If you want the database calls to be fully processed by HSSR Engine, use call type supported by APISET 2.

FABH0696I "SKIPAUTH YES" IS APPLICABLE ONLY TO HALDB

Explanation

SKIPAUTH YES is specified in the HSSROPT data set. This control statement is applicable only to HALDBs.

System action

HSSR continues processing. If the database is a HALDB, IMS DBRC database authorization request is bypassed.

User response

If you want to avoid DBRC authorization failure (DFS047A) for non-HALDBs, specify DBRC=N and then rerun the job.

FABH0697E INCORRECT CABBASE CONTROL STATEMENT IS SPECIFIED

Explanation

An incorrect operand on the CABBASE control statement is specified in the HSSROPT data set.

System action

The incorrect statement is ignored.

User response

Correct the CABBASE statement.

FABH0698E INCORRECT ZIIPMODE CONTROL STATEMENT IS SPECIFIED

Explanation

An incorrect operand on the ZIIPMODE control statement is specified in the HSSROPT data set.

System action

The incorrect statement is ignored.

User response

Correct the ZIIPMODE statement.

FABH0700E FIRST CONTROL STATEMENT ID IS NOT MAP

Explanation

In the FABHPSFM step, the first control statement ID is not MAP.

System action

Program FABHPSFM ends abnormally.

User response

Correct the control statement.

FABH0701E DBDNAME IS MISSING OR HAS IMBEDDED BLANKS

Explanation

A DBD name was not specified or it contains blank characters in the MAP control statement.

System action

Program FABHPSFM ends abnormally.

User response

Correct the control statement.

FABH0702E INDEX DBDNAME HAS IMBEDDED BLANKS

Explanation

An Index DBD name contains blank characters in the MAP control statement.

System action

Program FABHPSFM ends abnormally.

User response

Correct the control statement.

FABH0703E THIS SEGMENT ENTRY IS NOT ROOT SEGMENT

Explanation

Program FABHPSFM checked the segment code of the database and found that the segment code of the first segment type is not for root segment.

System action

FABHPSFM ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct version of DBD was being used.
- Ensure that the database processed by HSSR Engine was not being updated at the time when the error occurred.

FABH0704E NO FIELD IS PRESENT IN DBD

Explanation

The number of segment fields is zero in the DBD being used.

System action

Program FABHPSFM ends abnormally.

User response

See message FABH0703E, and take an appropriate action.

FABH0705E KEY FIELD IS NOT FOUND IN DBD

Explanation

There is not any sequence field in the DBD member being used.

System action

Program FABHPSFM ends abnormally.

User response

See message FABH0703E, and take an appropriate action.

FABH0708E KEY TYPE IS NOT C, X, OR BLANK

Explanation

The specified key type in the MAP control statement is not valid.

System action

Program FABHPSFM ends abnormally.

User response

Correct the control statement.

FABH0709E SELECT KEY OPTION IS NOT A, D, E, N, V, Y, OR BLANK

Explanation

The specified select key option in the MAP control statement is not correct.

System action

Program FABHPSFM ends abnormally.

User response

Correct the control statement.

FABH0710E RELATIVE BLOCK IS MISSING OR HAS IMBEDDED BLANKS

Explanation

A relative block was not specified or it contains blank characters in the MAP control statement.

System action

Program FABHPSFM ends abnormally.

User response

Correct the control statement.

FABH0711E DELTA IS MISSING OR HAS IMBEDDED BLANKS

Explanation

A search delta was not specified or it contains blank characters in the MAP control statement.

System action

Program FABHPSFM ends abnormally.

User response

Correct the control statement.

FABH0712E INVALID INDEX DBD NAME

Explanation

An index DBD name was specified in the MAP control statement, but this DBD is not an index DBD.

System action

Program FABHPSFM ends abnormally.

User response

Correct the control statement.

FABH0713E END CONTROL STATEMENT IS MISSING OR OUT OF PLACE

Explanation

The END control statement is missing or out of place.

System action

Program FABHPSFM ends abnormally.

User response

Add or replace the END control statement.

FABH0714E INDEX TARGET SEGMENT IS NOT A ROOT SEGMENT

Explanation

The target segment of the index database is not a root segment.

System action

Program FABHPSFM ends abnormally.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBD was being used.
- Ensure that the DBDGEN statements of this DBD are correct.

FABH0715E NO HIDAM INDEX DBD NAME IS SPECIFIED

Explanation

The DBD name of the HIDAM database was specified, but an index DBD name was not specified.

System action

Program FABHPSFM ends abnormally.

User response

Specify the index DBD name in the MAP control statement.

FABH0716E INDEX FIELD NAME IS NOT FOUND IN BASE DBD FIELD TABLE

Explanation

An index field name in the index DBD was not found in the base DBD field table.

System action

Program FABHPSFM ends abnormally with dump.

User response

See message FABH0714E, and take an appropriate action.

FABH0717E INDEX FIELD IS NOT AN XFLD IN BASE DBD

Explanation

An index field for an index DBD was not defined as XFLD in the base DBD.

System action

Program FABHPSFM ends abnormally with dump.

User response

See message FABH0714E, and take an appropriate action.

FABH0719E FIRST VSAM INDEX RECORD POINTER IS ZERO

Explanation

A VSAM index record is retrieved by GET macro, and the RBA of this record is zero.

System action

Program FABHPSFM ends abnormally with dump.

User response

Check the VSAM KSDS data set. Check if the versions of both IMS and z/OS are supported by IMS High Performance Unload.

FABH0720E DBD IS NOT HIDAM OR HDAM

Explanation

Parallel Scan Facility supports only HIDAM and HDAM databases.

System action

Program FABHPSFM ends abnormally.

User response

Specify the DBD name of HIDAM or HDAM.

FABH0721E VSAM INDEX FAILURE ON OPEN, SHOWCB OR MODCB

Explanation

Return code was not zero when OPEN, SHOWCB, or MODCB macro was issued for VSAM KSDS.

System action

Program FABHPSFM ends abnormally with dump.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the VSAM KSDS is correctly defined.
- Ensure that the VSAM DD statement is correct.
- Ensure that the versions of both IMS and z/OS are supported by IMS High Performance Unload.

FABH0722E **FIND FAILED FOR MEMBER *xxxxxxxx* IN DBDLIB - R15:*yy* - R0:*zz***

Explanation

A FIND macro failed for the DBD member *xxxxxxxx* in the DBD library (IMS DD in the JCL). *yy* is a return code and *zz* is a reason code from the FIND macro. If the return code is 4 and the reason code is 0, this message shows that DBD *xxxxxxxx* was not found in the DBD library on your IMS DD statement. Refer to the description of FIND macro for other cases.

System action

Program FABHPSFM ends abnormally.

User response

Correct the error and rerun the Job.

FABH0723I **INDEX DBD LRECL IS NOT ACTUAL LRECL**

Explanation

The logical record length of the index DBD differs from the logical record length of the actual data set.

System action

Program FABHPSFM continues processing.

User response

Correct the DBD or the DD statement.

FABH0724E **VSAM DATABASE I/O FAILURE**

Explanation

The return code was not zero when OPEN or SHOWCB macro was issued for VSAM ESDS.

System action

Program FABHPSFM ends abnormally with dump.

User response

Ensure that VSAM ESDS is correctly defined. To identify the cause, see message FABH0721E.

FABH0725E **VSAM INDEX I/O FAILURE**

Explanation

The return code was not zero when GET macro was issued for VSAM KSDS.

System action

Program FABHPSFM ends abnormally with dump.

User response

See message FABH0721E, and take an appropriate action.

FABH0727I **DBD BLOCK SIZE IS NOT ACTUAL BLOCK SIZE**

Explanation

This message is issued when either one of the following occurs:

- The block size specified in the DBD is not equal to the one that is specified on the DD statement in your JCL.
- The block size specified in the DBD is not equal to the one that is held in the VTOC.

System action

Program FABHPSFM continues processing, using the block size specified in the VTOC, or block size specified in your JCL if the BLKSIZE parameter is coded in the DCB parameter.

User response

Correct the DBD or the DD statement.

FABH0729E **FABHPSFM INTERNAL ERROR**

Explanation

Program FABHPSFM detected an unexpected error.

System action

FABHPSFM ends abnormally with dump.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct version of DBD was being used.
- Ensure that the versions of both IMS and z/OS are supported by IMS High Performance Unload.

FABH0730I NO MULTIPLE EXTENTS - USE SELECT KEY OPTION

Explanation

The primary data set group has no extent.

System action

Program FABHPSFM closes the data set and ends the job step.

User response

Correct the select key option in the MAP control statement.

FABH0734I INDEX RECORD POINTER IS ZERO

Explanation

Program FABHPSFM retrieved the index records and found that the record pointer is zero.

System action

FABHPSFM continues processing.

User response

None. This message is informational.

FABH0735I NO MULTIPLE VOLUMES - USE OTHER SELECT KEY OPTIONS

Explanation

The primary data set group was allocated within one volume.

System action

Program FABHPSFM closes the data set and ends the job step.

User response

Correct the select key option in the MAP control statement.

FABH0736E GETMAIN FAILURE

Explanation

This error was caused by insufficient virtual storage in the address space.

System action

HSSR Engine ends abnormally with dump.

