IBM IMS High Performance Pointer Checker for z/OS

User’s Guide

Version 2 Release 2

SC18-9974-01

IBM IMS High Performance Pointer Checker for z/OS

User's Guide

Version 2 Release 2

SC18-9974-01

Note:

Before using this information and the product it supports, read the information in Appendix F, “Notices,” on page 731.

This edition applies to Version 2 Release 2, Modification Level 0, of the program product IBM IMS High Performance Pointer Checker for z/OS (Program number: 5655-K53) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this information

This information is designed for system programmers, application programmers, system analysts, database administrators, computer operators, and technical support personnel, who are involved in database management, maintenance, and performance tuning, and who need to know how to operate the following five utilities of IBM® IMS™ High Performance Pointer Checker for z/OS® (also referred to as IMS HP Pointer Checker). Before using this book, you should understand basic IMS concepts, the IMS environment, and your installation's IMS system.

- HD Pointer Checker
- HD Tuning Aid
- DB Historical Data Analyzer
- Space Monitor
- DB Segment Restructure

This information contains Chapter 1, "Overview of IMS HP Pointer Checker" which gives an introduction to the five utilities:

Part 2, "HD Pointer Checker," provides an introduction to the HD Pointer Checker utility, and explains how to use it:

Part 3, "HD Tuning Aid," provides an introduction to the HD Tuning Aid utility, and explains how to use it.

Part 4, "DB Historical Data Analyzer," provides an introduction to the DB Historical Data Analyzer utility, and explains how to use it.

Part 5, "Space Monitor," provides an introduction to the Space Monitor utility, and explains how to use it.

Part 6, "DB Segment Restructure," provides an introduction to the DB Segment Restructure utility, and explains how to use it.

This information also contains the following appendixes:

- Appendix A, "Messages and codes"
- Appendix B, "Diagnostics Aid"
- Appendix C, "Migration considerations"
- Appendix D, "Layout of KEYSIN data set record"
- Appendix E, "Layout of HISTORY data set record"
- Appendix F, "Notices"

Specific changes since the previous edition of this document are indicated by a vertical bar (|) to the left of a change. Editorial changes that have no technical significance are not noted.

Always check the DB2® and IMS Tools Library page for the most current version of this publication:

www.ibm.com/software/data/db2imstools/library.html

Service updates and support information

To find service updates and support information, including software fix packs, PTFs, Frequently Asked Question (FAQs), technical notes, troubleshooting information, and downloads, refer to the following Web page:

www.ibm.com/software/data/db2imstools/support.html

Highlighting conventions

This information uses the following highlighting conventions:

- **Boldface** type indicates commands or user interface controls such as names of fields, folders, icons, or menu choices.
- `Monospace` type indicates examples of text that you enter exactly as shown.
- *Italic* type indicates variables that you should replace with a value, to indicate the titles of other publication, and to emphasize significant terms.

How to look up message explanations

You can use any of the following methods to search for messages and codes:

Searching an information center

In the search box that is located in the top left toolbar of any Eclipse help system, such as the IBM Information Management Software for z/OS Solutions Information Center, enter the number of the message that you want to locate. For example, you can enter DFS1065A in the search field.

Use the following tips to help you improve your message searches:

- You can search for information on codes by entering the code; for example, enter -327.
- Enter the complete or partial message number. You can use wild cards (* or ?) in the message number to broaden your search; for example, DFS20??I.

The information center contains the latest message information for all of the information management products that are included in the information center.

Using a Web search

You can use any of the popular search engines that are available on the Web to search for message explanations. When you type the specific message number or code into the search engine, you will be presented with links to the message information in IBM information centers.

Using LookAt

LookAt is an online facility that you can use to look up explanations for most of the IBM messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can use LookAt from the following locations to find IBM message explanations for z/OS elements and features, z/VM®, VSE/ESA™, and Clusters for AIX® and Linux®:

- The Internet. You can access IBM message explanations directly from the LookAt Web site at http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/.

- Your z/OS TSO/E host system. You can install code on your z/OS or z/OSe systems to access IBM message explanations, using LookAt from a TSO/E command line (for example, TSO/E prompt, ISPF, or z/OS UNIX® System Services running OMVS).
- Your Microsoft® Windows® workstation. You can install code to access IBM message explanations on the z/OS Collection (SK3T-4269) using LookAt from a Microsoft Windows command prompt (also known as the DOS command line).
- Your wireless handheld device. You can use the LookAt Mobile Edition with a handheld device that has wireless access and an Internet browser (for example, Internet Explorer for Pocket PCs, Blazer, or Eudora for Palm OS, or Opera for Linux handheld devices). Link to the LookAt Mobile Edition from the LookAt Web site.

You can obtain code to install LookAt on your host system or Microsoft Windows workstation from a disk on your z/OS Collection (SK3T-4269) or from the LookAt Web site (click **Download**, and select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

How to send your comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this information or any other IMS High Performance Pointer Checker for z/OS documentation, use either of the following options:

- Use the online reader comment form, which is located at:

  www.ibm.com/software/data/rcf/
- Send your comments by e-mail to comments@us.ibm.com. Be sure to include the name of the book, the part number of the book, the version of IMS High Performance Pointer Checker for z/OS, and, if applicable, the specific location of the text you are commenting on (for example, a page number or table number).

Part 1. Introduction

Chapter 1. Overview of IMS HP Pointer Checker

This chapter gives introduction to the following utilities of IBM IMS High Performance Pointer Checker for z/OS (also referred to as IMS HP Pointer Checker):

- HD Pointer Checker
- HD Tuning Aid
- DB Historical Data Analyzer
- Space Monitor
- DB Segment Restructure

These topics introduce you to the functionality that is provided by IMS HP Pointer Checker, its processing environment, and its compatibility with its former versions of the product.

Topics:

IMS HP Pointer Checker terminology

To make this information easier to read, the version and release levels of IMS are abbreviated, as follows:

- **IMS Version 8** refers to IMS Version 8 Release 1.
- **IMS Version 9** refers to IMS Version 9 Release 1.
- **IMS Version 10** refers to IMS Version 10 Release 1.

The various versions of IMS are referred to simply as IMS, except where distinctions among them need to be made.

The following abbreviations are used in this information.

Abbreviation	Meaning
DBHDA	DB Historical Analyzer
DBSR	DB Segment Restructure
HDPC	HD Pointer Checker
HDTA	HD Tuning Aid
IMS	Database Manager of one of the currently supported versions of IMS.
IMS Database Repair Facility, IMS DBRF	IBM IMS Database Repair Facility for z/OS, Version 1 (Program number: 5655-E03)
IMS DBT Version 2	IBM IMS System Utilities/Data Base Tools, Version 2 (Program number: 5685-093)

IMS Database Recovery Facility, IMS DRF	IBM IMS Database Recovery Facility for z/OS, Version 3 Release 1 (Program number: 5655-N47)
IMS HP Image Copy, IMS HPIC	IBM IMS High Performance Image Copy for z/OS, Version 3 (Program number: 5655-K96) and Version 4 (Program number: 5655-N45)
IMS HP Pointer Checker	IBM IMS High Performance Pointer Checker for z/OS, Version 2 Release 2 (Program number: 5655-K53) (this product)
IMS HP Pointer Checker V1	IBM IMS High Performance Pointer Checker for OS/390®, Version 1 Release 1 (Program number: 5655-E09)
IMS Image Copy Extensions, IMS ICE	IBM IMS Image Copy Extensions for z/OS
IMS Parallel Reorganization, IMS PR	IBM IMS Parallel Reorganization for z/OS, Version 3 Release 1 and Release 2 (Program number: 5655-M28)
IMS Library Integrity Utilities, IMS LIU	IBM IMS Library Integrity Utilities for z/OS, Version 1 Release 1 (Program number: 5655-I42)
IMS Tools KB Output repository	Output repository of IBM IMS Tools Knowledge Base
IMS Tools Knowledge Base, IMS Tools KB	IBM IMS Tools Knowledge Base for z/OS
IMS Tools KB server	Server of IBM IMS Tools Knowledge Base
SPMN	Space Monitor

What does IMS HP Pointer Checker do?

This section describes the overview of the five utilities of IMS HP Pointer Checker.

HD Pointer Checker detects and reports problems of direct and/or other types of pointers. These reports pinpoint both the errors and their locations within IMS databases. It also produces many reports to help in database tuning such as redundant spaces in IMS databases.

HD Tuning Aid produces reports that describe the distribution of root segments in HDAM, HIDAM, PHDAM, or PHIDAM databases. It also produces a report giving summary information about a High Availability Large Database (HALDB).

DB Historical Data Analyzer assists IMS users to analyze the status and historical trend of IMS full-function database data sets which HD Pointer Checker supports. Historical trend here means the change in various aspects of IMS full-function database data sets (for example, the use of space, size/number of database segments, or size/number of database blocks) from the past.

DB Historical Data Analyzer uses data collected by HD Pointer Checker and Space Monitor as follows:

- Statistics information produced by HD Pointer Checker
- Space allocation information produced by Space Monitor

DB Historical Data Analyzer has a utility called Export Utility. Export Utility exports the data collected by HD Pointer Checker to flat records, which can be processed by a user's application program.

Space Monitor assists the users to forecast potential space utilization problems of IMS full-function database data sets that HD Pointer Checker supports, and OS data sets (including VSAM data sets).

Note: The term OS data sets used hereafter in this guide includes VSAM data sets. It generally means non-IMS data sets in this guide.

DB Segment Restructure is used to change the format of a segment data within any existing full-function database including HALDB. Its main function is to modify databases in ways that exceed the capabilities of the standard IMS utilities. One such task would be the changing of the hierarchy in a database.

When you use DB Segment Restructure, there is no need to write a program to reformat segment data.

Utilities management solutions

IBM solutions help IT organizations maximize their investment in IMS databases while staying on top of some of today's toughest IT challenges. Utilities management solutions can help you achieve higher availability and better performance during database maintenance while enhancing the productivity of both database administrators and system programmers.

Underlying the operation of any database management system are the utilities. With the number of database objects growing exponentially, especially when dealing with ERP applications such as SAP, the impact of managing utility jobs, meeting service level agreements, and ensuring recoverability can be overwhelming.

IBM offers IMS Tools that assist in the Utilities Management process for example:

In an on demand world 24x7 data availability is a key requirement. Reorganization tools from IBM can help with the performance of key functions such as unloading and reloading IMS data, without impacting data access.

These and other IMS Tools can help you achieve higher availability and better performance during data maintenance while enhancing the productivity of both database administrators and system programmers.

Many IMS Tools products provide database management features that are not available in IMS itself or that provide enhancements to capabilities that are built into IMS.

IBM IMS High Performance Pointer Checker for z/OS is only one of several IMS Tools products that provide enhancements to the process of utility management operations for your databases.

For example, IMS High Performance Pointer Checker for z/OS helps you efficiently manage many database reorganization activities. When used regularly, IMS HP Pointer Checker helps ensure that your database pointers are error free—alerting you when it's time to perform a database reorganization. To help programmers analyze corrupt databases, IMS HP Pointer Checker provides important information that helps to reduce the time spent handling diagnostics and repairs. These reports reveal errors and their locations within the database—facilitating system tuning and optimization.

When IMS HP Pointer Checker is used in conjunction with IMS Database Repair Facility (a powerful tool that can interactively—and rapidly—repair VSAM- and OSAM-organized IMS databases containing pointer or data errors), their synergy creates a powerful set of productivity aids that helps you detect and correct database errors quickly—and repair them with minimum downtime to your IMS environment. These tools help you maintain the data availability you need to help your business thrive.

Other IMS Tools products that can assist with database utilities management include:

- IMS Database Control Suite
- IMS High Performance Fast Path Utilities
- IMS High Performance Load
- IMS High Performance Prefix Resolution
- IMS High Performance Unload
- IMS Index Builder
- IMS Parallel Reorganization

Support for IMS Tools Knowledge Base

IBM IMS Tools Knowledge Base for z/OS (also referred to as IMS Tools Knowledge Base or IMS Tools KB) is an IMS Tools product that provides common services for storing and viewing reports that are generated by other participating IMS Tools products.

To fully participate in the IMS Tools Knowledge Base information management environment, the code for each IMS tool must be enabled to communicate with the IMS Tools Knowledge Base server. An enabled IMS tool can automatically send its generated reports to the IMS Tools Knowledge Base repository. This version of IMS HP Pointer Checker is enabled to participate in the IMS Tools Knowledge Base environment.

Additionally, you must register the enabled IMS tool with IMS Tools Knowledge Base. The registration procedure allows you to use the ISPF user interface to view, print, and manage those reports that are generated by IMS HP Pointer Checker and that are stored in the IMS Tools Knowledge Base repository. Instructions for registering IMS Tools products can be found in the *IMS Tools Knowledge Base User's Guide* (SC18-9963).

For the reports that can be stored in the IMS Tools KB Output repository, see Table 15 on page 109, Table 21 on page 112 (for HD Pointer Checker), and "Output" on page 508 (for Space Monitor).

Processing environment

This section describes the software and hardware environments for the five utilities of IMS HP Pointer Checker.

Software requirements

IMS HP Pointer Checker for z/OS, Version 2 Release 2 operates in z/OS Version 1 Release 6 (5694-A01) or later.

IMS HP Pointer Checker for z/OS, Version 2 requires the currently supported versions of IMS.

HD Pointer Checker, **Space Monitor**, and **HD Tuning Aid** additionally require DFSORT™ (Data Facility Sort), part of z/OS, or a functionally equivalent sort/merge program.

HD Pointer Checker requires the following products:

- To use the MAPDBD or the DECODEDBD functions, you need IBM IMS Library Integrity Utilities for z/OS (also referred to as IMS Library Integrity Utilities) Version 1 Release 1 (5655-I42). If the functions are used in a DBB region, IMS Library Integrity Utilities requires APAR PQ75863 and the corresponding PTF UQ78591.
- To use the full-function database HASH check option with image copy job, you need one of the following:
  - IBM IMS High Performance Image Copy for z/OS (also referred to as IMS HP Image Copy), Version 3 Release 2 (5655-K96) or later
  - IBM IMS High Performance Image Copy for z/OS (also referred to as IMS HP Image Copy), Version 4 Release 1 (5655-N45)
- To use the HASH Check option in the IMS Parallel Reorganization job, you need IBM IMS Parallel Reorganization for z/OS (also referred to as IMS Parallel Reorganization), Version 3 Release 1 (5655-M28) or later.
- To use the HASH Check option in the IMS Database Recovery Facility, you need IBM IMS Database Recovery Facility for z/OS (also referred to as IMS Database Recovery Facility),Version 3 Release 1 (5655-N47).

DB Historical Data Analyzer additionally requires the following products to optionally produce charts on a display terminal:

- GDDM/MVS Version 3 Release 2 (5695-167)
- GDDM-PGF Version 2 Release 3 (5668-812) or the GDDM-PGF part of z/OS

Export Utility can process the historical records created by HD Pointer Checker, which runs in the following environment:

- IMS HP Pointer Checker Version 2 Release 1 or Release 2 (5655-K53)
- IMS HP Image Copy Version 3 Release 2 (5655-K96) or later
- IMS HP Image Copy Version 4 Release 1 (5655-N45)
- IMS Parallel Reorganization Version 3 Release 1 (5655-M28) or later

HD Pointer Checker and Space Monitor additionally require IBM IMS Tools Knowledge Base for z/OS, Version 1 Release 1 (5655-R34, hereafter also referred to as IMS Tools KB) to store reports in the IMS Tools KB Output repository.

Hardware requirements

The hardware requirements are the same as those for IMS.

For **DB Historical Data Analyzer**, the option to produce charts on a display terminal requires a terminal that is supported by GDDM® Interactive Chart Utility (ICU).

Compatibility

This section discusses the compatibility of the product with its earlier products and the compatibility of the HISTORY data set among different versions of the product and its usages.

Compatibility with IMS DBT Version 2

This section explains about the compatibility of the product with IMS DBT Version 2.

HD Pointer Checker

See also Appendix C, “Migration considerations,” on page 723.

For details about the control statements and JCL in IMS DBT Version 2, see *IMS System Utilities/Data Base Tools HD Pointer Checker User’s Guide* (SH21-0542).

Input control statements

PROCCTL: The control statements in the PROCCTL data set that is used in IMS DBT Version 2 (Program number: 5685-093) can be used with this product.

Cataloged procedures

To run the JCL of the IMS DBT Version 2 HD Pointer Checker under the load modules of this product, use new or modified cataloged procedures.

Cataloged procedures supplied by IBM: The IMS DBT Version 2 cataloged procedures originally provided have been replaced. If you use these procedures in the IMS DBT Version 2 JCL, you should replace the original procedures with those of this product, which have the same names.

Cataloged procedures that you have tailored: If you have tailored the IMS DBT Version 2 cataloged procedure to meet your requirements and fit your environment, it is recommended that you change the LRECL and BLKSIZE values of some of the DD statements in that procedure. Also, some work data sets have been removed; it is also recommended that you delete them.

If you have coded the scan, sort, merge, and check steps, you need to delete the sort and merge steps, because the sort and merge steps are removed from IMS HP Pointer Checker Version 2. Also, it is better that you delete the SORTMERG= option in the PROCCTL data set because the option is ignored and you will need to change DDs of the work data sets.

For more detail on DD statement of the work data set, see “FABPMAIN JCL” on page 37.

Recommendation for users running in multiple steps: We strongly recommend that you use the single-step--that is PROC TYPE=ALL--job as in FABPPD, instead of continue using your multiple-step--that is PROC TYPE=SCAN and CHECK--job by modifying it, because by using the single-step job, the performance is better, the work data sets become smaller, and the JCL statements are simpler.

For the same reason, we recommend that you use FABPPTA rather than FABPPTAM.

JCLs

Single-step (PROC TYPE=ALL in PROCCTL data set): The HD Pointer Checker JCL used in IMS DBT Version 2, with PROC TYPE=ALL in the PROCCTL data set, can be used with this product.

Some work data sets are no longer used in this version. It is recommended that you delete them.

Multiple-step (PROC TYPE=SCAN and TYPE=CHECK in PROCCTL data set): If there are any sort and merge steps between steps PROC TYPE=SCAN and TYPE=CHECK, the JCL cannot be used with this product, because the sort and merge steps are no longer used. The sort and merge steps are the steps invoked by PGM=SORT.

If there are no sort or merge steps between steps PROC TYPE=SCAN and TYPE=CHECK, the JCL can be used.

If more than two scan steps are invoked and the output work data sets are combined into one check step, the JCLs cannot be used. The input DD names of the check step have been changed.

For the procedure recommended by IBM, see “Recommendation for users running in multiple steps” on page 8.

OUTPUT data sets

KEYSIN and HISTORY data sets: The formats of some reports, the HISTORY data set, the KEYSIN data set, and the work data set are different from those in IMS DBT Version 2.

HD Tuning Aid

The compatibility of HD Tuning Aid is as follows:

Input control statements

CTL: The control statements, which are the input to HD Tuning Aid used in IMS DBT Version 2 can be used in HD Tuning Aid of this product.

SYSIN for DFSORT: The control statements in the SYSIN data set, which is the input to DFSORT used in IMS DBT Version 2 cannot be used.

For details, read Appendix C, “Migration considerations,” on page 723.

DB Historical Data Analyzer, Space Monitor, and DB Segment Restructure

The JCLs to run DB Historical Data Analyzer, Space Monitor, and DB Segment Restructure of IMS DBT Version 2 Release 3 can be used in this product.

HISTORY Data Set

The layout of the HISTORY data set records has been changed to support HALDB. HISTORY data sets that is used under IMS System Utilities/Data Base Tools (IMS DBT) Version 2 Release 3 Space Management Utilities (SMU) do not have upward compatibility for this product. When migrating from IMS DBT Version 2 Release 3, you will need to consider the following:

- The KEYS parameter in a DEFINE CLUSTER JCL to allocate the HISTORY data set must be KEYS (23,0).
- The RECORDSIZE parameter in a DEFINE CLUSTER JCL to allocate the HISTORY data set must be RECORDSIZE (240,240).
- HISTORY data sets that is used under IMS DBT Version 2 Release 3 must be migrated by the procedure described in “Migrating from DBT Version 2 Release 3 format” on page 370.
- Application programs that use the HISTORY data set must be modified to work with the format described in Appendix E, “Layout of HISTORY data set record,” on page 729. Even if the program uses the product-sensitive macro FABGHIR to map the HISTORY data set records, you must reassemble the application programs with the FABGHIR macro provided by this product.

For more details, see “HISTORY data set (HISTORY)” on page 367.

Compatibility with IMS HP Pointer Checker Version 1

This section explains about the compatibility of the product with IMS HP Pointer Checker Version 1.

HD Pointer Checker

See also Appendix C, “Migration considerations,” on page 723.

For details of control statements and job control language used in the Version 1, read *IMS High Performance Pointer Checker for OS/390, Version 1 Release 1 User’s Guide Volume 1* (SC27-0921).

Input control statements

PROCCTL: The control statements in the PROCCTL data set that is used in Version 1 can be used in this product.

Cataloged procedures

FABPP, FABPPD, and FABPPTA: The cataloged procedures FABPP, FABPPD, and FABPPTA supplied in Version 1 can be used with the load modules in this product. Some work data sets are no longer used. It is recommended that you delete them.

FABPPM, FABPPMD, and FABPPTAM: The cataloged procedures FABPPM, FABPPMD, FABPPTAM that IBM supplied in Version 1 cannot be used. They should be replaced with the procedures supplied with this product.

Recommendation for users running in multiple steps: We strongly recommend that you use the single-step—that is PROC TYPE=ALL—job as in FABPPD, instead of continue using your multiple-step—that is PROC TYPE=SCAN and CHECK— job by modifying it, because by using the single-step job, the performance is better, the work data sets become smaller, and the JCL statements are simpler.

For the same reason, we recommend that you use FABPPTA rather than FABPPTAM.

Cataloged procedures that you have tailored: If you have tailored cataloged procedures to meet your requirements and fit your environment, and have coded the scan, sort, merge, and check steps, you need to delete the sort and merge steps, because the sort and merge steps are removed from IMS HP Pointer Checker Version 2. Also, it is better that you delete the SORTMERG= option in the PROCCTL data set because the option is ignored and you will need to change DDs of the work data sets.

For more detail on DD statement of the work data set, see “FABPMAIN JCL” on page 37.

JCLs

Single-step (PROC TYPE=ALL in PROCCTL data set): HD Pointer Checker JCL with PROC TYPE=ALL in the PROCCTL data set used in Version 1 can be used with this product. Some work data sets are no longer used in Version 2; it is recommended that you delete them.

Multiple-step (PROC TYPE=SCAN and TYPE=CHECK in PROCCTL data set): If there are any sort and merge steps between steps PROC TYPE=SCAN and TYPE=CHECK, the JCL cannot be used, because the sort and the merge steps are no longer used. The sort and merge steps are the steps invoked by PGM=SORT.

If there are no sort or merge steps between steps PROC TYPE=SCAN and TYPE=CHECK, the JCL can be used.

If more than two scan steps are invoked and the output work data sets are combined into one check step, the JCLs cannot be used. The input DD names of the check step have been changed.

For the procedure recommended by IBM, see “Recommendation for users running in multiple steps” on page 10.

OUTPUT data sets

KEYSIN and HISTORY data sets: The KEYSIN and HISTORY data sets that have been used in Version 1 are also supported in this product. As for the compatibility of HISTORY data sets, see “Compatibility of HISTORY data sets.”

IMS Image Copy Extensions full-function database HASH check

The JCLs and catalog procedures of IBM IMS Image Copy Extensions for z/OS (also referred to as IMS Image Copy Extensions) full-function database HASH check function that were used with IMS HP Pointer Checker Version 1 can be used with the load modules of this product.

HD Tuning Aid

The compatibility of HD Tuning Aid is as follows:

Input control statements

CTL: The control statements in the CTL data set used in Version 1 can be used in this product.

SYSIN for DFSORT: The control statements in the SYSIN data set, which is the input to DFSORT used in Version 1, can be used in this product.

DB Historical Data Analyzer, Space Monitor, and DB Segment Restructure

The JCLs to run DB Historical Data Analyzer, Space Monitor, and DB Segment Restructure of IMS HP Pointer Checker Version 1 can be used in this product.

Compatibility of HISTORY data sets

Depending on the options you have activated or set to HISTORY data sets, HISTORY data sets have several incompatibilities. This section summarizes these incompatibilities. For the details of the options and values, see “OPTION control statement” on page 376 of DB Historical Data Analyzer.

Format (MULTIENT=YES/NO)

The usage of the key field of the HISTORY data sets depends on whether you have activated multiple entries option. The activation is controlled by specifying MULTIENT=YES or MULTIENT=NO when you update HISTORY data sets. If a user application program processes the HISTORY data sets, considerations for the key field and handling of the entries depending on the option are required.

For more details of the MULTIENT option, see “OPTION control statement” on page 376. For more details about the key field, see “Format” on page 367.

Record types (EXPORTABLE=YES/NO)

If exportable option is activated, a new format of history record entries will be created. The activation is controlled by specifying EXPORTABLE=YES or EXPORTABLE=NO when you update HISTROY data sets. The new format is compatible with the existing records, and it is not public. It is referred to as a non-public history record.

If a user application program processes the HISTORY data sets, it should bypass the non-public records. The non-public history record is identified by the public format flag in the record. For more details about the flag, see “Format” on page 367.

Once the non-public records are created, the records will not be deleted even if the exportable option is deactivated. To delete them, run the delete function of the FABGHIST program.

For more details of the EXPORTABLE option, see "OPTION control statement" on page 376.

Lock mechanism (HISTLOCK=GROUP/DATASET)

The HD Pointer Checker jobs are serialized to update a HISTORY data set by the ENQ macro. Different RNAME parameters of the ENQ macro are used depending on whether you have set the HISTORY lock option to GROUP or DATASET. The setting can be changed by specifying HISTLOCK=GROUP or HISTLOCK=DATASET when you update HISTORY data sets. If you define resources of the HISTORY data sets in Global Resource Serialization (GRS) or an equivalent product, you need to specify the correct names.

For details of the HISTLOCK option, see "OPTION control statement" on page 376.

Format (MULTIIMSID=YES/NO)

The usage of the key field of the HISTORY data sets depends on whether you have enabled the Multiple-IMSID option. The status is controlled by the specification of MULTIIMSID=YES or MULTIIMSID=NO while updating HISTORY data sets. If a user application program processes the HISTORY data sets, considerations for the key field and handling of the entries depending on the option are required.

For more details of the MULTIIMSID option, see "OPTION control statement" on page 376. For more details about the key field, see "Format" on page 367.

IMS HP Pointer Checker documentation and updates

This topic explains where to find DB2 and IMS Tools information on the Web, and explains how to receive information updates automatically.

IMS HP Pointer Checker information on the Web

The DB2 and IMS Tools Library Web page provides current product documentation that you can view, print, and download. To locate publications with the most up-to-date information, refer to the following Web page:

http://www.ibm.com/software/data/db2imstools/library.html

You can also access documentation for many DB2 for z/OS and IMS Tools from the Information Management Software for z/OS Solutions Information Center:

http://publib.boulder.ibm.com/infocenter/imzic

Documentation for many DB2 Tools that run on Linux, UNIX, and Windows systems can be found in the IBM DB2 Tools for Linux, UNIX, and Windows Information Center:

http://publib.boulder.ibm.com/infocenter/mptoolic/v1r0/index.jsp

IBM Redbooks® publications that cover DB2 and IMS Tools are available from the following Web page:

http://www.ibm.com/software/data/db2imstools/support.html

The Data Management Tools Solutions Web site shows how IBM solutions can help IT organizations maximize their investment in DB2 and IMS databases while staying ahead of today's top data management challenges:

http://www.ibm.com/software/data/db2imstools/solutions/index.html

Receiving information updates automatically

To automatically receive a weekly email that notifies 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 Support service. You can customize the service so that you receive information about only those IBM products that you specify.

To register with the My Support service:

1. Go to http://www.ibm.com/support/mysupport.
2. Enter your IBM ID and password, or create one by clicking **register now**.
3. When the My Support page is displayed, click **add products** to select those products that you want to receive information updates about. The DB2 and IMS Tools category is located under **Software** -> **Data and Information Management** -> **Database Tools & Utilities**.
4. Click **Subscribe to email** to specify the types of updates that you would like to receive.
5. Click **Update** to save your profile.

Accessibility features for IMS HP Pointer Checker

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 of the product is the keyboard-only operation for ISPF editors. It uses the standard TSO/ISPF interface.

Keyboard navigation

You can access the information center and 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 the following publications for information about accessing ISPF interfaces:

- *z/OS ISPF User's Guide, Volume 1*, SC34-4822
- *z/OS TSO/E Primer*, SA22-7787
- *z/OS TSO/E User's Guide*, SA22-7794

These guides describe how to use ISPF, including the use of keyboard shortcuts or function keys (PF keys), include the default settings for the PF keys, and explain 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.

Summary of changes

This topic describes the history of this information.

IMS High Performance Pointer Checker User's Guide

This section describes the history of IMS High Performance Pointer Checker User's Guide.

SC18-9974-01

This edition covers the changes of the following functional changes or new functions:

PK31298

By this APAR, IMS HP Pointer Checker supports the following two functions:

- Sending exception notification messages under IMS High Performance Image Copy (IMS HP Image Copy) Version 4 Release 1.
- Limiting the number of exception notification messages, to TSO users, to 50.

PK33114

By this APAR, IMS HP Pointer Checker supports all of the existing functions under IMS Version 10 and the following new functions:

- Parallel RECON access
- DBRC Migration/Coexistence
- z/OS Version 1 Release 7 DFSMSdfp™ large sequential data sets
- Data set FlashCopy
- ACBGEN in 31-Bit

PK33317

By this APAR, message FABP1098E is changed and message FABP1099E is added to report the details of an error of partition selection, and message FABK0009E is added to handle data set catalog error in Space Monitor (SPMN).

PK33419

By this APAR, HD Pointer Checker (HDPC) supports the following functions:

- Quick HASH evaluation of single-step HASH check under IMS HP Image Copy
- The DBALL keyword for the PROCCTL statement
- The EMPTYKEYSIN keyword for the HPSRETCD statement
- Three digits enabled for return code specification

PK40389

This APAR introduces a new message FABP2018I to notify that DBALL=YES option is in effect.

PK44413

By this APAR, message FABP2031E is added to handle an error in the TYPE=CHECK process.

PK45122

By this APAR, IMS HP Pointer Checker supports IBM IMS Tools Knowledge Base for z/OS.

PK50537

By this APAR, message FABP2119E is added to notify the error on search of standard image copy record.

PK51609

This APAR provides the following new functions that are added to the DB Historical Data Analyzer (DBHDA) component.

- KEYDATA format support for the TYPE=LIST report
- Multiple-IMSID support

PK57593

By this APAR, HD Pointer Checker (HDPC) supports the following functions:

- Secondary index HASH check function
- Index key HASH check support

PK57594

By this APAR, HD Pointer Checker (HDPC) supports the following functions:

- Recovery needed flag notification
- The RECOVNEEDED keyword of the HPSRETCD statement
- The TYPE=BLKMAP completion message

PK59723

By this APAR, the explanation of message FABP3630E is corrected.

SC18-9974-00

This edition covers the functions of IMS HP Pointer Checker for z/OS, Version 2 Release 2. This version improves usability and makes it easy and quick to detect database status as described below.

In this version, the two former versions, *IMS High Performance Pointer Checker User's Guide* Volumes 1 and 2, are combined into one book.

Improved usability:

- Sets the site default of HD Pointer Checker and Space Monitor
  - The HD Pointer Checker Site Default Generation utility is supported. You can set your own default values of the optional parameters of the PROCCTL control statements.
  - The Space Monitor Site Default Generation utility is supported. You can set your own default values for monitoring threshold values of the control statements.
- History data set maintenance is improved in DB Historical Data Analyzer
  - You can specify retention days for the History records. DB Historical Data Analyzer job will delete the History records of which life time exceeds the retention days.

Detecting database status quickly and easily:

- Shows error descriptions in the HASH Check
  - HD Pointer Checker will show the error description when it detects pointer errors during the HASH Check.
- Exception notification is sent in HD Pointer Checker and Space Monitor
  - HD Pointer Checker will send notification messages to TSO user IDs, when it detects a pointer error or a T2 error.
  - Space Monitor will send notification messages to TSO user IDs, when it detects threshold exceptions.
- Monitors the DBDS size in Space Monitor

  - Space Monitor checks and monitors the DBDS size and its percentage within the IMS database size limit. If the size or percentage exceeds the threshold values, Space Monitor reports a threshold exception.

The format of some reports are different from those of IMS HP Pointer Checker Version 1 and Version 2 Release 1.

IMS High Performance Pointer Checker User's Guide Volume 1

This section describes the history of IMS High Performance Pointer Checker User's Guide Volume 1.

SC18-7257-02

This edition covers the changes of the following functional changes or new functions:

PK22791
HD Pointer Checker supports the HALDB partition reorganization number validation.

PK20840
HD Pointer Checker supports Fast Recovery Image Copy data sets.

PK16212
The HISTLOCK option is introduced to specify how HD Pointer Checker locks HISTORY data sets.

PK16210
Statistical reports of HD Pointer Checker are changed.

- The HISAM data set statistics reports include additional information about the database data set size.
- The HD data set statistics reports include additional information about the database data set size.

PK16055
Message FABP1520W is changed to clarify whether the detected error is a T2 error or a pointer error.

PK12823
The way in which IMS HP Image Copy calls HD Pointer Checker is changed so as to enable the support for the selectable return code of Space Monitor in IMS HP Image Copy. HD Pointer Checker issues an error message when the maintenance level of IMS HP Image Copy is lower than expected.

PK11923
HD Pointer Checker reports and the HISTORY data set record type are changed.

- The following HD Pointer Checker reports are changed.
  - DB Record Distribution Statistics report
  - Partition Statistics report
  - Database Statistics report
- Under hash check options called from IMS HP Image Copy, the following reports are newly added or changed.
  - HD Data Set Statistics report
  - HISAM Data Set Statistics report
  - Partition Statistics report

  - Database Statistics report
- Some new types of record are added to the HISTORY data set.

PK11662
HD Pointer Checker can be called from IMS Database Recovery Facility.

PK11601
HD Pointer Checker performs more strict syntax checking on the HPSRETCD data set.

PK07981
Reclassification of message FABP0973E as FABP0973W.

PK07708
Support of IMS HP Image Copy Version 4.

PK05745
HD Pointer Checker supports pointer checking of secondary index databases.

PK04599
HD Pointer Checker calls Space Monitor to report space utilization.

PK04598
HD Pointer Checker supports checking of direct-address pointers in HISAM databases.

PK01212
Data set name is set to ″N/A″ in the HISTORY data set when HD Pointer Checker is called from IMS Parallel Reorganization.

PK00511
HD Pointer Checker supports selectable return code.

SC18-7257-01

This edition covers the changes of the following functional changes or new functions:

PQ98104
A new message FABP2120W is added by this APAR.

PQ97234
Added the support for the single step HASH checking during the Concurrent Image Copy process, which is invoked by CIC or ACIC statement of ICEIN in IMS HP Image Copy Version 3.2. A new message FABP3864E is added by this APAR.

PQ95904
Made the IMS2 DD statement optional for HD Pointer Checker.

PQ93811
Added new messages for the HASH checking in IMS Parallel Reorganization Version 3.

PQ94383
HD Pointer Checker enhancement for multiple entries per day in the HISTORY data set.

PQ91816
HD Pointer Checker enhancement on the partition and database statistics reports.

Compression factor
The compression factor is added to the rightmost column in the VL

SEGMENT LENGTH STATISTICS part of the Partition Statistics report and the Database Statistics report. The compression factor shows the data compression factor by the segment edit/compression exit routine.

Rate of segment I/O occurrence
This is a new part in the Partition Statistics report and the Database Statistics report. It shows the probability of doing I/O when getting access to a target segment from a source segment by a hierarchical database structure.

PQ80355
HD Pointer Checker enhancement for the symbolic pointer checking.

- Checks each value of the symbolic pointer in secondary index databases, and the sequence key in the index target segment (ITS). The check is activated by the new option SYMIXCHK=YES.
- Checks each LPCK value of the symbolic Logical Parent pointer and the sequence key of the logical parent segment. This check is activated by the new option SYMLPCHK=YES.
- Checks that the number of index pointer segments is equal to the number of index source segments for symbolic index pointers.
- Checks that the sum of CTR values of logical parents is equal to the number of logical child segments even if they have symbolic pointers. Before this APAR, this check could not be done when symbolic pointers were defined in the logical child segments.
- Checks the numbers of occurrences among the paired logical children in Bidirectional Physically Paired Logical Relationship and that they have symbolic pointers.

PQ83676
Increased the region size parameter in the HD Pointer Checker IVP job step to avoid insufficient storage.

PQ80186
IMS V9 support. The main changes are to support the HALDB Online Reorganization.

- HD Pointer Checker can process the active set of DBDS (A-J&X) or (M-V&Y).
- HD Tuning Aid can process the KEYSIN data set created from any side of DBDS, (A-J&X) or (M-V&Y).
- HD Pointer Checker can process the image copy data set created in a large block interface by the IMS Version 9 IC utility.
- The functional specification of PHIDAM primary index processing is changed. Before this APAR, when DATASET=IMAGECOPY is specified in the DATASET statement of the primary index in the PROCCTL data set, the database data set was processed. After this APAR, FABP2047W is issued and the primary index data set is not checked.

PQ78859
HD Tuning Aid is enhanced to support the new %OPTION statement.

PQ75900
HD Pointer Checker is enhanced to run with ACBLIB.

SC18-7257-00

This edition covers the functions of IMS HP Pointer Checker Version 2. This version improves the performance and decreases the DASD space needed. It also improves usability and the reporting function.

Performance is improved by:

- Parallel scans of databases
  - A new option, SCANGROUP=, is introduced.
  - Database data sets and image copy data sets are read in parallel among different scan groups.
- Introducing smaller sort records
  - Both the quantity and the record length of sort record are reduced. This makes the DASD space smaller.
  - When PROC TYPE=ALL, IXKEYCHK=NO, EPSCHK=NO is specified in the PROCCTL data set, only the MERGIN work data set remains.
- Eliminating the MERGE step
  - Eliminating the MERGE step of the segment and the pointer records.
  - Reading the segment records and pointer records sequentially, instead of processing the MERGE step.
- Optimizing the work buffers
  - Optimizing the work buffers of the work data sets automatically.
- Providing VSCR relief
  - The work buffers and some internal control blocks are moved to above 16M.

Usability is improved by simplifying the setup and operation procedures as follows:

- JCL
  - The number of work data sets is reduced. The estimation of DASD space is simpler.
  - There is no need to separate SCAN jobs from CHECK job when scanning the databases in parallel. All you need to do is to specify the SCANGROUP option in one job.
- Report only option
  - A new option, PTRCHK=NO, is introduced. When you choose this option, pointer checking is omitted and only statistics reports are generated.

Reporting is improved by:

- New reports
  - New reports for the DBD source and the DBD map are generated by choosing new options, DECODEDBD and MAPDBD.

IMS High Performance Pointer Checker User's Guide Volume 2

This section describes the history of IMS High Performance Pointer Checker User's Guide Volume 2.

SC18-7258-02

This edition covers the changes of the following functional changes or new functions:

PK16212

The HISTLOCK option is introduced to specify how HD Pointer Checker locks HISTORY data sets.

PK16210

Export utility is changed.

- Export utility shows the percentage of the database data set size to its maximum size.

PK11923

The HISTORY data set record type is changed.

- Export utility is added to DB Historical Data Analyzer.
- Some new type records are added to the HISTORY data set.

SC18-7258-01

This edition covers the following functional changes or new functions:

PK11923

Export Utility is added to DB Historical Data Analyzer.

A new program FABGXEXP, which is the main program of Export Utility is added.

A new optional attribute, EXPORTABLE is added to the HISTORY data set.

PK04599

Space Monitor can be run in the HD Pointer Checker job step.

The output reports data sets are allocated dynamically by Space Monitor.

PQ94383

A new optional attribute, MULTIENT is added to the HISTORY data set.

PQ81450

DB Segment Restructure is enhanced to accept maximum of 1500-byte literals per database for the CHG statement in the SYSIN data set of FABRRELD.

PQ80186

IMS V9 run under support and HALDB Online Reorganization supports:

- Space Monitor and DB Historical Data Analyzer to process the HISTORY data set created from any side of DBDS, (A-J&X) or (M-V&Y).
- DB Segment Restructure to process (A-J&X) or (M-V&Y) DBDS.
  - FABRUNLD program unloads the database from the active set of DBDS.
  - FABRRELD can reload the database to (A-J&X) or update the segment data in the active set of DBDS.

PQ78036

FABK3533E and FABK3542E are added to Space Monitor.

PQ75900

The ’ENVIRONMENT’ section is added to the HD Analysis report. The section contains information about IMSID.

SC18-7258-00

This book covers the following three utilities of IMS HP Pointer Checker Version 2:

- DB Historical Data Analyzer
- Space Monitor
- DB Segment Restructure

HD Pointer Checker utility has been changed in IMS HP Pointer Checker Version 2. These three utilities have the same functions as Version 1.

Part 2. HD Pointer Checker

Chapter 2. Overview of HD Pointer Checker

This chapter discusses the overview of the HD Pointer Checker utility.

Topics:

Program functions

This section explains about the program functions of the HD Pointer Checker utility.

Detecting errors in a database

This is the primary function of HD Pointer Checker.

It finds errors by checking direct pointers (physical pointers, logical pointers, hierarchical pointers, pointers to free space, and index pointers) in IMS full-function databases.

The supported databases are as follows:

- HDAM databases
- HIDAM primary and index databases
- PHDAM databases
- PHIDAM primary and index databases
- HISAM (including SHISAM) databases
- HDAM/HIDAM secondary index databases
- PHDAM/PHIDAM secondary index databases (PSINDEX)

HD Pointer Checker checks the consistency between direct pointers and the Relative Bytes Address (RBA) of segments and free space.

HD Pointer Checker has the following optional check other than the direct pointer check:

- **Index Key Check function**
  HD Pointer Checker detects incorrect index keys for the following databases:
  - HIDAM and PHIDAM primary index database
  - Secondary index database for HDAM, HIDAM, PHDAM, and PHIDAM
- **Symbolic Pointer Checking**
  HD Pointer Checker detects the missing and errors in the symbolic LP pointers and the secondary index symbolic pointers.
- **HALDB EPS Checking**
  HD Pointer Checker detects the inconsistency among the HALDB Extended Pointer Sets, indirect pointers in the Indirect List Data Set (ILDS), and the RBAs of pointed segments.
- **HALDB reorganization number verification**
  HD Pointer Checker detects errors of reorganization counts of HALDB partitions.

If some errors are detected in the direct pointer checking and the optional checking, HD Pointer Checker will report pointer error. When pointer error is reported, this means that the database is damaged.

Other than pointer errors, HD Pointer Checker will report T2 errors, if a database contains redundant spaces that are not segments nor free spaces. Such a space is called T2 record in HD Pointer Checker. When a T2 error is reported, it does not mean that the database is damaged but just indicates there are some redundant space in the database. When HD Pointer Checker detects a pointer error or a T2 error, it will notify it by a message and return code. Optionally, it will send a notification message to TSO user IDs.

HD Pointer Checker provides two methods to check pointers. One is called the Standard Check function, where HD Pointer Checker compares the pointers and RBAs of the pointed segments one by one. Another is called the HASH Check function, where HD Pointer Checker compares the sum of pointer values with the sum of RBAs of the pointed segments, rather than comparing each of the pointer values and segments one by one.

The Standard Check function takes more DASD and CPU resource and takes longer elapsed time, than the HASH Check function. The HASH Check function provides fast pointer checking. However, it does not show the details of errors, and some optional functions are not available for the HASH Check function. Therefore, the HASH check function is more suitable for regular checking while the Standard Check function is recommended when checking for some more important points. For example, you can use the Standard Check function to identify the location of an error that is found in a HASH Check.

Describing the database

HD Pointer Checker prints a detailed description of the condition of the IMS database. This includes comprehensive statistical information about pointers, segments, and free space.

Printing HDAM, HIDAM, PHDAM, or PHIDAM database blocks

HD Pointer Checker prints hexadecimal and character dumps of database control intervals or blocks. Each dump includes a map of all the segments and free space elements in the block, as well as the relative byte address (RBA) for each printed line.

Finding absolute disk addresses

HD Pointer Checker converts an RBA into the corresponding absolute disk address. It calculates the cylinder, track, record, and offset for each input RBA.

Finding all pointers to a target segment

HD Pointer Checker lists the RBAs of all pointers (in other segments, possibly in other databases) that point to a specific target segment.

Creating a history record

The results of a HD Pointer Checker run are stored in the HISTORY data set as a historical record. For the details, see “Creating HISTORY data set” on page 368.

Space Monitor

HD Pointer Checker can call the Space Monitor utility and process the following functions:

- Monitor and log data set space utilization
- Describe space utilization

For the details of the Space Monitor functions, see Part 5, “Space Monitor,” on page 489.

Space Monitor functions are called when the SPMNIN and SPMNSPDT DD statements are specified for FABPMAIN JCL or when SPMN=YES is specified on a DATABASE statement in the PROCCTL data set. For more details, see “DD statements” on page 38 or “OPTION statement” on page 87.

If you want to store the Space Monitor reports in the IMS Tools KB Output repository, you must call Space Monitor by setting SPMN=YES on a DATABASE statement.

Storing reports in the IMS Tool KB Output repository

You can store the reports of HD Pointer Checker and Space Monitor in the IMS Tools KB Output repository. To do this, you need to set up the environment with the following steps:

1. Install the IBM IMS Tools Knowledge Base product.
2. Set up an IMS Tools KB server.
3. Register IMS HP Pointer Checker to the registry of the IMS Tools KB server.
4. If needed, add the RECON.
5. Confirm that IMS Tools KB server is properly initialized before you run HD Pointer Checker jobs.

For detailed procedures, see *IMS Tools Knowledge Base for z/OS User’s Guide* (SC18-9963).

Typical uses

HD Pointer Checker can be used as follows:

- Monitor databases regularly, to detect either direct pointer errors or the need for a database reorganization.
- Analyze a corrupted database as part of the repair process, thus reducing the diagnostic and repair time spent by programmers or analysts.

Program structure

The HD Pointer Checker processor (FABPMAIN) runs under the IMS batch region controller (DFSRRC00) and controls all of the HD Pointer Checker processes except the FABPCHRO program.

The FABPCHRO program runs as a batch job. The FABPCHRO program prints the absolute disk address of user-specified relative byte address.

HD Pointer Checker can run with multiple IMS versions and releases without reinstalling the product, as far as the version/release is supported.

Restrictions and considerations

The HD Pointer Checker applies to the following IMS databases:

- HDAM
- PHDAM
- HIDAM and primary index databases
- PHIDAM and primary index databases
- HISAM databases
- HDAM/HIDAM/HISAM secondary index databases
- PHDAM/PHIDAM secondary index databases (PSINDEX)

HD Pointer Checker does not support IMS Partition DB (5697-A06, 5697-D85) or any other products with an equivalent function.

HD Pointer Checker checks pointers pairwise: It looks at a pointer and its target to determine if there is an error. It does not look at all segments in the entire twin chain at the same time. Errors can sometimes occur in a way that can ″fool″ the HD Pointer Checker into thinking there are no errors. For instance, the PP pointer of a child segment might point to the wrong parent—that is to say the pointer points to the correct segment type, but the wrong key. However, this is extremely rare.

Restrictions and considerations for SHISAM and HISAM databases:

- HD Pointer Checker supports SHISAM databases. Since a SHISAM database has no direct pointers, no pointer checking is done. However, several reports presenting various segment information on SHISAM databases are generated.
- HD Pointer Checker supports HISAM databases. However, the direct-address pointers in the HISAM databases can be validated correctly only after an initial load or a reorganization. DL/I does not set a delete flag in the logical record in an HISAM overflow data set when the record is deleted. Therefore, the deleted logical record remains in the HISAM database. On the other hands, DL/I deletes the direct-address pointer that points the logical record. Therefore, in such a case, there is no direct-address pointer pointing to the logical record. HD Pointer Checker cannot determine correctly, such status caused by the segment deletion or some kind of pointer error. HD Pointer Checker determines the status as correct. When some logical records in the overflow data set of HISAM database are not pointed to by any of the direct address pointers, HD Pointer Checker issues FABP0973W or FABP1992W in the report and returns RC=00.

Restrictions and considerations for HD database whose size is greater than 4 Giga bytes:

For an HDAM and HIDAM database using OSAM access method, a database data set with an even-numbered block size may exceed 4 GB. In such a case the following rule applies:

- The highest bit (bit 33) of an RBA with a capacity of 8 GB must be moved to the lowest unused bit position (bit 1). You should use the 32-bit value to show the original RBA.
  - For example, to specify the hexadecimal value x’10000F000’ as a 32-bit odd value, you should specify as follows:

    ```
    BLOCKDUMP=(0000F001,001)
    ```

  - Do not use a 32-bit odd value to specify an original odd RBA. An odd value is interpreted by the HD Pointer Checker as an RBA beyond 4 GB.

Restrictions and considerations for HALDB:

If an online reorganization has not completed, there will be a HALDB in two active sets of data sets (both A-J&X and M-V&Y), and HD Pointer Checker cannot be run for those HALDBs.

Restrictions and considerations for image copy data set:

- HD Pointer Checker accepts the following image copy types as input to the database data set:
  - A batch image copy
  - A compressed batch image copy taken with IMS Image Copy Extensions or IMS HP Image Copy
  - An SMSNOCIC type image copy and an SMSOFFLC type image copy
  - A Fast Recovery image copy taken with IMS HP Image Copy Version 4 or later
- HD Pointer Checker can process the following image copies. But, if the database is updated while image copy is being taken, HD Pointer Checker might detect pointer errors even if there are actually not pointer error.
  - An online image copy
  - A concurrent image copy taken by a batch image copy or IMS HP Image Copy
  - An SMSCIC type image copy and an SMSONLC type image copy
- HD Pointer Checker does not support a Database Image Copy 2 (IC2) image copy data set that contains multiple image copies. For more information on the online image copy and the concurrent image copy, refer to Chapter 8, “HD Pointer Checker online considerations,” on page 265.

Restrictions and considerations for the HASH Check function:

When the HASH Check function is used, the following restrictions should be considered:

- A pointer value has to correspond to the RBA of the segment to which the pointer points. The sum of the pointer values for a specific pointer type also has to correspond to the sum of the RBAs of the given segment type.
- The location of the errors cannot be determined precisely.
- Pointer value errors may theoretically compensate, but the probability to make such compensations is extremely low.
- Errors on the following pointer types cannot be detected by the HASH Check function:
  - The physical parent pointers that are not at the beginning or end of their twin chain.
  - Neither index pointers in PSINDEX, nor logical pointers in HALDB, because Index List Entry (ILE) in Indirect List Data Set (ILDS) is not checked.
  - Symbolic pointers

Restrictions and considerations for analyzing secondary index databases:

When the Index Key Check function is used, the following restriction applies:

- When a index source segment is split to the prefix part and data part, the index key check function cannot validate the key in the split segment.
- When the /CK field is specified on the SUBSEQ= operand of the XDFLD statement to make key fields unique, the Index Key Check option is not effective.

On checking the suppressed index pointer segment, the following restriction applies:

- The suppressed segment is checked during the scan of the INDEXed database.
- When a /CK field is specified on the SUBSEQ= operand of the XDFLD statement, HD Pointer Checker cannot get the field data. If the Secondary Index Database Maintenance Exit routine refers to a /CK field, the result is unpredictable.
- When a index source segment is split to the prefix part and data part, the index key check function cannot validate the key in the split segment.

Considerations for running HD Pointer Checker by multiple steps:

In TYPE=SCAN and TYPE=CHECK job steps, take note of the following:

- Performance is lower and sizes of the work data sets are larger than a TYPE=ALL job.
- The JCL statements are more complex than in the TYPE=ALL job.
- When the scan steps, which are composed of multiple jobs, run in a ULU region, be careful when specifying independent—that is, with neither logical nor index relation—databases. For details, read the description about message FABP2104E in Appendix A, “Messages and codes,” on page 595.

Since these restrictions or notices do not apply to the TYPE=ALL job, we strongly recommend that you select the TYPE=ALL job. To do so, specify PROC TYPE=ALL in the PROCCTL statement.

Considerations for HD Pointer Checker that is called in an IMS Parallel Reorganization job:

When HD Pointer Checker is called from IMS Parallel Reorganization Version 3, the word ″N/A″ is set to the database data set name fields in the HD Pointer Checker report. For the data set names, see the data set information report of IMS Parallel Reorganization.

Restrictions and Considerations for HD Pointer Checker calling Space Monitor:

Space Monitor can be called from the HD Pointer Checker FABPMAIN program and the single-step HASH checking option of IMS High Performance Image Copy (IMS HP Image Copy) Version 3 or later. However, Space Monitor cannot be called in HD Pointer Checker when it is run in the following environment:

- The multiple-step HASH checking option of IMS HP Image Copy
- IMS Parallel Reorganization
- IMS Database Recovery Facility

Data flow

The HD Pointer Checker processor (FABPMAIN) controls and invokes all or some of the following HD Pointer Checker processes and DFSORT processes as one job based on the control statements.

1. **SCAN Process**

This process reads the input database data sets or image copy data sets and also contains the following processes:

- **HISAM:** This process creates sort records, or sums hash values for pointers and segments in the HISAM databases.
- **INDEX/PSINDEX:** This process creates sort records and key records for pointers in the HIDAM and PHIDAM primary index and secondary index databases. It also sums up hash values for pointers to the HIDAM, PHIDAM, and primary databases.
- **HDAM/HIDAM/PHDAM/PHIDAM:** This process creates sort records, or sums up hash values for pointers and segments in the HDAM, HIDAM, PHDAM, and PHIDAM databases. It also creates both key records for source segments and root segment key records in the HDAM, HIDAM, PHDAM, and PHIDAM databases for the Index Key check. The root segment key records are also created in the KEYSIN data set.

This process reads RECON data sets and does the HALDB partition reorganization number validation.

This process prints statistical information, database error messages, and error-correction data (maps and dumps of database blocks).

This process runs DFSORT (or its equivalent program), and sorts all of the sort records.

2. **CHECK Process**

   This process contains the following processes:

   - **HASH Check:** This process matches the hash totals of pointers and segments generated by the SCAN process (HISAM, INDEX, and HDAM/HIDAM/PHDAM/PHIDAM processes) to give a general indication of database integrity.
   - **Validate/Evaluate:** This process validates pointers, evaluates segments, prints database error messages, and creates a control record for each database error.
   - **Index Key Check:** This process matches key data between index pointer segments and HIDAM/PHIDAM root segments, or HISAM, HDAM, HIDAM, PHDAM or PHIDAM pointer source segments. This process also prints database error messages.
   - **EPS Healing:** This process refers to the ILDS data sets and validates the pointers.
3. **BLOCKMAP Process:** This process prints error-correction data (pointers to segments with errors, and the maps and dumps of database blocks that contain the segments with errors). This process may run as a separate job (or job step) on the control statements.

In a typical job stream, the following HD Tuning Aid programs and sort program are also executed:

4. **FABTROOT:** This program prints the Actual Roots Per Block report and creates sort records to be used to create other reports.
5. **DFSORT:** DFSORT (or its equivalent program) sorts all of the sort records from the previous job step.
6. **FABTRAPS:** This program prints the Assigned Roots Per Rap report and the Assigned Roots Per Block report.

Figure 1 on page 34 shows the data flow for HD Pointer Checker.

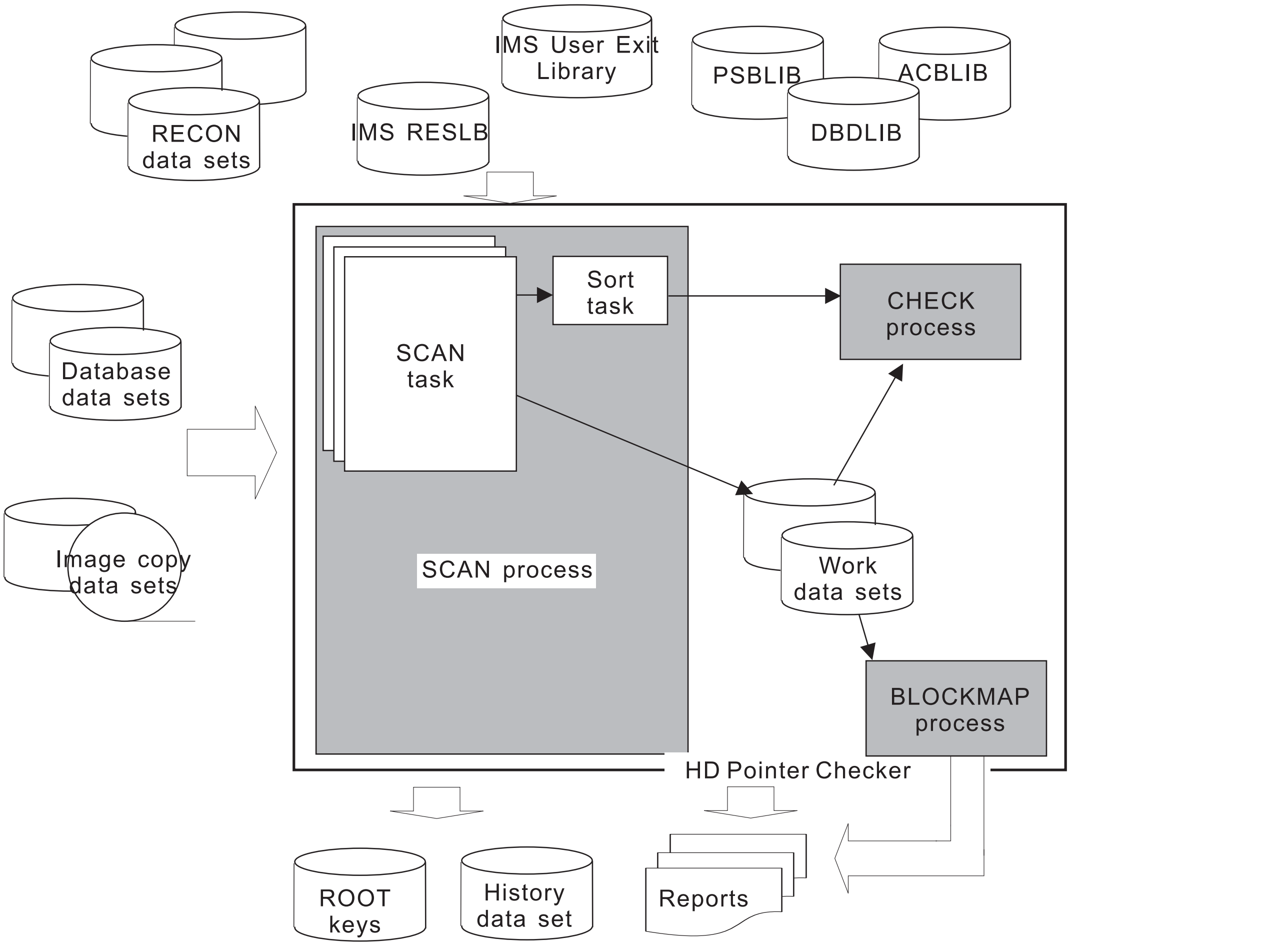


Figure 1. Data flow for HD Pointer Checker

The following HD Pointer Checker program is usually executed in a stand-alone job:

- **FABPCHRO:** This program prints the absolute disk address of user-specified relative byte addresses. For details, see Chapter 4, "Operating instructions for Disk Address Analyzer," on page 237.

You can run HD Pointer Checker with several databases in a single run. The best way is to run with a minimal set of logically related databases in one job.

Note: It is important to include *all* logically related databases and index databases in the same run. Otherwise, the pointer checking will be incomplete, and huge number of invalid error messages may result.

The scan and check processes can be run in separate job steps. However, it is strongly recommended that you use 'PROC TYPE=ALL' and not 'PROC TYPE=SCAN and CHECK,' because if you use 'PROC TYPE=ALL,' the JCL statements will be simpler, the performance will improve, and the sizes of work data sets will be smaller.

Table 1 can be used as a guide in selecting the proper job process.

Table 1. Selecting job steps for HD Pointer Checker and HD Tuning Aid

Job function	Job processes	Applicable database
Checking pointers	FABPMAIN	This process accepts any combination of databases.
Checking pointers and root-segment locations	FABPMAIN	This process accepts any combination of databases.
	FABTROOT DFSORT FABTRAPS	HDAM, HIDAM, PHDAM, and PHIDAM databases
Finding disk addresses	FABPCHRO	Any database

Chapter 3. Operating instructions for the HD Pointer Checker processor

The HD Pointer Checker processor (FABPMAIN) runs under the IMS batch region controller (DFSRRC00) as one job step, and controls all of the HD Pointer Checker processes.

Before HD Pointer Checker runs with the HISTORY option, the DB Historical Data Analyzer utility must be run to allocate and initialize the HISTORY data set which will be an input data set to HD Pointer Checker. (For an explanation of the HISTORY option, see the description of “HISTORY= option parameter” on page 90, and for the HISTORY data set, see the description of “HISTORY DD” on page 40.) For a complete documentation on the DB Historical Data Analyzer utility, see Part 4, “DB Historical Data Analyzer,” on page 355.

A typical job stream executes both HD Pointer Checker and the HD Tuning Aid. Refer to Part 3, “HD Tuning Aid,” on page 305 for a complete documentation on the HD Tuning Aid utility.

Topics:

- “Job control language”
- “Input” on page 71
- “Output” on page 109

Job control language

This section discusses the job control language (JCL) of HD Pointer Checker.

FABPMAIN JCL

To run the HD Pointer Checker processor (FABPMAIN), supply an EXEC statement with PARM parameters, and the appropriate DD statements.

EXEC statement

It is required that you run the FABPMAIN program in the IMS region controller program (DFSRRC00). Specify PGM=DFSRRC00 in the EXEC statement and specify parameters for the IMS region controller program as follows:

```
//     EXEC PGM=DFSRRC00,
//          PARM=(region,FABPMAIN,psbname,,,,,,,,,,,dbrc,N)
//
```

The number of commas after *psbname* is 11. The PARM parameter has the same format as that used in the DLIBATCH or the DBBBATCH procedure. It is documented in the following manuals:

- *IMS Version 8 Installation Volume 2: System Definition and Tailoring*
- *IMS Version 9 Installation Volume 2: System Definition and Tailoring*
- *IMS Version 10 Installation Volume 2: System Definition and Tailoring*

The following are parameters in parentheses of PARM=:

region

Specifies an IMS region type. It is a required parameter for specifying the region type. The possible values are ULU, DLI, or DBB.

- ULU is for running HD Pointer Checker as utility batch program region (ULU region) by using the DBD library.

- DLI is for running an offline DL/I batch processing program region (DLI region) by using the PSB and DBD libraries.
- DBB is for running an offline DL/I batch processing program region (DBB region) by using the ACB library.

psbname

Specifies the PSB name that contains the PCB of the processing databases. The PSB name must be defined as a PSB with LANG=ASSEM or LANG=COBOL. It must refer directly or indirectly to all input databases. The PSB, in which less than 2,500 of database data sets are referred to, can be used in the HD Pointer Checker run.

It is required if DLI or DBB is specified in the region parameter. It is not required if ULU is specified in the region parameter.

dbrc

Specifies whether database recovery control is to be used. The possible values are Y or N.

- Y is for using the database recovery control
- N is for not using the database recovery control

When HD Pointer Checker runs in the following condition, you have to specify Y:

- HISTORY=Y is specified in the PROCCTL data set.
- The processing database is a HALDB.
- If you specify an image copy data set as input and you omit the image copy data set DD statement. HD Pointer Checker refers to the RECON data set for the image copy data set name and allocates the data set dynamically.

DD statements

This section describes the DD statements.

As a general rule, DD names that start with HKT are reserved for IBM use.

DD statements for input and output data set: Table 2 summarizes the DD statements for input and output data set.

Table 2. FABPMAIN DD statements

DDNAME	Use: Input (In), Output (Out), or Work	Format	PROC TYPE= Required (R) or Optional (O)				Dynamically allocatable?	Description
			ALL	SCAN	CHECK	BLKMAP		
STEPLIB	In	PDS	R	R	R	R	No	
IMS	In	PDS	O	O	O	O	No	A data set contains PSB and DBD load module.
IMSACB	In	PDS	O	O	O	O	No	A data set contains ACB.
IMS2	In	PDS	O	O			No	User exit load modules.
DFSRESLB	In	PDS	R	R	R	R	No	IMS load module.
DFSVSAMP	In		R	R	R	R	No	Buffer information required by the DL/I buffer handler
RECON*x*	In		O	O	O	O	No	RECON data sets. Required if image copy data set is allocated dynamically, or if HALDB is processed.
HISTORY	In/Out	VSAM KSDS	O	O	O		No	Input to Database Historical Data Analyzer.
KEYSIN	Out		O	O			No	Input data set to HD Tuning Aid.
PROCCTL	In	LRECL=80	R	R	R	R	No	User-specified control statement.

Table 2. FABPMAIN DD statements (continued)

DDNAME	Use: Input (In), Output (Out), or Work	Format	PROC TYPE= Required (R) or Optional (O)				Dynamically allocatable?	Description
			ALL	SCAN	CHECK	BLKMAP		
ddname	In		O	O	O	O	Yes	The input database data set that is to be processed. If the ddname DD statement is omitted, dynamic allocation is available.
HPSRETCD	In	LRECL=80	O	O	O	O	No	Return code statement that you specify.
IEFRDER	Not used	DUMMY	R	R	R	R	No	Primary system log data set.
SYSOUT*nn* (*nn*=01 to 04)	Out	SYSOUT	O	O	O		Yes	Data set contains messages produced by DFSORT.
PRIMAPRT	Out	LRECL=133	R	R	R	R	No	Output data set contains primary reports.
STATIPRT	Out	LRECL=133	R	R	R	R	No	Output data set contains statistical reports.
VALIDPRT	Out	LRECL=133	R	R	R	R	No	Output data set contains legends and validation reports.
EVALUPRT	Out	LRECL=133	R	R	R	R	No	Output data set contains evaluation reports.
EVALUPR2	Out	LRECL=133	O	O				Output data set contains evaluation reports.
EVALIPRT	Out	LRECL=133	O	O	O		No	
SNAPPIT	Out	LRECL=133	R	R	R	R	No	Output data set contains block maps and block dumps.
SUMMARY	Out	LRECL=133	R	R	R	R	No	Output data set contains summary reports.
DBSRCPRT	Out	SYSOUT	O	O	O	O	Yes	Used if DECODEDBD=YES.
DBMAPPRT	Out	SYSOUT	O	O	O	O	Yes	Used if MAPDBD=YES.
SYSUDUMP	Out	SYSOUT	O	O	O	O	No	
SORTWK*nn* SRTSWK*nn* SRTXWK*nn* SRTEWK*nn*	Work			O	O	O	Yes	Used by DFSORT program.

Besides the data sets in this table, HD Pointer Checker allocates the following data sets dynamically and uses them as temporary data sets. You do not need to specify them in the JCLs, but you must not use these names in your JCL statements, because they will be used by HD Pointer Checker.

- STATIP*nn* DD (*nn*=01, 02, 03, ...)
- VALIDP*nn* DD (*nn*=01, 02, 03, ...)
- SNAPPI*nn* DD (*nn*=01, 02, 03, ...)
- FSESTA*nn* DD (*nn*=01, 02, 03, ...)
- FSESTAT DD
- LIUIN/LIUOUT/LIUPRINT/LIUPUNCH

For other DD statements used by IMS Library Integrity Utilities when REPORT DECODEDBD=YES or MAPDBD=YES is specified, see *IMS Library Integrity Utilities User's Guide*.

STEPLIB DD

This defines the input data sets as follows:

- IMS HP Pointer Checker production library (mandatory)
- IMS RESLIB (mandatory)
- IMS Library Integrity Utilities production library (optional)
- The library that contains the data compression module (optional)
- The library that contains the partition selection exit routine (optional)
- The library that contains DFSMDA members for dynamic allocation (optional)

Note: If the input you are using is an image copy data set compressed by IMS Image Copy Extensions for z/OS, the STEPLIB DD data set also needs to contain the data compression module, FABJCMP1 or FABJCMP2, that was used when the image copy was taken.

IMS DD

This optional input data set is a library (partitioned data set) that contains your PSB and DBD load modules. It is required when HD Pointer Checker runs with ULU or DLI specified in the EXEC parameter. If ULU is specified, it must contain the DBD library. If DLI is specified, it must contain the PSB and DBD libraries. It must contain all DBDs that are referenced (either directly or indirectly) by your PSB. If your PSB and DBDs are not in the same library, all appropriate libraries must be concatenated.

IMSACB DD

This optional input data set is an IMS.ACBLIB (partitioned data set). It is required if HD Pointer Checker runs in a DBB region.

IMS2 DD

This optional input data set is a library (a partitioned data set) that contains the following user exit load modules.

- HDAM and PHDAM randomizing modules. It is processed when you process the HDAM/PHDAM databases and when you check the HDAM/PHDAM home addresses.
- Segment edit/compression exit routine, if it is specified in the DBD.
- Secondary index maintenance exit routine, if it is specified in the DBD.
- FABPZWTO. if you want to use your FABPZWTO routine.

Note: When IMS2 DD statement is omitted, the user exit load modules are loaded from STEPLIB, JOBLIB, or LINKLIST library.

DFSRESLB DD

This required input data set contains the IMS load modules.

DFSVSAMP DD

This required input data set contains the buffer information required by the DL/I buffer handler.

Note: It is recommended that you specify the minimum required number of buffers and the minimum required buffer size in the DFSVSAMP data set.

RECON*x* DD

These optional data sets make up the DBRC RECON data set. They are required if HALDB is to be processed. If the RECON data sets are allocated dynamically by DFSMDA macro, these DD statements can be omitted.

HISTORY DD

This optional data set defines a HISTORY data set (VSAM KSDS), which is the input to the IMS HP Pointer Checker DB Historical Data Analyzer and Space Monitor. When the HISTORY option is specified, this data set is required, and must be allocated and initialized by the DB Historical Data Analyzer utility before you start an HD Pointer Checker run. DISP=SHR must be used.

If a multiple entries option of the HISTORY data set is activated by the DB Historical Data Analyzer, one or more database data set entries are taken per day.

For more detail information about the HISTORY data set, see "HISTORY data set (HISTORY)" on page 367.

KEYSIN DD

This optional output data set contains root segment keys that are used by module FABTROOT of the HD Tuning Aid. The LRECL, BLKSIZE, and RECFM must be coded on your DD statement. The data set is required when KEYSIN=YES is specified on the OPTION statement in the PROCCTL data set.

RECFM=VB is recommended.

PROCCTL DD

This required input data set contains the user-specified control statements that define the process type, process options, and databases to be processed by the HD Pointer Checker processor.

ddname **DD**

This optional input data set is an IMS database data set. There is one DD statement for each HISAM, index (the HIDAM index or the secondary index), HDAM, HIDAM, PHDAM, PHIDAM, or PSINDEX database data set to be processed. You must use the ddname that is specified in the DBD. The actual data set can be a real database, an image copy, an image copy compressed with IMS Image Copy Extensions for z/OS, a Fast Recovery image copy taken with IMS HP Image Copy for z/OS, or an image copy taken with the IMS Database Image Copy 2 utility. The DATASET=REAL|IMAGECOPY parameter of the DATABASE statement in the PROCCTL data set must match the type of the specified data set. If *ddname* DD is not specified, HD Pointer Checker allocates dynamically the database data sets or the image copy data sets to be processed. For details, see "Dynamic allocation for database data set and image copy data set" on page 69.