User response

Ensure that sufficient virtual storage is available.

FABH0737E OBTAIN MACRO FAILED FOR DDNAME *ddname* (RC=*xx*)

Explanation

The OBTAIN macro failed with a return code *xx* for the data set that is specified for the DDNAME *ddname*.

System action

Program FABHPSFM ends abnormally.

User response

Check the job log messages and other printed output for an indication that some system service has failed. If none can be found, it is likely that IMS High Performance Unload has had an internal logic error. Contact IBM Software Support.

FABH0738E RDJFCB MACRO FAILED FOR DDNAME *ddname* (RC=*xx*)

Explanation

The RDJFCB macro failed with a return code *xx* for the data set that is specified for the DDNAME *ddname*.

System action

Program FABHPSFM ends abnormally.

User response

Check the job log messages and other printed output for an indication that some system service has failed. If none can be found, it is likely that IMS High Performance Unload has had an internal logic error. Contact IBM Software Support.

FABH0740E FIRST CONTROL STATEMENT ID IS NOT DBD

Explanation

In the FABHPSFC step, the first control statement ID is not DBD.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0741E DBDNAME IS MISSING OR HAS IMBEDDED BLANKS

Explanation

A DBD name was not specified, or it contains blank characters in the DBD control statement.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0742E INDEX DBDNAME HAS IMBEDDED BLANKS

Explanation

An index DBD name contains blank characters in the DBD control statement.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0743E SEQUENCE CHECK OPTION IS NOT Y, N, OR BLANK

Explanation

The sequence check option in the DBD control statement is not Y, N, or blank.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0744E SEQUENCE ERROR PRINT OPTION IS NOT Y, N, OR BLANK

Explanation

The sequence error print option in the DBD control statement is not Y, N, or blank.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0745E SEQUENCE ERROR OPTION IS NOT A, B, OR BLANK

Explanation

The sequence error option in the DBD control statement is not A, B, or blank.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0746E SEQUENCE ERROR THRESHOLD IS NOT NUMERIC

Explanation

The value of the sequence error threshold in the DBD control statement is not numeric.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statements.

FABH0747E INVALID OPTION FIELD IS USED

Explanation

Columns 26 and 27 in the DBD control statement are not blank. Any entry is not allowed in columns 26 and 27 of the DBD control statement. This field must be blank.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0748E PSB NAME IS MISSING OR HAS IMBEDDED BLANKS

Explanation

A PSB name was not specified correctly in the PSB control statement.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0749E PCB NUMBER IS NOT NUMERIC

Explanation

The relative PCB number in the PSB control statement is not numeric.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0750E OUTPUT FORMAT IS NOT SPECIFIED

Explanation

The format of output data set to be created was not specified in the PSB control statement.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0751E DDNAME IS MISSING OR HAS IMBEDDED BLANKS

Explanation

A DD name was not specified, or it contains blank characters in the PSB control statement.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0752E EXIT ROUTING NAME HAS IMBEDDED BLANKS

Explanation

The name of the exit routine contains blank characters in the PSB control statement.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0753E SEGMENT MODIFICATION OPTION IS NOT Y, N, OR BLANK

Explanation

The segment modification option in the PSB control statement is not Y, N, or blank.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0754E CONCATENATED KEY OPTION IS NOT Y, N, OR BLANK

Explanation

The concatenated key option in the PSB control statement is not Y, N, or blank.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0755E EXIT ROUTINE CONTROL OPTION IS NOT Y, N, OR BLANK

Explanation

The exit routine control option in the PSB control statement is not Y, N, or blank.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0756E DBR SKIP OPTION IS NOT Y, N, OR BLANK

Explanation

The DBR skip option in the PSB control statement is not Y, N, or blank.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0758E END CONTROL STATEMENT IS MISSING OR OUT OF PLACE

Explanation

The END control statement is missing or it is out of place.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0759E NO PSB CONTROL STATEMENT IS PRESENT

Explanation

The PSB control statement is not specified.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0760E SCAN CONTROL DATA SET RECFM IS NOT VB

Explanation

The record format of the scan control data set is not VB.

System action

Program FABHPSFC ends abnormally.

User response

Correct the record format of the scan control data set.

FABH0761E FIND FAILED FOR MEMBER *xxxxxxxx* IN DBDLIB - R15:*yy* - R0:*zz*

Explanation

A FIND macro failed for the DBD member *xxxxxxxx* in the DBD library (IMS DD in the JCL). *yy* is a return code and *zz* is a reason code from FIND macro. If the return code is 4 and the reason code is 0, this message shows that DBD *xxxxxxxx* was not found in the DBD library on your IMS DD statement. Refer to the description of FIND macro for other cases.

System action

Program FABHPSFC ends abnormally.

User response

Correct the error and rerun the Job.

FABH0765E INVALID OPTION FIELD IS USED

Explanation

Column 30 in the DBD control statement is not blank. Any entry is not allowed in column 30 of the DBD control statement. This field must be blank.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0766E INCORRECT CONTROL STATEMENT PLACEMENT

Explanation

The sequence of control statements is not valid. The CTL control statement does not follow the DBD control statement.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0767E PARALLEL SCAN NAME IS INVALID

Explanation

The parallel scan name was not specified correctly in the CTL control statement.

System action

Program FABHPSFC ends abnormally.

User response

Correct the parallel scan name.

FABH0768E SCAN CONTROL DATA SET EXPIRATION DATE IS NOT VALID

Explanation

The expiration date of the scan control data set was not specified correctly in the CTL control statement.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0769E NUMBER OF PARALLEL SCANS IS NOT VALID

Explanation

The total number of parallel scan phases was not specified correctly in the CTL control statement.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0770E DSN SEQUENCE OFFSET IN NOT VALID

Explanation

The DSN sequence offset option was not specified correctly in the CTL control statement.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0771E FABHPSFC INTERNAL ERROR

Explanation

Program FABHPSFC detected an unexpected error.

System action

FABHPSFC ends abnormally with dump.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBDs and PSBs were being used.
- Ensure that the versions of both IMS and z/OS are supported by IMS High Performance Unload.

If necessary, contact IBM Software Support.

FABH0772E FABHPSFC INTERNAL ERROR

Explanation

Program FABHPSFC detected an unexpected error.

System action

FABHPSFC ends abnormally with dump.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the correct DBDs and PSBs were being used.
- Ensure that the database processed by FABHPSFC was not being updated at the time when the error occurred.

If necessary, contact IBM Software Support.

FABH0773E PHASE COUNT AND NODE POINT OR HIGH KEY ARE MISMATCH

Explanation

Program FABHPSFC detected an unexpected error.

System action

FABHPSFC ends abnormally.

User response

See message FABH0772E, and take an appropriate action.

FABH0774E LIMIT FIELD IS NOT NUMERIC

Explanation

The specified node point value in the NPT control statement or the high key value in the HKY control statement is not numeric.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0775E SPECIFIED KEY IS TOO LONG OR ZERO LENGTH

Explanation

The key value specified with the node point value in the NPT control statement or with the high key value in the HKY control statement is too long or zero length.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0776E NODE POINT VALUE TYPE IS NOT R, C, OR X

Explanation

The node point value type in the NPT control statement is not R, C, or X.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0777E INVALID OPTION FIELD IS USED

Explanation

Columns 31 - 38 in the DBD control statement are not blank. Any entry is not allowed in columns 31 - 38 of the DBD control statement. This field must be blank.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0778E POINTER BYPASS OPTION IS NOT 1, 2, OR BLANK

Explanation

The specified pointer bypass option in the DBD control statement is not 1, 2, or blank.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0779E INCORRECT CONTROL STATEMENT PLACEMENT

Explanation

The specified sequence of control statements is not valid. The NPT control statement does not follow the CTL control statement.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0780E FABHPSFC INTERNAL ERROR

Explanation

Program FABHPSFC detected an unexpected error.

System action

FABHPSFC ends abnormally with dump.

User response

See message FABH0771E.

FABH0781E DBD IS NOT HIDAM OR HDAM

Explanation

Parallel Scan Facility supports only HIDAM and HDAM databases.

System action

Program FABHPSFC ends abnormally.

User response

Specify the DBD name of HIDAM or HDAM.

FABH0782E SEGMENT TABLE BUILD ERROR

Explanation

Program FABHPSFC built the segment table and the segment table extension, and detected an internal error.

System action

FABHPSFC ends abnormally with dump.

User response

See message FABH0772E, and take an appropriate action.

FABH0783E INVALID OPTION FIELD IS USED

Explanation

Column 36 in the PSB control statement is not blank. Any entry is not allowed in column 36 of the PSB control statement. This field must be blank.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0784E NUMBER OF PARALLEL SCANS IS NOT EQUAL NUMBER OF NPT PLUS ONE

Explanation

The total number of scan phases specified with the CTL control statement is not equal to the number of the NPT control statements plus one.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0785E DSN CHECK OPTION IS NOT Y, N, OR BLANK

Explanation

The specified DSN check option in the CTL control statement is not Y, N, or blank.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0786E WTO OPTION IS NOT Y, N, OR BLANK

Explanation

The specified WTO option in the CTL control statement is not Y, N, or blank.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0787E INCONSISTENT TYPES OF NODE POINT OR HIGH KEY VALUE

Explanation

Program FABHPSFC detected an unexpected error.