It is recommended that you omit the DD statement of database data set or image copy data set, because an appropriate database data set or the latest image copy data set will be selected and allocated dynamically by HD Pointer Checker.

Consideration for an Online Reorganization (OLR) of HALDB: If the database is online reorganization capable and you need to process the data set other than the one allocated dynamically, do as follows:

- For the real database data set, specify the DD name of the active DBDS.
- For the image copy data set, specify the DD name that matches the DD= parameter in the DATABASE statement in the PROCCTL data set. If DD=*ALL is specified or the DD= parameter is omitted, the DD name is assumed as that of the A through J or X data set group.

HPSRETCD DD

This optional input data set contains the control statements that you specify to define the return code of the HD Pointer Checker processor.

IEFRDER DD

This required statement defines the primary system log data set. It should be coded as DUMMY.

SYSOUT*nn* DD (nn=01 to 04)

This optional output data set contains messages produced by DFSORT. If the DD statements in the JCL are omitted, these data sets are dynamically allocated by HD Pointer Checker.

PRIMAPRT DD

This required output data set contains the primary reports produced by the HISAM, INDEX, HDAM/HIDAM, PHDAM/PHIDAM, and PSINDEX processes. If the BLKSIZE is coded on the DD statement, it must be a multiple of 133.

STATIPRT DD

This required output data set contains some of the statistical reports that the HISAM, HDAM/HIDAM, and PHDAM/PHIDAM processes produce. If BLKSIZE is coded on the DD statement, it must be a multiple of 133.

VALIDPRT DD

This required output data set contains the legends and the validation reports produced by the HISAM, INDEX, HDAM/HIDAM, PHDAM/PHIDAM, PHIDAM, and CHECK processes. If BLKSIZE is coded on the DD statement, it must be a multiple of 133.

EVALUPRT DD

This required output data set contains evaluation reports (produced by the CHECK process, the HASH option, or the Index Key option) and the Pointer Chain Reconstruction report (produced by the BLOCKMAP process). If BLKSIZE is coded on the DD statement, it must be a multiple of 133.

EVALUPR2 DD

This optional output data set contains evaluation reports (produced by the SYMIXCHK and SYMLPCHK options). If this DD statement is omitted, HD Pointer Checker writes the reports to SYSOUT=*. If BLKSIZE is coded on the DD statement, it must be a multiple of 133.

EVALIPRT DD

This is an optional data set that includes the reports issued by the EPS healing process and the evaluation of ILDS process. If HALDB is processed, this data set is mandatory. If BLKSIZE is coded, it must be a multiple of 133.

SNAPPIT DD

This required output data set contains the block maps and block dumps that the HDAM/HIDAM, PHDAM/PHIDAM, and BLOCKMAP processed produce. It also contains a dump of some internal control blocks used in the HISAM, INDEX, HDAM/HIDAM, PHDAM/PHIDAM, and PSINDEX processes. If BLKSIZE is coded on the DD statement, it must be a multiple of 133.

SUMMARY DD

This required output data set contains the HD Pointer Checker Summary report produced by the HD Pointer Checker processor. If BLKSIZE is coded on the DD statement, it must be a multiple of 133.

DBSRCPRT DD

This optional output data set contains the messages and the DBD sources produced by IMS Library Integrity Utilities if DECODEDBD=YES is specified on the REPORT statement. This data set is dynamically allocated if the DD statement in the JCL is omitted and if DECODEDBD=YES is specified on the REPORT statement.

DBMAPPRT DD

This optional output data set contains the messages and the DBD maps produced by IMS Library Integrity Utilities if MAPDBD=YES is specified on the REPORT statement. This data set is dynamically allocated if the DD statement in the JCL is omitted and if MAPDBD=YES is specified on the REPORT statement.

SORTWK*nn* DD, SRTSWK*nn* DD, SRTXWK*nn* DD, SRTEWK*nn* DD (*nn*=01 or greater)

DFSORT uses intermediate storage data sets. If the DD statements in the JCL are omitted, these data sets are dynamically allocated by DFSORT. The value *nn* is numbered by DFSORT. For more information on how to code SORTWK*nn* DD statements, see *DFSORT Application Programming Guide*.

SYSUDUMP DD

This optional output data set defines the output from a system ABEND dump routine. It is used only when a dump is required. Although optional, it is recommended that you include this data set.

DD statements for work data set: This section presents tables listing the work data sets that HD Pointer Checker uses by the TYPE= specification in the PROC statement.

It is strongly recommended that you use 'PROC TYPE=ALL' and not 'PROC TYPE=SCAN and CHECK,' because by using 'PROC TYPE=ALL,' the performance and the size of work data sets will improve and the JCL statements are more simple. Multiple jobs is not recommended, because the specification of work data sets is complicated. If you must do so, however, refer to the examples in “Example 4: Standard database analysis—Multiple jobs” on page 260.

Notes for Table 3 on page 44 through Table 6 on page 46

DDNAME

- For the TYPE=ALL or SCAN process, *nn* in ddnames corresponds to scan group number specified by the SCANGROUP option of the DATABASE statement. The default value of the option is 1. Each scan task uses one work data set. For example, if SCANGROUP=1 and SCANGROUP=2 are specified, *xxxxxx*01 and *xxxxxx*02 data sets are used.
- For the TYPE=CHECK process, *nn* should be specified a serial number starting from 01 for each data set generated by the preceding TYPE=SCAN processes. The *nn* might be different from the number used in the preceding scan processes. For how to specify it, see “Example 4: Standard database analysis—Multiple jobs” on page 260.
- The ddname in brackets is used in HD Pointer Checker Version 1. If the old-version ddname is specified in the JCL, it is used instead of the *xxxxxx*01 data set. If ddnames in both styles, for example MERGIN and MERGIN01, are specified in the JCL, only the *xxxxxx*01 data set is used.

Dynamically allocatable

- If DD statements marked ’YES’ are omitted in the JCL, HD Pointer Checker allocates the temporary data sets dynamically. The temporary data sets are allocated on the disk by using the UNIT=SYSALLDA parameter. The number of requested volumes that is dynamically allocated is calculated based on the input data set size. The minimum number of volumes is two. If dynamic allocation fails, the parameter information is shown in message FABP3988E. Check the required volume counts specified in the message.

However, if the target database is so large, the size of each temporary work data set may not be enough. In such a case, you must specify the size needed on the DD statement in the JCL. See “Estimating the work data set size for HD Pointer Checker” on page 289.

PROC TYPE=ALL

Table 3 shows the work data sets used by HD Pointer Checker when PROC TYPE=ALL is specified.

Table 3. Work data sets used when PROC TYPE=ALL is specified

DDNAME	Use: Input (In), Output (Out), or Work	Format	Required (R) or Optional (O)	Dynamically allocatable?	Description
MERGIN*nn* (MERGIN)	Work	RECFM=VB	O	Yes	Used if no HASH Check function is used.
CHECKREC	Work	RECFM=FB LRECL=40	O	Yes	Used if CHECKREC=NO.
	Out		R	No	Used if CHECKREC=YES.
MERGI2*nn* (MERGIN2)	Work	RECFM=VB	O	Yes	Used if IXKEYCHK=YES and HASH=NO.
SORTE2*nn* (SORTEX2)	Work	RECFM=VB	O	Yes	Used if IXKEYCHK=YES and HASH=NO.
IXKEY	Work	RECFM=VB	O	Yes	Used if IXKEYCHK=YES and HASH=NO.
SORTIL*nn* (SORTIL)	Work	RECFM=VB	O	Yes	Used if EPSCHK=YES.
SORTOL	Work	RECFM=VB	O	Yes	Used if EPSCHK=YES.
JRM	Work	RECFM=FB LRECL=40	R	No	Required for the BLOCKMAP process.

PROC TYPE=SCAN

Table 4 on page 45 shows the work data sets used by HD Pointer Checker when PROC TYPE=SCAN is specified. These data sets are generated by the scan process and become the input for the CHECK process specified with PROC TYPE=CHECK that follows. You must specify them on the JCL DD statement because they will not be allocated dynamically by HD Pointer Checker.

Table 4. Work data sets used when PROC TYPE=SCAN is specified

DDNAME	Use: Input (In), Output (Out), or Work	Format	Required (R) or Optional (O)	Dynamically allocatable?	Description
SORTEX01 (SORTEX)	Out	RECFM=FB LRECL=40	R	No	
MERGIN*nn* (MERGIN)	Out	RECFM=VB	O	No	Required if HASH=NO or FORCE. (See note below.)
MERGI2*nn* (MERGIN2)	Out	RECFM=VB	O	No	Required if IXKEYCHK=YES and HASH=NO. (See note below.)
SORTE2*nn* (SORTEX2)	Out	RECFM=VB	O	No	Required if IXKEYCHK=YES and HASH=NO. (See note below.)
SORTIL*nn* (SORTIL)	Out	RECFM=VB	O	No	Required if EPSCHK=YES. (See note below.)

Note: If the scan of multiple database data sets or ICDSs are run in multiple jobs, specify the same keyword for the option in each SCAN job.

PROC TYPE=CHECK

Table 5 shows the work data sets used by HD Pointer Checker when PROC TYPE=CHECK is specified.

Note: For *nn* in ddnames, specify serial numbers starting from 01. The number might be different from the number assigned by the scan job. For how to specify it, see “Example 4: Standard database analysis—Multiple jobs” on page 260.

Table 5. Work data sets used when PROC TYPE=CHECK is specified

DDNAME	Use: Input (In), Output (Out), or Work	Format	Required (R) or Optional (O)	Dynamically allocatable?	Description
SORTEX*nn* (SORTEX)	In	RECFM=FB LRECL=40	R	No	
MERGIN*nn* (MERGIN)	In	RECFM=VB	O	No	Required if HASH=NO is used for the preceding scan jobs.
MERGI2*nn* (MERGIN2)	In	RECFM=VB	O	No	Required if IXKEYCHK=YES and HASH=NO are used for the preceding scan jobs.

Table 5. Work data sets used when PROC TYPE=CHECK is specified (continued)

DDNAME	Use: Input (In), Output (Out), or Work	Format	Required (R) or Optional (O)	Dynamically allocatable?	Description
SORTE2*nn* (SORTEX2)	In	RECFM=VB	O	No	Required if IXKEYCHK=YES and HASH=NO are used for the preceding scan jobs.
IXKEY	Work	RECFM=VB	O	Yes	Used if IXKEYCHK=YES and HASH=NO are used for the preceding scan jobs.
SORTIL*nn* (SORTIL)	In	RECFM=VB	O	No	Required if EPSCHK=YES is used for the preceding scan jobs.
SORTOL	Work	RECFM=VB	O	Yes	Used if EPSCHK=YES is used for the preceding scan jobs.
JRM	Work	RECFM=FB LRECL=40	R	No	Required for the BLOCKMAP process.
CHECKREC	Work	RECFM=FB LRECL=40	O	Yes	Used if CHECKREC=NO.
	Out		R	No	Required if CHECKREC=YES.

PROC TYPE=BLKMAP

Table 6 shows the work data sets used by HD Pointer Checker when PROC TYPE=BLKMAP is specified.

Table 6. Work data sets used when PROC TYPE=BLKMAP is specified

DDNAME	Input or Output	Format	Required (R) or Optional (O)	Dynamically allocatable?	Description
BLKMAPIN	In	LRECL=80	R	No	
CHECKREC	In	RECFM=FB LRECL=40	R	No	Created in the preceding CHECK process.

The following describes the DD statements listed in Table 3 on page 44 through Table 6.

MERGIN*nn* DD

This is a data set generated by the scan process of HISAM, HDAM, HIDAM, PHDAM, and PHIDAM (excluding primary index). It becomes the input for the CHECK process. It is not used for the following:

- HASH check
- Scan tasks solely of primary or secondary indexes

CHECKREC DD

This is a data set generated by the CHECK process to create maps and dumps of the database. When you specify PROC TYPE=ALL or PROC TYPE=CHECK, the specification of the DD statement differs depending on what you specify for the keyword of CHECKREC=.

CHECKREC=YES

This data set is required for the input of the process PROC TYPE=BLKMAP. You must specify this DD on the JCL statement, because HD Pointer Checker will not allocate it dynamically.

CHECKREC=NO

Because the size of the data set is small, HD Pointer Checker will allocate it dynamically. You do not need to specify this DD on the JCL statement.

If you specify PROC TYPE=BLKMAP, the CHECKREC data set generated for CHECKREC=YES in the preceding process PROC TYPE=ALL or TYPE=CHECK is required as input.

MERGI2*nn* DD

This is an optional work data set that is generated by the scan process of the index target segment in HDAM, HIDAM, PHDAM, or PHIDAM when PROC TYPE=ALL or TYPE=SCAN is specified with IXKEYCHK=YES and HASH=NO. It becomes an input for the CHECK process.

SORTE2*nn* DD

This is an optional work data set that is generated by the scan process of the index source segment in HISAM, HDAM, HIDAM, PHDAM, PHIDAM, and the index database when PROC TYPE=ALL or TYPE=SCAN is specified with IXKEYCHK=YES and HASH=NO. It becomes an input for the CHECK process.

IXKEY DD

This is a work data set used in the CHECK process when PROC TYPE=ALL or TYPE=SCAN is specified with IXKEYCHK=YES and HASH=NO.

SORTIL*nn* DD

This is an optional work data set that is generated by the scan process when PROC TYPE=ALL or TYPE=SCAN is specified with EPSCHK=YES (this is the default for HALDBs) and the target PHDAM/PHIDAM or PSINDEX has logical relationships. It becomes an input for the CHECK process.

SORTOL DD

This is an optional work data set that is used by the CHECK process when PROC TYPE=ALL or TYPE=SCAN is specified with EPSCHK=YES (this is the default for HALDBs) and the target PHDAM/PHIDAM or PSINDEX has logical relationships.

SORTEX01 DD

This is a data set generated by the scan process when PROC TYPE=SCAN is specified. One data set is created in each TYPE=SCAN step. It becomes an input for the CHECK process specified with PROC TYPE=CHECK.

SORTEX*nn* DD

This is the input data set that is used by the CHECK process when PROC TYPE=CHECK is specified. The SORTEX01 data set is generated in advance by each scan job. Specify that data set as SORTEX*nn* DD (where *nn* is a serial number, starting from 01).

JRM DD
This is a required work data set that contains control statements generated by the CHECK process and used by the BLOCKMAP process. LRECL must be 40, and BLKSIZE must be a multiple of 40.

BLKMAPIN DD
This is an input data set containing the user-specified control statements that are to be processed by the BLOCKMAP processor. It is required when TYPE=BLKMAP is specified.

DD statements to call Space Monitor: If the SPMNIN and SPMNSPDT DD statements are specified for TYPE=ALL or TYPE=SCAN of FABPMAIN JCL, HD Pointer Checker calls Space Monitor internally.

The SPMNMBR DD statement cannot be specified for the FABPMAIN JCL. Only SPMNIN can be used.

For details of each DD statement, see "Job control language" on page 495.

JCL procedures

To run HD Pointer Checker, use the IBM-supplied cataloged procedures shown in and Figure 2, through Figure 8 on page 65 or prepare a similar procedure of your own. The use of the IBM-supplied, cataloged procedures avoids the need for maintaining a lot of large sets of HD Pointer Checker JCL.

Note

Maintaining many large sets of JCL is an error-prone activity which can be avoided by using cataloged JCL procedures.

Table 7 summarizes the use of cataloged procedures. Three sets of procedure names are shown: those of this HD Pointer Checker, those of DBT Version 2 Release 1, and those of DBT Version 2 Release 2/Release 3.

Table 7. Selection of HD Pointer Checker JCL procedure

Job function	Name	DBT V2R1 name	DBT V2R2 or V2R3 name
Checking pointers (with PSB and work data sets defined in the procedure)	FABPP	HDPCP	FABPP
Checking pointers (with DBD and work data sets not defined in the procedure. The work data sets are allocated by HD Pointer Checker dynamically)	FABPPD	None	None
Checking pointers (with PSB in the DBB region by using ACBLIB. The work data sets are allocated dynamically)	FABPPA	None	None
Checking pointers and root-segment locations (with PSB)	FABPPTA	HDPCPTA	FABPPTA
Checking pointers in multiple job steps (with PSB)	FABPPM	HDPCPM	FABPPM

Table 7. Selection of HD Pointer Checker JCL procedure (continued)

Job function	Name	DBT V2R1 name	DBT V2R2 or V2R3 name
Checking pointers in multiple job steps (with DBD)	FABPPMD	None	None
Checking pointers in multiple job steps and root-segment locations (with PSB)	FABPPTAM	HDPCPTAM	FABPPTAM
Finding disk addresses	No procedure needed	No procedure needed	No procedure needed

Note: The names of IBM-supplied cataloged procedures and sample control statement member names for HD Pointer Checker have been changed in DBT Version 2 Release 2. Refer to Table 7 on page 48 and Table 8 for the details. Also refer to “Migrating from DBT Version 2 Releases 1 or Version 2 Release 2” on page 723 and “Migrating from DBT Version 2 Release 3” on page 724 for the considerations on migration.

The JCL procedures FABPV1, FABPV1D, and FABPV1TA, which were provided with DBT Version 1, are not supported in this version of HD Pointer Checker.

Table 8. Examples of control statements

Name	DBT V2R1 name
FABPVSAM	FABVSAMP
FABPSORT	CTLSORT
FABPSRT2	None
(None)	CTLMERG1

Figure 5 on page 56 is a typical example of a procedure running HD Pointer Checker and HD Tuning Aid in a same job. To run only the HD Pointer Checker, use a procedure similar to that in Figure 2 on page 50, Figure 3 on page 52, and Figure 4 on page 54. These procedures are examples of general-purpose procedures that can be used in verifying and analyzing IMS full-function databases.

The examples in Chapter 6, “JCL examples for HD Pointer Checker,” on page 253 assume that the IBM-supplied, cataloged procedures are used.

Procedure FABPP

This procedure, shown in Figure 2 on page 50, runs only the HD Pointer Checker. FABPP includes control statements for allocating the work data sets. The user supplies the PSB used in this procedure.

```
//********************************************************************  00010000
//*   Licensed Materials - Property of IBM                           *  00020000
//*                                                                  *  00030000
//*   5655-K53                                                       *  00040000
//*                                                                  *  00050000
//*   (c) Copyright IBM Corporation 2000, 2006. All rights reserved. *  00060000
//*                                                                  *  00070000
//*   US Government Users Restricted Rights - Use,                   *  00080000
//*   duplication or disclosure restricted by GSA ADP                *  00090000
//*   Schedule Contract with IBM Corp.                               *  00100000
//*                                                                  *  00110000
//********************************************************************  00120000
//        PROC PSB=,                        PSBNAME                     00130000
//        DBRC=N,                           DBRC=Y IF HALDB PROCESS     00140000
//        KEYS='NULLFILE',                  DS NAME IF GENERATE KEYSIN  00150000
//        KEYSVOL=,                         VOL NAME OF KEYSIN          00160000
//        KEYSU='SYSDA',                    UNIT  FOR  KEYSIN DATA SET  00170000
//        KEYSCYL='1,1',                    SPACE FOR  KEYSIN DATA SET  00180000
//        U=SYSDA,                          UNIT  FOR  WORK DATA SETS   00190000
//        CYL='1,1',                        SPACE FOR  WORK DATA SETS   00200000
//        SORT2=,            SORT2='DUMMY,' IF NO SORTEX2 DATA SET      00210000
//        MERG2=,            MERG2='DUMMY,' IF NO MERGIN2 DATA SET      00220000
//        SORTIL='&&SORTIL',                DS NAME IF HALDB PROCESS    00230000
//        SORTOL='&&SORTOL',                DS NAME IF HALDB PROCESS    00240000
//        PRTBLK=0,                         BLKSIZE OF PRINT DATA SETS  00250000
//        PRTBLK2=0,                        BLKSIZE OF FSESTAT DATA SET 00260000
//        SORTBLK=0,                        BLKSIZE OF SORT RECORS      00270000
//        IXKBLK=0,                         BLKSIZE OF IXKEY RECORS     00280000
//        HISTORY='NULLFILE',               HISTORY DATA SET            00290000
//        USERLIB='USER.LOADLIB',           USER RANDOMIZER             00300000
//        DBDLIB='IMSVS.DBDLIB',            <<--------<                 00310000
//        PSBLIB='IMSVS.PSBLIB',            <<--------<                 00320000
//        RESLIB='IMSVS.RESLIB',            <<--------<                 00330000
//        DBTLIB='HPS.SHPSLMD0',            <<--------<                 00340000
//        DBTSRC='HPS.SHPSSAMP(FABPVSAM)'   <<--------<                 00350000
//*-------------------------------------------------------------------* 00360000
//HDPCPRO EXEC PGM=DFSRRC00,                                            00370000
//             PARM='DLI,FABPMAIN,&PSB,,,,,,,,,,,&DBRC,N'               00380000
//STEPLIB   DD DSN=&DBTLIB,DISP=SHR                                     00390000
//          DD DSN=&RESLIB,DISP=SHR                                     00400000
//          DD DSN=&USERLIB,DISP=SHR                                    00410000
//*-------------------------------------------------------------------* 00420000
//* FOR IMS DATA SETS                                                   00430000
//*-------------------------------------------------------------------* 00440000
//IMS       DD DSN=&PSBLIB,DISP=SHR                                     00450000
//          DD DSN=&DBDLIB,DISP=SHR                                     00460000
//IMS2      DD DSN=&RESLIB,DISP=SHR                                     00470000
//          DD DSN=&USERLIB,DISP=SHR                                    00480000
//DFSRESLB  DD DSN=&RESLIB,DISP=SHR                                     00490000
//DFSVSAMP  DD DSN=&DBTSRC,DISP=SHR                                     00500000
//IEFRDER   DD DUMMY                                                    00510000
```

Figure 2. HD Pointer Checker JCL procedure using the HD Pointer Checker processor (FABPP) (Part 1 of 2)

```
//*-------------------------------------------------------------------* 00530000
//* REPORTS                                                             00540000
//*-------------------------------------------------------------------* 00550000
//PRIMAPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00560000
//STATIPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00570000
//VALIDPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00580000
//EVALUPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00582000
//EVALIPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00585000
//SNAPPIT   DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00610000
//SUMMARY   DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00620000
//SYSUDUMP  DD SYSOUT=A                                                 00640000
//*-------------------------------------------------------------------* 00650000
//* HISTORICAL ANALYSIS DATA SETS                                       00660000
//*-------------------------------------------------------------------* 00670000
//HISTORY   DD DSN=&HISTORY,DISP=SHR                                    00680000
//*-------------------------------------------------------------------* 00690000
//* SORT RECORDS                                                        00700000
//*-------------------------------------------------------------------* 00710000
//MERGIN01  DD DSN=&&MERGIN,DISP=(NEW,PASS,DELETE),                     00810000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                              00820000
//             DCB=(BLKSIZE=&SORTBLK)                                   00840000
//*-------------------------------------------------------------------* 00870000
//* SPECIFY SORTEX2 AND MERGIN2 FOR SCAN WITH IXKEYCHK = YES            00880000
//*-------------------------------------------------------------------* 00890000
//SORTE201  DD &SORT2.DSN=&&SORTEX2,DISP=(NEW,PASS,DELETE),             00900000
//             UNIT=&U,SPACE=(CYL,(&CYL))                               00910000
//MERGI201  DD &MERG2.DSN=&&MERGIN2,DISP=(NEW,PASS,DELETE),             00930000
//             UNIT=&U,SPACE=(CYL,(&CYL))                               00940000
//*-------------------------------------------------------------------* 00960000
//* FOR SCAN WITH HDAM/HIDAM PROCESS                                    00970000
//*-------------------------------------------------------------------* 00980000
//KEYSIN    DD DSN=&KEYS,DISP=(NEW,CATLG,DELETE),                       00990000
//             UNIT=&KEYSU,SPACE=(CYL,(&KEYSCYL)),VOL=SER=&KEYSVOL,     01000000
//             DCB=(RECFM=VB)                                           01010000
//*-------------------------------------------------------------------* 01040000
//* FOR EPS HEALING PROCESS                                           * 01050000
//*-------------------------------------------------------------------* 01060000
//SORTIL01  DD DSN=&SORTIL,DISP=(NEW,PASS,DELETE),                      01070000
//             UNIT=&U,SPACE=(CYL,(&CYL))                               01080000
//*-------------------------------------------------------------------* 01220000
//* FOR CHECK PROCESS                                                   01230000
//*-------------------------------------------------------------------* 01240000
//SORTOL    DD DSN=&SORTOL,DISP=(NEW,DELETE,DELETE),                    01250000
//             UNIT=&U,SPACE=(CYL,(&CYL))                               01270000
//IXKEY     DD DSN=&&IXKEY,DISP=(NEW,DELETE,DELETE),                    01290000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                              01310000
//             DCB=(BLKSIZE=&IXKBLK)                                    01315000
//FSESTAT   DD DSN=&&FSESTAT,DISP=(NEW,DELETE,DELETE),                  01320000
//             UNIT=&U,SPACE=(CYL,(2,2)),DCB=BLKSIZE=&PRTBLK2           01325000
//*-------------------------------------------------------------------* 01330000
//* FOR CHECK AND BLKMAP PROCESS                                        01340000
//*-------------------------------------------------------------------* 01350000
//JRM       DD DSN=&&JRM,DISP=(NEW,DELETE,DELETE),                      01360000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                              01370000
//             DCB=(BLKSIZE=&SORTBLK)                                   01380000
//*-------------------------------------------------------------------* 01390000
```

Figure 2. HD Pointer Checker JCL procedure using the HD Pointer Checker processor (FABPP) (Part 2 of 2)

Procedure FABPPD

The FABPPD procedure, shown in Figure 3, corresponds to FABPP. Both procedures check pointers; FABPPD, though, provides for dynamic PSB generation and allocation of dynamic sort work data set.

Note: The DBD parameter of the FABPPD procedure is not required.

```
//********************************************************************  00010000
//*   Licensed Materials - Property of IBM                           *  00020000
//*                                                                  *  00030000
//*   5655-K53                                                       *  00040000
//*                                                                  *  00050000
//*   (c) Copyright IBM Corporation 2000, 2006. All rights reserved. *  00060000
//*                                                                  *  00070000
//*   US Government Users Restricted Rights - Use,                   *  00080000
//*   duplication or disclosure restricted by GSA ADP                *  00090000
//*   Schedule Contract with IBM Corp.                               *  00100000
//*                                                                  *  00110000
//********************************************************************  00120000
//        PROC DBD=,                        DBDNAME                     00130000
//        DBRC=N,                           DBRC=Y IF HALDB PROCESS     00140000
//        KEYS='NULLFILE',                  DS NAME IF GENERATE KEYSIN  00150000
//        KEYSVOL=,                         VOL NAME OF KEYSIN          00160000
//        KEYSU='SYSDA',                    UNIT  FOR  KEYSIN DATA SET  00170000
//        KEYSCYL='1,1',                    SPACE FOR  KEYSIN DATA SET  00180000
//        U=SYSDA,                          UNIT  FOR  WORK DATA SETS   00190000
//        CYL='1,1',                        SPACE FOR  WORK DATA SETS   00200000
//        PRTBLK=0,                         BLKSIZE OF PRINT DATA SETS  00210000
//        PRTBLK2=0,                        BLKSIZE OF FSESTAT DATA SET 00220000
//        SORTBLK=0,                        BLKSIZE OF SORT RECORS      00230000
//        HISTORY='NULLFILE',               HISTORY DATA SET            00240000
//        USERLIB='USER.LOADLIB',           USER RANDOMIZER             00250000
//        DBDLIB='IMSVS.DBDLIB',            <<--------<                 00260000
//        RESLIB='IMSVS.RESLIB',            <<--------<                 00270000
//        DBTLIB='HPS.SHPSLMD0',            <<--------<                 00280000
//        DBTSRC='HPS.SHPSSAMP(FABPVSAM)'   <<--------<                 00290000
//*-------------------------------------------------------------------* 00300000
//HDPCPRO EXEC PGM=DFSRRC00,                                            00310000
//             PARM='ULU,FABPMAIN,&DBD,,,,,,,,,,,&DBRC,N'               00320000
//STEPLIB   DD DSN=&DBTLIB,DISP=SHR                                     00330000
//          DD DSN=&RESLIB,DISP=SHR                                     00340000
//          DD DSN=&USERLIB,DISP=SHR                                    00350000
//*-------------------------------------------------------------------* 00360000
//* FOR IMS DATA SETS                                                   00370000
//*-------------------------------------------------------------------* 00380000
//IMS       DD DSN=&DBDLIB,DISP=SHR                                     00390000
//IMS2      DD DSN=&RESLIB,DISP=SHR                                     00400000
//          DD DSN=&USERLIB,DISP=SHR                                    00410000
//DFSRESLB  DD DSN=&RESLIB,DISP=SHR                                     00420000
//DFSVSAMP  DD DSN=&DBTSRC,DISP=SHR                                     00430000
//IEFRDER   DD DUMMY                                                    00440000
```

Figure 3. HD Pointer Checker JCL procedure with dynamic allocation (FABPPD) (Part 1 of 2)

```
//*-------------------------------------------------------------------* 00460000
//* REPORTS                                                             00470000
//*-------------------------------------------------------------------* 00480000
//PRIMAPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00490000
//STATIPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00500000
//VALIDPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00510000
//EVALUPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00512000
//EVALIPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00515000
//SNAPPIT   DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00540000
//SUMMARY   DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00550000
//SYSUDUMP  DD SYSOUT=A                                                 00570000
//*-------------------------------------------------------------------* 00580000
//* HISTORICAL ANALYSIS DATA SETS                                       00590000
//*-------------------------------------------------------------------* 00600000
//HISTORY   DD DSN=&HISTORY,DISP=SHR                                    00610000
//*-------------------------------------------------------------------* 00620000
//* FOR SCAN WITH HDAM/HIDAM PROCESS                                    00630000
//*-------------------------------------------------------------------* 00640000
//KEYSIN    DD DSN=&KEYS,DISP=(NEW,CATLG,DELETE),                       00650000
//             UNIT=&KEYSU,SPACE=(CYL,(&KEYSCYL)),VOL=SER=&KEYSVOL,     00660000
//             DCB=(RECFM=VB)                                           00670000
//*-------------------------------------------------------------------* 00675000
//* FOR CHECK PROCESS                                                   00680000
//*-------------------------------------------------------------------* 00685000
//FSESTAT   DD DSN=&&FSESTAT,DISP=(NEW,DELETE,DELETE),                  00690000
//             UNIT=&U,SPACE=(CYL,(2,2)),DCB=BLKSIZE=&PRTBLK2           00695000
//*-------------------------------------------------------------------* 00700000
//* FOR CHECK AND BLKMAP PROCESS                                        00710000
//*-------------------------------------------------------------------* 00720000
//JRM       DD DSN=&&JRM,DISP=(NEW,DELETE,DELETE),                      00730000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                              00740000
//             DCB=(BLKSIZE=&SORTBLK)                                   00750000
//*-------------------------------------------------------------------* 00760000
```

Figure 3. HD Pointer Checker JCL procedure with dynamic allocation (FABPPD) (Part 2 of 2)

Procedure FABPPA

The FABPPA procedure, shown in Figure 4, runs only the HD Pointer Checker in an IMS DBB region. The user specifies a PSB name and a ACBLIB data set, and HD Pointer Checker obtains information about the PSB and all DBDs referred to by the PSB from the ACBLIB. This procedure contains no DD statement for the work data set; HD Pointer Checker allocates them dynamically.

Figure 4 shows the procedure FABPPA provides in the HPS.SHPSSAMP library.

```
//********************************************************************* 00010000
//*   Licensed Materials - Property of IBM                           *  00020000
//*                                                                  *  00030000
//*   5655-K53                                                       *  00040000
//*                                                                  *  00050000
//*   (c) Copyright IBM Corporation 2000, 2006. All rights reserved. *  00060000
//*                                                                  *  00070000
//*   US Government Users Restricted Rights - Use,                   *  00080000
//*   duplication or disclosure restricted by GSA ADP                *  00090000
//*   Schedule Contract with IBM Corp.                               *  00100000
//*                                                                  *  00110000
//********************************************************************* 00120000
//        PROC PSB=,                        PSB NAME                    00130000
//        DBRC=N,                           DBRC=Y IF HALDB PROCESS     00140000
//        KEYS='NULLFILE',                  DS NAME IF GENERATE KEYSIN  00150000
//        KEYSVOL=,                         VOL NAME OF KEYSIN          00160000
//        KEYSU='SYSDA',                    UNIT  FOR  KEYSIN DATA SET  00170000
//        KEYSCYL='1,1',                    SPACE FOR  KEYSIN DATA SET  00180000
//        U=SYSDA,                          UNIT  FOR  WORK DATA SETS   00190000
//        CYL='1,1',                        SPACE FOR  WORK DATA SETS   00200000
//        PRTBLK=0,                         BLKSIZE OF PRINT DATA SETS  00210000
//        PRTBLK2=0,                        BLKSIZE OF FSESTAT DATA SET 00220000
//        SORTBLK=0,                        BLKSIZE OF SORT RECORS      00230000
//        HISTORY='NULLFILE',               HISTORY DATA SET            00240000
//        USERLIB='USER.LOADLIB',           USER RANDOMIZER             00250000
//        ACBLIB='IMSVS.ACBLIB',            <<--------<                 00260000
//        RESLIB='IMSVS.RESLIB',            <<--------<                 00270000
//        DBTLIB='HPS.SHPSLMD0',            <<--------<                 00280000
//        DBTSRC='HPS.SHPSSAMP(FABPVSAM)'   <<--------<                 00290000
//*-------------------------------------------------------------------* 00300000
//HDPCPRO EXEC PGM=DFSRRC00,                                            00310000
//             PARM='DBB,FABPMAIN,&PSB,,,,,,,,,,,&DBRC,N'               00320000
//STEPLIB   DD DSN=&DBTLIB,DISP=SHR                                     00330000
//          DD DSN=&RESLIB,DISP=SHR                                     00340000
//          DD DSN=&USERLIB,DISP=SHR                                    00350000
//*-------------------------------------------------------------------* 00360000
//* FOR IMS DATA SETS                                                   00370000
//*-------------------------------------------------------------------* 00380000
//IMSACB    DD DSN=&ACBLIB,DISP=SHR                                     00390000
//IMS2      DD DSN=&RESLIB,DISP=SHR                                     00400000
//          DD DSN=&USERLIB,DISP=SHR                                    00410000
//DFSRESLB  DD DSN=&RESLIB,DISP=SHR                                     00420000
//DFSVSAMP  DD DSN=&DBTSRC,DISP=SHR                                     00430000
//IEFRDER   DD DUMMY                                                    00440000
```

Figure 4. HD Pointer Checker JCL procedure running with ACBLIB (FABPPA) (Part 1 of 2)

```
//*-------------------------------------------------------------------* 00450000
//* REPORTS                                                             00460000
//*-------------------------------------------------------------------* 00470000
//PRIMAPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00480000
//STATIPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00490000
//VALIDPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00500000
//EVALUPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00510000
//EVALIPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00520000
//SNAPPIT   DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00530000
//SUMMARY   DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                    00540000
//SYSUDUMP  DD SYSOUT=A                                                 00550000
//*-------------------------------------------------------------------* 00560000
//* HISTORICAL ANALYSIS DATA SETS                                       00570000
//*-------------------------------------------------------------------* 00580000
//HISTORY   DD DSN=&HISTORY,DISP=SHR                                    00590000
//*-------------------------------------------------------------------* 00600000
//* FOR SCAN WITH HDAM/HIDAM PROCESS                                    00610000
//*-------------------------------------------------------------------* 00620000
//KEYSIN    DD DSN=&KEYS,DISP=(NEW,CATLG,DELETE),                       00630000
//             UNIT=&KEYSU,SPACE=(CYL,(&KEYSCYL)),VOL=SER=&KEYSVOL,     00640000
//             DCB=(RECFM=VB)                                           00650000
//*-------------------------------------------------------------------* 00660000
//* FOR CHECK PROCESS                                                   00670000
//*-------------------------------------------------------------------* 00680000
//FSESTAT   DD DSN=&&FSESTAT,DISP=(NEW,DELETE,DELETE),                  00690000
//             UNIT=&U,SPACE=(CYL,(2,2)),DCB=BLKSIZE=&PRTBLK2           00700000
//*-------------------------------------------------------------------* 00710000
//* FOR CHECK AND BLKMAP PROCESS                                        00720000
//*-------------------------------------------------------------------* 00730000
//JRM       DD DSN=&&JRM,DISP=(NEW,DELETE,DELETE),                      00740000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                              00750000
//             DCB=(BLKSIZE=&SORTBLK)                                   00760000
//*-------------------------------------------------------------------* 00770000
```

Figure 4. HD Pointer Checker JCL procedure running with ACBLIB (FABPPA) (Part 2 of 2)

Procedure FABPPTA

This procedure, shown in Figure 5, runs HD Pointer Checker and HD Tuning Aid sequentially. The procedure includes control statements for the allocation of the work data sets. The user supplies the PSB used in this procedure.

```
//********************************************************************  00010000
//*   Licensed Materials - Property of IBM                           *  00020000
//*                                                                  *  00030000
//*   5655-K53                                                       *  00040000
//*                                                                  *  00050000
//*   (c) Copyright IBM Corporation 2000, 2006. All rights reserved. *  00060000
//*                                                                  *  00070000
//*   US Government Users Restricted Rights - Use,                   *  00080000
//*   duplication or disclosure restricted by GSA ADP                *  00090000
//*   Schedule Contract with IBM Corp.                               *  00100000
//*                                                                  *  00110000
//********************************************************************  00120000
//        PROC PSB=,                        PSB NAME                    00130000
//        DBRC=Y,                           DBRC=Y IF HALDB PROCESS     00140000
//        HDTA02P=,                         PARAM FOR STEP HDTA02       00150000
//        U=SYSDA,                          UNIT  FOR  WORK DATA SETS   00160000
//        CYL='1,1',                        SPACE FOR  WORK DATA SETS   00170000
//        SORT2=,            SORT2='DUMMY,' IF NO SORTEX2 DATA SET      00180000
//        MERG2=,            MERG2='DUMMY,' IF NO MERGIN2 DATA SET      00190000
//        SORTIL='&&SORTIL',                DS NAME IF HALDB PROCESS    00200000
//        SORTOL='&&SORTOL',                DS NAME IF HALDB PROCESS    00220000
//        PRTBLK=0,                         BLKSIZE OF PRINT DATA SETS  00224000
//        PRTBLK2=0,                        BLKSIZE OF FSESTAT DATA SET 00228000
//        SORTBLK=0,                        BLKSIZE OF SORT RECORS      00230000
//        IXKBLK=0,                         BLKSIZE OF IXKEY RECORS     00236000
//        HISTORY='NULLFILE',               HISTORY DATA SET            00240000
//        USERLIB='USER.LOADLIB',           USER RANDOMIZER             00270000
//        DBDLIB='IMSVS.DBDLIB',            <<--------<                 00280000
//        PSBLIB='IMSVS.PSBLIB',            <<--------<                 00290000
//        RESLIB='IMSVS.RESLIB',            <<--------<                 00300000
//        DBTLIB='HPS.SHPSLMD0',            <<--------<                 00310000
//        DBTSRC='HPS.SHPSSAMP'             <<--------<                 00320000
//*-------------------------------------------------------------------* 00330000
//HDPCPRO EXEC PGM=DFSRRC00,                                            00340000
//             PARM='DLI,FABPMAIN,&PSB,,,,,,,,,,,&DBRC,N'               00350000
//STEPLIB   DD DSN=&DBTLIB,DISP=SHR                                     00360000
//          DD DSN=&RESLIB,DISP=SHR                                     00370000
//          DD DSN=&USERLIB,DISP=SHR                                    00380000
//*-------------------------------------------------------------------* 00390000
//* FOR IMS DATA SETS                                                   00400000
//*-------------------------------------------------------------------* 00410000
//IMS       DD DSN=&PSBLIB,DISP=SHR                                     00420000
//          DD DSN=&DBDLIB,DISP=SHR                                     00430000
//IMS2      DD DSN=&RESLIB,DISP=SHR                                     00440000
//          DD DSN=&USERLIB,DISP=SHR                                    00450000
//DFSRESLB  DD DSN=&RESLIB,DISP=SHR                                     00460000
//DFSVSAMP  DD DSN=&DBTSRC(FABPVSAM),DISP=SHR                           00470000
//IEFRDER   DD DUMMY                                                    00480000
```

Figure 5. HD Pointer Checker/HD Tuning Aid JCL procedure using the HD Pointer Checker processor (FABPPTA) (Part 1 of 3)

```
//*-------------------------------------------------------------------* 00500000
//* REPORTS                                                             00510000
//*-------------------------------------------------------------------* 00520000
//PRIMAPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00530000
//STATIPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00540000
//VALIDPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00550000
//EVALUPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00560000
//EVALIPRT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00570000
//SNAPPIT   DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00580000
//SUMMARY   DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00590000
//SYSUDUMP  DD SYSOUT=A                                                 00610000
//*-------------------------------------------------------------------* 00620000
//* HISTORICAL ANALYSIS DATA SETS                                       00630000
//*-------------------------------------------------------------------* 00640000
//HISTORY   DD DSN=&HISTORY,DISP=SHR                                    00650000
//*-------------------------------------------------------------------* 00660000
//* SORT RECORDS                                                        00670000
//*-------------------------------------------------------------------* 00680000
//MERGIN01  DD DSN=&&MERGIN,DISP=(NEW,PASS,DELETE),                     00780000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                              00790000
//             DCB=(BLKSIZE=&SORTBLK)                                   00800000
//*-------------------------------------------------------------------* 00840000
//* SPECIFY SORTEX2 AND MERGIN2 FOR SCAN WITH IXKEYCHK = YES            00850000
//*-------------------------------------------------------------------* 00860000
//SORTE201  DD &SORT2.DSN=&&SORTEX2,DISP=(NEW,PASS,DELETE),             00870000
//             UNIT=&U,SPACE=(CYL,(&CYL))                               00880000
//MERGI201  DD &MERG2.DSN=&&MERGIN2,DISP=(NEW,PASS,DELETE),             00900000
//             UNIT=&U,SPACE=(CYL,(&CYL))                               00910000
//*-------------------------------------------------------------------* 00930000
//* FOR SCAN WITH HDAM/HIDAM PROCESS                                    00940000
//*-------------------------------------------------------------------* 00950000
//KEYSIN    DD DSN=&&KEYSIN,DISP=(NEW,PASS,DELETE),                     00960000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                              00970000
//             DCB=(RECFM=VB)                                           00980000
//*-------------------------------------------------------------------* 01010000
//* FOR EPS HEALING PROCESS                                           * 01020000
//*-------------------------------------------------------------------* 01030000
//SORTIL01  DD DSN=&SORTIL,DISP=(NEW,PASS,DELETE),                      01040000
//             UNIT=&U,SPACE=(CYL,(&CYL))                               01050000
//*-------------------------------------------------------------------* 01190000
//* FOR CHECK PROCESS                                                   01200000
//*-------------------------------------------------------------------* 01210000
//SORTOL    DD DSN=&SORTOL,DISP=(NEW,DELETE,DELETE),                    01220000
//             UNIT=&U,SPACE=(CYL,(&CYL))                               01240000
//IXKEY     DD DSN=&&IXKEY,DISP=(NEW,DELETE,DELETE),                    01260000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                              01280000
//             DCB=(BLKSIZE=&IXKBLK)                                    01285000
//FSESTAT   DD DSN=&&FSESTAT,DISP=(NEW,DELETE,DELETE),                  01290000
//             UNIT=&U,SPACE=(CYL,(2,2)),DCB=BLKSIZE=&PRTBLK2           01295000
```

Figure 5. HD Pointer Checker/HD Tuning Aid JCL procedure using the HD Pointer Checker processor (FABPPTA) (Part 2 of 3)

```
//*-------------------------------------------------------------------* 01300000
//* FOR CHECK AND BLKMAP PROCESS                                        01310000
//*-------------------------------------------------------------------* 01320000
//JRM       DD DSN=&&JRM,DISP=(NEW,DELETE,DELETE),                      01330000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                              01340000
//             DCB=(BLKSIZE=&SORTBLK)                                   01350000
//*-------------------------------------------------------------------* 01360000
//HDTA01  EXEC PGM=DFSRRC00,                                            01370000
//             PARM='DLI,FABTROOT,&PSB,,,,,,,,,,,&DBRC,N',              01380000
//             REGION=1000K,TIME=(,30)                                  01390000
//STEPLIB   DD DSN=&DBTLIB,DISP=SHR                                     01400000
//          DD DSN=&RESLIB,DISP=SHR                                     01410000
//IMS       DD DSN=&DBDLIB,DISP=SHR                                     01420000
//          DD DSN=&PSBLIB,DISP=SHR                                     01430000
//IMS2      DD DSN=&RESLIB,DISP=SHR                                     01440000
//          DD DSN=&USERLIB,DISP=SHR                                    01450000
//IEFRDER   DD DUMMY,DCB=BLKSIZE=1408                                   01460000
//DFSRESLB  DD DSN=&RESLIB,DISP=SHR                                     01470000
//DFSVSAMP  DD DSN=&DBTSRC(FABPVSAM),DISP=SHR                           01480000
//RAPSIN    DD DSN=&&RAPSIN,UNIT=&U,                                    01490000
//             DISP=(NEW,PASS,DELETE),                                  01500000
//             SPACE=(CYL,(&CYL)),                                      01510000
//             DCB=(LRECL=42,RECFM=FB)                                  01520000
//PR8       DD SYSOUT=A                                                 01530000
//PR10      DD SYSOUT=A                                                 01540000
//KEYSIN    DD DSN=&&KEYSIN,DISP=(OLD,DELETE,DELETE)                    01550000
//SYSUDUMP  DD SYSOUT=A                                                 01560000
//*-------------------------------------------------------------------* 01570000
//HDTA02  EXEC PGM=SORT,PARM='&HDTA02P',COND=((2,LT,HDPCPRO),           01580000
//             (0,LT,HDTA01))                                           01590000
//SORTIN    DD DSN=&&RAPSIN,DISP=(OLD,DELETE,DELETE)                    01600000
//SORTOUT   DD DSN=&&KEYSOUT,UNIT=&U,                                   01610000
//             DISP=(NEW,PASS,DELETE),                                  01620000
//             SPACE=(CYL,(&CYL)),                                      01630000
//             DCB=(LRECL=42,RECFM=FB)                                  01640000
//SORTWK01  DD UNIT=&U,SPACE=(CYL,(&CYL))                               01650000
//SORTWK02  DD UNIT=&U,SPACE=(CYL,(&CYL))                               01660000
//SORTWK03  DD UNIT=&U,SPACE=(CYL,(&CYL))                               01670000
//SORTWK04  DD UNIT=&U,SPACE=(CYL,(&CYL))                               01680000
//SORTWK05  DD UNIT=&U,SPACE=(CYL,(&CYL))                               01690000
//SORTWK06  DD UNIT=&U,SPACE=(CYL,(&CYL))                               01700000
//SYSOUT    DD SYSOUT=A                                                 01710000
//SYSIN     DD DSN=&DBTSRC(FABPSORT),DISP=SHR                           01720000
//*-------------------------------------------------------------------* 01730000
//HDTA03  EXEC PGM=FABTRAPS,COND=((2,LT,HDPCPRO),(0,LT,HDTA01),         01740000
//             (0,LT,HDTA02))                                           01750000
//STEPLIB   DD DSN=&DBTLIB,DISP=SHR                                     01760000
//PR9       DD SYSOUT=A                                                 01770000
//PR9X      DD SYSOUT=A                                                 01780000
//KEYSOUT   DD DSN=&&KEYSOUT,DISP=(OLD,DELETE,DELETE)                   01790000
//SYSUDUMP  DD SYSOUT=A                                                 01800000
//*-------------------------------------------------------------------* 01810000
```

Figure 5. HD Pointer Checker/HD Tuning Aid JCL procedure using the HD Pointer Checker processor (FABPPTA) (Part 3 of 3)

Procedure FABPPM

This procedure, shown in Figure 6, runs only HD Pointer Checker. FABPPM includes control statements for allocating the work data sets. The user supplies the PSB used in this procedure. This procedure checks pointers in multiple job steps. EPSCHK=YES cannot be specified with this procedure.

```
//********************************************************************  00010000
//*   Licensed Materials - Property of IBM                           *  00020000
//*                                                                  *  00030000
//*   5655-K53                                                       *  00040000
//*                                                                  *  00050000
//*   (c) Copyright IBM Corporation 2000, 2006. All rights reserved. *  00060000
//*                                                                  *  00070000
//*   US Government Users Restricted Rights - Use,                   *  00080000
//*   duplication or disclosure restricted by GSA ADP                *  00090000
//*   Schedule Contract with IBM Corp.                               *  00100000
//*                                                                  *  00110000
//********************************************************************  00120000
//        PROC PSB=,                      PSBNAME                       00130000
//        DBRC=N,                         DBRC=Y IF HALDB PROCESS       00140000
//        KEYS='NULLFILE',                DS NAME IF GENERATE KEYSIN    00150000
//        KEYSVOL=,                       VOL NAME OF KEYSIN            00160000
//        KEYSU='SYSDA',                  UNIT FOR KEYSIN DATA SET      00170000
//        KEYSCYL='1,1',                  SPACE FOR KEYSIN DATA SET     00180000
//        U=SYSDA,                        UNIT FOR WORK DATA SETS       00190000
//        CYL='1,1',                      SPACE FOR WORK DATA SETS      00200000
//        SORT2=,            SORT2='DUMMY,' IF NO SORTEX2 DATA SET      00208000
//        MERG2=,            MERG2='DUMMY,' IF NO MERGIN2 DATA SET      00210000
//        SORTIL='&&SORTIL',                 DS NAME IF HALDB PROCESS   00220000
//        SORTOL='&&SORTOL',                 DS NAME IF HALDB PROCESS   00230000
//        PRTBLK=0,                       BLKSIZE OF PRINT DATA SETS    00232000
//        PRTBLK2=0,                      BLKSIZE OF FSESTAT DATA SET   00235000
//        SORTBLK=0,                      BLKSIZE OF SORT RECORDS       00237000
//        HISTORY='NULLFILE',             HISTORY DATA SET              00240000
//        USERLIB='USER.LOADLIB',         USER RANDOMIZER ,             00250000
//*                                       SEGMENT COMPACTION EXIT AND   00260000
//*                                       INDEX MAINTENANCE EXIT        00265000
//        DBDLIB='IMSVS.DBDLIB',          DBD LIBRARY                   00270000
//        PSBLIB='IMSVS.PSBLIB',          PSB LIBRARY                   00280000
//        RESLIB='IMSVS.RESLIB',                                        00290000
//        DBTLIB='HPS.SHPSLMD0',          HPS LIBRARY                   00300000
//        DBTSRC='HPS.SHPSSAMP'           BUFF PARM DATA SET            00310000
//********************************************************************* 00320000
//* DB SCAN STEP                                                        00330000
//********************************************************************* 00340000
//HDPCSCAN EXEC PGM=DFSRRC00,                                           00350000
//             PARM='DLI,FABPMAIN,&PSB,,,,,,,,,,,&DBRC,N'               00360000
//STEPLIB  DD DSN=&DBTLIB,DISP=SHR                                      00370000
//         DD DSN=&RESLIB,DISP=SHR                                      00380000
//         DD DSN=&USERLIB,DISP=SHR                                     00390000
```

Figure 6. HD Pointer Checker JCL procedure using the HD Pointer Checker processor (FABPPM) (Part 1 of 3)

```
//*--------------------------------------------------------------------* 00400000
//* FOR IMS DATA SETS                                                    00410000
//*--------------------------------------------------------------------* 00420000
//IMS      DD DSN=&PSBLIB,DISP=SHR                                       00430000
//         DD DSN=&DBDLIB,DISP=SHR                                       00440000
//IMS2     DD DSN=&RESLIB,DISP=SHR                                       00450000
//         DD DSN=&USERLIB,DISP=SHR                                      00460000
//DFSRESLB DD DSN=&RESLIB,DISP=SHR                                       00470000
//DFSVSAMP DD DSN=&DBTSRC(FABPVSAM),DISP=SHR                             00480000
//IEFRDER  DD DUMMY                                                      00490000
//*--------------------------------------------------------------------* 00510000
//* REPORTS                                                              00520000
//*--------------------------------------------------------------------* 00530000
//PRIMAPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                      00540000
//STATIPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                      00550000
//VALIDPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                      00560000
//EVALUPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                      00570000
//EVALIPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                      00580000
//SNAPPIT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                      00590000
//SUMMARY  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                      00600000
//SYSUDUMP DD SYSOUT=A                                                   00610000
//*--------------------------------------------------------------------* 00620000
//* HISTORY ANALYSIS DATA SET                                            00630000
//*--------------------------------------------------------------------* 00640000
//HISTORY  DD DSN=&HISTORY,DISP=SHR                                      00650000
//*--------------------------------------------------------------------* 00660000
//* SORT RECORDS                                                         00670000
//*--------------------------------------------------------------------* 00680000
//SORTEX01 DD DSN=&&SORTEX,DISP=(NEW,PASS,DELETE),                       00720000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                               00730000
//             DCB=(BLKSIZE=&SORTBLK)                                    00750000
//MERGIN01 DD DSN=&&MERGIN,DISP=(NEW,PASS,DELETE),                       00780000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                      @PN04801 00790000
//             DCB=(BLKSIZE=&SORTBLK)                                    00790500
//*--------------------------------------------------------------------* 00791000
//* SPECIFY SORTEX2 AND MERGIN2 FOR SCAN WITH IXKEYCHK = YES             00793000
//*--------------------------------------------------------------------* 00794000
//SORTE201 DD &SORT2.DSN=&&SORTEX2,DISP=(NEW,PASS,DELETE),               00796000
//             UNIT=&U,SPACE=(CYL,(&CYL))                                00797000
//MERGI201 DD &MERG2.DSN=&&MERGIN2,DISP=(NEW,PASS,DELETE),               00799000
//             UNIT=&U,SPACE=(CYL,(&CYL))                                00800000
//*--------------------------------------------------------------------* 00802000
//* SPECIFY SORTIL FOR SCAN WITH EPSCHK = YES                            00803000
//*--------------------------------------------------------------------* 00805000
//SORTIL01 DD DSN=&SORTIL,DISP=(NEW,PASS,DELETE),                        00806000
//             UNIT=&U,SPACE=(CYL,(&CYL))                                00808000
//*--------------------------------------------------------------------* 00810000
//* FOR SCAN WITH HDAM/HIDAM PROCESS                                     00820000
//*--------------------------------------------------------------------* 00830000
//KEYSIN   DD DSN=&KEYS,DISP=(NEW,CATLG,DELETE),                         00840000
//             UNIT=&KEYSU,SPACE=(CYL,(&KEYSCYL)),VOL=SER=&KEYSVOL,      00850000
//             DCB=(RECFM=VB)                                            00860000
```

Figure 6. HD Pointer Checker JCL procedure using the HD Pointer Checker processor (FABPPM) (Part 2 of 3)

```
//********************************************************************* 01350000
//* CHECK STEP                                                          01360000
//********************************************************************* 01370000
//HDPCCHK  EXEC PGM=DFSRRC00,                                           01380000
//             PARM='DLI,FABPMAIN,&PSB,,,,,,,,,,&DBRC,N',               01390000
//             COND=(4,LT,HDPCSCAN)                                     01400000
//STEPLIB  DD DSN=&DBTLIB,DISP=SHR                                      01410000
//         DD DSN=&RESLIB,DISP=SHR                                      01420000
//         DD DSN=&USERLIB,DISP=SHR                                     01430000
//*-------------------------------------------------------------------* 01440000
//* FOR IMS DATA SETS                                                   01450000
//*-------------------------------------------------------------------* 01460000
//IMS      DD DSN=&PSBLIB,DISP=SHR                                      01470000
//         DD DSN=&DBDLIB,DISP=SHR                                      01480000
//IMS2     DD DSN=&RESLIB,DISP=SHR                                      01490000
//         DD DSN=&USERLIB,DISP=SHR                                     01500000
//DFSRESLB DD DSN=&RESLIB,DISP=SHR                                      01510000
//DFSVSAMP DD DSN=&DBTSRC(FABPVSAM),DISP=SHR                            01520000
//IEFRDER  DD DUMMY                                                     01530000
//*-------------------------------------------------------------------* 01560000
//* REPORTS                                                             01570000
//*-------------------------------------------------------------------* 01580000
//PRIMAPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01590000
//STATIPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01600000
//VALIDPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01610000
//EVALUPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01620000
//EVALIPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01630000
//SNAPPIT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01640000
//SUMMARY  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01650000
//SYSUDUMP DD SYSOUT=A                                                  01660000
//*-------------------------------------------------------------------* 01710000
//* HISTORY ANALYSIS DATA SET                                           01720000
//*-------------------------------------------------------------------* 01730000
//HISTORY  DD DSN=&HISTORY,DISP=SHR                                     01740000
//*-------------------------------------------------------------------* 01740600
//* FOR CHECK PROCESS                                                   01741000
//*-------------------------------------------------------------------* 01741900
//SORTEX01 DD DSN=&&SORTEX,DISP=(OLD,DELETE,DELETE)                     01742000
//MERGIN01 DD DSN=&&MERGIN,DISP=(OLD,DELETE,DELETE)                     01743000
//SORTE201 DD &SORT2.DSN=&&SORTEX2,DISP=(OLD,DELETE,DELETE)             01743900
//MERGI201 DD &MERG2.DSN=&&MERGIN2,DISP=(OLD,DELETE,DELETE)             01744000
//SORTIL01 DD DSN=&SORTIL,DISP=(OLD,DELETE,DELETE)                      01745000
//SORTOL   DD DSN=&SORTOL,DISP=(NEW,DELETE,DELETE),                     01745900
//             UNIT=&U,SPACE=(CYL,(&CYL))                               01746000
//IXKEY    DD DSN=&&IXKEY,DISP=(NEW,DELETE,DELETE),                     01747000
//             UNIT=&U,SPACE=(CYL,(&CYL))                               01747900
//FSESTAT  DD DSN=&&FSESTAT,DISP=(NEW,DELETE,DELETE),                   01748000
//             UNIT=&U,SPACE=(CYL,(2,2)),DCB=BLKSIZE=&PRTBLK2           01749000
//*-------------------------------------------------------------------* 01750000
//* FOR CHECK AND BLKMAP PROCESS                                        01760000
//*-------------------------------------------------------------------* 01770000
//JRM      DD DSN=&&JRM,DISP=(NEW,DELETE,DELETE),                       01790000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                              01800000
//             DCB=(BLKSIZE=&SORTBLK)                                   50000000
```

Figure 6. HD Pointer Checker JCL procedure using the HD Pointer Checker processor (FABPPM) (Part 3 of 3)

Procedure FABPPMD

The FABPPMD procedure, shown in Figure 7, corresponds to FABPPM. Both procedures check pointers in multiple job steps; FABPPMD, though, provides for dynamic PSB generation.

Note: The DBD parameter of the FABPPMD procedure is not required.

```
//********************************************************************  00010000
//*   Licensed Materials - Property of IBM                           *  00020000
//*                                                                  *  00030000
//*   5655-K53                                                       *  00040000
//*                                                                  *  00050000
//*   (c) Copyright IBM Corporation 2000, 2006. All rights reserved. *  00060000
//*                                                                  *  00070000
//*   US Government Users Restricted Rights - Use,                   *  00080000
//*   duplication or disclosure restricted by GSA ADP                *  00090000
//*   Schedule Contract with IBM Corp.                               *  00100000
//*                                                                  *  00110000
//********************************************************************  00120000
//        PROC DBD=,                     DBDNAME                        00130000
//        DBRC=N,                        DBRC=Y IF HALDB PROCESS        00140000
//        KEYS='NULLFILE',               DS NAME IF GENERATE KEYSIN     00150000
//        KEYSVOL=,                      VOL NAME OF KEYSIN             00160000
//        KEYSU='SYSDA',                 UNIT FOR KEYSIN DATA SET       00170000
//        KEYSCYL='1,1',                 SPACE FOR KEYSIN DATA SET      00180000
//        U=SYSDA,                       UNIT FOR WORK DATA SETS        00190000
//        CYL='1,1',                     SPACE FOR WORK DATA SETS       00200000
//        SORT2=,            SORT2='DUMMY,' IF NO SORTEX2 DATA SET      00210000
//        MERG2=,            MERG2='DUMMY,' IF NO MERGIN2 DATA SET      00220000
//        SORTIL='&&SORTIL',                DS NAME IF HALDB PROCESS    00230000
//        PRTBLK=0,                      BLKSIZE OF PRINT DATA SETS     00232000
//        PRTBLK2=0,                     BLKSIZE OF FSESTAT DATA SET    00235000
//        SORTBLK=0,                     BLKSIZE OF SORT RECORDS        00237000
//        HISTORY='NULLFILE',            HISTORY DATA SET               00240000
//        USERLIB='USER.LOADLIB',        USER RANDOMIZER ,              00250000
//*                                      SEGMENT COMPACTION EXIT AND    00260000
//*                                      INDEX MAINTENANCE EXIT         00265000
//        DBDLIB='IMSVS.DBDLIB',         DBD LIBRARY                    00270000
//        RESLIB='IMSVS.RESLIB',                                        00280000
//        DBTLIB='HPS.SHPSLMD0',         HPS LIBRARY                    00290000
//        DBTSRC='HPS.SHPSSAMP'          BUFF PARM DATA SET             00300000
//********************************************************************** 00310000
//* DB SCAN STEP                                                        00320000
//********************************************************************** 00330000
//HDPCSCAN EXEC PGM=DFSRRC00,                                           00340000
//             PARM='ULU,FABPMAIN,&DBD,,,,,,,,,,,&DBRC,N'               00350000
//STEPLIB  DD DSN=&DBTLIB,DISP=SHR                                      00360000
//         DD DSN=&RESLIB,DISP=SHR                                      00370000
//         DD DSN=&USERLIB,DISP=SHR                                     00380000
```

Figure 7. HD Pointer Checker JCL procedure using the HD Pointer Checker processor (FABPPMD) (Part 1 of 3)

```
//*-------------------------------------------------------------------* 00390000
//* FOR IMS DATA SETS                                                   00400000
//*-------------------------------------------------------------------* 00410000
//IMS      DD DSN=&DBDLIB,DISP=SHR                                      00420000
//IMS2     DD DSN=&RESLIB,DISP=SHR                                      00430000
//         DD DSN=&USERLIB,DISP=SHR                                     00440000
//DFSRESLB DD DSN=&RESLIB,DISP=SHR                                      00450000
//DFSVSAMP DD DSN=&DBTSRC(FABPVSAM),DISP=SHR                            00460000
//IEFRDER  DD DUMMY                                                     00470000
//*-------------------------------------------------------------------* 00490000
//* REPORTS                                                             00500000
//*-------------------------------------------------------------------* 00510000
//PRIMAPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00520000
//STATIPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00530000
//VALIDPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00540000
//EVALUPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00550000
//EVALIPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00560000
//SNAPPIT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00570000
//SUMMARY  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00580000
//SYSUDUMP DD SYSOUT=A                                                  00590000
//*-------------------------------------------------------------------* 00600000
//* HISTORY ANALYSIS DATA SET                                           00610000
//*-------------------------------------------------------------------* 00620000
//HISTORY  DD DSN=&HISTORY,DISP=SHR                                     00630000
//*-------------------------------------------------------------------* 00640000
//* SORT RECORDS                                                        00650000
//*-------------------------------------------------------------------* 00660000
//SORTEX01 DD DSN=&&SORTEX,DISP=(NEW,PASS,DELETE),                      00700000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                              00710000
//             DCB=(BLKSIZE=&SORTBLK)                                   00730000
//MERGIN01 DD DSN=&&MERGIN,DISP=(NEW,PASS,DELETE),                      00760000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                     @PN04801 00770000
//             DCB=(BLKSIZE=&SORTBLK)                                   00770500
//*-------------------------------------------------------------------* 00771000
//* SPECIFY SORTEX2 AND MERGIN2 FOR SCAN WITH IXKEYCHK = YES            00773000
//*-------------------------------------------------------------------* 00774000
//SORTE201 DD &SORT2.DSN=&&SORTEX2,DISP=(NEW,PASS,DELETE),              00776000
//             UNIT=&U,SPACE=(CYL,(&CYL))                               00777000
//MERGI201 DD &MERG2.DSN=&&MERGIN2,DISP=(NEW,PASS,DELETE),              00779000
//             UNIT=&U,SPACE=(CYL,(&CYL))                               00780000
//*-------------------------------------------------------------------* 00782000
//* SPECIFY SORTIL FOR SCAN WITH EPSCHK = YES                           00783000
//*-------------------------------------------------------------------* 00785000
//SORTIL01 DD DSN=&SORTIL,DISP=(NEW,PASS,DELETE),                       00786000
//             UNIT=&U,SPACE=(CYL,(&CYL))                               00788000
//*-------------------------------------------------------------------* 00790000
//* FOR SCAN WITH HDAM/HIDAM PROCESS                                    00800000
//*-------------------------------------------------------------------* 00810000
//KEYSIN   DD DSN=&KEYS,DISP=(NEW,CATLG,DELETE),                        00820000
//             UNIT=&KEYSU,SPACE=(CYL,(&KEYSCYL)),VOL=SER=&KEYSVOL,     00830000
//             DCB=(RECFM=VB)                                           00840000
```

Figure 7. HD Pointer Checker JCL procedure using the HD Pointer Checker processor (FABPPMD) (Part 2 of 3)

```
//********************************************************************* 01330000
//* CHECK STEP                                                          01340000
//********************************************************************* 01350000
//HDPCCHK  EXEC PGM=DFSRRC00,                                           01360000
//             PARM='ULU,FABPMAIN,&DBD,,,,,,,,,,,&DBRC,N',              01370000
//             COND=(4,LT,HDPCSCAN)                                     01380000
//STEPLIB  DD DSN=&DBTLIB,DISP=SHR                                      01390000
//         DD DSN=&RESLIB,DISP=SHR                                      01400000
//         DD DSN=&USERLIB,DISP=SHR                                     01410000
//*-------------------------------------------------------------------* 01420000
//* FOR IMS DATA SETS                                                   01430000
//*-------------------------------------------------------------------* 01440000
//IMS      DD DSN=&DBDLIB,DISP=SHR                                      01450000
//IMS2     DD DSN=&RESLIB,DISP=SHR                                      01460000
//         DD DSN=&USERLIB,DISP=SHR                                     01470000
//DFSRESLB DD DSN=&RESLIB,DISP=SHR                                      01480000
//DFSVSAMP DD DSN=&DBTSRC(FABPVSAM),DISP=SHR                            01490000
//IEFRDER  DD DUMMY                                                     01500000
//*-------------------------------------------------------------------* 01530000
//* REPORTS                                                             01540000
//*-------------------------------------------------------------------* 01550000
//PRIMAPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01560000
//STATIPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01570000
//VALIDPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01580000
//EVALUPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01590000
//EVALIPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01600000
//SNAPPIT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01610000
//SUMMARY  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01620000
//SYSUDUMP DD SYSOUT=A                                                  01630000
//*-------------------------------------------------------------------* 01680000
//* HISTORY ANALYSIS DATA SET                                           01690000
//*-------------------------------------------------------------------* 01700000
//HISTORY  DD DSN=&HISTORY,DISP=SHR                                     01710000
//*-------------------------------------------------------------------* 01710900
//* FOR CHECK PROCESS                                                   01711000
//*-------------------------------------------------------------------* 01712000
//SORTEX01 DD DSN=&&SORTEX,DISP=(OLD,DELETE,DELETE)                     01713000
//MERGIN01 DD DSN=&&MERGIN,DISP=(OLD,DELETE,DELETE)                     01714000
//SORTE201 DD &SORT2.DSN=&&SORTEX2,DISP=(OLD,DELETE,DELETE)             01715000
//MERGI201 DD &MERG2.DSN=&&MERGIN2,DISP=(OLD,DELETE,DELETE)             01716000
//SORTIL01 DD DSN=&SORTIL,DISP=(OLD,DELETE,DELETE)                      01717000
//FSESTAT  DD DSN=&&FSESTAT,DISP=(NEW,DELETE,DELETE),                   01718000
//             UNIT=&U,SPACE=(CYL,(2,2)),DCB=BLKSIZE=&PRTBLK2           01719000
//*-------------------------------------------------------------------* 01720000
//* FOR CHECK AND BLKMAP PROCESS                                        01730000
//*-------------------------------------------------------------------* 01740000
//JRM      DD DSN=&&JRM,DISP=(NEW,DELETE,DELETE),                       01760000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                              01770000
//             DCB=(BLKSIZE=&SORTBLK)                                   50000000
```

Figure 7. HD Pointer Checker JCL procedure using the HD Pointer Checker processor (FABPPMD) (Part 3 of 3)

Procedure FABPPTAM

This procedure, shown in Figure 8, runs HD Pointer Checker and HD Tuning Aid sequentially. The procedure includes control statements for the for allocating of the work data sets. The user supplies the PSB used in this procedure. This procedure checks pointers in multiple job steps. EPSCHK=YES cannot be specified with this procedure.

```
//********************************************************************  00010000
//*   Licensed Materials - Property of IBM                           *  00020000
//*                                                                  *  00030000
//*   5655-K53                                                       *  00040000
//*                                                                  *  00050000
//*   (c) Copyright IBM Corporation 2000, 2006. All rights reserved. *  00060000
//*                                                                  *  00070000
//*   US Government Users Restricted Rights - Use,                   *  00080000
//*   duplication or disclosure restricted by GSA ADP                *  00090000
//*   Schedule Contract with IBM Corp.                               *  00100000
//*                                                                  *  00110000
//********************************************************************  00120000
//        PROC PSB=,                     PSBNAME                        00130000
//        DBRC=Y,                        FOR HDTA WHEN HALDB            00140000
//        HDTA02P=,                      PERM FOR STEP HDTA02P          00150000
//        KEYSVOL=,                      VOL NAME OF KEYSIN             00160000
//        KEYSU='SYSDA',                 UNIT FOR KEYSIN DATA SET       00170000
//        KEYSCYL='1,1',                 SPACE FOR KEYSIN DATA SET      00180000
//        U=SYSDA,                       UNIT FOR WORK DATA SETS        00190000
//        CYL='1,1',                     SPACE FOR WORK DATA SETS       00200000
//        SORT2=,            SORT2='DUMMY,' IF NO SORTEX2 DATA SET      00208000
//        MERG2=,            MERG2='DUMMY,' IF NO MERGIN2 DATA SET      00210000
//        SORTIL='&&SORTIL',                DS NAME IF HALDB PROCESS    00220000
//        SORTOL='&&SORTOL',                DS NAME IF HALDB PROCESS    00230000
//        PRTBLK=0,                      BLKSIZE OF PRINT DATA SETS     00232000
//        PRTBLK2=0,                     BLKSIZE OF FSESTAT DATA SET    00235000
//        SORTBLK=0,                     BLKSIZE OF SORT RECORDS        00237000
//        HISTORY='NULLFILE',            HISTORY DATA SET               00240000
//        USERLIB='USER.LOADLIB',        USER RANDOMIZER ,              00250000
//*                                      SEGMENT COMPACTION EXIT AND    00260000
//*                                      INDEX MAINTENANCE EXIT         00265000
//        DBDLIB='IMSVS.DBDLIB',         DBD LIBRARY                    00270000
//        PSBLIB='IMSVS.PSBLIB',         PSB LIBRARY                    00280000
//        RESLIB='IMSVS.RESLIB',                                        00290000
//        DBTLIB='HPS.SHPSLMD0',         HPS LIBRARY                    00300000
//        DBTSRC='HPS.SHPSSAMP'          BUFF PARM DATA SET             00310000
```