System action

FABHPSFC ends abnormally.

User response

See message FABH0772E, and take an appropriate action. If necessary, contact IBM Software Support.

FABH0788E HDAM RELATIVE BLOCK NUMBER IS SPECIFIED BEYOND RAA

Explanation

The HDAM relative block number is specified out-of-read addressable area.

System action

Program FABHPSFC ends abnormally.

User response

Correct the node point value in the NPT control statement.

FABH0789E HDAM RELATIVE BLOCK NUMBER IS NOT IN ASCENDING SEQUENCE

Explanation

The HDAM relative block number is not in ascending order.

System action

Program FABHPSFC ends abnormally.

User response

Correct the node point value in the NPT control statement.

FABH0790E HDAM RELATIVE BLOCK NUMBER IS ZERO OR NEGATIVE

Explanation

The HDAM relative block number is zero or a negative value.

System action

Program FABHPSFC ends abnormally.

User response

Correct the node point value in the NPT control statement.

FABH0791E VALUE TYPE C OR X IS NOT VALID FOR HDAM WITHOUT INDEX

Explanation

The specified node point value type in the NPT control statement or the high key value type in the HKY control statement is not valid for an HDAM database that has no secondary index.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0792E NODE POINT KEY OR HIGH KEY IS NOT IN ASCENDING SEQUENCE

Explanation

The node point key or the high key that is specified with the value type C or X is not in ascending order.

System action

Program FABHPSFC ends abnormally.

User response

Correct the node point value in the NPT control statement or the high key value in the HKY control statement.

FABH0793E BLOCK SIZE IS TOO SMALL FOR CONTROL SCAN PHASES

Explanation

The block size of the scan control data set is too small to build control records for all scan phases.

System action

Program FABHPSFC ends abnormally.

User response

Create a new scan control data set.

FABH0794E SEPARATE HEADER OPTION IS NOT Y, N, OR BLANK

Explanation

The specified separate header option in the CTL control statement is not Y, N, or blank.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0795E GETMAIN FAILURE

Explanation

Virtual storage in the address space was insufficient.

System action

Program FABHPSFC ends abnormally with dump.

User response

Ensure that sufficient virtual storage is available.

FABH0796E NUMBER OF PARALLEL SCANS IS NOT EQUAL TO THE NUMBER OF HKY

Explanation

The total number of scan phases that was specified in columns 19 and 20 of the CTL control statement is not equal to the number of the HKY control statements. These numbers must be equal.

System action

Program FABHPSFC ends abnormally.

User response

Make the number of scan phases that is specified on the CTL control statement the same as the number of the HKY control statements and rerun the job.

FABH0797E A HIGH KEY CANNOT BE SPECIFIED FOR THIS UNLOAD FORMAT

Explanation

A high key (HKY) is specified for an unload format that is not supported. A high key can only be specified when the MI option is specified for the PSB control statement.

System action

Program FABHPSFC ends abnormally.

User response

Correct the control statement.

FABH0800E FIRST CONTROL STATEMENT ID IS NOT SUM

Explanation

In FABHPSFS step, the first control statement ID is not SUM.

System action

Program FABHPSFS ends abnormally.

User response

Correct the control statement.

FABH0801E DBDNAME IS NOT VALID

Explanation

A DBD name was not specified correctly in the SUM control statement.

System action

Program FABHPSFS ends abnormally.

User response

Correct the control statement.

FABH0802E SCAN NAME IS NOT VALID

Explanation

The specified parallel scan name in the SUM control statement is not valid.

System action

Program FABHPSFS ends abnormally.

User response

Correct the control statement.

FABH0803E SUM OPTION IS NOT STATUS, RERUN, OR FORCE

Explanation

The acceptable option keyword RERUN, STATUS, or FORCE is not specified in the SUM control statement.

System action

Program FABHPSFS ends abnormally.

User response

Correct the control statement.

FABH0804E HEADER/TRAILER RE-CREATE OPTION IS NOT Y, N, H, T, OR BLANK

Explanation

The specified header/trailer re-create option is not Y, N, H, T, or blank in the SUM control statement.

System action

Program FABHPSFS ends abnormally.

User response

Correct the control statement.

FABH0806E END CONTROL STATEMENT IS MISSING OR OUT OF PLACE

Explanation

The END control statement is missing or is out of place.

System action

Program FABHPSFS ends abnormally.

User response

Correct the control statement.

FABH0807E DBD IS NOT HIDAM OR HDAM

Explanation

Parallel Scan Facility supports only HIDAM and HDAM databases.

System action

Program FABHPSFS ends abnormally.

User response

Specify the DBD name of HIDAM or HDAM.

FABH0808E INVALID SEGMENT WAS FOUND

Explanation

Program FABHPSFS detected an unexpected segment.

System action

FABHPSFS ends abnormally with dump.

User response

See message FABH0772E, and take an appropriate action.

FABH0810E HEADER/TRAILER DATA SET RECFM IS NOT VB

Explanation

The record format of the header or trailer data set is not VB.

System action

Program FABHPSFS ends abnormally.

User response

Correct the record format of the header or trailer data set.

FABH0811E TRAILER DATA SET RECFM IS NOT VB

Explanation

The record format of the trailer data set is not VB.

System action

Program FABHPSFS ends abnormally.

User response

Correct the record format of the trailer data set.

FABH0812E FIND FAILED FOR MEMBER *xxxxxxxx* IN DBDLIB - R15:*yy* - R0:*zz*

Explanation

A FIND macro failed for the DBD member *xxxxxxxx* in the DBD library (IMS DD in the JCL). *yy* is a return code and *zz* is a reason code from FIND macro. If the return code is 4 and the reason code is 0, this message shows that DBD *xxxxxxxx* was not found in the DBD library on your IMS DD statement. Refer to the description of FIND macro for other cases.

System action

Program FABHPSFS ends abnormally.

User response

Correct the error and rerun the Job.

FABH0813E END CONTROL STATEMENT IN SCAN CONTROL DATA SET IS NOT FOUND

Explanation

The END control statement in the scan control data set cannot be found, or it is incorrect.

System action

Program FABHPSFS ends abnormally with dump.

User response

Check the result of the previous FABHPSFC step and the FABHFSU parallel scan steps.

FABH0814E SEGMENT NAME IS MISMATCH

Explanation

Program FABHPSFS built segment tables and updated them from statistics records, but detected that the segment name was not a match.

System action

FABHPSFS ends abnormally with dump.

User response

See message FABH0772E, and take an appropriate action.

FABH0815E READ JFCB FAILURE

Explanation

Program FABHPSFS issued RDJFCB command but did not complete successfully.

System action

FABHPSFS ends abnormally with dump.

User response

See message FABH0772E, and take an appropriate action.

FABH0816E HEADER/TRAILER DSN IS NOT VALID

Explanation

DSN of the header or trailer data set is not valid.

System action

Program FABHPSFS ends abnormally with dump.

User response

Correct the control statement.

FABH0817I TRAILER PREVIOUSLY CREATED - NO RERUN OPTION SPECIFIED

Explanation

The trailer data set was already created but the RERUN option was not specified in the SUM control statement.

System action

Program FABHPSFS continues processing.

User response

None. This message is informational.

FABH0818E FABHPSFS INTERNAL ERROR

Explanation

Program FABHPSFS detected an unexpected error.

System action

FABHPSFS ends abnormally with dump.

User response

See message FABH0772E, and take an appropriate action.

FABH0819E SCAN NAME IS MISMATCH ON SCAN CONTROL DATA SET

Explanation

The specified parallel scan name in the SUM control statement is not equal to the scan name in the scan control data set.

System action

Program FABHPSFS ends abnormally.

User response

Ensure that the scan control data set is correct. Correct the DD statement of the scan control data set or correct the control statement.

FABH0820E DBD NAME IS MISMATCH ON SCAN CONTROL DATA SET

Explanation

The specified DBD name in the SUM control statement is not equal to the DBD name in the scan control data set.

System action

Program FABHPSFS ends abnormally.

User response

See message FABH0819E, and take an appropriate action.

FABH0821E SCAN CONTROL DATA SET OPEN FAILURE

Explanation

The scan control data set open failure occurred.

System action

Program FABHPSFS ends abnormally with dump.

User response

Complete the following tasks to identify the cause, and take appropriate actions:

- Ensure that the DD statement is correct.
- Ensure that the versions of both IMS and z/OS are supported by IMS High Performance Unload.

FABH0822E HEADER DATA SET OPEN FAILURE

Explanation

The header data set open failure occurred.

System action

Program FABHPSFS ends abnormally with dump.

User response

See message FABH0821E.

FABH0823E TRAILER DATA SET OPEN FAILURE

Explanation

The trailer data set open failure occurred.

System action

Program FABHPSFS ends abnormally with dump.

User response

See message FABH0821E, and take an appropriate action.

FABH0824E GETMAIN FAILURE

Explanation

This error was caused by insufficient virtual storage in the address space.

System action

Program FABHPSFS ends abnormally with dump.

User response

Ensure that sufficient virtual storage is available.

FABH0825I FORCE OPTION REJECTED - A STARTED PHASE INCOMPLETE

Explanation

The FORCE option was specified in the SUM control statement, but the phase process specified by the DD statement was not completed.

System action

Program FABHPSFS continues processing.

User response

None. This message is informational.

FABH0826I RUN TIME ENVIRONMENT EXIT ROUTINE IS BEING INVOKED, MODULE=*xxxxxxxx*

Explanation

The runtime exit routine whose name is *xxxxxxxx* is invoked.

System action

HSSR Engine continues processing.

User response

None. This message is informational.

FABH0827E ERROR OCCURRED IN RTEXIT, MODULE=*xxxxxxxx*, FUNC=*yyyy*, RC=*zz*

Explanation

HSSR Engine detected a nonzero return code *zz* from the runtime environment exit routine *xxxxxxxx*.

System action

HSSR Engine ends abnormally.

User response

Determine the reason for the nonzero return code from the runtime exit routine. Run the job again after correcting the problem, if necessary.

FABH0828W NO SEGMENT WAS RETRIEVED

Explanation

No segment was retrieved in all PSF phases.

System action

Program FABHPSFS normally ends with a return code of 01.

User response

Check whether it is acceptable that the input database has no valid segments. If a wrong database data set or a wrong DBD or PSB was specified, specify the appropriate ones, and rerun the job.

FABH0829W INCONSISTENT DEC OPTIONS WERE SPECIFIED IN SCAN PHASES

Explanation

The segment decompression option "DEC" specified in the CARDIN data set of FABHFSU in each scan phase was not consistent.

System action

Program FABHPSFS continues processing.

User response

The unloaded data sets created by these scan phases cannot be used to load the database. The DEC option specified in each scan phase must be consistent if you want to use the created data sets for reloading.

FABH0830E FIRST PCB IS NOT A VALID HSSR PCB

Explanation

Program FABHLDBR detected that the first PCB in the PSB was not a valid HSSR PCB.

System action

FABHLDBR ends abnormally.

User response

See "HSSR PCB requirements" on page 80, and correct the error.

FABH0831E AN INVALID STATEMENT IS FOUND IN SYSIN DATA SET

Explanation

An incorrect control statement type was detected in the FABHLDBR SYSIN data set.

System action

Program FABHLDBR ends abnormally.

User response

Correct the control statement.

FABH0832E NON-POSITIVE NUMBER IS SPECIFIED FOR *xxxxxxxx* CONTROL STATEMENT

Explanation

A non-positive numeric value is specified for the operand of the *xxxxxxxx* control statement in the SYSIN data set of the FABHLDBR utility.

System action

Program FABHLDBR ends abnormally.

User response

Correct the control statement.

FABH0833E THE OPERAND OF *xxxxxxxx* CONTROL STATEMENT IS NOT NUMERIC

Explanation

A non-numeric value is specified for the operand of the *xxxxxxxx* control statement in the SYSIN data set of the FABHLDBR utility.

System action

Program FABHLDBR ends abnormally.

User response

Correct the control statement.

FABH0834E OPEN OF SYSUT1 HAS FAILED

Explanation

Program FABHLDBR could not open SYSUT1.

System action

FABHLDBR ends abnormally.

User response

Correct the SYSUT1 DD statement.

FABH0835E INVALID PARAMETER IS FOUND IN AN LOUT CONTROL STATEMENT

Explanation

An undefined parameter was detected in the LOUT control statement. Or, the value specified by the LENGTH= or IO= parameter of the LOUT control statement was incorrect.

System action

The incorrect LOUT parameter and any parameters that follow it on the same control statement, are ignored.

User response

Correct the LOUT control statement.

FABH0836E A NON-NUMERIC VALUE IS SPECIFIED IN HSSRLDEF DATA SET

Explanation

A record that contains a non-numeric value is found in the HSSRLDEF data set.

System action

The incorrect record is ignored.

User response

Correct the incorrect record.

FABH0837E A NUMERIC FIELD IS TOO LONG IN HSSRLDEF DATA SET

Explanation

A numeric value that is too long is specified in the HSSRLDEF data set.

System action

The record that has the incorrect numeric field is ignored.

User response

Correct the statement that contains the incorrect field.

FABH0838E LENGTH-DEFINITIONS IN HSSRLDEF ARE NOT IN ASCENDING SEQUENCE

Explanation

The numeric values in the HSSRLDEF data set were not in ascending sequence.

System action

The incorrect numeric field is ignored.

User response

Correct the statement that contains the incorrect field, so that the values will be in ascending sequence.

FABH0839I *work_area* EXCEEDS 4GB (*yyyyyyyyyyyyyyyyyy*)

Explanation

The value of the indicated internal work area exceeds the maximum limit. As a result, the database record length that is shown in some reports might be shorter than the actual length.

System action

HSSR Engine continues processing.

User response

None. This message is informational.

FABH0840E FABHEXTR FAILED TO OPEN HSSREXTR DATA SET

Explanation

FABHEXTR could not open the HSSREXTR data set.

System action

Program FABHURG1 ends abnormally.

User response

Correct the HSSREXTR DD statement.

FABH0841E INVALID STATEMENT IS FOUND IN HSSREXTR DATA SET

Explanation

An incorrect control statement was specified in the HSSREXTR data set.

System action

Program FABHURG1 ends abnormally.

User response

Remove or correct the control statement.

FABH0842E NUMERIC FIELD IS NOT NUMERIC

Explanation

A non-numeric value was specified for a numeric field in a control statement in the HSSREXTR data set.

System action

Program FABHURG1 ends abnormally.

User response

Correct the control statement.

FABH0843E NUMBER OF DB RECORDS TO BE EXTRACTED IS [NOT POSITIVE| TOO LARGE]

Explanation

The EXTR or PARTEXTR control statement specified a numeric value that was not positive or was too large.

System action

Program FABHURG1 ends abnormally.

User response

Correct the control statement.

FABH0844E NEITHER EXTR NOR PARTEXTR IS SPECIFIED

Explanation

Neither the EXTR control statement nor the PARTEXTR control statement is specified in the HSSREXTR data set. One or the other must be specified.

System action

Program FABHURG1 ends abnormally.

User response

Specify either EXTR or PARTEXTR.

FABH0845E DATABASE IS EMPTY

Explanation

Program FABHEXTR detected that FABHURG1 reached the end of a database, but no segment was retrieved. The database might be empty.

System action

FABHURG1 ends abnormally.

User response

Check whether the correct DBD and database data set are specified.

FABH0846E TOO MANY DB RECORDS ARE SKIPPED

Explanation

Program FABHEXTR skipped too many database records and reached the end of the database without retrieving any segments. The unloaded data set will be empty.

System action

Program FABHURG1 ends abnormally.

User response

Check whether a correct DBD and database data set are specified. If the database specification is correct, check whether the value specified on the SKIP control statement in the HSSREXTR data set is less than the total number of database records in the database.

FABH0847E *xxxxxxxx* IS SPECIFIED FOR NON-HALDB

Explanation

xxxxxxxx control statement is specified for a database that is not a HALDB.

System action

Program FABHURG1 ends abnormally.

User response

Check whether the correct DBD was specified.

FABH0848E DUPLICATE STATEMENTS: *xxxxxxxx*

Explanation

The control statement *xxxxxxxx* is specified more than once.

System action

Program FABHURG1 ends abnormally.

User response

Remove unnecessary statements, and rerun the job.

FABH0849E MUTUALLY EXCLUSIVE CONTROL STATEMENTS ARE SPECIFIED: *xxxxxxxx* AND *yyyyyyyy*

Explanation

The control statements *xxxxxxxx* and *yyyyyyyy* are mutually exclusive.

System action

Program FABHURG1 ends abnormally.

User response

Select only one of these statements, and remove the other.

FABH0850E LOAD FAILED FOR RTEXIT (*xxxxxxxx*), CC=*yyyy*, RC=*zz*

Explanation

A LOAD error occurred when HSSR Engine tried to load the runtime exit routine *xxxxxxxx*. As a result, the ERRET routine received the control. *yyyy* indicates the system completion code at time of error, and *zz* indicates the error reason code for the LOAD request.

System action

HSSR Engine ends abnormally.

User response

Check the system completion code and the error reason code. If you find the reason for the error, correct it and rerun the job.

FABH0851E LOAD FAILED FOR APPL PGM (*xxxxxxxx*), CC=*yyyy*, RC=*zz*

Explanation

A LOAD error occurred when HSSR Engine tried to load the application program *xxxxxxxx*. As a result, the ERRET routine received the control. *yyyy* indicates the system completion code at time of error, and *zz* indicates the error reason code for the LOAD request.

System action

HSSR Engine ends abnormally.

User response

Check the system completion code and the error reason code. If you find the reason for the error, correct it and rerun the job.

FABH0852E LOAD FAILED FOR MODULE DFSDBUX1

Explanation

The user specified DATXEXIT=YES at DBDGEN time, but DFSDBUX1 is not found in any STEPLIB library. At execution time, while processing a DBD requiring the exit, HSSR determined that the exit could not be loaded, and issued the message.

System action

HSSR Engine ends abnormally.

User response

If the user exit (DFSDBUX1) is required for the job, make sure that DFSDBUX1 is placed into a STEPLIB library, and rerun the job. If you do not need DFSDBUX1 for this job, but need it for another application, specify 'DATXEXIT NO' in the HSSROPT data set for the job. If you do not need DFSDBUX1 at all for the database you are processing, remove the DATXEXIT=YES from DBDGEN.