Figure 8. HD Pointer Checker JCL procedure using the HD Pointer Checker processor (FABPPTAM) (Part 1 of 4)

```
//********************************************************************* 00320000
//* DB SCAN STEP                                                        00330000
//********************************************************************* 00340000
//HDPCSCAN EXEC PGM=DFSRRC00,                                           00350000
//             PARM='DLI,FABPMAIN,&PSB,,,,,,,,,,&DBRC,N'                00360000
//STEPLIB  DD DSN=&DBTLIB,DISP=SHR                                      00370000
//         DD DSN=&RESLIB,DISP=SHR                                      00380000
//         DD DSN=&USERLIB,DISP=SHR                                     00390000
//*-------------------------------------------------------------------* 00400000
//* FOR IMS DATA SETS                                                   00410000
//*-------------------------------------------------------------------* 00420000
//IMS      DD DSN=&PSBLIB,DISP=SHR                                      00430000
//         DD DSN=&DBDLIB,DISP=SHR                                      00440000
//IMS2     DD DSN=&RESLIB,DISP=SHR                                      00450000
//         DD DSN=&USERLIB,DISP=SHR                                     00460000
//DFSRESLB DD DSN=&RESLIB,DISP=SHR                                      00470000
//DFSVSAMP DD DSN=&DBTSRC(FABPVSAM),DISP=SHR                            00480000
//IEFRDER  DD DUMMY                                                     00490000
//*-------------------------------------------------------------------* 00510000
//* REPORTS                                                             00520000
//*-------------------------------------------------------------------* 00530000
//PRIMAPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00540000
//STATIPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00550000
//VALIDPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00560000
//EVALUPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00570000
//EVALIPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00580000
//SNAPPIT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00590000
//SUMMARY  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     00600000
//SYSUDUMP DD SYSOUT=A                                                  00610000
//*-------------------------------------------------------------------* 00620000
//* HISTORY ANALYSIS DATA SET                                           00630000
//*-------------------------------------------------------------------* 00640000
//HISTORY  DD DSN=&HISTORY,DISP=SHR                                     00650000
//*-------------------------------------------------------------------* 00660000
//* SORT RECORDS                                                        00670000
//*-------------------------------------------------------------------* 00680000
//SORTEX01 DD DSN=&&SORTEX,DISP=(NEW,PASS,DELETE),                      00720000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                              00730000
//             DCB=(BLKSIZE=&SORTBLK)                                   00750000
//MERGIN01 DD DSN=&&MERGIN,DISP=(NEW,PASS,DELETE),                      00780000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                     @PN04801 00790000
//             DCB=(BLKSIZE=&SORTBLK)                                   00790500
//*-------------------------------------------------------------------* 00791000
//* SPECIFY SORTEX2 AND MERGIN2 FOR SCAN WITH IXKEYCHK = YES            00793000
//*-------------------------------------------------------------------* 00794000
//SORTE201 DD &SORT2.DSN=&&SORTEX2,DISP=(NEW,PASS,DELETE),              00796000
//             UNIT=&U,SPACE=(CYL,(&CYL))                               00797000
//MERGI201 DD &MERG2.DSN=&&MERGIN2,DISP=(NEW,PASS,DELETE),              00799000
//             UNIT=&U,SPACE=(CYL,(&CYL))                               00800000
//*-------------------------------------------------------------------* 00802000
//* SPECIFY SORTIL FOR SCAN WITH EPSCHK = YES                           00803000
//*-------------------------------------------------------------------* 00805000
//SORTIL01 DD DSN=&SORTIL,DISP=(NEW,PASS,DELETE),                       00806000
//             UNIT=&U,SPACE=(CYL,(&CYL))                               00808000
//*-------------------------------------------------------------------* 00810000
//* FOR SCAN WITH HDAM/HIDAM PROCESS                                    00820000
//*-------------------------------------------------------------------* 00830000
//KEYSIN   DD DSN=&KEYSIN,DISP=(NEW,CATLG,DELETE),                      00840000
//             UNIT=&KEYSU,SPACE=(CYL,(&KEYSCYL)),VOL=SER=&KEYSVOL,     00850000
//             DCB=(RECFM=VB)                                           00860000
```

Figure 8. HD Pointer Checker JCL procedure using the HD Pointer Checker processor (FABPPTAM) (Part 2 of 4)

```
//********************************************************************* 01350000
//* CHECK STEP                                                          01360000
//********************************************************************* 01370000
//HDPCCHK  EXEC PGM=DFSRRC00,                                           01380000
//             PARM='DLI,FABPMAIN,&PSB,,,,,,,,,,&DBRC,N',               01390000
//             COND=(4,LT,HDPCSCAN)                                     01400000
//STEPLIB  DD DSN=&DBTLIB,DISP=SHR                                      01410000
//         DD DSN=&RESLIB,DISP=SHR                                      01420000
//         DD DSN=&USERLIB,DISP=SHR                                     01430000
//*-------------------------------------------------------------------* 01440000
//* FOR IMS DATA SETS                                                   01450000
//*-------------------------------------------------------------------* 01460000
//IMS      DD DSN=&PSBLIB,DISP=SHR                                      01470000
//         DD DSN=&DBDLIB,DISP=SHR                                      01480000
//IMS2     DD DSN=&RESLIB,DISP=SHR                                      01490000
//         DD DSN=&USERLIB,DISP=SHR                                     01500000
//DFSRESLB DD DSN=&RESLIB,DISP=SHR                                      01510000
//DFSVSAMP DD DSN=&DBTSRC(FABPVSAM),DISP=SHR                            01520000
//IEFRDER  DD DUMMY                                                     01530000
//*-------------------------------------------------------------------* 01560000
//* REPORTS                                                             01570000
//*-------------------------------------------------------------------* 01580000
//PRIMAPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01590000
//STATIPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01600000
//VALIDPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01610000
//EVALUPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01620000
//EVALIPRT DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01630000
//SNAPPIT  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01640000
//SUMMARY  DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK,OUTLIM=0                     01650000
//SYSUDUMP DD SYSOUT=A                                                  01660000
//*-------------------------------------------------------------------* 01710000
//* HISTORY ANALYSIS DATA SET                                           01720000
//*-------------------------------------------------------------------* 01730000
//HISTORY  DD DSN=&HISTORY,DISP=SHR                                     01740000
//*-------------------------------------------------------------------* 01740600
//* FOR CHECK PROCESS                                                   01741000
//*-------------------------------------------------------------------* 01741900
//SORTEX01 DD DSN=&&SORTEX,DISP=(OLD,DELETE,DELETE)                     01742000
//MERGIN01 DD DSN=&&MERGIN,DISP=(OLD,DELETE,DELETE)                     01743000
//SORTE201 DD &SORT2.DSN=&&SORTEX2,DISP=(OLD,DELETE,DELETE)             01743900
//MERGI201 DD &MERG2.DSN=&&MERGIN2,DISP=(OLD,DELETE,DELETE)             01744000
//SORTIL01 DD DSN=&SORTIL,DISP=(OLD,DELETE,DELETE)                      01745000
//SORTOL   DD DSN=&SORTOL,DISP=(NEW,DELETE,DELETE),                     01745900
//             UNIT=&U,SPACE=(CYL,(&CYL))                               01746000
//IXKEY    DD DSN=&&IXKEY,DISP=(NEW,DELETE,DELETE),                     01747000
//             UNIT=&U,SPACE=(CYL,(&CYL))                               01747900
//FSESTAT  DD  DSN=&&FSESTAT,DISP=(NEW,DELETE,DELETE),                  01748000
//             UNIT=&U,SPACE=(CYL,(2,2)),DCB=BLKSIZE=&PRTBLK2           01749000
//*-------------------------------------------------------------------* 01750000
//* FOR CHECK AND BLKMAP PROCESS                                        01760000
//*-------------------------------------------------------------------* 01770000
//JRM      DD DSN=&&JRM,DISP=(NEW,DELETE,DELETE),                       01790000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                              01800000
//             DCB=(BLKSIZE=&SORTBLK)                                   01820000
```

Figure 8. HD Pointer Checker JCL procedure using the HD Pointer Checker processor (FABPPTAM) (Part 3 of 4)

```
//********************************************************************* 01840000
//* FABTROOT STEP                                                       01850000
//********************************************************************* 01860000
//HDTA01  EXEC PGM=DFSRRC00,                                            01870000
//             PARM='DLI,FABTROOT,&PSB,,,,,,,,,,&DBRC,N',               01880000
//             REGION=1000K,TIME=(,30),                                 01890000
//             COND=((4,LT,HDPCSCAN),(2,LT,HDPCCHK))                    01900000
//STEPLIB  DD DSN=&DBTLIB,DISP=SHR                                      01910000
//         DD DSN=&RESLIB,DISP=SHR                                      01920000
//IMS      DD DSN=&DBDLIB,DISP=SHR                                      01930000
//         DD DSN=&PSBLIB,DISP=SHR                                      01940000
//IMS2     DD DSN=&RESLIB,DISP=SHR                                      01950000
//         DD DSN=&USERLIB,DISP=SHR                                     01960000
//IEFRDER  DD DUMMY,DCB=BLKSIZE=1408                                    01970000
//DFSRESLB DD DSN=&RESLIB,DISP=SHR                                      01980000
//DFSVSAMP DD DSN=&DBTSRC(FABPVSAM),DISP=SHR                            01990000
//RAPSIN   DD DSN=&&RAPSIN,UNIT=&U,                                     02000000
//             DISP=(NEW,PASS,DELETE),                                  02010000
//             SPACE=(CYL,(&CYL)),                                      02020000
//             DCB=(LRECL=42,BLKSIZE=8400,RECFM=FB)                     02030000
//PR8      DD SYSOUT=A                                                  02040000
//PR10     DD SYSOUT=A                                                  02050000
//KEYSIN   DD DSN=&&KEYSIN,DISP=(OLD,DELETE,DELETE)                     02060000
//SYSUDUMP DD SYSOUT=A                                                  02070000
//********************************************************************* 02080000
//* RAPSIN SORT STEP                                                    02090000
//********************************************************************* 02100000
//HDTA02   EXEC PGM=SORT,PARM='&HDTA02P',COND=((2,LT,HDPCSCAN),         02110000
//             (2,LT,HDPCCHK),(0,LT,HDTA01))                            02120000
//*-------------------------------------------------------------------* 02130000
//* SORT RECORDS                                                        02140000
//*-------------------------------------------------------------------* 02150000
//SORTIN   DD DSN=&&RAPSIN,DISP=(OLD,DELETE,DELETE)                     02160000
//SORTOUT  DD DSN=&&KEYSOUT,UNIT=&U,DISP=(NEW,PASS,DELETE),             02170000
//             SPACE=(CYL,(&CYL)),                                      02180000
//             DCB=(LRECL=42,BLKSIZE=8400,RECFM=FB)                     02190000
//*-------------------------------------------------------------------* 02200000
//* SORT STATEMENT AND OUTPUT                                           02210000
//*-------------------------------------------------------------------* 02220000
//SYSOUT   DD SYSOUT=A                                                  02230000
//SYSIN    DD DSN=&DBTSRC(FABPSORT),DISP=SHR                            02240000
//*-------------------------------------------------------------------* 02250000
//* SORT WORK                                                           02260000
//*-------------------------------------------------------------------* 02270000
//SORTWK01 DD UNIT=&U,SPACE=(CYL,(&CYL))                                02280000
//SORTWK02 DD UNIT=&U,SPACE=(CYL,(&CYL))                                02290000
//SORTWK03 DD UNIT=&U,SPACE=(CYL,(&CYL))                                02300000
//SORTWK04 DD UNIT=&U,SPACE=(CYL,(&CYL))                                02310000
//SORTWK05 DD UNIT=&U,SPACE=(CYL,(&CYL))                                02320000
//SORTWK06 DD UNIT=&U,SPACE=(CYL,(&CYL))                                02330000
//********************************************************************* 02340000
//* FABTRAPS STEP                                                       02350000
//********************************************************************* 02360000
//HDTA03   EXEC PGM=FABTRAPS,COND=((2,LT,HDPCSCAN),                     02370000
//             (2,LT,HDPCCHK),(0,LT,HDTA01),(0,LT,HDTA02))              02380000
//STEPLIB  DD DSN=&DBTLIB,DISP=SHR                                      02390000
//KEYSOUT  DD DSN=&&KEYSOUT,DISP=(OLD,DELETE,DELETE)                    02400000
//PR9      DD SYSOUT=A                                                  02410000
//PR9X     DD SYSOUT=A                                                  02420000
//SYSUDUMP DD SYSOUT=A                                                  02430000
```

Figure 8. HD Pointer Checker JCL procedure using the HD Pointer Checker processor (FABPPTAM) (Part 4 of 4)

Dynamic allocation for database data set and image copy data set

The HD Pointer Checker provides the dynamic allocation function that is invoked when you omit the DD statement for the database data set or the image copy data set.

DISP=SHR is used to allocate the data sets.

The following sections describe how the HD Pointer Checker dynamically allocates the data sets required for its run.

Database data set of Non-HALDB

The database data sets of the HDAM, HIDAM, HISAM, SHISAM, and index databases are dynamically allocated by use of the DFSMDA member in the libraries specified in the STEPLIB DD statement. The NODYNALLOC statement in DFSVSMxx member is not referred to.

DISP=SHR is used, regardless of the definitions in DFSMDA, to allocate the data sets.

The data sets need to be cataloged. DBRC=Y is not required.

A shared secondary index database has multiple database names. If only some database names are defined in the DFSMDA macro, specify those databases before other databases in the DATABASE statement in PROCCTL.

Database data set of HALDB

The database data sets of the PHDAM, PHIDAM, and PSINDEX databases are dynamically allocated by use of the information in the RECON data set.

DISP=SHR is used, regardless of the definitions in RECON, to allocate the data sets.

The data sets need to be cataloged. DBRC=Y is required.

When a database is online reorganization capable, the active set of DBDS registered in the RECON is allocated dynamically with the active DD name.

Image copy data set

The image copy data sets of both non-HALDB and HALDB are dynamically allocated by use of the information in the RECON data set.

- If two or more image copies are registered, the latest data set name is used.
- If secondary image copy is registered, the primary image copy is used.
- If the image copy is a GDG data set, the absolute generation and version numbers is used as the data set name.

DISP=SHR is used, regardless of definitions in RECON, to allocate the data sets.

DBRC=Y is required.

- If CATDS is specified in the RECON, HD Pointer Checker uses the information in the system catalog. The data sets need to be cataloged.
- If NOCATDS is specified in the RECON, HD Pointer Checker uses the volume serial number and the file sequence number that are registered in the RECON. If ICUNIT is specified in the OPTION statement in the PROCCTL data set, the unit

name is used. If ICUNIT is not specified, the unit name is retrieved from the RECON. The data sets need not be cataloged.

When a database is online reorganization capable, the newest image copy data set of the data set groups (A-J&X) and (M-V&Y), which is registered in the RECON is allocated dynamically. And the DD name is used as same name at the time of the image copy was taken.

Consideration for using tapes

If two or more data sets are stacked in a tape, all data sets in the tape have to be allocated dynamically or all DD statements of data sets in the tape have to be specified apparently in JCL.

Input

This section describes all of the input that you must specify to run the HD Pointer Checker processor (FABPMAIN). It includes the conventions for coding control statements for the PROCCTL data set.

FABPMAIN PROCCTL data set

This section explains the FABPMAIN PROCCTL data set.

Function

The PROCCTL data set contains your description of the processing to be done by HD Pointer Checker.

Format

This control data set usually resides in the input stream. However, it can also be defined 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.

This data set can contain one or more combinations of five types of control statements; PROC, DATABASE, OPTION, REPORT, and END. These control statements can be coded as shown in Figure 9.

```
//PROCCTL DD *
 PROC     TYPE=ALL,DBORG=ALL
 OPTION   ERRLIMIT=YES,DIAGDUMP=ERROR,HISTORY=YES,KEYSIN=NO,
          T2CHK=(0,7),ZEROCTR=NO    COMMENT
 REPORT   RUNTM=YES,INTST=YES,INTFS=YES,DBDIST=YES,BITMAP=NO,
          FSEMAP=YES,MAXFSD=YES
* COMMENT STATEMENTS
 DATABASE DB=HDAMDB01,DD=HDAMDS01
  OPTION  ERRLIMIT=YES,DIAGDUMP=ERROR,HISTORY=YES,KEYSIN=YES,
          T2CHK=(0,20),ZEROCTR=YES
  REPORT  RUNTM=YES,INTST=YES,INTFS=YES,DBDIST=YES,BITMAP=YES,
          FSEMAP=YES,MAXFSD=YES
/*
```

Figure 9. Sample control statement format in PROCCTL data set

Note: On the control statement, code uppercase alphabetical characters, numerical characters, and the following special characters:

asterisk	*
comma	,
equal sign	=
parenthesis	( )

Control statement syntax

The following describes the coding conventions that you must follow in writing control statements in the PROCCTL data set:

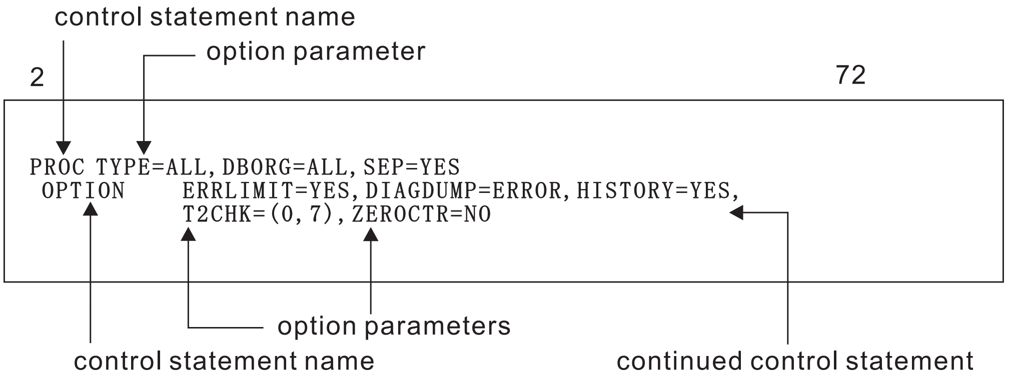
- A control statement can be coded onto one or more control statement records. Control statement names (those are, PROC, DATABASE, OPTION, REPORT, and END) and option parameters must be coded within column 2 and column 72. A control statement name must be the first entry in the control statement.
- Option parameters follow the control statement name separated by one or more blanks. A control statement name and the first option parameter must be written in the same control statement record. When more than one option parameter is coded, they must be separated by commas. No blanks are allowed between the option parameters and the commas, or within the option parameters.

  Option parameters can be continued onto one or more following control statement records. The control statement starting with a control statement name must be completed with an option parameter including the comma that follows it. Then option parameters can be continued onto another control statement records begin in any column from 2.

  Option parameters are not the positional parameter; they can be specified in any order of sequence.

  A null value is not allowed for any option parameter.
- Comments may follow the last option parameter on each control statement record separated by at least one blank.
- A comment line must begin with an asterisk in column 1.

Figure 10 shows the control statement syntax.



```
PROC TYPE=ALL,DBORG=ALL,SEP=YES
 OPTION    ERRLIMIT=YES,DIAGDUMP=ERROR,HISTORY=YES,
           T2CHK=(0,7),ZEROCTR=NO
```

Figure 10. Control statement syntax—PROCCTL

PROC statement

The PROC statement specifies the options for all databases to be processed. There must be only one PROC statement, and it must be the first control statement in the PROCCTL data set.

The control statement format illustrations in this book use the following conventions:

- Brackets [ ] are option parameters.
- An UNDERSCORED value is a default value. If no option parameter is specified, the underscored value is assumed to be used.
- Items separated by a vertical bar | represent alternative items. No more than one of these items must be selected.
- Braces { } are a choice of entry. At least one of the entries must be specified.
- Parentheses ( ) are required if multiple entries are specified.

The PROC statement contains the keywords shown in Figure 11.

```
PROC  TYPE=ALL|SCAN|CHECK|BLKMAP

   ┌                                                                    ┐
   │ ,DBORG={ALL,HDAM,HIDAM,DAM,HISAM,INDEX,PHDAM,PHIDAM,PDAM,PSINDX}   │
   │ ,HASH=YES|NO|FORCE                                                 │
   │ ,EPSCHK=YES|NO                                                     │
   │ ,IXKEYCHK=YES|NO                                                   │
   │ ,SEP=YES|NO                                                        │
   │ ,VLSSUMM=YES|NO                                                    │
   │ ,CHECK=ALL|(CHK,nnnnnn)                                            │
   │           |(CHK,111111)                                            │
   │ ,CHECKREC=YES|NO                                                   │
   │ ,SYMIXCHK=YES|NO                                                   │
   │ ,SYMLPCHK=YES|NO                                                   │
   │ ,ICRG#CHK=YES|NO                                                   │
   │ ,USER=userid|(userid1,userid2,......)|*NO                          │
   │ ,RETCDDSN=dsn|dsn(member)|*NO                                      │
   │ ,ITKBSRVR=servername|*NO                                           │
   │ ,ITKBLOAD=dsn|*NO                                                  │
   └                                                                    ┘
```

Figure 11. PROC statement syntax

TYPE=

Specifies the type of the process to be done.

ALL

Specifies that following processes are to be done by HD Pointer Checker:

1. **SCAN Process**
2. **CHECK Process**
3. **BLOCKMAP Process:** This process is run only when some pointer errors are detected.

SCAN

Specifies that the following process is to be done by HD Pointer Checker:

1. **SCAN Process**

CHECK

Specifies that the following processes are to be done by HD Pointer Checker:

2. **CHECK Process**

3. **BLOCKMAP Process:**This process is run only when some pointer errors are detected.

Note: It is strongly recommended that you use PROC TYPE=ALL and not PROC TYPE=SCAN and CHECK, because by using PROC TYPE=ALL, the performance and the size of work data sets will improve and the JCL statements are more simple.

BLKMAP

Specifies that a stand-alone BLOCKMAP process is run by HD Pointer Checker.

3. **BLOCKMAP Process**

(Use the BLKMAPIN data set which contains the user specified control statement records, and the CHECKREC data set which contains all sorted records as the input.)

Note: Before you run the BLKMAP process, you must run either the TYPE=ALL process or the TYPE=CHECK process, with CHECKREC=YES, and you must retain the CHECKREC data set.

DBORG=

Specifies one or more database types to be processed. At least one database data set group of the DBORG=parameter must be specified by the succeeding the DATABASE statements.

This option can be specified when TYPE=ALL or TYPE=SCAN is specified. If you specify TYPE=ALL and INDEX or PSINDEX for this parameter to process an index database, you must include a database type for an indexed database. This means, you cannot process only an index database.

ALL

Specifies that all types of databases (HDAM, HIDAM, HISAM, INDEX, PHDAM, PHIDAM, and PSINDEX) specified by the succeeding the DATABASE statements are to be processed. This is the default value. If DBORG=ALL is specified with any other parameters, other parameters are ignored.

HDAM

Specifies that the HDAM databases specified by the DATABASE statements are to be processed.

HIDAM

Specifies that the primary databases of the HIDAM databases specified by the DATABASE statements are to be processed.

DAM

Specifies that the HDAM, HIDAM, PHDAM and PHIDAM databases specified by the DATABASE statements are to be processed. If DAM is specified with HDAM, HIDAM, PHDAM, or PHIDAM parameters, these parameters are ignored.

HISAM

Specifies that the HISAM. (including SHISAM) databases specified by the DATABASE statements are to be processed.

INDEX
Specifies that the HIDAM index, the secondary index databases, the PHIDAM index, and the PSINDEX databases specified by the DATABASE statements are to be processed.

PHDAM
This option specifies that PHDAM databases specified by the DATABASE statement are to be processed.

PHIDAM
This option specifies that PHIDAM databases specified by the DATABASE statement are to be processed. The primary index databases pointed to by the PHIDAM database are also to be processed.

PDAM
Specifies that the PHDAM and PHIDAM databases specified by the DATABASE statements are processed. If PDAM is specified with PHDAM or PHIDAM parameters, PHDAM or PHIDAM parameters are ignored.

PSINDX
This option specifies that PSINDEX databases specified by the DATABASE statement are to be processed.

HASH=
Specifies whether you want to check the pointers by use of the HASH Check function. This option can be specified when TYPE=ALL or SCAN is specified. When HASH=YES or FORCE is specified, INCORE=YES and EPSCHK=YES are ignored and set to NO.

YES
Specifies that all databases specified by the succeeding the DATABASE statements are to be processed by the HASH Check function without in-core pointer checking. The HASH Check function provides an extra fast pointer-checking capability. If the HASH Check function is not applicable to some databases because of the restrictions, the job will end. (See “Restrictions and considerations” on page 30, and “Restrictions and considerations” on page 310.)

NO
Specifies that the HD Pointer Checker creates the segment and pointer sort records without using the HASH Check function. This is the comprehensive technique for pointer checking; however, more CPU and I/O time is required. This is the default value.

FORCE
Specifies that the HASH Check function applies to the databases that are not subject to restrictions, but the databases that are not applicable to the HASH Check function are processed using the standard pointer checking technique. The HD Pointer Checker analyzes each specified database data set group to check if the HASH Check function is applicable or not.

EPSCHK=

This keyword specifies whether EPS evaluation is to be done for HALDB.

The EPS evaluation validates the following pointers:

- The LP pointers or the paired LC pointers
- Pointers in secondary indexes (PSINDEXs)

After the target partition is reorganized or migrated, some of the target RBAs in these pointers may be outdated. The latest RBA information is stored in an ILE (indirect list entry). The ILEs are in an ILDS. HD Pointer Checker validates EPS by referring to the ILEs as follows:

- EPS healing

  If the pointer has the outdated RBA value, HD Pointer Checker revises the RBA with reference to the corresponding ILE. The revised RBA value is validated in the CHECK process. If there is no corresponding ILE, the reason is probably that either an ILK (indirect list key) in the source segment, or the index of ILDS has been corrupted, and HD Pointer Checker identifies the trouble as a pointer error.
- ILK checking

  ILKs are contained in the source segment, ILE, and the target segment. These three objects are related to each other by the ILK value. During the CHECK process, HD Pointer Checker checks whether the ILK in the source segment and the ILK in the target segments are equivalent. As was just explained, the ILK in the ILE is checked during EPS healing.

This keyword can be specified if TYPE=ALL or TYPE=SCAN is specified. If HASH=YES is specified, this keyword is ignored.

EPS evaluation is effective only for a HALDB. If EPSCHK=YES is specified and non-HALDB is included in the DATABASE statement, this keyword does not affect a non-HALDB.

YES
EPS is checked. This is the default.

NO
EPS is not checked.

IXKEYCHK=
Specifies whether you want to check the pointers and the key data by use of the Index Key Check function. If this option is not specified, only the pointers of the primary or the secondary index databases are checked. The Index Key Check function checks not only the pointers, but also the key data of the primary and secondary index databases, against the associated segment RBA and the source key data of the primary databases.

| This option can be specified when TYPE=ALL or SCAN is specified.

Abbreviation **KEYCHK** and **IKEYCHK** can be used for **IXKEYCHK**.

YES
Specifies that the Index Key check is to be applied to the databases. Index databases and associated primary databases must be specified in the succeeding DATABASE statements. If a primary database is indexed in more than one index database or vice versa, this option applies only to the specified database.

NO
Specifies that the Index Key Check function is not to be performed. Only pointers in a primary or a secondary index are checked. This is the default value.

SEP=
Specifies whether you want to generate the separator pages for your reports. This option can be specified with any TYPE= specification.

YES

Specifies that the separator pages are to be generated for each data set for the reports. This is the default value.

NO

Specifies that the separator pages are not to be generated for the reports.

VLSSUMM=

Specify whether "Summary of VL Segment Sizes" is printed or not in the Partition Statistics report and Database Statistics report. This option can be specified when TYPE=ALL or SCAN in specified. If YES is specified, "Summary of VL Segment Sizes" is printed for every partition and database.

YES

Print "Summary of VL Segment Sizes".

NO

Do not print "Summary of VL Segment Sizes". This is the default value.

CHECK=

Specifies the certain kinds of pointers that are to be turned off. It is also used to specify whether to print all input records.

This option can be specified when TYPE=ALL or TYPE=CHECK is specified.

ALL

This option specifies that all input records in the CHECK process, and all error messages, are to be printed. If this option is specified, you will get an extremely large report.

(CHK,*nnnnnn*)

This option specifies that the following pointer checking is to be turned off by the CHECK processor. *n* is 1 or 0. For details, see Chapter 14, “HD Pointer Checker options for debugging.”

- HIDAM/PHIDAM index pointers
- physical pointers
- logical pointers
- the counter field versus the number of logical child segments
- physically paired segments
- printing of hash formulas

CHECKREC=

Specifies the CHECKREC data set to be created. This option can be specified with TYPE=ALL or CHECK. The CHECKREC data set contains work records for printing the maps and dumps of database blocks, and is used in step TYPE=BLKMAP, which follows it.

YES

Specifies to create the CHECKREC data set.

NO

Specifies not to create the CHECKREC data set. This is the default value.

SYMIXCHK=

Specifies whether to validate the symbolic pointers in secondary index databases and the key data. This option can be specified when TYPE=ALL and HASH=NO are specified. The symbolic pointers can be validated when the target is a root segment.

YES

Specifies to validate the symbolic pointers. Index databases and associated primary databases must be specified in the DATABASE statements.

NO
Specifies not to validate the symbolic pointers. This is the default value.

SYMLPCHK=
Specifies whether to check the symbolic LP pointers. If both direct LP and symbolic LP pointers are defined in the logical child, both are validated. This option can be specified when TYPE=ALL and HASH=NO is specified. The symbolic LP pointers can be validated when the logical parent is a root segment.

YES
Specifies to validate the symbolic LP pointers. Both the databases having logical parent segment and logical child segments must be specified in the DATABASE statements.

NO
Specifies not to validate the symbolic LP pointers. This is the default value.

ICRG#CHK=
Specifies whether to do the HALDB partition reorganization number validation for image copy data sets.

IMS maintains the HALDB partition reorganization numbers in the RECON data set and the partition data set. IMS validates these numbers to maintain the correctness of HALDB partitions. The HALDB partition reorganization validation is enabled by the INIT.RECON REORGV command or the CHANGE.RECON REORGV command.

HD Pointer Checker also validates the reorganization numbers, when the HALDB partition reorganization number validation is enabled. If the partition data set contains a lower reorganization number than the RECON, HD Pointer Checker issues message FABP1097E in the Validation of a Pointer to a Target at Scan report telling that the ILDS rebuild is required.

HD Pointer Checker always validates the reorganization numbers when the input is a real database data set. In the mean time, HD Pointer Checker does the validation optionally for the image copy data set, because the reorganization number in the image copy data set is obsolete if the database is reorganized. It is recommended that you validate the reorganization numbers immediately after taking the image copy.

This option can be specified when TYPE=ALL or TYPE=SCAN is specified. This option is available for the HALDB (PHDAM or PHIDAM database) image copy data set. This option is ignored for the real database data set or a non-HALDB.

YES
Specifies to do the HALDB partition reorganization number validation for the image copy data set.

NO
Specifies not to do the HALDB partition reorganization number validation for the image copy. This is the default.

USER=
Specifies TSO user IDs. HD Pointer Checker will send a notification message to the TSO users, when it detects pointer error or T2 error in a database.

userid
or
(*userid1,userid2,....., userid20***)**
Specify up to 20 TSO user IDs. If the specified TSO user is not logged on to TSO or is disconnected from the terminal, the message will be discarded.

You can also specify the special value *JOBUSR as one of these user IDs. This value will be converted to the user ID of the submitter of a JOB when HD Pointer Checker runs.

HD Pointer Checker does not check whether the specified TSO user is correct. If an incorrect ID is specified, HD Pointer Checker will attempt to send the notification to the user ID and the message will be discarded.

TSO user ID is effective when TYPE=ALL, SCAN, or CHECK is specified to the PROC control statement.

***NO**
Notification message will not be sent to the TSO user. This is the default value.

RETCDDSN=
Specify a data set name or a data set name and a member name that includes HPSRETCD control statements. You can change the return codes of FABPMAIN by the RETCDDSN= control statements. HD Pointer Checker provides two methods to change the return codes: an HPSRETCD DD statement in JCL or the RETCDDSN= parameter. If both the HPSRETCD DD statement and the RETCDDSN= parameter are specified, HD Pointer Checker uses the specification in the HPSRETCD DD statement. For more details of HPSRETCD, see “FABPMAIN HPSRETCD data set” on page 106. This keyword is optional.

dsn
or
dsn(member)
Specify the data set name, or specify the data set name and the member name for the partitioned data set.

***NO**
RETCDDSN data set is not provided. This is the default value.

ITKBSRVR=
Specifies the name of the IMS Tools KB server. HDPC reports are stored in the IMS Tools KB Output repository that is owned by the specified IMS Tools KB server. This keyword is optional.

servername
HDPC stores reports in the IMS Tools KB Output repository of the specified server.

***NO**
HDPC does not store reports in the IMS Took KB Output repository. This is the default.

ITKBLOAD=
Specifies the IMS Tools KB load module data set that is to be used by HDPC. This keyword is optional.

dsn
Specifies the name of the IMS Tools KB load module data set that is to be used by HDPC.

***NO**
The IMS Tools KB modules are loaded from the private library or the system library of the job. This is the default.

DATABASE statement

A DATABASE statement specifies the database name and data set groups to be processed, and the options.

One or more DATABASE statements can be specified in any order in the PROCCTL data set, but must follow a PROC statement.

It is not necessary to specify the DATABASE statement if the TYPE=CHECK or BLKMAP is specified in the PROC statement.

If TYPE=ALL is specified in the PROC statement, you must specify the DATABASE statements of all logically related and index related databases. Otherwise, the job ends with an error message. Figure 12 shows the DATABASE statement syntax.

```
DATABASE   DB=dbdname
          [,PART=partition name|*ALL]
          [,NUM=number]
          [,DD=ddname|dsg-id|*ALL]
         ┌                                ┐
         │ ,OVERFLOW=ddname               │
         │ ,PRIMEDB=dbdname               │
         │ ,DATASET=REAL|IMAGECOPY        │
         │ ,SCANGROUP=nn|1                │
         │ ,BLOCKDUMP=(rba,nnn|1)         │
         │ ,DBALL=YES|NO                  │
         │      .                         │
         │      .                         │
         │      .                         │
         └                                ┘
```

Figure 12. DATABASE statement syntax

DB=*dbdname*
: Specifies the dbdname (as coded in your DBD) of the HISAM, HIDAM index or the secondary index databases, the HDAM, the HIDAM, the PHDAM, PHIDAM, or the PSINDEX to be processed.

When the HIDAM index or the secondary index database is specified, the PRIMEDB=*dbdname* must be specified with this DATABASE statement.

PART=

Specifies the partition to be processed.

This statement is valid for HALDB only.

***ALL** All partitions are to be processed. This is the default.

partition name

Only the specified partition is to be processed.

NUM=

Specifies the number of partitions to be processed. The partitions specified with the NUM= keyword are processed in the partition selection order. If the specified number with the NUM= keyword is larger than the actual partition number, HD Pointer Checker processes up to the last partition without issuing an error message.

This keyword is valid for HALDB only.

This value is applicable only if PART=*partition name* is specified.

DD=

ddname
: Specifies the ddname (as coded in your DBD) of the HDAM, HIDAM, KSDS part of the HISAM data set group, or the index database to be processed.

***ALL** For a non-HALDB, this option specifies to process all data sets in a database. For a HALDB, it specifies to process all data sets in the partition specified by PART=. This is the default.

dsg-id **or (***dsg-id1***,** *dsg-id2***,...)**

Specifies data set group to be processed. This option is valid for HALDB only. Letters A through J, M through V, X and Y can be specified for dsg-id. The letter L cannot be specified. If EPSCHK=YES is effective, ILDS can be evaluated automatically.

Consideration for the Online Reorganization capable HALDB: When DATASET=REAL is specified, HD Pointer Checker ignores the DD= parameter and assumes that the data set to process is the active DBDS. For what happens when DATASET=IMAGECOPY is specified, read about “Consideration for an Online Reorganization (OLR) of HALDB” on page 41 and “Dynamic allocation for database data set and image copy data set” on page 69.

Examples of combination of PART=, NUM=, and DD= for HALDB:

- In the following example all data set groups in all partitions are processed:

```
DATABASE DB=HALDB1   ... The default value *ALL is used for PART= and DD=
```

- In the following example data set groups specified on DD= in all partitions are processed:

```
DATABASE DB=HALDB1,PART=*ALL,DD=A ... Each DSG A is processed in All
partitions.
```

- In the following example all data set groups in PART1 and the following partition are processed:

```
DATABASE DB=HALDB1,PART=PART1,NUM=2,DD=*ALL
```

For more information for specifying the DATABASE statements for HALDB, see “How to specify multiple DATABASE statements for a HALDB” on page 85.

OVERFLOW=*ddname*

Specifies the ddname (as coded in your DBD) of the ESDS of the HISAM data set group, or the index database.

Abbreviations **OFLOW** and **OVFLW** can be used for **OVERFLOW**.

PRIMEDB=*dbdname*

Specifies the dbdname of the primary database indexed by the HIDAM index or the secondary index database to be processed.

DATASET=

Specifies the type of input database.

Abbreviations **DS** for **DATASET**, and **IC** and **ICOPY** for **IMAGECOPY** can be used.

REAL

Specifies that your input database is a real database. This is the default value.

IMAGECOPY

Specifies that your input database is an image copy. The image copy can be one of following:

- A batch image copy taken by the IMS Database Image Copy utility (RECFM=VB)
- A compressed batch image copy taken with IMS Image Copy Extensions or IMS HP Image Copy (RECFM=VB)
- An SMSNOCIC type image copy taken by the IMS Database Image Copy 2 utility (RECFM=U)

HD Pointer Checker identifies the image copy type by checking its record format (RECFM).

When the input database is an image copy, you must specify this parameter even if TYPE=CHECK or BLKMAP is specified.

Note: An SMSNOCIC type image copy has no DBD name or DD name information in the header record. Therefore, HD Pointer Checker cannot verify the image copy with it. The HD Pointer Checker issues the message FABP1365I with the name of the dumped data set name in the image copy so that you can verify the image copy with it.

Consideration for the image copy of HALDB: If you reorganize the HALDB after taking an image copy, specify EPSCHK=NO in the PROC statement, because Index list entries (ILE) in an Index List data set (ILDS) are updated by the reorganization and they are inconsistent with the image copy. The ILDS is not referred to if EPSCHK=NO is specified.

If IMAGECOPY is specified for the PHIDAM database, the primary index database is not checked, because an image copy cannot be created for the primary index database.

SCANGROUP=

Specify this parameter when you want to read in parallel the DATABASE data sets or image copy data sets. Specify a 1- or 2-digit number for *nn*, where *nn* is a number greater than 01. If you specify different numbers for the DATABASE statements, they are read in parallel. If you specify the same number for them, they are read sequentially.

In the following case, for example, HD Pointer Checker reads *dd1* and *dd2* in parallel:

Example 1

```
DATABASE DB=database1,DD=dd1,SCANGROUP=01
DATABASE DB=database1,DD=dd2,SCANGROUP=02
```

In the following case, for example, HD Pointer Checker reads *dd1* and *dd2* sequentially in SCANGROUP=01, and *dd3* and *dd4* sequentially in SCANGROUP=02, but processes SCANGROUP 01 and 02 in parallel:

Example 2

```
DATABASE DB=database1,DD=dd1,SCANGROUP=01
DATABASE DB=database1,DD=dd2,SCANGROUP=01
DATABASE DB=database2,DD=dd3,SCANGROUP=02
DATABASE DB=database2,DD=dd4,SCANGROUP=02
```

HD Pointer Checker decides, and so you cannot specify, the order of processing within a scan group (SCANGROUP).

A subtask is attached for each scan group. This subtask is called a scan task. The number specified for each scan group is also used as the suffix of the work data set of each scan task. The ddname of the MERGIN*nn* data set used by the scan task of SCANGROUP=2 would be MERGIN02.

To process an image copy on a tape, use the same scan group number for data sets on the same volume serial. The number of tape units must be more than or equal to the total number of scan groups. In the previous examples, you must prepare at least two tape units.

If you specify KEYSIN=YES, specify the same scan group number for data sets that have root segments.

You can specify the scan group number only if TYPE=ALL or TYPE=SCAN is specified. The maximum number you can specify for scan group is 99, and the default is 1. If you do not specify the scan group number, or if you specify the same number for all DATABASE statements, all data sets are read sequentially.

Running many scan tasks causes more resources to be used by HD Pointer Checker. For the details, read "Estimating the storage needed for HD Pointer Checker" on page 295.

BLOCKDUMP=

Specifies a request to print the maps, dumps, and the addresses of the database blocks. A maximum of 100 BLOCKDUMP parameters can be specified in the DATABASE statement. The format of the dump is specified by the DUMPFORM parameter in the OPTION statement.

When this option is specified, HISTORY=YES cannot be specified.

This option can be specified when TYPE=ALL or SCAN is specified. It is only effective for the HDAM, HIDAM, PHDAM, and PHIDAM databases. **If it is specified, the pointer is not checked for the specified database data set.**

Abbreviations **BDUMP** and **BLKDUMP** can be used for **BLOCKDUMP**.

rba

Specifies the relative byte address (RBA) of the first block to be dumped. The RBA of any byte within the block may be used. Code eight hexadecimal digits, with leading zeros, if necessary.

nnn

Specifies the number of consecutive blocks that are mapped, dumped, and started at the relative byte address specified by the RBA parameter. The maximum value is 999 and the minimum value is 1. The default value is 1.

For an RBA beyond 4 GB, you should specify a 32-bit odd value based on the over 4-GB RBA rule.

For example, to specify the hexadecimal value x'10000F000' as a 32-bit odd value, you should specify as follows:

```
BLOCKDUMP=(0000F001,001)
```

The value BDUMP=(0000F000) is equivalent to BLOCKDUMP=(0000F000,001).

DBALL=

Specifies whether you want to process all logically related databases to the one specified in the DB keyword. This option can be specified when TYPE=ALL or SCAN is specified in the PROC statement.

YES

All logically related databases and index databases are processed. However, the following restrictions apply:

- When HASH=YES is specified in the PROC statement, secondary index database and PSINDEX database are not processed.
- When the DBORG= keyword is specified, only databases of the specified organization are processed.

DB=*dbdname* must be specified. DATASET= and SCANGROUP= are available. PART=, NUM=, DD=, OVERFLOW=, PRIMEDB=, and BLOCKDUMP= are not effective even if they are specified.

NO

Only the databases that are specified in the DB keyword are processed. This is the default value.

When multiple DATABASE statements are specified, either DBALL=YES or DBALL=NO can be selected for each DATABASE statement. However, the following restrictions apply when databases that are associated with DBALL=YES are specified by another DATABASE statement.

- When DBALL=YES is specified by another statement, the first DBALL=YES in the logically related databases is effective and the subsequent DBALL=YES specifications are ignored.
- When DBALL=NO is specified by another statement, only database data sets that are specified with DBALL=NO are processed with the options specified in the DATABASE statements.
- Two or more DATABASE statements with same DBD name and DBALL=YES option will result in error.

How to specify multiple DATABASE statements for a HALDB

Multiple DATABASE statements specified for a HALDB are allowed, but there are some points to be careful.

- When multiple DATABASE statements are specified for a HALDB, you must specify the partition name by partition selection order. The partition selection order is the order returned from the partition selection.
- When HASH=NO and EPSCHK=YES are specified in the PROCCTL data set and a logical relationship or a PSINDEX database is defined in a HALDB, you cannot check the subset partitions of the HALDB.

  Because logical pointers and index pointers point to target segments across databases or partition boundaries, all partitions and databases must be checked at once. The examples presented in this section are for the HALDBs without logical relationships or PSINDEX databases.
- The same partition can be specified in different DATABASE statements. When you specify PART=*ALL or NUM=, however, make sure it is as follows:

<Example>

Error case

```
//PROCCTL DD *
    PROC TYPE=ALL
    DATABASE DB=HALDB1,PART=*ALL
    DATABASE DB=HALDB1,PART=HALDBP1
/*
```

This is an error; the specification for HALDBP1 on PART= is a duplicate.

```
//PROCCTL DD *
    PROC TYPE=ALL
    DATABASE DB=HALDB1,PART=HALDBP1,NUM=2
    DATABASE DB=HALDB1,PART=HALDBP2
/*
```

Suppose HALDB1 has two partitions in the following order: HALDBP1, HALDBP2. This is an error; the specification for HALDBP2 on PART= is a duplicate.

Normal case

```
//PROCCTL DD *
    PROC TYPE=ALL
    DATABASE DB=HALDB1,PART=HALDBP1,NUM=2
    DATABASE DB=HALDB1,PART=HALDBP4
/*
```

Suppose HALDB1 has four partitions in the following order: HALDBP1, HALDBP2, HALDBP3, HALDBP4. This is valid. The partition name HALDBP1, HALDBP2, and HALDBP4 are processed. The partition name HALDBP3 is skipped.

```
//PROCCTL DD *
    PROC TYPE=ALL
    DATABASE DB=HALDB1,PART=HALDBP1,DD=(A,B,C)
    DATABASE DB=HALDB1,PART=HALDBP2,DD=*ALL
/*
```

Suppose HALDB1 has two partitions in the following order: HALDBP1, HALDBP2. This is valid. The data set group A, B, and C of HALDBP1 are processed. All data set groups in HALDBP2 are processed.

```
//PROCCTL DD *
    PROC TYPE=ALL
    DATABASE DB=HALDB1,PART=HALDBP1,DD=(A,B),SCANGROUP=01
    DATABASE DB=HALDB1,PART=HALDBP1,DD=C,SCANGROUP=02
```

To read the data set groups in parallel, the same partition name is specified in two DATABASE statements. Each data set group can be specified in only one statement.

OPTION statement

An OPTION statement can be specified following a PROC statement or a DATABASE statement.

Explicitly specified option parameters and default option values in an OPTION statement that follows a PROC statement specify the options for the entire database to be processed.

Explicitly specified option parameters in an OPTION statement that follows a DATABASE statement override a previously specified option of an OPTION statement that follows a PROC statement. These override options affect only one database data set group in a preceding DATABASE statement.

The OPTION statement contains the keywords shown in Figure 13.

```
OPTION    ┌                                                                    ┐
          │  ERRLIMIT=YES|NO                                                   │
          │ ,DIAGDUMP=NO|ERROR|FIRST100                                        │
          │ ,DSSIZE={nnnn(.n)S}                                                │
          │ ,DUMPFORM=FORMAT|UNFORMAT                                          │
          │ ,HISTORY=YES|NO                                                    │
          │ ,HOMECHK=(YES,-mmm,[+]nnn)|NO                                      │
          │ ,ICUNIT=unit name                                                  │
          │ ,INCORE=YES|NO                                                     │
          │ ,INTERVAL=DATASET|BITMAP|(BLOCK,nn)                                │
          │ ,VSAMBF=nnnnn                                                      │
          │ ,IBUFF=nnnn|0                                                      │
          │ ,KEYSIN=YES|NO                                                     │
          │ ,T2CHK=(T2num,T2len)|(0,7)                                         │
          │ ,ZEROCTR=YES|NO                                                    │
          │ ,DIAG=YES|NO                                                       │
          │ ,PRINTDATA=YES|NO                                                  │
          │ ,NOCHKP={HF,HB,PTF,PTB,PP,LTF,LTB,LP,LC,PC,RAP,VLS}                │
          │ ,SPIXCHK=YES|NO                                                    │
          │ ,PTRCHK=YES|NO                                                     │
          │ ,SPMN=YES|NO                                                       │
          │ ,THRESHOLDS=keyword=value(keyword1=value1,keyword2=value2,...)     │
          └                                                                    ┘
```

Figure 13. OPTION statement syntax

ERRLIMIT=

Specifies whether you want to limit the number of database error messages that will be printed during the SCAN or CHECK processes. It is only effective for the HDAM, HIDAM, PHDAM, and PHIDAM databases.

This option can be specified when TYPE=ALL, SCAN, or CHECK is specified in the PROC statement. If this option is specified with TYPE=SCAN, this option is passed to the CHECK process for the database data set that will run as the another job. If this option is specified with TYPE=CHECK, this option is effective during the CHECK process.

YES

The number of printed database error messages is limited to 100 in each of the SCAN process and CHECK process. This is the default value.

NO

All database error messages are printed.

DIAGDUMP=

Specifies whether you want to print the block maps and dumps for the database blocks.

The format of the dump is specified by the DUMPFORM parameter of the OPTION statement.

This option can be specified with any TYPE= specification. It is only effective for the HDAM and HIDAM databases.

Abbreviations **DDUMP** for **DIAGDUMP**, and **F100** for **FIRST100** can be used.

NO

No block map and dump is printed for the database data set.

ERROR

A block map and dump for a database data set block having an error is printed.

A maximum of 100 maps and dumps reports will be printed for each database data set through the HD Pointer Checker run. If ERRLIMIT=YES is specified and the total number of error messages for the data set reaches 100, the block map and dump function for the data set is suppressed, even if the number of maps and dumps reports does not reach to 100. This is the default value.

FIRST100

Block maps and dumps of the first 100 blocks, except the first IMS control block and bit-map blocks for the database data set, are printed unconditionally. This option is only effective with TYPE=ALL and SCAN. If this option is specified with TYPE=CHECK or BLKMAP, ERROR parameter is used to override for this parameter.

DSSIZE=

This parameter is used in conjunction with dynamic space allocation for the work data sets.

If the OPTION statement follows the DATABASE statement, specify the total of the database data set sizes in the DATABASE statement.

If the OPTION statement follows the PROC statement, specify the maximum size for database data sets of all sizes.

If HDAMDD1 is 1 GB and HDAMDD2 is 2 GB, specify as follows:

Example 1:

```
PROC TYPE=ALL
DATABASE DB=HDAM1,DD=(HDAMDD1,HDAMDD2)
OPTION DSSIZE=3G
```

If all of the following is true, specify as shown in Example 2.

- HDAMDD1 is less than 2G
- the sum of HIDAMDD1 and HIDAMDD2 is 2G
- the sum of the data sets of HALDB1 is less than 2G

Example 2

```
PROC TYPE=ALL
OPTION DSSIZE=2G
DATABASE DB=HDAM1,DD=HDAMDD1
DATABASE DB=HIDAM1,DD=(HIDAMDD1,HIDAMDD2)
DATABASE DB=HALDB1,PART=*ALL,DD=*ALL
```

CAUTION:
In example 2, the sizes of HDAM1 and HALDB1 are assumed to be 2G each. Therefore, the IMS HP Pointer Checker prepares the work data sets for a total size of 6G. This may waste too much DASD space. We recommend that you specify the DSSIZE values after each DATABASE statement.

The override convention for the DSSIZE parameter follows the established protocol for the parameters in the OPTIONS statement. The parameters in an OPTIONS statement which follows a DATABASE statement override the parameters in any OPTIONS statement which may follow the PROC statement.

If the input data set is a tape Image Copy, specify the original database data set size for the DSSIZE parameter.

If no DSSIZE parameter is provided and the database data set or image copy data set is on DASD, HD Pointer Checker estimates the space requirements for work data sets on the basis of the database data set or image copy data set size found in the VSAM catalog or in the DSCB in the VTOC. If the image copy data set is on tape, the data set labels contains no size information. Then for a tape data set HD Pointer Checker uses one of the following as the default size:

- If the input data set is a tape Image Copy of the index database data set, the default size is 1 GB.
- If the input data set is a tape Image Copy of data set other than the index database data set, the default size is 4 GB.

nnnn **or** *nnnn.n*

The size can be specified either as a 1-4 digit integer value or as a decimal value with a maximum of one decimal position.

The maximum value of *nnnn.n* is dependent on the scale factor.

- For **K**, **M**, or **G** scales, the maximum value is 9999.9
- For **X** scales, the maximum value is 9.9

S The scale value can be represented by one of the following characters:

K Kilobytes: The size value is 1000 times *nnnn.n*

M Megabytes: The size value is 1,000,000 times *nnnn.n*

G Gigabytes: The size value is 1,000,000,000 times *n.n*

X Times: The size value is *n.t* times the default or calculated value.

Use the **X** option when the dynamic allocation of a data set is too small, resulting in frequent reallocation and restart.

The size of the work data set is adjusted by use of the primary quantity and the secondary quantity for allocation. For each of these, specify the value one-tenth the size of the estimated total. For example, if DSSIZE is 4 GB, the primary quantity has the ability to process 400 MB. The work data set will reach the estimated size of 4 GB after allocation is done ten times.

The size of each primary and secondary quantity is given in an FABP1101I message in the PROCCTL Statement report or the DMB Directory report.

The SORTWK*nn*, SRTXWK*nn*, and SRTEWK*nn* data sets are allocated dynamically by the DFSORT program, and so the size of these work data sets are not controlled by the DSSIZE parameter.

DUMPFORM=

Specifies the dump format you want to print block dumps with.

This option can be specified when DIAGDUMP=ERROR or FIRST100 in the OPTION statement or the BLOCKDUMP parameter is specified in the DATABASE statement. This option can be specified with any TYPE= specification.

Abbreviations **DF**, **DUMPF**, and **DFORM** for **DUMPFORM**, **UF** and **UNFMT** for **UNFORMAT**, and **F** and **FMT** for **FORMAT** can be used.

FORMAT
Specifies the formatted dumps to be printed. This is the default value.

UNFORMAT
Specifies the unformatted dumps to be printed.

HISTORY=
Specifies whether you want to update the HISTORY data set for the database data sets to be analyzed. When HISTORY=YES is specified, the HISTORY data set is required.

If the multiple entries option of the HISTORY data set is active, one or more database data set records are recorded per day. For more detail information about the multiple entries option, see “MULTIENT=” on page 376.

This option can be specified when TYPE=ALL or SCAN is specified.

This option cannot be specified when the following parameters are effective.

- On the DATABASE statement:
  - BLOCKDUMP parameter
- On the OPTION statement:
  - HOMECHK=NO
  - INCORE=NO
  - NOCHKP parameter

An abbreviation **HIST** can be used for **HISTORY**.

YES
The HISTORY data set for the database data sets to be analyzed is updated.

NO
The HISTORY data set is not updated. This is the default value.

Serialization of HISTORY data sets:

The HD Pointer Checker jobs are serialized to update the HISTORY data sets by using the ENQ macro. The RNAME parameter of the macro depends on the HISTORY lock option. The HISTORY lock option is set by the HISTLOCK parameter of the FABGHIST program of DB Historical Data Analyzer.

For details of the HISTLOCK option, see “HISTLOCK=” on page 378.

HOMECHK=
Specifies whether you want to print the DISTRIBUTION OF ROOT SEGMENTS part in the DB Record Distribution Statistics report. This option can be specified when TYPE=ALL or SCAN is specified. It is printed for HDAM or PHDAM, and only for the data set group that contains root segments. This option is effective when REPORT DBDIST=YES is specified (default value of DBDIST=YES).

When HOMECHK=NO is specified, HISTORY=YES cannot be specified.

(YES,*-mmm***,***+nnn***)**
The DISTRIBUTION OF ROOT SEGMENTS is printed. Yes is the default value. If Yes, *mmm* is the backward distance from Home Block (1-999) and

nnn is the forward distance from Home Block (1-999). The sign "+" can be omitted. The default value is (YES,-10,+10).

NO

The DISTRIBUTION OF ROOT SEGMENTS is not printed.

ICUNIT=

Specifies the name of the unit in which the image copy data set resides, if the uncataloged image copy data set is to be allocated dynamically. This option can be specified when TYPE=ALL or SCAN is specified.

This option is referred to only when ddname DD of the image copy data set is omitted from the JCL, and NOCATDS is specified in the RECON data set.

For details, refer to “Dynamic allocation for database data set and image copy data set” on page 69.

INCORE=

Specifies whether you use the in-core pointer checking where possible, or you postpone the pointer checking until the CHECK process is executed. With the in-core checking option, pointers are checked as many times as possible in the SCAN process. This minimizes the CPU, I/O, and the run time used by HD Pointer Checker.

The pointers that are not checked by using the in-core checking are checked in the CHECK process. The CHECKREC data set used input for CHECK process and the BLOCKMAP process does not contain the sorted records for the pointers that were checked with the in-core checking.

If this run finds some unexpected pointer errors, the listing produced by the BLOCKMAP process might be incomplete.

If the BLOCKMAP processor runs as a separate job or a job step that uses the BLKMAPIN and CHECKREC data set as input, the pointers that were checked with the in-core checking option are not listed.

This option can be specified when TYPE=ALL or SCAN is specified. It is effective for HDAM, HIDAM, PHDAM, or PHIDAM databases. When the HASH option is effective, INCORE=NO is forced.

YES

The in-core pointer checking is used where possible. This is the default value.

NO

The in-core pointer checking is not used.

INTERVAL=

Specifies whether you want to define the interval at which the Interval Statistics report and the Interval Free Space Summary report are produced.

This option can be specified when TYPE=ALL or SCAN is specified on the PROC statement, and when INTFS=YES or INTST=YES is specified on the REPORT statement. It is effective only for HDAM, HIDAM, PHDAM, or PHIDAM databases.

Abbreviations **DS** for **DATASET**, **BM**, **BITM**, and **BMAP** for **BITMAP**, and **BLK** for **BLOCK** can be used.

DATASET

The reports are produced for the entire database data set only once. This is the default value.

BITMAP

The reports are produced each time a bit-map block is processed.

(BLOCK,*nn*)

The number times 100 is the number of blocks that is processed between statistics interval. To code this field, you must include two decimal digits. Use leading zeros, if necessary.

VSAMBF=

Specifies the number of I/O buffers that VSAM is to use for the data records of the VSAM database data sets. The maximum value is 99999, and the minimum is 1.

If no AMP parameter or IBUFF keyword is specified in a DD statement, the buffers specified with this operand apply to all the VSAM database data sets. For a VSAM database, those buffers are obtained when the data set is opened and freed when it is closed. The total number of buffers depends on the block size of the VSAM database data set. If you do not specify this parameter or the AMP parameter, HD Pointer Checker uses the IBUFF keyword by default. The AMP parameter is substituted for the IBUFF and the VSAMBF keyword. If both VSAMBF and IBUFF keywords are specified, the value of IBUFF keyword is used. This option can be used only on the TYPE=ALL and SCAN parameters and is effective only for a VSAM database data set.

You should be careful when specifying this parameter, since:

- The region size will increase when buffer space is specified
- An excessive number of buffers may cause an open error for the database data set.

IBUFF=

Specifies the size, in kilobytes, of the I/O buffer area for an input database data set or image copy data set. The maximum value is 9999, and the minimum is 0. However, the maximum number of buffers in a non-VSAM data set is 255. Thus the buffer sizes can actually range up to 255 times the size of a block.

The default size for OSAM, VSAM, and the image copy data set on DASD is an equivalent size of 10 tracks on DASD space. The default size for an image copy data set on tape is 640 KB. If 0 is specified, or no value is specified, the default is assumed. If BUFNO in DCB, or BUFND in AMP, is specified on the database DD statement in the JCL, the IBUFF option is ignored.

For an Indirect List Data Set (ILDS) of HALDB, the IBUFF option does not apply. HD Pointer Checker uses 100 buffers for an ILDS unless BUFND in AMP is specified on the JCL DD statement.

This option can be used only on the TYPE=ALL and SCAN parameters.

KEYSIN=

Specifies whether you want to write the KEYSIN data set. This data set is used as the input to HD Tuning Aid.

This option can be specified when TYPE=ALL or SCAN is specified. It is effective only for HDAM, HIDAM, PHDAM, or PHIDAM databases.

YES

The KEYSIN data set is generated.

NO

The KEYSIN data set is not generated. This is the default value.

T2CHK=

Specifies the two threshold values that will define how the slack bytes or unknown data is treated as T2 records.

This option can be specified when TYPE=ALL or SCAN is specified.

T2num

Specifies the maximum number of T2 records (whose length is more than T2len) that are ignored (suppressed) and not considered to be errors. If the number of generated T2 records exceeds the threshold value, all T2 records are considered to be errors. The maximum value is 99 and the minimum value is 0. The default value is 0.

This option is effective only for HISAM, HDAM, HIDAM, PHDAM, or PHIDAM databases.

T2len

Specifies the maximum length of the T2 which is not considered to be an error. The maximum value is 99, and the minimum value is 1. The default value is 7.

This option is effective only for HDAM and HIDAM databases. The value specified for the HISAM database does not cause an error, but is ineffective.

For example: T2CHK=(,3) is equivalent to T2CHK=(0,3). T2CHK=(10,) is equivalent to T2CHK=(10,7). T2CHK=(10) is equivalent to T2CHK=(10,7).

SPIXCHK=

Specifies whether you want to check the suppressed segment by use of the secondary index maintenance exit routine. This option can be specified when TYPE=ALL or SCAN is specified. It is effective only for the sparse indexing database when the secondary index maintenance exit routine is used.

YES

Suppressed segment check performs using a secondary index maintenance Exit routine. This is the default value.

NO

Suppressed segment check is not performed.

ZEROCTR=

Specifies request to report the segment which has a counter field with zero value.

You can specify this option when you have also specified TYPE=ALL or SCAN. This option is effective only for HDAM, HIDAM, PHDAM, or PHIDAM databases. This option is not effective when you have also specified HASH=YES or HASH=FORCE on the PROC statement.

YES

The segment with zero counter field is reported to the Evaluation of All Pointers to the Same Target report. This option may result in an extremely large report.

NO

The segment with zero counter field is not reported. This is the default value.

PTRCHK=

Specifies whether you want to check the pointers. This option can be specified when TYPE=SCAN is specified.

YES
The pointers are checked. This is the default value.

NO
The pointers are not checked. Only database statistics reports are printed.

When INCORE=YES is specified, the incore pointer checking is done even if PTRCHK=NO is specified. The default value of the INCORE option is YES. If you want to suppress the incore pointer checking, specify not only PTRCHK=NO but also INCORE=NO.

DIAG=, PRINTDATA=, NOCHKP=
These options must be used for debugging purpose only. For details, see Chapter 14, “HD Pointer Checker options for debugging,” on page 299.

SPMN=
Specifies whether HDPC calls Space Monitor.

YES
HDPC calls Space Monitor. Space Monitor monitors the data sets of the databases that are specified by the associated DATABASE statements.

NO
HDPC does not call Space Monitor.

THRESHOLDS=
Specifies the threshold values for Space Monitor. You can specify the following keyword and its value as a pair. When you specify more than two pairs, separate each pair with a comma and enclose the entire specification of the pairs with parentheses.

Table 9. Keywords and values for the THRESHOLDS parameter

Keyword	Value	Default value	Description
EXTENTS=	0-99 or "NO"	10	Specifies the warning threshold value for the number of extents.
FREESPC%=	0-100 or "NO"	10	Specifies the warning threshold value for the percentage of free space.
AVAILEXT=	0-50 or "NO"	10	Specifies the warning threshold value for the number of available extents.
LASTEXT=	"YES" or "NO"	"YES"	If YES is specified and the data set uses the last extent, a warning message for this data set is shown in the Space Monitor Exception report.
USEDSPC%=	0-100 or "NO"	90	Specifies the warning threshold value for the percentage of space used.
VOLEXT=	"YES" or "NO"	"YES"	If YES is specified and not enough space is left on the DASD volume for the data set to extend, a warning message for this data set is shown in the Space Monitor Exception report.
CASPLIT%=	0-100 or "NO"	50	Specifies the warning threshold value for the percentage of CA splits.
CISPLIT%=	0-100 or "NO"	50	Specifies the warning threshold value for the percentage of CI splits.

Table 9. Keywords and values for the THRESHOLDS parameter (continued)

Keyword	Value	Default value	Description
REORGINTVL=	0-999 or "NO"	"NO"	Specifies the warning threshold value for the number of days that can pass without a database reorganization.
DSSIZE%=	0-100 or "NO"	90	Specifies the warning threshold value for the percentage of the space used by data sets within the maximum size.
DSSIZE=	0-9999 or "NO"	"NO"	Specifies the warning threshold value for the data set size in the units of MB.

REPORT statement

A REPORT statement can be specified following a PROC statement and a DATABASE statement.

Explicitly specified option parameters and default option values of a REPORT statement that follows a PROC statement specify the options for all databases to be processed.

Explicitly specified option parameters of a REPORT statement following a DATABASE statement override a previously specified option of a REPORT statement following a PROC statement. These override options affect only one database data set group in a DATABASE statement.

REPORT statement contains the keywords shown in Figure 14.

```
REPORT    ┌                         ┐
          │  RUNTM=YES|NO           │
          │ ,INTST=YES|NO           │
          │ ,BITMAP=YES|NO          │
          │ ,FSEMAP=YES|NO          │
          │ ,MAXFSD=YES|NO          │
          │ ,INTFS=YES|NO           │
          │ ,DBDIST=YES|NO          │
          │ ,CHAINDIST=YES|NO       │
          │ ,DECODEDBD=YES|NO       │
          │ ,MAPDBD=YES|NO          │
          │ ,COMPFACT=YES|NO        │
          │ ,SEGIO=YES|NO           │
          └                         ┘
```

Figure 14. REPORT statement syntax

RUNTM=

Specifies whether you want to generate the separator page for DB/DSG reports with Run Time Option.

This option can be specified when TYPE=ALL or SCAN is specified.

YES

The report is generated. This is the default value.

NO

The report is not generated.

INTST=

Specifies whether you want to generate the Interval Statistics report for the HDAM, HIDAM, PHDAM, or PHIDAM database. The report is produced each time an interval is processed; the information in the report is added to the next report. That is, the Nth report provides the total information of the 1st-Nth reports.

This option can be specified when TYPE=ALL or SCAN is specified.

YES

The report is generated. This is the default value.

NO

The report is not generated.

BITMAP=

Specifies whether you want to generate the Bit Map Display report for the HDAM, HIDAM, PHDAM, or PHIDAM database.

This option can be specified when TYPE=ALL or SCAN is specified.

YES

The report is generated. This is the default value.

NO

The report is not generated.

FSEMAP=

Specifies whether you want to generate the Free Space Map report for the HDAM, HIDAM, PHDAM, or PHIDAM database.

This option can be specified when TYPE=ALL or SCAN is specified.

YES

The report is generated. This is the default value.

NO

The report is not generated.

MAXFSD=

Specifies whether you want to generate the Maximum Free Space Distribution report for the HDAM, HIDAM, PHDAM, or PHIDAM database.

This option can be specified when TYPE=ALL or SCAN is specified.

YES

The report is generated. This is the default value.

NO

The report is not generated.

INTFS=

Specifies whether you want to generate the Interval Free Space Summary report for the HDAM, HIDAM, PHDAM, or PHIDAM database. The report is produced each time an interval is processed and the information in the report is added to the next report. That is, the Nth report provides the total information of the 1st-Nth reports.

This option can be specified when TYPE=ALL or SCAN is specified.

YES

The report is generated. This is the default value.

NO

The report is not generated.

DBDIST=

Specifies whether you want to generate the DB Record Distribution Statistics report for the HDAM, HIDAM, PHDAM, or PHIDAM database.

This option can be specified when TYPE=ALL or SCAN is specified.

YES

The report is generated. This is the default value.

NO

The report is not generated.

CHAINDIST=

Specifies whether you want to print the DISTRIBUTION OF RAP CHAIN LENGTHS part in the DB Record Distribution Statistics report. This option can be specified when TYPE=ALL or SCAN is specified. It is printed for HDAM or PHDAM, and only for the data set group that contains root segments. This option is effective when REPORT DBDIST=YES is specified (default value of DBDIST is DBDIST=YES).

YES
The DISTRIBUTION OF RAP CHAIN LENGTHS is printed. Yes is the default value.

NO
The DISTRIBUTION OF RAP CHAIN LENGTHS is not printed.

DECODEDBD=
Specifies whether you want to print a DBD source code for each database processed by HD Pointer Checker. If you specify DECODEDBD=YES for the REPORT statement under the PROC statement, decoded DBDs are generated for all databases that are processed. If you specify DECODEDBD=YES for the REPORT statement under the DATABASE statement, a decoded DBD is generated for the specified database.

To use this function, you must install IMS Library Integrity Utilities, Version 1 and specify the library of IMS Library Integrity Utilities in STEPLIB.

YES
The decoded DBD is generated in a report.

NO
The decoded DBD is not generated. This is the default.

If HD Pointer Checker runs in a DBB region, IMS Library Integrity Utilities obtains DBD information from IMS.ACBLIB specified in the IMSACB DD statement.

MAPDBD=
Specifies whether you want to print a DBD map to a report for each database processed by HD Pointer Checker. To use this function, you must install IMS Library Integrity Utilities, and specify its library in STEPLIB.

YES
DBD map is generated in a report.

NO
DBD map is not generated. This is the default.

If HD Pointer Checker runs in a DBB region, IMS Library Integrity Utilities obtains DBD information from IMS.ACBLIB specified in the IMSACB DD statement.

COMPFACT=
Specifies whether you want to print a compression factor in the VL SEGMENT LENGTH STATISTICS part of the Partition Statistics report and the Database Statistics report.

This option can be specified when TYPE=ALL or SCAN is specified.

YES
The compression factor is generated.

NO
The compression factor is not generated. This is the default value.

The compression factor is generated, not for every data set group, but for every database. If COMPFACT=YES and NO is specified for the same database in different DATABASE statements, YES is taken.

The compression factor is printed for HISAM, HDAM, HIDAM, PHDAM, and PHIDAM databases. If COMPFACT=YES is specified to the other database than the above, it is ignored.

SEGIO=

Specifies whether you want to generate a RATE OF SEGMENT I/O OCCURRENCE part of the Partition Statistics report and the Database Statistics report.

This option can be specified when TYPE=ALL or SCAN is specified.

YES

The RATE OF SEGMENT I/O OCCURRENCE part is generated.

NO

The RATE OF SEGMENT I/O OCCURRENCE part is not generated. This is the default value.

The RATE OF SEGMENT I/O OCCURRENCE part is generated, not for every data set group, but for every database. If SEGIO=YES and NO is specified for the same database in different DATABASE statements, YES is taken.

The RATE OF SEGMENT I/O OCCURRENCE part is generated for HDAM, HIDAM, PHDAM, and PHIDAM databases. If SEGIO=YES is specified to the other database than the above, it is ignored.

END statement

The END statement can be specified in any order in the PROCCTL data set. The END statement is used to discontinue reading the PROCCTL data set. Any control statements and comments that follow the END statement are ignored in the PROCCTL data set.

It is not necessary to specify the END statement, unless you want to specify the end of the PROCCTL data set explicitly. The END statement is the optional statement which does not have any parameter.

Summary of index database checking

HD Pointer Checker checks several items regarding primary and secondary indexes. What you can check varies depending on how it is run and what you specify on the control statement.