FABH0853E IMODULE ERROR (*rc*) *function* COMPRESSION EXIT (*xxxxxxxx*)

Explanation

An IMODULE *function* (LOADING or DELETING) error occurred when HSSR Engine tried to load or delete the segment edit/compression routine *xxxxxxxx*. *rc* is the error return code from IMODULE. For the meanings of the return codes, read about IMODULE return codes in *IMS Messages and Codes*.

System action

HSSR Engine ends abnormally.

User response

Check the error return code. If you find the reason for the error, correct it and rerun the job.

FABH0854E LOAD FAILED FOR FABHRCEX EXIT, CC=*xxxx*, RC=*yy*

Explanation

A LOAD error occurred when HSSR Engine tried to load the Return Code Edit exit routine (FABHRCEX). As a result, the ERRET routine received control. *xxxx* is the system completion code at time of error, and *yy* is the error reason code for the LOAD request.

System action

HSSR Engine ends abnormally.

User response

Check the system completion code and the error reason code. If you find the reason for the error, correct it and rerun the job.

FABH0855E COMPRESSION EXIT ROUTINE *nnnnnnnn* INITIALIZATION ERROR - U*aaaa* REASON *rrrrrrrr*

Explanation

An initialization error was detected by a Segment Edit/Compression exit routine. In the message text:

nnnnnnnn
Name of the exit routine.

U*aaaa*
The IMS user abend code generated by the exit routine.

rrrrrrrr
The unique label at which the error was detected. This label corresponds to the error reason code. Find the meaning of the reason code in the user's guide of the Segment Edit/Compression exit routine or contact the supplier of the Segment Edit/Compression exit routine.

System action

HSSR Engine ends abnormally.

User response

Determine the cause of the error and correct the problem.

FABH0856E LOAD FAILED FOR GEXAPI00

Explanation

A LOAD error occurred when HSSR Engine tried to load the GEXAPI00 module.

System action

If ZIIPMODE=FORCE is specified, HSSR Engine ends abnormally. If ZIIPMODE=COND is specified, processing continues.

User response

Ensure that the SGLXLOAD library of IMS Tools Base is specified to the STEPLIB or JOBLIB.

If the SGLXLOAD library is specified, ensure that the GEXAPI00 module exists and is available. If it is not available, make it available and rerun the job.

FABH0860E X'FF' IS SET IN SRCHFLAG BY DFSDBUX1

Explanation

The user specified DATXEXIT=YES at DBDGEN time, but the Data Conversion exit set SRCHFLAG to X'FF'.

System action

HSSR Engine ends abnormally.

User response

If you do not need DFSDBUX1 for this HSSR job, but need it for another application, specify 'DATXEXIT NO' in the HSSROPT data set for the HSSR job. If you need DFSDBUX1 for the database you are processing, define table entries for the database in the translation table of the Data Conversion exit routine.

FABH0861I DBDGEN REQUIRED FOR DATABASE *xxxxxxxx* TO SET DATXEXIT INDICATOR

Explanation

While HSSR Engine was processing the first HSSR call for a database whose DATXEXIT=YES flag is not on, the Data Conversion exit routine (DFSDBUX1) was called and the routine returned to HSSR Engine without SRCHFLAG set to X'FF'. This indicates that the Data Conversion exit routine was required for this database. HSSR Engine dynamically sets the DATXEXIT=YES flag on and continues processing for this database, but issues this message to warn the user that a DBDGEN with DATXEXIT=YES needs to be done for this database.

System action

HSSR Engine continues processing.

User response

The database administrator needs to be notified that a DBDGEN is required for this database.

FABH0862E NON-ZERO RC (*xx*) RETURNED FROM CONV EXIT

Explanation

Non-zero return code *xx* is returned from the Data Conversion exit routine (DFSDBUX1).

System action

HSSR Engine ends abnormally.

User response

Check whether the correct Data Conversion exit routine is used. If a conditional exit routine is specified for the database you are processing, check also whether the correct one is used. If you cannot determine the cause of the problem, record the message and contact IBM Software Support.

FABH0870E LOAD FAILED FOR IDCAMS

Explanation

A LOAD error occurred when HSSR Engine tried to load the IDCAMS module.

System action

HSSR Engine ends abnormally.

User response

Ensure that the IDCAMS module is available on your JOB, and also that it exists in the LPA list, LINK list, or STEPLIB or JOBLIB path.

FABH0871E LISTCAT FAILED FOR DATA SET *dsname* (*dbdname*, *partname*, *ddname*), RC=*rc*

Explanation

An error occurred when the LISTCAT command of the program IDCAMS was run for data set *dsname*. Value *rc* shows the return code from program IDCAMS.

System action

HSSR Engine ends abnormally.

User response

See *DFSMS Access Method Services for Catalogs* for the meaning of the return code. Correct the error, and rerun the job. If the cause is not clear, collect the dump and contact IBM Software Support.

FABH0872I DATA SET *dsname* IS NOT CATALOGED (DDN=*ddname*)

Explanation

The data set specified is not cataloged. The string *ddname* shows the DD name with which the data set is associated by the partition definition.

System action

HSSR Engine continues processing.

User response

See the explanation and the programmer response of message FABH0873I.

FABH0873I PARTITION *partname* OF HALDB *dbdname* WILL NOT BE PROCESSED

Explanation

This message is preceded by one or more FABH0872I messages. HSSR will not process the partition *partname* of HALDB *dbdname* because one or more data sets that are defined for partition *partname* are not cataloged.

System action

HSSR Engine continues processing.

User response

If the partition needs to be processed, you must catalog all data sets for partition *partname*.

FABH0881I ***applname*** **ENDED WITH RC=*xx*, WHICH MIGHT BE CHANGED BY FABHRCEX EXIT**

Explanation

The HSSR application program *applname* ended with a return code of *xx*, but the return code has been processed and might be edited by the user specified Return Code Edit exit routine (FABHRCEX). If the return code is changed by the routine, you will see a code other than *xx* at the completion of IMS High Performance Unload job step.

System action

HSSR Engine continues processing.

User response

If the original return code *xx* is not zero, check the meaning of the return code by referring to "Return codes" on page 379.

FABH1000I DFSISVI0 EXIT IS CALLED BUT FABHURG1 CANNOT BE INVOKED

Explanation

This message notifies the user that the IMS exit DFSISVI0 is called but FABHURG1 unload utility cannot be invoked because of the error indicated in the prior message.

System action

IMS continues the processing and DFSURGU0 will be invoked.

User response

None. This message is informational.

FABH1001I DFSISVI0 EXIT IS CALLED AND FABHURG1 IS BEING INVOKED

Explanation

This message notifies the user that the IMS exit DFSISVI0 is called, and FABHURG1 unload utility will be invoked by IMS batch region controller as the unload program.

System action

IMS continues processing. Unload function is provided by the FABHURG1 unload utility.

User response

None. This message is informational.

FABH2001E INVALID COMPAT= PARAMETER IS SPECIFIED

Explanation

The value of the COMPAT= keyword parameter is incorrect. It must be one of HPU, DBT, and 5787LAC.

System action

The assembly of the option table ends with a return code of 8. The option table is not replaced.

User response

Correct the error.

FABH2002E INVALID DIAGG= PARAMETER IS SPECIFIED

Explanation

The value of the DIAGG= keyword parameter is incorrect. It must be DIAGONLY, CB, BUF, or NOINT.

System action

The assembly of the option table ends with a return code of 8. The option table is not replaced.

User response

Correct the error.

FABH2003E INVALID CABSTAT= PARAMETER IS SPECIFIED

Explanation

The value of the CABSTAT= keyword parameter is incorrect. It must be either YES or NO.

System action

The assembly of the option table ends with a return code of 8. The option table is not replaced.

User response

Correct the error.

FABH2004E INVALID LSR= PARAMETER IS SPECIFIED

Explanation

The value of the LSR= keyword parameter is incorrect. It must be either YES or NO.

System action

The assembly of the option table ends with a return code of 8. The option table is not replaced.

User response

Correct the error.

FABH2005E INVALID URG1DEC= PARAMETER IS SPECIFIED

Explanation

The value of the URG1DEC= keyword parameter is incorrect. It must be either YES or NO.

System action

The assembly of the option table ends with a return code of 8. The option table is not replaced.

User response

Correct the error.

FABH2006E INVALID FSUDEC= PARAMETER SPECIFIED

Explanation

The value of the FSUDEC= keyword parameter is incorrect. It must be either YES or NO.

System action

The assembly of the option table ends with a return code of 8. The option table is not replaced.

User response

Correct the error.

FABH2007E DUPLICATED FABHTOPT STATEMENTS ARE FOUND

Explanation

The FABHTOPT statement was specified more than once. Only one FABHTOPT statement is allowed.

System action

The assembly of the option table ends with a return code of 12. The option table is not replaced.

User response

Correct the error.

FABH2008E INVALID BUFDEFAULT= PARAMETER IS SPECIFIED

Explanation

The value of the BUFDEFAULT= keyword parameter is incorrect. It must be either CAB or BB.

System action

The assembly of the option table ends with return code 8. The option table is not replaced.

User response

Correct the error.

FABH2009E INVALID CABDEFAULT= PARAMETER IS SPECIFIED

Explanation

The value of the CABDEFAULT= keyword parameter is incorrect. It must be either HPU or DBT.

System action

The assembly of the option table ends with return code 8. The option table is not replaced.

User response

Correct the error.

FABH2010E INCORRECT APISET= PARAMETER IS SPECIFIED

Explanation

The value of the APISET= keyword parameter is incorrect. It must be 1, 2, or 3.

System action

The assembly of the option table ends with return code 8. The option table is not replaced.

User response

Correct the error.

FABH2011E INCORRECT PCBLIST= PARAMETER IS SPECIFIED

Explanation

The value specified for the PCBLIST= keyword parameter is incorrect. It must be HSSR or IMS.

System action

The assembly of the option table ends with return code 8. The option table is not replaced.

User response

Correct the error.

FABH2012E INCORRECT URG1BUFNO= PARAMETER IS SPECIFIED

Explanation

The value specified for the URG1BUFNO= keyword parameter is incorrect. It must be in the range of 1 - 255.

System action

The assembly of the option table ends with return code 8. The option table is not replaced.

User response

Correct the error.

FABH2013E INCORRECT FSUBUFNO= PARAMETER IS SPECIFIED

Explanation

The value specified for the FSUBUFNO= keyword parameter is incorrect. It must be in the range of 1 - 255.

System action

The assembly of the option table ends with return code 8. The option table is not replaced.

User response

Correct the error.

FABH2014E INCORRECT URG1CHKRC= PARAMETER IS SPECIFIED

Explanation

The value specified for the URG1CHKRC= keyword parameter is incorrect. It must be either YES or NO.

System action

The assembly of the option table ends with return code 8. The option table is not replaced.

User response

Correct the error.

FABH2015E INCORRECT COMPAUTH= PARAMETER IS SPECIFIED

Explanation

The value specified for the COMPAUTH keyword is incorrect. It must be YES or NO.

System action

The assembly of the option table ends with return code 8. The option table is not replaced.

User response

Correct the value of the COMPAUTH keyword.

FABH2016E INCORRECT CABBASE_*xx*= PARAMETER IS SPECIFIED

Explanation

The value specified for the CABBASE_*xx*= keyword parameter is incorrect. It must be in the range of 1 - 255.

System action

The assembly of the option table ends with return code 8. The option table is not replaced.

User response

Correct the error.

FABH2017E INCORRECT ZIIPMODE= PARAMETER IS SPECIFIED

Explanation

The value specified for the ZIIPMODE keyword parameter is incorrect.

System action

The assembly of the option table ends with return code 8. The option table is not replaced.

User response

See "ZIIPMODE control statement" on page 178 and correct the value on the ZIIPMODE keyword.

FABI messages

Use the information in these messages to help you diagnose and solve Sequential Subset Randomizer utility problems.

FABI0001E IDTYPE= PARAMETER IS NEITHER C NOR X

Explanation

The value of the IDTYPE= keyword parameter must be either C or X.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

Correct the error.

FABI0002E VALTYPE= PARAMETER IS NEITHER E NOR H

Explanation

The value of the VALTYPE= keyword parameter must be either E or H.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

Correct the error.

FABI0003E BYTES= PARAMETER IS NOT NUMERIC

Explanation

The value of the BYTES= keyword parameter must be numeric.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

Correct the error.

FABI0004E START= PARAMETER IS NOT NUMERIC

Explanation

The value of the START= keyword parameter must be numeric.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

Correct the error.

FABI0005E START= PARAMETER CANNOT BE ZERO

Explanation

The value of the START= keyword parameter must be a number greater than zero.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

Correct the error.

FABI0006E DBNBYTES= PARAMETER IS NOT NUMERIC

Explanation

The value of the DBNBYTES= keyword parameter must be numeric.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

This parameter is not supported by the IMS High Performance Unload Sequential Subset Randomizer utility.

FABI0007E DBNBYTES= PARAMETER IS TOO LONG

Explanation

The value of the DBNBYTES= keyword parameter cannot be greater than 8.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

This parameter is not supported by the IMS High Performance Unload Sequential Subset Randomizer utility.

FABI0008E DBNBYTES= PARAMETER CANNOT BE ZERO

Explanation

The value of the DBNBYTES= keyword parameter cannot be 0.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

This parameter is not supported by the IMS High Performance Unload Sequential Subset Randomizer utility.

FABI0009E DBNSTART= PARAMETER IS NOT NUMERIC

Explanation

The value of the DBNSTART= keyword parameter must be a number.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

This parameter is not supported by the IMS High Performance Unload Sequential Subset Randomizer utility.

FABI0010E DBNSTART= PARAMETER CANNOT BE ZERO

Explanation

The value of the DBNSTART= keyword parameter cannot be zero.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

This parameter is not supported by the IMS High Performance Unload Sequential Subset Randomizer utility.

FABI0011E DBNSTART= PARAMETER IS TOO LONG

Explanation

The value of the DBNSTART= keyword parameter cannot be greater than 8.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

This parameter is not supported by the IMS High Performance Unload Sequential Subset Randomizer utility.

FABI0021E DBNAME= KEYWORD MISSING ON FABIDBN MACRO

Explanation

The DBNAME= macro is missing on the FABIDBN macro.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

This parameter is not supported by the IMS High Performance Unload Sequential Subset Randomizer utility.

FABI0022E DBNAME= LONGER THAN DBNBYTES OF FABITAB MACRO

Explanation

The length of DBNAME= parameter value must not exceed the length specified on the DBNBYTES= parameter of the FABITAB macro.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

This parameter is not supported by the IMS High Performance Unload Sequential Subset Randomizer utility.

FABI0023E SPLIT= KEYWORD MISSING ON FABIDBN MACRO

Explanation

The SPLIT= keyword parameter is missing on the FABIDBN macro.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

This parameter is not supported by the IMS High Performance Unload Sequential Subset Randomizer utility.

FABI0024E SPLIT= KEYWORD NOT NUMERIC

Explanation

The value of the SPLIT= keyword parameter must be numeric.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

This parameter is not supported by the IMS High Performance Unload Sequential Subset Randomizer utility.

FABI0025E SPLIT= KEYWORD CANNOT BE ZERO

Explanation

The value of the SPLIT= keyword parameter must not be zero.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

This parameter is not supported by the IMS High Performance Unload Sequential Subset Randomizer utility.

FABI0031E ID= KEYWORD MISSING ON FABIDEF MACRO

Explanation

The mandatory ID= keyword parameter is missing on the FABIDEF macro.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

Correct the error.

FABI0032E LENGTH OF ID= PARAMETER NOT EQUAL TO THE BYTES= VALUE ON THE FABITAB MACRO

Explanation

The length of the ID= parameter value on the FABIDEF macro is not equal to the value specified on the BYTES= parameter of the FABITAB macro.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

Correct the error.

FABI0033E FABIDEF MACROS NOT IN ASCENDING ID-SEQUENCE

Explanation

The FABIDEF macros must be provided in the ascending sequence of the ID= parameter values.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

Correct the error.

FABI0034E ID= PARAMETER NOT A VALID HEXADECIMAL VALUE

Explanation

The value of the ID= parameter is not a valid hexadecimal value.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

Correct the error.

FABI0035E FABIDEF MACROS NOT IN ASCENDING HEX ID-SEQUENCE

Explanation

The FABIDEF macros must be provided in the ascending sequence of the hexadecimal values of the ID= parameters.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

Correct the error.

FABI0036E UNITS= KEYWORD MISSING ON FABIDEF MACRO

Explanation

The mandatory UNITS= keyword was not specified on the FABIDEF macro.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

Correct the error.

FABI0037E VALUE OF UNITS= PARAMETER NOT NUMERIC

Explanation

The UNITS= parameter must specify a numeric value.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

Correct the error.

FABI0041E NO CORRECT FABIDEF MACRO

Explanation

Any correct FABIDEF macro was not provided by the user before the FABIGEN macro.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

Correct the error.

FABI0042E NO CORRECT FABISPL MACRO

Explanation

The FABISPL macro is not for the use of users.

System action

The generation of the randomizer will not be successful. The return code is 8.

User response

This parameter is not supported by the IMS High Performance Unload Sequential Subset Randomizer utility.

FABI0043E NUMBER OF FABIDEF MACROS IS EXCESSIVE

Explanation

More FABIDEF macros than currently supported were specified.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

See "FABIDEF macro statement" on page 283 and ensure that the number of FABIDEF macro statements does not exceed the allowable maximum.

FABI0044E **SPLIT-NUMBER FOR DB *xxxxxxxx(n)* IS HIGHER THAN THE NUMBER OF DB-SPLITS DEFINED BY FABISPL MACRO**

Explanation

The FABISPL macro is not for the use of users.

System action

The generation of the randomizer will not be successful. The return code is 12.

User response

This parameter is not supported by the IMS High Performance Unload Sequential Subset Randomizer utility.

FABI0051E **NO SUCCESSFUL FABIDEF MACRO**

Explanation

Any correct FABIDEF macro was not specified by the user.

System action

The generation of the randomizer will not be successful. The return code is 8.

User response

Correct the error.

FABI0101E **DATABASE IS NOT AN HDAM DATABASE WITH THE SS-RANDOMIZER**

Explanation

The FABIUNLS utility can be used only to split an unloaded database, if the database is an HDAM database using a Sequential Subset Randomizer.