Table 10 through Table 13 on page 103 show what you can check for each index database type and how to specify it on the control statement. ″Yes″ shows that the check is available, and how to specify it is shown in brackets. Even if ″Yes″ is specified, there are restrictions, which are described on page 103.

For details on how to specify the PROCCTL control statement, see each part in “FABPMAIN PROCCTL data set” on page 71.

For details on how to specify the ICEIN control statement when running HD Pointer Checker on IMS HP Image Copy or on IMS Parallel Reorganization, see the User’s Guides of each product.

For running these functions, a segment/edit compression routine and a secondary index database maintenance exit routine are called if they are defined in the DBD. Specify the data set that contains the load modules of the exits in the IMS2 or the STEPLIB DD statements.

Table 10. HIDAM primary index

	Function		**HD Pointer Checker processor (FABPMAIN)**		**Single Step HASH Check Option run from IMS HPIC processor (FABJMAIN) V3 or later, or IMS PR V3**
			PROCCTL PROC HASH=NO	**PROCCTL PROC HASH=YES**	**ICEIN HDPC=Y**
1-1	Basic function	• Check the number of root segments and index pointer segments. • Validate the pointer values (RBA of the root segment) of the index pointer segments.	Yes (DATABASE statement of the index database) (Required)	Yes (DATABASE statement of the index database)	Yes (Specify the index database in ICEIN)
1-2	Additional function	**Index Key Check** Validate the key values of the index pointer segments.	Yes (PROC IXKEYCHK= YES) *3	YES (PROC IXKEYCHK= YES) *10	YES (GLOBAL IXKEYCHK= YES) *10

Note: See Notes for Table 10 to Table 13 on page 103 on page 103.

Table 11. Secondary index for non-HALDB

	Function		**HD Pointer Checker processor (FABPMAIN)**		**Single Step HASH Check Option run from IMS HPIC processor (FABJMAIN) V3 or later, or IMS PR V3**
			PROCCTL PROC HASH=NO	**PROCCTL PROC HASH=YES**	**ICEIN HDPC=Y**
2-1	Basic function	• Check the number of index source segments and index pointer segments. • Validate the pointer values (RBA of the index target segment) of the index pointer segments.	Yes (DATABASE statement of the index database) *4 *6 *9	YES (DATABASE statement of the index database)*1 *5 *6	Yes (Specify the index database in ICEIN) *1 *2 *5
2-2	Additional function	**Index Key Check** Validate the key values of the index pointer segments.	Yes (PROC IXKEYCHK= YES) *3 *7	YES (PROC IXKEYCHK= YES)*7 *10	YES (GLOBAL IXKEYCHK= YES) *10
2-3		**Symbolic Pointer Check** Validate the symbolic index pointers. (Only when the index target segment is the root segment)	Yes (PROC SYMIXCHK= YES)	No	No
2-4		**Suppressed Segment Check** Check that the index pointer segments are suppressed correctly by using the secondary index maintenance exit routine.	Yes (OPTION SPIXCHK= YES)	No	Yes (This check is done with the 2-1 check)

Note: See Notes for Table 10 on page 100 to Table 13 on page 103 on page 103.

Table 12. Primary index for HALDB PHIDAM

	Function		**HD Pointer Checker processor (FABPMAIN)**				**Single Step HASH Check Option run from IMS HPIC processor (FABJMAIN) V3 or later, or IMS PR V3**
			PROCCTL PROC HASH=NO DATABASE DATASET= REAL	**PROCCTL PROC HASH=NO DATABASE DATASET= IMAGECOPY**	**PROCCTL PROC HASH=YES DATABASE DATASET= REAL**	**PROCCTL PROC HASH= YES DATABASE DATASET= IMAGECOPY**	**ICEIN HDPC=Y**
3-1	Basic function	• Check the number of root segments and index pointer segments. • Validate the pointer values (RBA of the root segment) of the index pointer segments.	Yes (DATABASE statement of the index database) (Required)	No (Will be ignored even if the DATABASE statement of the index database is specified.)	Yes (DATABASE statement of the index database) (Required)	No (Will be ignored even if specified.)	No
3-2	Additional function	**Index Key Check** Validate the key values of the index pointer segments.	Yes (PROC IXKEYCHK= YES) [*3]	No	YES (PROC IXKEYCHK= YES)[*10]	No	No

Note: See Notes for Table 10 on page 100 to Table 13 on page 103 on page 103.

Table 13. Secondary index for HALDB (PSINDEX)

	Function		**HD Pointer Checker processor (FABPMAIN)**		**Single Step HASH Check Option run from IMS HPIC processor (FABJMAIN) V3 or IMS PR V3**
			PROCCTL PROC HASH=NO	**PROCCTL PROC HASH=YES**	**ICEIN HDPC=Y**
4-1	Basic function	• Check the number of index source segments and index pointer segments. • Validate the pointer values of the index pointer segments.	Yes (DATABASE statement of the index database) [*4] [*6]	No	No
4-2	Additional function	**Index Key Check** Validate the key values of the index pointer segments.	Yes (PROC IXKEYCHK= YES)[*3] [*7]	No	No
4-3		**EPS Check** Validate whether the relationship among the index target segments, the index list entries (ILE) in the index list data set (ILDS), and the index pointer segments are correct.	Yes (PROC EPSCHK= YES)[*8]	No	No
4-4		**Suppressed Segment Check** Check that the index pointer segments are suppressed correctly by using the secondary index maintenance exit routine.	Yes (OPTION SPIXCHK= YES)	No	No

Note: See Notes for Table 10 on page 100 to Table 13 on page 103.

Notes for Table 10 on page 100 to Table 13

***1** RBA check is possible only when index source segment and index target segment are the in the same segment, or when the index target segment is the parent segment of index source segment. However, if index target segment is the parent segment of index source segment, and the index source segment does not have a PP pointer, RBA check is not done.

***2** If segment edit/compression exit routine is defined on the index source segments, the HASH check is not done for the index database.

***3, *4, and *5**

If the index source segment is variable length or the segment edit/compression exit routine is defined, a segment split might occur while the database is updated. When there is no split segment in the database, the checks for each case are done, but if a split segment is created, the checks are disabled.

When a split segment is found, HD Pointer Checker prints the following messages for each case, and returns with a return code of 00.

***3** If there is a split segment in any of the index source segments, the Index Key Check will not be done for any index pointer segments, and message FABP2083W is issued.

***4** If there is a segment that is split and physically deleted in any of the index source segments, and if there is a segment whose index is suppressed in any of the index source segments, the number of index pointer segments cannot be checked, and message FABP0324I is issued.

***5** For variable-length segment, if there is a segment that is split and physically deleted in any of the index source segments, and if there is a segment whose index is suppressed in any of the index source segments, the HASH check for the index database cannot be done, and message FABP4025W is issued.

***6 and *7**
When the secondary index database maintenance exit is defined on the index source segment, there are the following restrictions:

***6** Check is done only when SPIXCHK=YES is also specified.

***7** Index Key Check is done only when SPIXCHK=YES is also specified.

***8** If Image Copy is the input, specify EPSCHK=NO because the contents of ILDS will be changed by the HALDB reorganization after taking the image copy.

If the HALDB reorganization was not done after taking an image copy, EPS Check will become effective. HD Pointer Checker does not check whether the HALDB was reorganized or not after taking an image copy. Therefore, select EPSCHK=YES or NO depending on whether reorganization was done. Note that the default value is EPSCHK=YES. To disable EPS Check, specify EPSCHK=NO explicitly.

***9** If symbolic index pointer is used, the RBA of the index target segment is not validated.

***10** If the following index is used, HASH check of index key is not done.

- A secondary index that cannot be checked due to restriction.
- A /CK field is specified on the SUBSEQ= operand of the XDFLD statement.
- Some of the source segments are split to prefix and data portions and physically deleted.

FABPMAIN BLKMAPIN data set

This section explains the FABPMAIN BLKMAPIN data set.

Function

The BLKMAPIN data set contains your description of the processing to be done by the BLOCKMAP processor. It contains the target relative byte addresses (RBAs). The BLOCKMAP processor prints a list of all pointers that point to each target RBA.

The target RBAs specified can be the RBAs of segments that have undamaged as well as damaged pointers.

Usually, sufficient information about pointer chaining for a pointer error is automatically produced during an HD Pointer Checker run in the TYPE=ALL or TYPE=CHECK process. If you need information about the pointer chains of other segments, use the BLKMAPIN data set and the CHECKREC data set as the input in TYPE=BLKMAP- a stand-alone run of the BLOCKMAP process.

To list pointers that are free of error, you must include as input the CHECKREC data set created by specifying CHECKREC=YES in the TYPE=ALL or TYPE=SCAN process. Also be careful about the following:

- When IN-CORE pointer checking is done, all pointers validated in memory are discarded from the CHECKREC data set and they are not printed in the BLOCKMAP process.
- When HASH pointer checking is done, pointer information is not generated in the CHECKREC data set or printed in the BLOCKMAP process.

If you need to see all the pointers, rerun the pointer checker with no HASH pointer checking or no IN-CORE pointer checking.

Format

This control data set usually resides in the input stream. However, it can also be defined as a sequential data set or as a member of a partitioned data set. It must contain one 80-byte fixed-length record for each target RBA to be processed. BLKSIZE, if coded, must be a multiple of 80. The BLKMAPIN data set can be coded as shown in Figure 15.

```
//BLKMAPIN DD *
00100001 A0003FC7D
00100001 B00600CD8
002     0100000400
002     0200000812
/*
1  4    9 11
```

Figure 15. Format of the BLKMAPIN data set

Record format

There is only one record type in the BLKMAPIN data set.

Position	Description
1	The 3-digit field contains the database number. The field contains hexadecimal digits.
4	If HALDB, the 5-digit field contains the partition ID. The field contains decimal digits. If non-HALDB, this field must be blank.
9	If non-HALDB, the 2-digit field contains data set group number. The field contains hexadecimal digit. If HALDB, the 1-digit field contains data set group number contains a character.
11	This 8-digit field contains the target RBA. This field contains eight hexadecimal digits with leading zeros (if needed).

FABPMAIN HPSRETCD data set

This section explains the function, format, and the control statements of the HPSRETCD PROCCTL data set.

Function

The HPSRETCD data set contains your specification for the return code that is returned by HD Pointer Checker.

Format

This control data set usually resides in the input stream. However, it can also be defined as a sequential data set or as a member of a partitioned data set. It must contain 80-byte, fixed-length records. If the optional BLKSIZE is coded, it must be a multiple of 80.

The HPSRETCD data set should contain (HDPC) statements. These control statements can be coded as shown in Figure 16.

```
//HPSRETCD  DD  *
(HDPC)
  T2ERROR=12,DBERROR=16,
  PROCERROR=20
```

Figure 16. Sample control statement format in the HPSRETCD data set

Note: On the control statement, use uppercase alphabetical characters, numerical characters, and the following special characters:

asterisk	*
comma	,
equal sign	=
parenthesis	( )

Control statement syntax

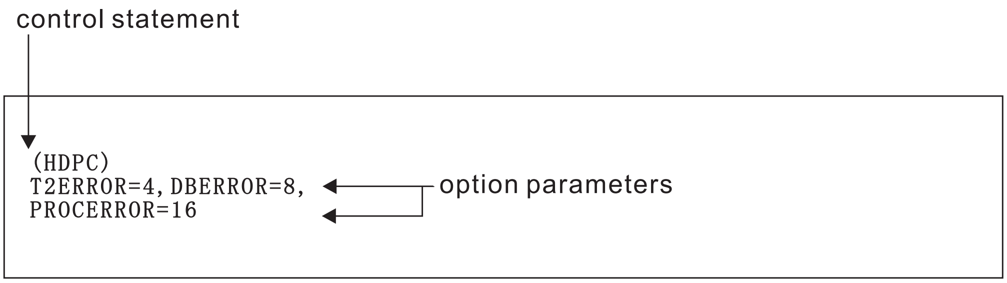
The following describes the coding conventions that you must follow in writing control statements in the HPSRETCD data set:

- You must code an (HDPC) control statement in the first line of the HPSRETCD data set, and you must code optional parameters must on the second or subsequent lines.

- You must code the (HDPC) control statement and option parameters within columns 2 and 72.
- When you code multiple option parameters, they must be separated by commas.

  You are not allowed to place blanks between the option parameters and the commas, or within the option parameters. You can continue option parameters onto one or more of the following control statement records.
- Option parameters are not positional; you can specify them in any order of sequence. You cannot specify a null value for any option parameter.
- You can code comments after the last option parameter on each control statement record separated by at least one blank.
- You must code a comment line to begin with an asterisk in column 1.
- The only control statement name that you can use within parentheses is HDPC, as in (HDPC). If (HPIC) is specified, HD Pointer Checker recognizes that the statement is for IMS HP Image Copy, and ignores the subsequent parameters. If none of the above word is specified in parentheses, HD Pointer Checker recognizes it as a syntax error.

Figure 17 shows the control statement syntax for the HPSRETCD data set.



```
(HDPC)
T2ERROR=4,DBERROR=8,
PROCERROR=16
```

Figure 17. Control statement syntax—HPSRETCD

(HDPC) statement

The (HDPC) statement specifies the options for return codes. You are allowed only one (HDPC) statement, and it must be the first control statement in the HPSRETCD data set.

The (HDPC) statement must contain optional parameters that are specified by the keywords shown in Figure 18.

```
(HDPC)

       ┌                                ┐
           T2ERROR=nnn|2
           ,DBERROR=nnn|4
           ,PROCERROR=nnn|8
           ,SPMNWARN=nnn|4
           ,SPMNERROR=nnn|8
           ,EMPTYKEYSIN=nnn|0
           ,RECOVNEEDED=nnn|0
       └                                ┘
```

Figure 18. (HDPC) statement

T2ERROR=

Specify the return code when the T2 (unknown data) error is detected. *nnn* is from 0 to 999. The default value is 2.

DBERROR=

Specify the return code when a database error is detected. *nnn* is from 0 to 999. The default value is 4.

PROCERROR=

Specify the return code when the PROCCTL statement error is detected. *nnn* is from 0 to 999. The default value is 8.

SPMNWARN=

If Space Monitor detects warning return code 4, HD Pointer Checker returns the return code specified by this parameter. The original return code is 4. *nnn* is from 0 to 999. The default value is 4.

SPMNERROR=

If Space Monitor detects warning return code 8, HD Pointer Checker returns the return code specified by this parameter. The original return code is 8. *nnn* is from 0 to 999. The default value is 8.

EMPTYKEYSIN=

Specify the return code that is to be issued when no KEYSIN record is generated. *nnn* is from 0 to 999. The default value is 0. IBM recommends that you use this keyword only when KEYSIN=YES is specified in the PROCCTL statement of HDPC. HDPC returns the return code of the EMPTYKEYSIN keyword even in the case of normal end, because no KEYSIN record is generated when KEYSIN=NO is specified.

RECOVNEEDED=

Specify the return code when a database or a partition to be checked is registered in the RECON data set and is marked as recovery needed. When the value is not 0 and a database or a partition is marked as recovery needed, message FABP2122I is issued. This keyword is available when HDPC runs with DBRC=YES. *nnn* is from 0 to 999. The default value is 0.

Notes:

1. SPMNWARN and SPMNERROR are effective when they are specified in the FABPMAIN TYPE=ALL or TYPE=SCAN JCL.
2. If more than one of the above errors are detected, the highest return code is returned by HD Pointer Checker.

Output

HD Pointer Checker primary output consists of messages and reports. These are described in detail in this section. HD Pointer Checker also produces some work data sets which will not be described.

The HD Pointer Checker reports are contained in the data sets shown in Table 14 through Table 23 on page 112.

Important:

> The order that the reports are produced is the same order of the DD statements specified in JCL. However, the order of the tables, Table 14 to Table 23 on page 112, is the recommended order to get the efficient reports.

Table 14. HD Pointer Checker reports that are printed in the PRIMAPRT data set

Report name	**Reports printed in FABPMAIN**			**Report printed in HASH Check option in IMS HPIC or IMS PR**	
	PROCCTL statement and parameter for printing report	**Whether a report is printed when PROC HASH=NO is specified** **Y: Printed** **N: Not printed**	**Whether a report is printed when PROC HASH=YES is specified** **Y: Printed** **N: Not printed**	**ICEIN HDPC= keywords and (HOMECHK=) keyword for printing report**	**Whether a report is printed** **Y: Printed** **N: Not printed**
Separator page for start of HD Pointer Checker		Y	Y		Y
DMB Directory		Y	Y		Y
PROCCTL Statements		Y	Y		Y (Format is different from FABPMAIN)

Table 15. HD Pointer Checker reports that are printed in the STATIPRT data set

Report name	**Reports printed in FABPMAIN**			**Report printed in HASH Check option in IMS HPIC or IMS PR**		**Report stored in the IMS Tools KB Output repository when PROC ITKBSRVR=*servername* is specified**
	PROCCTL statement and parameter for printing report	**Whether a report is printed when PROC HASH=NO is specified** **Y: Printed** **N: Not printed**	**Whether a report is printed when PROC HASH=YES is specified** **Y: Printed** **N: Not printed**	**ICEIN HDPC= keywords and (HOMECHK=) keyword for printing report**	**Whether a report is printed** **Y: Printed** **N: Not printed**	
Separator page for statistics	PROC SEP=YES	Y	Y		Y	
Separator page for DB/DSG	REPORT RUNTM=YES	Y	Y		Y	Y
HISAM Data Set Statistics		Y	Y		Y	Y
HISAM Segment Level Statistics		Y	Y		Y	Y
Interval Statistics	REPORT INTST=YES	Y	Y		Y	Y
Bit Map Display	REPORT BITMAP=YES	Y	Y	BITMAP	Y	Y
Free Space Map	REPORT FSEMAP=YES	Y	Y	FSEMAP	Y	Y
Maximum Free Space Distribution	REPORT MAXFSD=YES	Y	Y	MAXFSD	Y	Y
Interval Free Space Summary	REPORT INTFS=YES	Y	Y		Y	Y
HD Data Set Statistics		Y	Y		Y	Y

Table 15. HD Pointer Checker reports that are printed in the STATIPRT data set (continued)

Report name	Reports printed in FABPMAIN			Report printed in HASH Check option in IMS HPIC or IMS PR		Report stored in the IMS Tools KB Output repository when PROC ITKBSRVR=*servername* is specified
	PROCCTL statement and parameter for printing report	**Whether a report is printed when PROC HASH=NO is specified Y: Printed N: Not printed**	**Whether a report is printed when PROC HASH=YES is specified Y: Printed N: Not printed**	**ICEIN HDPC= keywords and (HOMECHK=) keyword for printing report**	**Whether a report is printed Y: Printed N: Not printed**	
DB Record Distribution Statistics	REPORT DBDIST=YES OPTION INCORE=YES OPTION HOMECHK=YES REPORT CHAINDIST=YES	Y	Y	DBDIST * HOMECHK= YES * CHAINDIST *	Y	Y
Separator page for Partition Statistics	PROC SEP=YES	Y	Y		Y	
Partition Statistics	PROC VLSSUMM=YES REPORT COMPFACT=YES REPORT SEGIO=YES	Y	Y	VLSSUMM COMPFACT * SEGIO	Y	Y
Separator page for Database Statistics	PROC SEP=YES	Y	Y		Y	
Database Statistics	PROC VLSSUMM=YES REPORT COMPFACT=YES REPORT SEGIO=YES	Y	Y	VLSSUMM COMPFACT * SEGIO	Y	Y

Note: * means that the keyword is not available in IMS Parallel Reorganization.

Table 16. HD Pointer Checker reports that are printed in the VALIDPRT data set

Report name	Reports printed in FABPMAIN			Report printed in HASH Check option in IMS HPIC or IMS PR	
	PROCCTL statement and parameter for printing report	**Whether a report is printed when PROC HASH=NO is specified Y: Printed N: Not printed**	**Whether a report is printed when PROC HASH=YES is specified Y: Printed N: Not printed**	**ICEIN HDPC= keywords and (HOMECHK=) keyword for printing report**	**Whether a report is printed Y: Printed N: Not printed**
Separator page for validation	PROC SEP=YES	Y	Y		Y
Scan of HISAM Database		Y	Y		Y
Scan of Index Database		Y	Y		Y
Validation of a Pointer to a Target at SCAN (HDAM/HIDAM/PHDAM/ PHIDAM)		Y	Y		Y
Legend for Scan and Validation		Y	N		Y
Description of All Scanned Databases		Y	N		N
Validation of a Pointer to a Target at CHECK		Y	N		N
Legend for Check Process Validation		Y	N		N

Table 17. HD Pointer Checker reports that are printed in the EVALUPRT data set

Report name	Reports printed in FABPMAIN			Report printed in HASH Check option in IMS HPIC or IMS PR	
	PROCCTL statement and parameter for printing report	**Whether a report is printed when PROC HASH=NO is specified Y: Printed N: Not printed**	**Whether a report is printed when PROC HASH=YES is specified Y: Printed N: Not printed**	**ICEIN HDPC= keywords and (HOMECHK=) keyword for printing report**	**Whether a report is printed Y: Printed N: Not printed**
Separator page for evaluation	PROC SEP=YES	Y	Y		Y
Evaluation of All Pointers to the Same Target		Y	N		N
Legend for Check Process Evaluation		Y	N		N
Check Process Total		Y	N		N
HASH Evaluation	PROC HASH=YES	N	Y		Y
Evaluation of Index Pointers and Keys	PROC IXKEYCHK=YES and HASH=NO	Y	N		N
Database Repair Guidelines	(printed only if database error is detected)	Y	Y		Y
Separator page for reconstruction	PROC SEP=YES	Y	N		N
DMB Directory and Control Card Format	(printed only if FABPMAIN TYPE=BLKMAP)	N	N		N
Pointer Chain Reconstruction	(printed only if database error is detected)	Y	N		N
Legend for Reconstruction		Y	N		N

Table 18. HD Pointer Checker reports that are printed in the EVALUPR2 data set

Report name	Reports printed in FABPMAIN			Report printed in HASH Check option in IMS HPIC or IMS PR	
	PROCCTL statement and parameter for printing report	**Whether a report is printed when PROC HASH=NO is specified Y: Printed N: Not printed**	**Whether a report is printed when PROC HASH=YES is specified Y: Printed N: Not printed**	**ICEIN HDPC= keywords and (HOMECHK=) keyword for printing report**	**Whether a report is printed Y: Printed N: Not printed**
Evaluation of Symbolic Pointer	PROC SYMLPCHK=YES PROC SYMIXCHK=YES	Y	N		N

Table 19. HD Pointer Checker reports that are printed in the EVALIPRT data set

Report name	Reports printed in FABPMAIN			Report printed in HASH Check option in IMS HPIC or IMS PR	
	PROCCTL statement and parameter for printing report	**Whether a report is printed when PROC HASH=NO is specified Y: Printed N: Not printed**	**Whether a report is printed when PROC HASH=YES is specified Y: Printed N: Not printed**	**ICEIN HDPC= keywords and (HOMECHK=) keyword for printing report**	**Whether a report is printed Y: Printed N: Not printed**
Separator page for EPS Healing and Evaluation of ILKs	PROC SEP=YES PROC EPSCHK=YES	Y	N		N
EPS Healing	PROC EPSCHK=YES	Y	N		N
Evaluation of ILKs		Y	N		N

Table 20. HD Pointer Checker reports that are printed in the SNAPPIT data set

Report name	**Reports printed in FABPMAIN**			**Report printed in HASH Check option in IMS HPIC or IMS PR**	
	PROCCTL statement and parameter for printing report	**Whether a report is printed when PROC HASH=NO is specified** **Y: Printed** **N: Not printed**	**Whether a report is printed when PROC HASH=YES is specified** **Y: Printed** **N: Not printed**	**ICEIN HDPC= keywords and (HOMECHK=) keyword for printing report**	**Whether a report is printed** **Y: Printed** **N: Not printed**
Separator page for Block Map and Dumps	PROC SEP=YES	Y	Y		Y
Block Map and Block Dump	DATABASE BLOCKDUMP= (and printed if database error is detected)	Y	Y	(printed only if database error is detected)	Y

Table 21. HD Pointer Checker reports that are printed in the SUMMARY data set

Report name	**Reports printed in FABPMAIN**			**Report printed in HASH Check option in IMS HPIC or IMS PR**		**Report stored in the IMS Tools KB Output repository when PROC ITKBSRVR=*servername* is specified**
	PROCCTL statement and parameter for printing report	**Whether a report is printed when PROC HASH=NO is specified** **Y: Printed** **N: Not printed**	**Whether a report is printed when PROC HASH=YES is specified** **Y: Printed** **N: Not printed**	**ICEIN HDPC= keywords and (HOMECHK=) keyword for printing report**	**Whether a report is printed** **Y: Printed** **N: Not printed**	
Separator page for summary	PROC SEP=YES	Y	Y		Y	
HD Pointer Checker Summary		Y	Y		Y	Y

Table 22. HD Pointer Checker reports that are printed in the DBSRCPRT data set

Report name	**Reports printed in FABPMAIN**			**Report printed in HASH Check option in IMS HPIC or IMS PR**	
	PROCCTL statement and parameter for printing report	**Whether a report is printed when PROC HASH=NO is specified** **Y: Printed** **N: Not printed**	**Whether a report is printed when PROC HASH=YES is specified** **Y: Printed** **N: Not printed**	**ICEIN HDPC= keywords and (HOMECHK=) keyword for printing report**	**Whether a report is printed** **Y: Printed** **N: Not printed**
Messages	REPORT DECODEDBD= YES	Y	Y		N
DBD Source		Y	Y		N

Table 23. HD Pointer Checker reports that are printed in the DBMAPPRT data set

Report name	**Reports printed in FABPMAIN**			**Report printed in HASH Check option in IMS HPIC or IMS PR**	
	PROCCTL statement and parameter for printing report	**Whether a report is printed when PROC HASH=NO is specified** **Y: Printed** **N: Not printed**	**Whether a report is printed when PROC HASH=YES is specified** **Y: Printed** **N: Not printed**	**ICEIN HDPC= keywords and (HOMECHK=) keyword for printing report**	**Whether a report is printed** **Y: Printed** **N: Not printed**
Messages	REPORT MAPDBD=YES	Y	Y		N
DBD Map		Y	Y		N

Note: If HD Pointer Checker runs in a DBB region of IMS Version 6 and scans one or more segments that are not sensitive in the specified PSB. The segment names will be written as "*UNKNOWN" in the following reports:

- HISAM Data Set Statistics report
- HISAM Segment Level Statistics report
- Interval Statistics report

- Database Statistics report
- Scan of Index Database report
- Evaluation of All Pointers to the Same Target report
- Block Map and Block Dump report

If input DD statements SPMNIN and SPMNSPDT are specified, additional output data sets are generated by Space Monitor.

For output DD statements of Space Monitor, see “Job control language” on page 495.

Messages to operator

This section explains the HD Pointer Checker messages.

Function

The HD Pointer Checker writes some messages to inform the operator about the status of the job. These messages also appear on the Job Log in the output run listing.

The customized messages can be written to the operator console by the FABPZWTO user exit routine by editing the information in HD Pointer Checker summary report. See the member FABPZWTO of the sample exit routine provided in HPS.AHPSSAMP library.

Format

The messages that are written to the operator console are described in Appendix A, “Messages and codes,” on page 595. Figure 19 illustrates the kinds of messages that the operator sees.

```
21.45.36 JOB02419  +FABP1440I SCAN OF DB: HDAMDB2  DSG: 01 IN PROGRESS     @ BLOCK          1  TIME=21.45.36
21.45.36 JOB02419  +FABP1440I SCAN OF DB: TPFOH3   PID: 00001 DSG:  A IN PROGRESS @ BLOCK         1 TIME=21.45.36
21.45.36 JOB02419  +FABP1440I SCAN OF DB: TPFOH1   PID: 00001 DSG:  A IN PROGRESS @ BLOCK         1 TIME=21.45.36
21.45.37 JOB02419  +FABP1440I SCAN OF DB: TPFOH2   PID: 00001 DSG:  A IN PROGRESS @ BLOCK         1 TIME=21.45.37
21.45.37 JOB02419  +FABP1410I SCAN OF DB: TPFOX1   PID: 00001 DSG:  A IN PROGRESS                   TIME=21.45.37
21.45.37 JOB02419  +FABP1370I SCAN OF DB: HISAMDB1 DSG: 01 IN PROGRESS                         TIME=21.45.37
21.45.37 JOB02419  +FABP1450I SCAN OF DB: TPFOH3   PID: 00001 DSG:  A COMPLETED   @ BLOCK      1469 TIME=21.45.37
21.45.37 JOB02419  +FABP1420I SCAN OF DB: TPFOX1   PID: 00001 DSG:  A COMPLETED                     TIME=21.45.37
21.45.37 JOB02419  +FABP1450I SCAN OF DB: HDAMDB2  DSG: 01 COMPLETED       @ BLOCK       2474  TIME=21.45.37
21.45.38 JOB02419  +FABP1440I SCAN OF DB: TPFOH1   PID: 00001 DSG:  A IN PROGRESS @ BLOCK      3977 TIME=21.45.38
21.45.38 JOB02419  +FABP1440I SCAN OF DB: TPFOH2   PID: 00001 DSG:  A IN PROGRESS @ BLOCK      4009 TIME=21.45.38
21.45.38 JOB02419  +FABP1450I SCAN OF DB: TPFOH2   PID: 00001 DSG:  A COMPLETED   @ BLOCK      4409 TIME=21.45.38
21.45.38 JOB02419  +FABP1380I SCAN OF DB: HISAMDB1 DSG: 01 COMPLETED                           TIME=21.45.38
21.45.38 JOB02419  +FABP1440I SCAN OF DB: TPFOH1   PID: 00001 DSG:  A IN PROGRESS @ BLOCK      7953 TIME=21.45.38
21.45.39 JOB02419  +FABP1440I SCAN OF DB: TPFOH1   PID: 00001 DSG:  A IN PROGRESS @ BLOCK     11929 TIME=21.45.39
21.45.39 JOB02419  +FABP1410I SCAN OF DB: TPFOH2   PID: 00001 DSG:  X IN PROGRESS                   TIME=21.45.39
21.45.39 JOB02419  +FABP1440I SCAN OF DB: TPFOH1   PID: 00001 DSG:  A IN PROGRESS @ BLOCK     15905 TIME=21.45.39
21.45.39 JOB02419  +FABP1420I SCAN OF DB: TPFOH2   PID: 00001 DSG:  X COMPLETED                     TIME=21.45.39
21.45.39 JOB02419  +FABP1450I SCAN OF DB: TPFOH1   PID: 00001 DSG:  A COMPLETED   @ BLOCK     19844 TIME=21.45.39
21.45.40 JOB02419  +FABP1440I SCAN OF DB: TPFOH1   PID: 00001 DSG:  B IN PROGRESS @ BLOCK         1 TIME=21.45.40
21.45.40 JOB02419  +FABP1450I SCAN OF DB: TPFOH1   PID: 00001 DSG:  B COMPLETED   @ BLOCK       734 TIME=21.45.40
21.45.43 JOB02419  +FABP1480I EVAL OF DB: HDAMDB2  DSG: 01 IN PROGRESS                         TIME=21.45.43
21.45.43 JOB02419  +FABP1490I EVAL OF DB: HDAMDB2  DSG: 01 COMPLETED                           TIME=21.45.43
21.45.43 JOB02419  +FABP1480I EVAL OF DB: HISAMDB1 DSG: 01 IN PROGRESS                         TIME=21.45.43
21.45.43 JOB02419  +FABP1490I EVAL OF DB: HISAMDB1 DSG: 01 COMPLETED                           TIME=21.45.43
21.45.43 JOB02419  +FABP1480I EVAL OF DB: TPFOH1   PID: 00001 DSG:  A IN PROGRESS                   TIME=21.45.43
21.45.44 JOB02419  +FABP1490I EVAL OF DB: TPFOH1   PID: 00001 DSG:  A COMPLETED                     TIME=21.45.44
21.45.44 JOB02419  +FABP1480I EVAL OF DB: TPFOH1   PID: 00001 DSG:  B IN PROGRESS                   TIME=21.45.44
21.45.44 JOB02419  +FABP1490I EVAL OF DB: TPFOH1   PID: 00001 DSG:  B COMPLETED                     TIME=21.45.44
21.45.44 JOB02419  +FABP1480I EVAL OF DB: TPFOH3   PID: 00001 DSG:  A IN PROGRESS                   TIME=21.45.44
21.45.44 JOB02419  +FABP1490I EVAL OF DB: TPFOH3   PID: 00001 DSG:  A COMPLETED                     TIME=21.45.44
21.45.44 JOB02419  +FABP1480I EVAL OF DB: TPFOH2   PID: 00001 DSG:  A IN PROGRESS                   TIME=21.45.44
21.45.44 JOB02419  +FABP1490I EVAL OF DB: TPFOH2   PID: 00001 DSG:  A COMPLETED                     TIME=21.45.44
21.45.45 JOB02419  +FABP0001I***********************************************************************************
21.45.45 JOB02419  +FABP0001I                                                                                  *
21.45.45 JOB02419  +FABP0001I HD POINTER CHECKER ENDED NORMALLY                                                *
21.45.45 JOB02419  +FABP0001I                                                                                  *
21.45.45 JOB02419  +FABP0001I***********************************************************************************
21.45.45 JOB02419  DFS627I IMS RTM CLEANUP ( EOT ) COMPLETE FOR JS MANUHPC2.HDPCRUN .HDPCPRO ,RC=00
```

Figure 19. HD Pointer Checker job log

If Space Monitor is called, the result of Space Monitor is shown by the WTO message displayed after the HD Pointer Checker’s end message. Figure 20 on page 115 shows the HD Pointer Checker job log with Space Monitor.

```
FABP0001I******************************************************
FABP0001I
FABP0001I HD POINTER CHECKER ENDED NORMALLY
FABP0001I
FABP0001I******************************************************
FABP0007I SPACE MONITOR ENDED NORMALLY
```

Figure 20. HD Pointer Checker job log with Space Monitor

PRIMAPRT data set

This section explains the PRIMAPRT data set.