System action

The FABIUNLS utility will issue an abend.

User response

Correct the error.

FABI0102E **PROBLEMS WITH TABLE OF SS-RANDOMIZER**

Explanation

The FABIUNLS utility detected an unexpected situation while trying to analyze the control blocks of Sequential Subset Randomizer.

System action

The FABIUNLS utility will issue an abend.

User response

Save the dump, the JES log, the DBDGEN output, and the output of the randomizer generation. Then contact IBM Software Support.

FABI0103E ***xxxxxxxxx* COULD NOT BE OPENED**

Explanation

The FABIUNLS utility could not open the indicated *xxxxxxxxx* data set (such as the HDR, TRL, FKDx, SYSPRINT, or SYSUT1 data set).

System action

The FABIUNLS utility will issue an abend.

User response

Check for other error messages that describe reasons for the problems with opening the data sets.

FABI0104E **SYSUT1 DOES NOT BEGIN WITH A HEADER RECORD**

Explanation

The FABIUNLS utility detected that the SYSUT1 data set does not start with a header record.

System action

The FABIUNLS utility will issue an abend.

User response

Provide an appropriate SYSUT1 data set. The SYSUT1 data set must contain a database unloaded in the format of the IMS HD Unload Utility.

FABI0105E SYSUT1 DOES NOT END WITH A TRAILER RECORD

Explanation

The FABIUNLS utility detected that the SYSUT1 data set does not end with a trailer record.

System action

The FABIUNLS utility will issue an abend.

User response

Provide an appropriate SYSUT1 data set. The SYSUT1 data set must contain a database unloaded in the format of the IMS HD Unload Utility.

FABI0106E HEADER OR TRAILER RECORD MISSING ON SYSUT1

Explanation

The FABIUNLS utility detected that a header record or trailer record was missing in the SYSUT1 data set.

System action

The FABIUNLS utility will issue an abend.

User response

Provide an appropriate SYSUT1 data set. The SYSUT1 data set must contain a database unloaded in the format of the IMS HD Unload Utility.

FABI0107E ROOT SEGMENT HAS NO SEQUENCE-FIELD

Explanation

The FABIUNLS utility detected that the root segment of the unloaded database has no sequence field. Root segments without a sequence field are not supported by Sequential Subset Randomizer.

System action

The FABIUNLS utility will issue an abend.

User response

If the root segment has no sequence-field, then a randomizer other than Sequential Subset Randomizer must be used.

FABI0111E BINARY SEARCH IMPLEMENTED IN SEQUENTIAL SUBSET RANDOMIZER FAILED

Explanation

The binary search implemented in Sequential Subset Randomizer failed.

System action

Sequential Subset Randomizer will request either an abend (batch region) or a pseudo-abend (online region).

User response

Save the dump, the JES log, the DBDGEN output, and the output of the randomizer generation. Contact IBM Software Support.

FABI0112E THE NUMBER OF RAPS DEFINED DURING DBDGEN IS TOO SMALL

Explanation

The number of RAPs defined during the DBDGEN is too small. If the subset ID does not start at the first position of the root key, the number of RAPs must be at least equal to the number of subsets defined during the generation of Sequential Subset Randomizer.

System action

Sequential Subset Randomizer will request either an abend (batch region) or a pseudo-abend (online region).

User response

Increase, in the DBDGEN specifications, either the number of RAPs per block/CI or the number of blocks/CIs located in the root addressable area.

FABI0113E THERE IS AN INCONSISTENCY BETWEEN DBDGEN AND SEQUENTIAL SUBSET RANDOMIZER GENERATION

Explanation

There is an inconsistency between:

- The length of the root key field (as defined during DBDGEN)
- The length and start position of the subset ID (as defined during the generation of Sequential Subset Randomizer) within the root key field.

Notice that the subset ID must be completely within the root key field.

System action

Sequential Subset Randomizer will request either an abend (batch region) or a pseudo abend (online region).

User response

Resolve the inconsistency between DBDGEN and Sequential Subset Randomizer generation.

FABI0121E FABISTAT INTERNAL ERROR

Explanation

The binary search implemented in the FABISTAT failed. A root segment that is required by application program was not found in the range of this segment's subset.

System action

The FABISTAT will issue an abend.

User response

Contact IBM Software Support.

FABI0122I SEQUENTIAL SUBSET STATISTICS WAS NOT CREATED

Explanation

The FABISTAT exit routine did not create a Sequential Subset Statistics, because there is no statistics record.

System action

None.

User response

Confirm that the correct name of randomizer was specified on the HSSROPT SSSTATS control statement or the DBDGEN source statement.

Chapter 36. Gathering diagnostic information

Before you report a problem with IMS High Performance Unload to IBM Software Support, you need to gather the appropriate diagnostic information.

Procedure

Provide the following information for all IMS High Performance Unload problems:

- A clear description of the problem and the steps that are required to re-create the problem
- The version of IMS that you are using and the version of the operating system that you are using
- A complete log of the job
- A Load Module/Macro APAR Status report

For information about creating a Load Module/Macro APAR Status report, see Chapter 37, “Diagnostics Aid,” on page 511.

Chapter 37. Diagnostics Aid

If you have a problem that you think is not a user error, you should run the Diagnostics Aid program (FABHDIAG), obtain the Load Module/Macro APAR Status report, attach it to the other diagnostic documents (such as job dump list or I/O of the utility), and report the error to IBM.

The Diagnostics Aid generates a Load Module/Macro APAR Status report. This report shows the latest APAR fixes applied to each module and macro.

The Diagnostics Aid is not applicable for any other versions or releases.

Topics:

Running the Diagnostics Aid with JCL

To run the Diagnostics Aid program (FABHDIAG), supply an EXEC statement and DD statements that define the input and the output data sets.

Procedure

1. Specify the EXEC statement. It must be in the following form:

```
    //stepname  EXEC PGM=FABHDIAG
```

2. Specify the DD statements.

DD statement	Description
STEPLIB DD	This statement defines the library containing the load modules (usually HPS.SHPSLMD0).
SHPSLMD DD	This statement defines the library containing the load modules (usually HPS.SHPSLMD0) for which you have a problem. If this DD statement is not provided or if DD DUMMY is specified, the Load Module APAR Status report is not generated. It is always recommended that you specify this DD statement.
SHPSMAC DD	This statement defines the library containing the provided macros (usually HPS.SHPSMAC0) for which you have a problem. If this DD statement is not provided or if DD DUMMY is specified, the Macro APAR Status report is not generated.
SYSPRINT DD	This output data set contains the Load Module/Macro APAR Status report. The data set contains 133-byte, fixed-length records. It can reside on a tape, direct-access device, or printer; or it can be routed through the output stream. If BLKSIZE is coded in the DD statement, it must be a multiple of 133. However, it is recommended that you use: `//SYSPRINT  DD SYSOUT=A`

3. Run the job.

Load Module/Macro APAR Status report

The Diagnostics Aid generates two reports for maintenance by IBM.

The generated reports are:

- Load Module APAR Status report
- Macro APAR Status report

Load Module APAR Status report

The Load Module APAR Status report contains information about the modules and their applied APARs.

This report contains the following information:

MODULE LIBRARY
This field includes the data set names specified in the SHPSLMD DD statement. If more than 30 data sets are concatenated, only the first 30 data sets are listed.

MODULE NAME
This field shows the name of the load module member or the alias.

ALIAS-OF
This field shows the name of the original member of the alias. If the module name is not an alias, this field is left blank.

CSECT NAME
This field shows the name of the included CSECT in the module. The CSECT names are reported in the included order in the module.

APAR NUMBER
This field shows the latest APAR number applied to the module represented by the CSECT name. If no APAR is applied, NONE is shown.

APAR FIX-DATE
This field shows the date when the modification was prepared for the module represented by the CSECT name. If no APAR is applied, N/A is shown.

Notes:

1. If the CSECT name does not start with FAB, HPS, or the program structure of the CSECT does not conform to the IMS High Performance Unload module standard to identify the APAR number and the APAR fixed date, the fields APAR NUMBER and APAR FIX-DATE are filled with asterisks (*).
2. If the load module is a member of the PDSE library, the following statement is shown on the report line and the job completes with a return code of 4.

```
** IT CAN NOT BE ANALYZED DUE TO PDSE LIBRARY MEMBER **
```

3. If the load macro fails for a utility member, the following statement is shown on the report line and the job completes with a return code of 8.

```
** IT CAN NOT BE ANALYZED DUE TO LOAD FAILED MEMBER  **
```

Macro APAR Status report

The Macro APAR Status report contains information about macros and their applied APARs.

This report contains the following information:

MACRO LIBRARY
This field includes the data set names specified in the SHPSMAC DD statement. If more than 30 data sets are concatenated, only the first 30 data sets are listed.

MACRO NAME
This field shows the name of the macro member or the alias.

ALIAS-OF
This field shows the name of the original member of the alias. If the macro name is not an alias, this field is left blank.

APAR NUMBER
This field shows the latest APAR number applied to the macro. If no APAR is applied, NONE is shown.

APAR FIX-DATE
This field shows the date when the modification was prepared for the macro. If no APAR is applied, N/A is shown.

Note: If the macro source statement structure does not conform to the IMS High Performance Unload macro standard to identify the APAR number and the APAR fixed date, the fields APAR NUMBER and APAR FIX-DATE are filled with asterisks (*).

Messages and codes

The following topics describe the return codes, abend codes, and messages issued by the Diagnostics Aid (FABHDIAG).

Return codes

FABHDIAG ends with one of the following return codes:

0
Successful completion of the program.

4
Warning messages were issued, but the requested operation was completed.

8
Error messages were issued, but the request operation was completed.

Abend codes

All 36*xx* abend codes are accompanied by a FABU36*xx* message. Refer to the appropriate message for problem determination.

Messages

Use the information in these messages to help you diagnose and solve FABHDIAG problems.

FABU1001I DIAG ENDED NORMALLY

Explanation

This message is generated when Diagnostic Aid has been completed successfully.

System action

Diagnostic Aid completes the job successfully with a return code of 0.

User response

None. This message is informational.

FABU1002W DIAG ENDED WITH WARNINGS

Explanation

This message is generated when trivial error conditions are encountered by Diagnostic Aid.

System action

Diagnostic Aid ends with a return code of 4.

User response

Refer to other messages generated by Diagnostic Aid to determine the nature and the cause of the detected errors. Correct the problem and rerun the job.

FABU1003E DIAG ENDED WITH ERRORS

Explanation

This message is generated when severe error conditions are encountered by Diagnostic Aid.

System action

Diagnostic Aid ends with a return code of 8.

User response

Refer to other messages generated by Diagnostic Aid to determine the nature and the cause of the detected errors. Correct the problem and rerun the job.

FABU1005W [SHPSLMD | SHPSMAC] DD STATEMENT NOT FOUND

Explanation

Diagnostic Aid could not find the SHPSLMD/SHPSMAC DD statement.

System action

Diagnostic Aid sets an end-of-job return code of 4 and continues processing. Diagnostic Aid does not generate a report for the load module or the macro.

User response

If you intended to specify the indicated DD statement, correct the error and rerun the job.

FABU1006W DUPLICATE *member name* IN LIBRARY DDNAME *ddname*

Explanation

Diagnostic Aid found a duplicated member in the concatenated libraries.

System action

Diagnostic Aid uses the member which is first found in the concatenated libraries. Diagnostic Aid sets an end-of-job return code of 4 and continues processing.

User response

Ensure which libraries have correct module/macro libraries. Correct the error and rerun the job if necessary.

FABU1007W DUMMY SPECIFIED FOR [SHPSLMD | SHPSMAC] DD STATEMENT

Explanation

DUMMY was specified for the SHPSLMD/SHPSMAC DD statement.

System action

Diagnostic Aid sets an end-of-job return code of 4 and continues processing. Diagnostic Aid does not generate a report for the load module or the macro.

User response

If you did not intend to specify the dummy DD statement, correct the error and rerun the job.

FABU1008W NO [MODULE | MACRO] MEMBERS FOUND IN DDNAME [SHPSLMD | SHPSMAC]

Explanation

Diagnostic Aid could not find any utility modules or macros members from the DD ddname data set.

System action

Diagnostic Aid sets an end-of-job return code of 4 and continues processing.

User response

Ensure that the libraries have correct utility module or macro libraries. Correct the error and rerun the job.

FABU2001E LOAD FAILED FOR DDNAME *ddname* MODULE *member*

Explanation

Diagnostic Aid could not load a *member name* from *ddname*.

System action

Diagnostic Aid sets an end-of-job return code of 8 and continues processing.

User response

Ensure that the member indicated exists in the data set specified for the indicated *ddname*. Correct the error and rerun the job.

FABU3600E OPEN FAILED FOR DDNAME *ddname*

Explanation

The named DCB could not be opened.

System action

Diagnostic Aid ends with an abend code of U3600.

User response

Ensure that a *ddname* DD statement exists, and that it specifies the correct DD parameter. Correct any errors and rerun the job.

FABU3601E GET FAILED FOR DDNAME *ddname*

Explanation

The GET failed for a directory from the DD *ddname* data set.

System action

Diagnostic Aid ends with an abend code of U3601.

User response

Refer to the MVS system message and its programmer response. Correct the error and rerun Diagnostic Aid. If the error persists, contact IBM Software Support.

FABU3602E READ FAILED FOR DDNAME *ddname* MEMBER *member*

Explanation

The READ failed for a *member* from the DD *ddname* data set.

System action

Diagnostic Aid ends with an abend code of U3602.

User response

Refer to the MVS system message and its programmer response. Correct the error and rerun Diagnostic Aid. If the error persists, contact IBM Software Support.

FABU3603E BLDL FAILED FOR DDNAME *ddname* MEMBER *member*

Explanation

The *member* was not found when the BLDL macro searched the PDS directory for the *ddname*.

System action

Diagnostic Aid ends with an abend code of U3603.

User response

Ensure that the member indicated exists in the data set specified for the indicated ddname. Correct the error and rerun the job. If the error persists, contact IBM Software Support.

FABU3604E LOAD FAILED FOR DDNAME *ddname* MODULE *member*

Explanation

Diagnostic Aid could not load the *member name* from the *ddname*.

System action

Diagnostic Aid ends with an abend code of U3604.

User response

Refer to the MVS system message and its programmer response. Correct the error and rerun Diagnostic Aid. If the error persists, contact IBM Software Support.

FABU3605E DELETE FAILED FOR MODULE *member*

Explanation

Diagnostic Aid could not delete a *member name*.

System action

Diagnostic Aid ends with an abend code of U3605.

User response

Contact IBM Software Support.

FABU3606E PUT FAILED FOR SYSPRINT

Explanation

Diagnostic Aid could not put report data in SYSPRINT.

System action

Diagnostic Aid ends with an abend code of U3606.

User response

Refer to the MVS system message and its programmer response. Correct the error and rerun Diagnostic Aid. If the error persists, contact IBM Software Support.

FABU3607E OPEN FAILED FOR SYSPRINT

Explanation

SYSPRINT DCB could not be opened.

System action

Diagnostic Aid ends with an abend code of U3607.

User response

Ensure that a *ddname* SYSPRINT DD statement exists, and that it specifies the correct DD parameter. Correct any errors and rerun the job.

FABU3608E FIND FAILED FOR DDNAME *ddname* MEMBER *member*

Explanation

The FIND failed for a *member* from DDNAME *ddname* data set.

System action

Diagnostic Aid ends with an abend code of U3608.

User response

Ensure that the member indicated exists in the data set specified for the indicated ddname. Correct the error and rerun the job. If the error persists, contact IBM Software Support.

FABU3609E DEVTYPE FAILED FOR DDNAME *ddname*

Explanation

The DEVTYPE failed for a DDNAME *ddname* data set.

System action

Diagnostic Aid ends with an abend code of U3609.

User response

Contact IBM Software Support.

FABU3610E RDJFCB FAILED FOR DDNAME *ddname*

Explanation

The READJFCB failed for a DDNAME *ddname* data set.

System action

Diagnostic Aid ends with an abend code of U3610.

User response

Contact IBM Software Support.

FABU3611E GETMAIN FAILED. INSUFFICIENT STORAGE TO RUN THE JOB

Explanation

Work space for Diagnostic Aid could not be obtained.

System action

Diagnostic Aid ends with an abend code of U3611.

User response

Increase the region size and rerun the job.

FABU3612E TOO MANY [MODULE | MACRO] MEMBERS DETECTED IN DDNAME [SFABMOD | SHPSMAC]

Explanation

There are too many utility members in the SFABMOD/SHPSMAC DD data set.

System action

Diagnostic Aid ends with an abend code of U3612.

User response

Specify the correct data set for the indicated DD statement and rerun the job.

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 give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this 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 supply 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 information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

This publication primarily documents information that is NOT intended to be used as Programming Interfaces of IMS High Performance Unload.

This publication also documents intended general-use programming interface and associated guidance information, product-sensitive programming interface and associated guidance information, and diagnosis, modification, or tuning information for IMS High Performance Unload.

General-use programming interface enables the customer to write programs that obtain the services of IMS High Performance Unload. This information is identified where it occurs, either by an introductory statement to a topic or by the following marking:

General-use programming interface and associated guidance information...

Product-sensitive programming interfaces are provided to allow the customer installation to perform tasks such as tailoring, monitoring, modification, or diagnosis of this IBM product. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM product. Product-sensitive interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces might need to be changed in order to run with new product releases or versions, or as a result of service. This information is identified where it occurs, either by an introductory statement to a topic or by the following marking:

Product-sensitive programming interface and associated guidance information...

Diagnosis, modification, or tuning information is provided to help the customer diagnose, modify, or tune IMS High Performance Unload. This information is identified where it occurs, either by an introductory statement to a topic or by the following marking:

Diagnosis, modification, or tuning information...

Attention: Do not use this diagnosis, modification, or tuning information as a programming interface.

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.

Other company, product, and service names may be trademarks or service marks of others.

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.

Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you

to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at http://www.ibm.com/privacy and IBM's Online Privacy Statement at www.ibm.com/privacy/details the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at http://www.ibm.com/software/info/product-privacy.

Index

Special Characters

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Z

Product Number: 5655-E06

SC27-0936-12