Function

The PRIMAPRT data set contains the following reports produced by the HD Pointer Checker processor (FABPMAIN):

- Separator page for start of HD Pointer Checker reports
- DMB Directory report
- PROCCTL Statements report
- HPSRETCD Statements report

Separator page for the start of HD Pointer Checker reports

This separator page (see Figure 21) contains the title, “HD Pointer Checker,” and indicates that HD Pointer Checker started the processing.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                  "SEPARATOR PAGE FOR START OF HDPC"                         PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                         FABPMAIN - V2.R2




     HHH      HHH  DDDDDDDD
     HHH      HHH  DDDDDDDDDD
      HH      HH    DD     DDD
      HH      HH    DD      DD
      HH      HH    DD      DD
      HH      HH    DD      DD
      HHHHHHHHHH    DD      DD
      HHHHHHHHHH    DD      DD
      HHHHHHHHHH    DD      DD
      HH      HH    DD      DD
      HH      HH    DD      DD
      HH      HH    DD      DD
      HH      HH    DD     DDD
     HHH      HHH  DDDDDDDDDD
     HHH      HHH  DDDDDDDD

     PPPPPPPP          OOOO       IIIIIIIIII   NNN     NNNN  TTTTTTTTTTTT  EEEEEEEEEEE   RRRRRRRR
     PPPPPPPPPP      OOOOOOOO     IIIIIIIIII   NNN     NNNN  TTTTTTTTTTTT  EEEEEEEEEEE   RRRRRRRRRR
      PP     PPP    OOO    OOO        II        NN      NN   TT   TT   TT   EE            RR     RRR
      PP      PP    OO      OO        II        NNN     NN        TT        EE            RR      RR
      PP      PP    OO      OO        II        NNNN    NN        TT        EE            RR      RR
      PP      PP    OO      OO        II        NNNNN   NN        TT        EE            RR      RR
      PP     PPP    OO      OO        II        NNNNNN  NN        TT        EE   EEE      RR     RRR
      PPPPPPPPP     OO      OO        II        NN NNNN NN        TT        EEEEEEEE      RRRRRRRRR
      PPPPPPP       OO      OO        II        NN  NNNNNN        TT        EE   EEE      RRRRRRR
      PP            OO      OO        II        NN   NNNNN        TT        EE            RR RRR
      PP            OO      OO        II        NN    NNNN        TT        EE            RR  RRR
      PP            OO      OO        II        NN     NNN        TT        EE            RR   RRR
      PP            OOO    OOO        II        NN      NN        TT        EE            RR    RRR
     PPPP            OOOOOOOO     IIIIIIIIII   NNNN     NNN      TTTT      EEEEEEEEEEE   RRRR   RRRR
     PPPP              OOOO       IIIIIIIIII   NNNN     NNN      TTTT      EEEEEEEEEEE   RRRR   RRRR

         CCCC      HHH      HHH  EEEEEEEEEEE       CCCC      KKKK    KKKK  EEEEEEEEEEE   RRRRRRRR
       CCCCCCCC    HHH      HHH  EEEEEEEEEEE     CCCCCCCC    KKKK    KKKK  EEEEEEEEEEE   RRRRRRRRRR
      CCC    CCC    HH      HH    EE            CCC    CCC    KK      KK    EE            RR     RRR
      CC      CC    HH      HH    EE            CC      CC    KK     KK     EE            RR      RR
      CC     CCC    HH      HH    EE            CC     CCC    KK    KK      EE            RR      RR
      CC     CCC    HH      HH    EE            CC     CCC    KK   KK       EE            RR      RR
      CC            HHHHHHHHHH    EE   EEE      CC            KK  KK        EE   EEE      RR     RRR
      CC            HHHHHHHHHH    EEEEEEEE      CC            KKKKK         EEEEEEEE      RRRRRRRRR
      CC            HHHHHHHHHH    EE   EEE      CC            KK  KK        EE   EEE      RRRRRRR
      CC     CCC    HH      HH    EE            CC     CCC    KK   KK       EE            RR RRR
      CC     CCC    HH      HH    EE            CC     CCC    KK    KK      EE            RR  RRR
      CC      CC    HH      HH    EE            CC      CC    KK     KK     EE            RR   RRR
      CCC    CCC    HH      HH    EE            CCC    CCC    KK      KK    EE            RR    RRR
       CCCCCCCC    HHH      HHH  EEEEEEEEEEE     CCCCCCCC    KKKK    KKKK  EEEEEEEEEEE   RRRR   RRRR
         CCCC      HHH      HHH  EEEEEEEEEEE       CCCC      KKKK    KKKK  EEEEEEEEEEE   RRRR   RRRR
```

Figure 21. PRIMAPRT—Separator page for the start of HD Pointer Checker reports

DMB Directory report

This report (see Figure 22 on page 118) contains environment information and a list of all the databases that are defined (either explicitly or implicitly) by your PSB or DBD. This report is generated by the SCAN process.

The report fields are as follows:

ENVIRONMENT:
This part shows environment information as follows:

CPU MODEL and OPERATING SYSTEM
Self-explanatory.

IMS LEVEL
The IMS version and release level of the specified IMS.RESLIB.

IMS REGION
The IMS region type such as DLI, DBB, and ULU.

IMS ID
The IMS subsystem ID.

The value of the IMS REGION and IMS ID fields are shown when HD Pointer Checker runs under the IMS batch region controller (DFSRRC00). Otherwise, 'N/A' is shown.

The columns of the succeeding list is as follows:

DBD NAME
The name of a DBD load module.

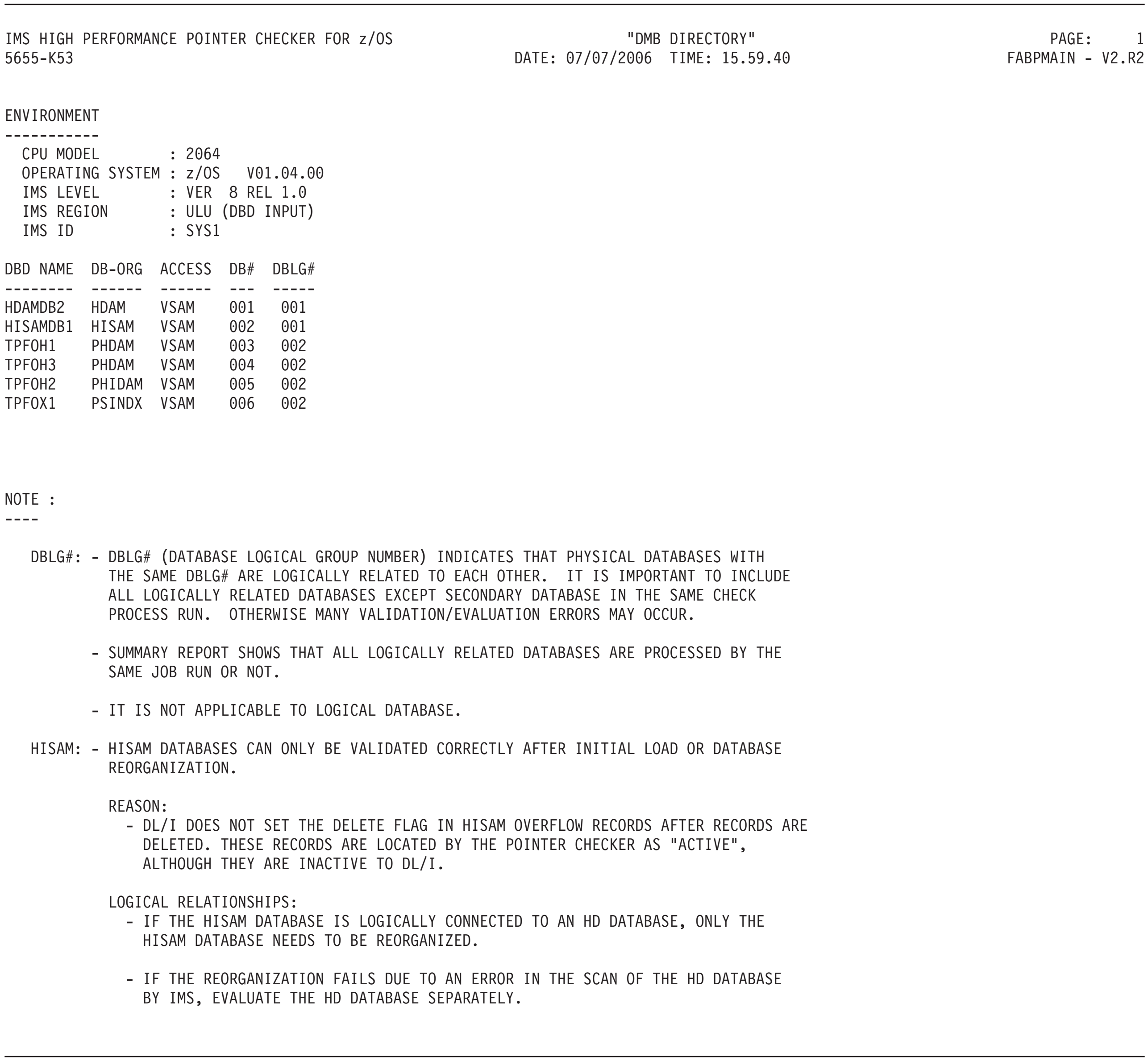
DB-ORG
The type of database organization—HISAM, SHISAM, HIDAM, HDAM, PHIDAM, PHDAM, PSINDEX, INDEX, or LOGICAL.

ACCESS
The access method used by the database (VSAM or OSAM)

DB#
The database number (in hexadecimal) used to identify the database throughout the HD Pointer Checker run.

DBLG#
The database logical group number which indicates that physical databases with the same database logical group number are logically related to each other.

See the note in this report.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                          "DMB DIRECTORY"                                     PAGE:     1
5655-K53                                                         DATE: 07/07/2006  TIME: 15.59.40                      FABPMAIN - V2.R2


ENVIRONMENT
-----------
  CPU MODEL        : 2064
  OPERATING SYSTEM : z/OS   V01.04.00
  IMS LEVEL        : VER  8 REL 1.0
  IMS REGION       : ULU (DBD INPUT)
  IMS ID           : SYS1

DBD NAME  DB-ORG  ACCESS  DB#  DBLG#
--------  ------  ------  ---  -----
HDAMDB2   HDAM    VSAM    001   001
HISAMDB1  HISAM   VSAM    002   001
TPFOH1    PHDAM   VSAM    003   002
TPFOH3    PHDAM   VSAM    004   002
TPFOH2    PHIDAM  VSAM    005   002
TPFOX1    PSINDX  VSAM    006   002




NOTE :
----

   DBLG#: - DBLG# (DATABASE LOGICAL GROUP NUMBER) INDICATES THAT PHYSICAL DATABASES WITH
            THE SAME DBLG# ARE LOGICALLY RELATED TO EACH OTHER.  IT IS IMPORTANT TO INCLUDE
            ALL LOGICALLY RELATED DATABASES EXCEPT SECONDARY DATABASE IN THE SAME CHECK
            PROCESS RUN.  OTHERWISE MANY VALIDATION/EVALUATION ERRORS MAY OCCUR.

          - SUMMARY REPORT SHOWS THAT ALL LOGICALLY RELATED DATABASES ARE PROCESSED BY THE
            SAME JOB RUN OR NOT.

          - IT IS NOT APPLICABLE TO LOGICAL DATABASE.

   HISAM: - HISAM DATABASES CAN ONLY BE VALIDATED CORRECTLY AFTER INITIAL LOAD OR DATABASE
            REORGANIZATION.

            REASON:
              - DL/I DOES NOT SET THE DELETE FLAG IN HISAM OVERFLOW RECORDS AFTER RECORDS ARE
                DELETED. THESE RECORDS ARE LOCATED BY THE POINTER CHECKER AS "ACTIVE",
                ALTHOUGH THEY ARE INACTIVE TO DL/I.

            LOGICAL RELATIONSHIPS:
              - IF THE HISAM DATABASE IS LOGICALLY CONNECTED TO AN HD DATABASE, ONLY THE
                HISAM DATABASE NEEDS TO BE REORGANIZED.

              - IF THE REORGANIZATION FAILS DUE TO AN ERROR IN THE SCAN OF THE HD DATABASE
                BY IMS, EVALUATE THE HD DATABASE SEPARATELY.
```

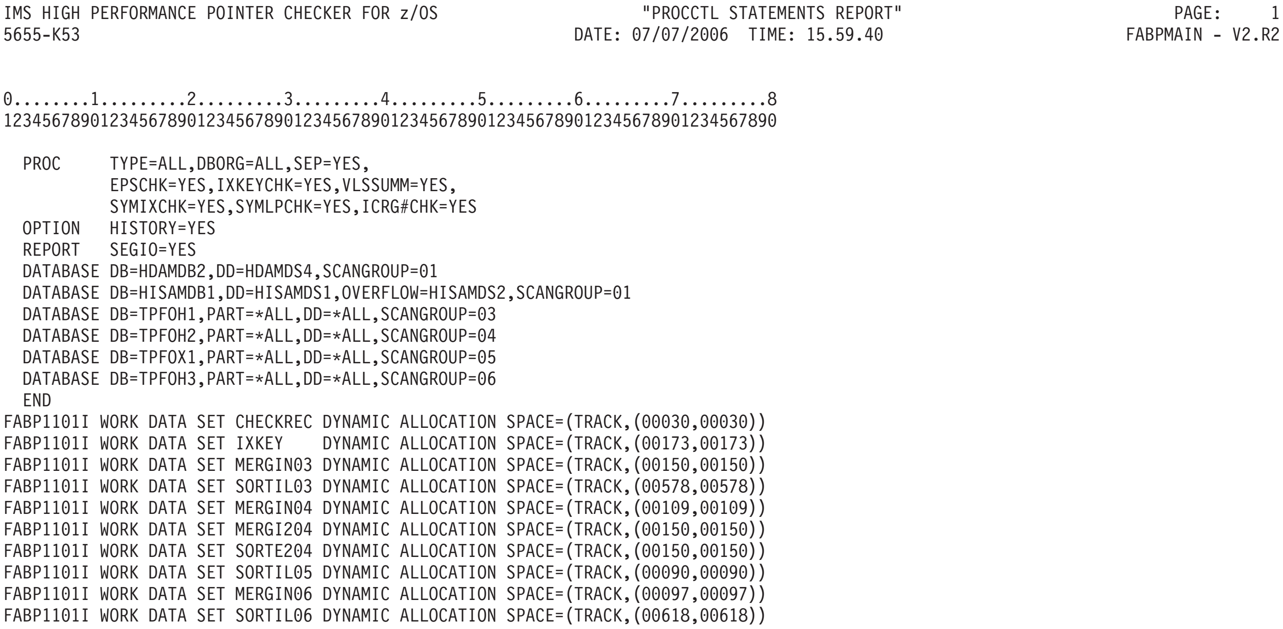
Figure 22. PRIMAPRT—DMB Directory report

PROCCTL Statements report

This report (see Figure 23) indicates all control statements of the current HD Pointer Checker run. These control statements have been specified in the PROCCTL data set. Following control statements are reported:

- PROC statement
- OPTION statement
- REPORT statement
- DATABASE statement
- END statement

This report provides the space information of work data sets at the time of dynamic allocation. The SPACE= parameter passed to the dynamic allocation macro is shown in message FABP1101I.

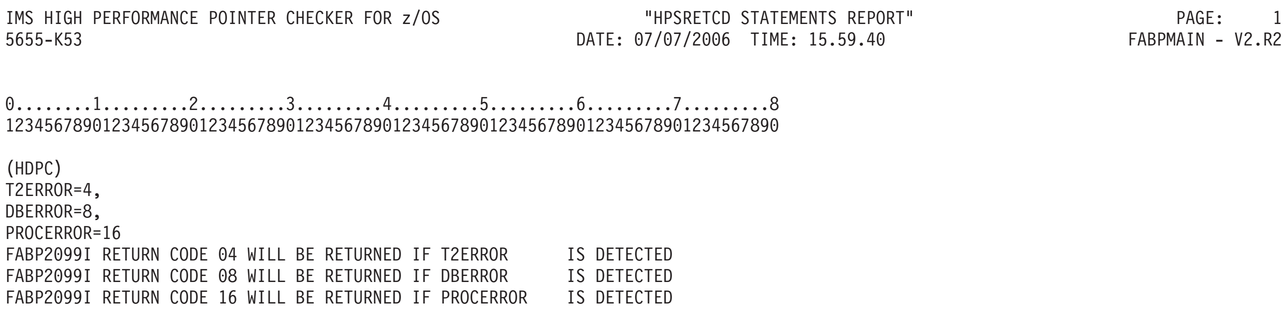


```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                     "PROCCTL STATEMENTS REPORT"                           PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                          FABPMAIN - V2.R2


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

  PROC     TYPE=ALL,DBORG=ALL,SEP=YES,
           EPSCHK=YES,IXKEYCHK=YES,VLSSUMM=YES,
           SYMIXCHK=YES,SYMLPCHK=YES,ICRG#CHK=YES
  OPTION   HISTORY=YES
  REPORT   SEGIO=YES
  DATABASE DB=HDAMDB2,DD=HDAMDS4,SCANGROUP=01
  DATABASE DB=HISAMDB1,DD=HISAMDS1,OVERFLOW=HISAMDS2,SCANGROUP=01
  DATABASE DB=TPFOH1,PART=*ALL,DD=*ALL,SCANGROUP=03
  DATABASE DB=TPFOH2,PART=*ALL,DD=*ALL,SCANGROUP=04
  DATABASE DB=TPFOX1,PART=*ALL,DD=*ALL,SCANGROUP=05
  DATABASE DB=TPFOH3,PART=*ALL,DD=*ALL,SCANGROUP=06
  END
FABP1101I WORK DATA SET CHECKREC DYNAMIC ALLOCATION SPACE=(TRACK,(00030,00030))
FABP1101I WORK DATA SET IXKEY    DYNAMIC ALLOCATION SPACE=(TRACK,(00173,00173))
FABP1101I WORK DATA SET MERGIN03 DYNAMIC ALLOCATION SPACE=(TRACK,(00150,00150))
FABP1101I WORK DATA SET SORTIL03 DYNAMIC ALLOCATION SPACE=(TRACK,(00578,00578))
FABP1101I WORK DATA SET MERGIN04 DYNAMIC ALLOCATION SPACE=(TRACK,(00109,00109))
FABP1101I WORK DATA SET MERGI204 DYNAMIC ALLOCATION SPACE=(TRACK,(00150,00150))
FABP1101I WORK DATA SET SORTE204 DYNAMIC ALLOCATION SPACE=(TRACK,(00150,00150))
FABP1101I WORK DATA SET SORTIL05 DYNAMIC ALLOCATION SPACE=(TRACK,(00090,00090))
FABP1101I WORK DATA SET MERGIN06 DYNAMIC ALLOCATION SPACE=(TRACK,(00097,00097))
FABP1101I WORK DATA SET SORTIL06 DYNAMIC ALLOCATION SPACE=(TRACK,(00618,00618))
```

Figure 23. PRIMAPRT—PROCCTL Statements report

HPSRETCD Statements report

This report (Figure 24) is generated when you specify the HPSRETCD data set. The report shows all of the control statements that were specified in the HPSRETCD data set.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                     "HPSRETCD STATEMENTS REPORT"                          PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                        FABPMAIN - V2.R2


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

(HDPC)
T2ERROR=4,
DBERROR=8,
PROCERROR=16
FABP2099I RETURN CODE 04 WILL BE RETURNED IF T2ERROR      IS DETECTED
FABP2099I RETURN CODE 08 WILL BE RETURNED IF DBERROR      IS DETECTED
FABP2099I RETURN CODE 16 WILL BE RETURNED IF PROCERROR    IS DETECTED
```

Figure 24. PRIMAPRT—HPSRETCD Statements report

STATIPRT data set

This section explains the STATIPRT data set.

Function

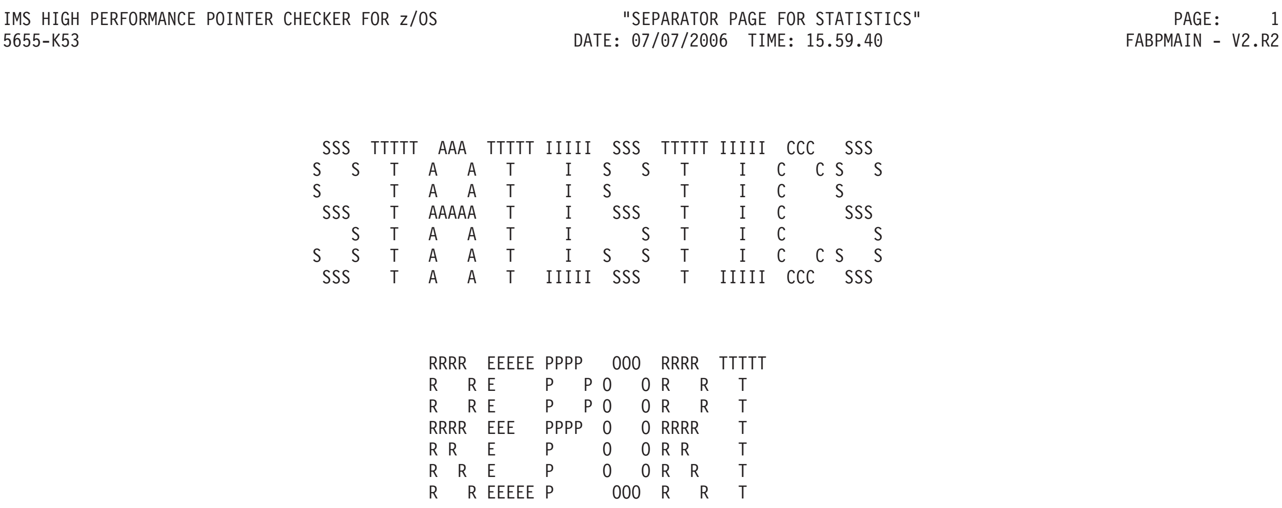
The STATIPRT data set contains the following reports produced by the HD Pointer Checker processor (FABPMAIN):

- Separator page for statistics reports
- Separator page for DB/DSG reports
- HISAM Data Set Statistics report
- HISAM Segment Level Statistics report
- Interval Statistics report
- Bit Map Display report
- Free Space Map report
- Maximum Free Space Distribution report
- Interval Free Space Summary report
- HD Data Set Statistics report
- DB Record Distribution Statistics report
- Separator page for Partition Statistics reports
- Partition Statistics report
- Separator page for Database Statistics reports
- Database Statistics report

These reports are printed as one set for each database data set group except the separator page for statistics reports.

Separator page for statistics reports

This separator page (see Figure 25) contains the title of “Statistics Report,” and indicates that statistics reports will follow this page. It is produced unless SEP=NO is specified on the PROC statement as input for the PROCCTL data set.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                   "SEPARATOR PAGE FOR STATISTICS"                           PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                          FABPMAIN - V2.R2



                                 SSS  TTTTT  AAA  TTTTT IIIII  SSS  TTTTT IIIII  CCC   SSS
                                S   S   T   A   A   T     I   S   S   T     I   C   C S   S
                                S       T   A   A   T     I   S       T     I   C     S
                                 SSS    T   AAAAA   T     I    SSS    T     I   C      SSS
                                    S   T   A   A   T     I       S   T     I   C         S
                                S   S   T   A   A   T     I   S   S   T     I   C   C S   S
                                 SSS    T   A   A   T   IIIII  SSS    T   IIIII  CCC   SSS



                                            RRRR  EEEEE PPPP   OOO  RRRR  TTTTT
                                            R   R E     P   P O   O R   R   T
                                            R   R E     P   P O   O R   R   T
                                            RRRR  EEE   PPPP  O   O RRRR    T
                                            R R   E     P     O   O R R     T
                                            R  R  E     P     O   O R  R    T
                                            R   R EEEEE P      OOO  R   R   T
```

Figure 25. STATIPRT—Separator page for statistics reports

Separator page for DB/DSG reports

This separator page (see Figure 26 on page 123 and Figure 27 on page 124) indicates that HD Pointer Checker started processing of the database/data set group described in this report. This separator has two types of formats. Figure 26 on page 123 is the general format, which is generated when TYPE=ALL or SCAN is specified on the PROC statement as input for the PROCCTL data set. Figure 27 on page 124 is generated when BLOCKDUMP= keyword is specified on the DATABASE statement. The separator page is produced unless RUNTM=NO is specified on the REPORT statement.

The report fields are as follows:

DBNAME
The name of the DBD as coded on the NAME= keyword of the DBD macro

DB#
The database number (in hexadecimal) that identifies the database being processed

PARTNAME
Partition name

PARTID
Partition ID

REORG#
Partition reorganization number (reorg number).

The reorganization number is shown only for data set group A. "N/A" is shown for groups other than group A, as same as for other reports.

DSG#
For non-HALDB, the data set group number (in hexadecimal). For HALDB that identifies the database being processed, the data set group ID (in an alphabetical character).

DDNAME
The ddname as coded on the DD1= keyword or OVFLW= keyword of the DATASET macro in the DBD

DSNAME
The name of the data sets being processed

DBORG
The organization type of the database.

The above heading fields are common among the HD Pointer Checker reports described later in this chapter. The detailed descriptions of these headings will not be repeated where they appear again. The remaining report fields are as follows:

PROC STATEMENT
Actual options being processed, which are specified on the PROC statement

DATABASE STATEMENT
Actual options being processed, which are specified on the DATABASE statement

OPTION STATEMENT
Actual options being processed, which are specified on the OPTION statement

REPORT STATEMENT
Actual options being processed, which are specified on the REPORT statement

BLOCKDUMP
The values specified with BLOCKDUMP=(rba,nnn) on the DATABASE statement

DUMP NO.
The dump number to be reported in the succeeding Block Map and Block Dump report

TOTAL BLOCKS DUMPED
The total number of blocks to be reported in the succeeding Block Map and Block Dump report.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                     "SEPARATOR PAGE FOR DB/DSG"                             PAGE:     1
5655-K53                                                   DATE: 07/28/2006  TIME: 09.56.33                          FABPMAIN - V2.R2


  TTTTT PPPP  FFFFF  OOO  H   H   1                                 TTTTT PPPP  FFFFF  OOO  H   H   1    AAA   AAA
    T   P   P F     O   O H   H  11                                   T   P   P F     O   O H   H  11   A   A A   A
    T   P   P F     O   O H   H   1                                   T   P   P F     O   O H   H   1   A   A A   A
    T   PPPP  FFF   O   O HHHHH   1                                   T   PPPP  FFF   O   O HHHHH   1   AAAAA AAAAA
    T   P     F     O   O H   H   1                                   T   P     F     O   O H   H   1   A   A A   A
    T   P     F     O   O H   H   1                                   T   P     F     O   O H   H   1   A   A A   A
    T   P     F      OOO  H   H 11111                                 T   P     F      OOO  H   H 11111 A   A A   A


  DDDD  BBBB   # #               000   000   333                    DDDD   SSS   GGG   # #                     AAA
  D   D B   B #####             0   0 0   0 3   3                   D   D S   S G   G #####                   A   A
  D   D B   B  # #              0  00 0  00     3                   D   D S     G      # #                    A   A
  D   D BBBB   # #              0 0 0 0 0 0   33                    D   D  SSS  G GGG  # #                    AAAAA
  D   D B   B  # #              00  0 00  0     3                   D   D     S G G G  # #                    A   A
  D   D B   B #####             0   0 0   0 3   3                   D   D S   S G   G #####                   A   A
  DDDD  BBBB   # #               000   000   333                    DDDD   SSS   GGG   # #                    A   A

RUN TIME OPTION FOR DB/DSG
--------------------------

  DBNAME: TPFOH1    DB#: 003 PARTNAME: TPFOH1A   PART ID: 00001 REORG#: 00001
                    DSG#:  A PRIM DD:  TPFOH1AA  DSNAME: TESTDS.PUBLIC.SAMPLE.TPFOH1.A00001

  PROC STATEMENT                   DATABASE STATEMENT                   OPTION STATEMENT                   REPORT STATEMENT
  --------------                   ------------------                   ----------------                   ----------------

  TYPE     = ALL                   DB       = TPFOH1                    ERRLIMIT  = YES                    RUNTM     = YES
  DBORG    = PHDAM                 DD       = TPFOH1AA                  DIAGDUMP  = ERROR                  INTST     = YES
  HASH     = NO                    OVERFLOW = N/A                       DUMPFORM  = FORMAT                 BITMAP    = YES
  IXKEYCHK = NO                    DATASET  = REAL                      HISTORY   = YES                    FSEMAP    = YES
  SEP      = YES                   SCANGROUP= 03                        HOMECHK   = (YES,-010,010)         MAXFSD    = YES
  VLSSUMM  = YES                                                        INTERVAL  = DATASET                INTFS     = YES
  CHECK    = (CHK,111111)                                               INCORE    = YES                    DBDIST    = YES
  EPSCHK   = YES                                                        KEYSIN    = NO                     CHAINDIST = YES
  CHECKREC = NO                                                         T2CHK     = (00,07)                DECODEDBD = NO
  ICRG#CHK = N/A                                                        ZEROCTR   = NO                     MAPDBD    = NO
  RETCDDSN = N/A (USE SPECIFICATIONS IN HPSRETCD)                       DIAG      = NO                     COMPFACT  = NO
  USER     = *NO                                                        PRINTDATA = NO                     SEGIO     = NO
                                                                        PTRCHK    = YES
                                                                        NOCHKP    =
                                                                        VSAMBF    = N/A
                                                                        DSSIZE    =
                                                                        ICUNIT    = N/A
                                                                        IBUFF     = 10TRK

FABP2084I I/O BUFFER OF DD: TPFOH1AA (CI/BLOCK SIZE:    512) IS    245 KBYTE
```

Figure 26. STATIPRT—Separator page for DB/DSG reports (General format)

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                     "SEPARATOR PAGE FOR DB/DSG"                             PAGE:     1
5655-K53                                                   DATE: 07/10/2006  TIME: 09.44.00                          FABPMAIN - V2.R2



  H   H DDDD   AAA  M   M DDDD  BBBB   222                          H   H DDDD   AAA  M   M DDDD   SSS    4
  H   H D   D A   A M   M D   D B   B 2   2                         H   H D   D A   A M   M D   D S   S 4 4
  H   H D   D A   A MM MM D   D B   B    2                          H   H D   D A   A MM MM D   D S     4 4
  HHHHH D   D AAAAA M M M D   D BBBB    2                           HHHHH D   D AAAAA M M M D   D  SSS  4 4
  H   H D   D A   A M   M D   D B   B  2                            H   H D   D A   A M   M D   D     S 44444
  H   H D   D A   A M   M D   D B   B 2                             H   H D   D A   A M   M D   D S   S   4
  H   H DDDD  A   A M   M DDDD  BBBB  22222                         H   H DDDD  A   A M   M DDDD   SSS    4



  DDDD  BBBB   # #               000   000    1                     DDDD   SSS   GGG   # #               000    1
  D   D B   B #####             0   0 0   0  11                     D   D S   S G   G #####             0   0  11
  D   D B   B  # #              0  00 0  00   1                     D   D S     G      # #              0  00   1
  D   D BBBB   # #              0 0 0 0 0 0   1                     D   D  SSS  G GGG  # #              0 0 0   1
  D   D B   B  # #              00  0 00  0   1                     D   D     S G G G  # #              00  0   1
  D   D B   B #####             0   0 0   0   1                     D   D S   S G   G #####             0   0   1
  DDDD  BBBB   # #               000   000  11111                   DDDD   SSS   GGG   # #               000  11111

RUN TIME OPTION FOR DB/DSG
--------------------------

  DBNAME: HDAMDB2   DB#: 001 DSG#: 01  PRIM DD: HDAMDS4   DSNAME: TESTDS.PUBLIC.SAMPLE.HDAMDS4
                                       OVER DD:           DSNAME:


  BLOCKDUMP= (00002400,001)


  DUMP NO.    FROM    =      1
  DUMP NO.    TO      =      1
  TOTAL BLOCKS DUMPED =      1
```

Figure 27. STATIPRT—Separator page for DB/DSG reports (Blockdump option format)

HISAM Data Set Statistics report

This report (see Figure 28 on page 129) contains information about the HISAM (including SHISAM) data set.

The report contains the following part:

- Data Set Summary: This shows the database data set information.
- Space Use Analysis: This shows status of the use of the data set.
- Segment Occurrences: This shows various totals for each segment type.

This report is produced for each data set specified on the DATABASE statement in the PROCCTL data set.

The report fields are as follows:

DBNAME DB# DSG#
The name of the DBD, the database number (in hexadecimal), the data set group number (in hexadecimal).

DATA SET SUMMARY: This part shows the database data set information. If the data set is an image copy, DSNAME OF DATA SET and DATA SET CREATION DATE TIME show the information about the image copy data set.

DBD NAME
The name of the DBD as coded on the NAME= keyword of the DBD macro.

DB NUMBER
The database number (in hexadecimal) used to identify the database throughout the HD Pointer Checker run.

DATABASE ORGANIZATION
The IMS organization used for this database: HDAM, HIDAM, PHDAM, or PHIDAM.

ACCESS METHOD
The access method used for this database: VSAM or OSAM.

DDNAME OF DATA SET
The DD name of this database data set.

DSNAME OF DATA SET
The name of database data set. If the processed data set is an image copy, it is the name of the image copy data set.

BLOCK SIZE
The data set the block size for OSAM or CI size for VSAM.

RECORD SIZE
The data set record size.

DATA SET CREATION DATE TIME
The date and time when this data set was created. If the data set is an image copy, it shows the date and time of the image copy data set. If the data set is a real database of VSAM KSDS or a Fast Recovery image copy, it shows "N/A".

SPACE USE ANALYSIS: This part shows the status of the space use for each data set.

TOTAL NUMBER OF LOGICAL RECORDS
The number of logical records read by the SCAN processor in this data set.

NUMBER OF LOGICAL RECORDS WITH FREE SPACE
The number of logical records that has enough free space to store the smallest

segment that is registered in the database. The smallest segment is a segment whose minimum length is what is specified in "SEGMENT SIZE RANGE FOR THIS DSG (FROM DBD)".

SEGMENT SIZE RANGE FOR THIS DATA SET (FROM DBD)
The maximum and minimum segment lengths of the segments in this data set.

The segment length is the prefix length plus the data length. The data length is a numeric value specified in the SEGM statement in the DBD. For a variable-length segment, the maximum segment size is taken as the data length. For a fixed-length segment that has been made longer than the segment size in the DBD by the segment edit/compress facility, the segment size in the DBD is taken as the segment data length. The increase in the maximum length of the fixed-segment which is edited by the segment edit/compression facility is ignored.

If a fixed length segment is defined with a segment edit/compression exit, the minimum length of the segment is the prefix length plus the minimum value of BYTES= in DBD.

TOTAL BYTES OF SPACE
The following items are shown:

- The database data set size in bytes
- The percentage of the database data set size to the total size (always 100.0%)
- The database data set size in giga bytes
- The percentage of database data set size to the maximum size
- The database data set size limit

SEGMENT PREFIX
The number of bytes in the prefix and what percentage it makes of the total bytes.

SEGMENT DATA
The number of bytes of data and what percentage it makes of the total bytes.

SEGMENT PAD
The number of bytes of segment pad and what percentage it makes of the total bytes. If the length of the variable segment is less than the MIN value in the DBD, the difference is counted as a segment pad.

Note: A deleted segment that is marked as deleted in a delete byte is reported in segment prefix, segment data, and segment pad, because the segment is not physically deleted.

FREE SPACE (USABLE)
The sum of the following values and what percentage it makes of the total bytes.

- Unused area in a logical record that has enough length to store the minimum length dependent segment. The segment length is the prefix length plus the data length. The data length is a numeric value specified in the SEGM statement in the DBD.

  For a variable-length segment, the maximum segment size is taken as the data length.

  If a fixed length segment is defined with a segment edit/compression exit, the minimum length of the segment is the prefix length plus the minimum value of BYTES= in DBD.

For a fixed-length segment that has been made longer than the segment size in the DBD by the segment edit/compress facility, the segment size in the DBD is taken as the segment data length. The increase in the maximum length of the fixed-segment, which is edited by a segment edit/compression facility is ignored.

- Unused area in the VSAM CI that has enough length to contain one or more logical records.

FREE SPACE (NOT USABLE)
The sum of the following values and what percentage it makes of the total bytes:

- Unused area in a logical record that does not have enough length to contain the shortest dependent segment.
- Unused area in a VSAM CI that does not have enough length to contain a logical record.

UNKNOWN DATA
The number of bytes of unknown data, and what percentage it makes of the total bytes.

The unknown data is contiguous bytes that cannot be classified as segment, free space, DL/I OVERHEAD, or VSAM CI OVERHEAD.

DL/I OVERHEAD
The sum of the following values and what percentage it makes of the total bytes:

- The first 4 bytes in a logical record that are a direct-address pointer to the next logical record in the database record.
- A 1-byte segment code of 0, which shows the last segment in the logical record has been reached.
- The size of the first CI that is reserved for VSAM (only for VSAM)

VSAM CI OVERHEAD
The sum of RDFs and CIDFs in the VSAM CIs and what percentage it makes of the total bytes.

NUMBER OF LOGICAL RECORDS ERROR DETECTED
The number of logical records in which some error has been detected by HD Pointer Checker.

The third part of this report shows various totals for each segment type:

DBNAME DB# DSG# DSNAME
The name of the DBD, the database number (in hexadecimal), the data set group number (in hexadecimal), and the name of the primary data set.

ROOT SEGMENT NAME
The value coded on the NAME= keyword of the SEGM macro that defines the root segment.

ROOT SEG KEY NAME
The value coded on the NAME= keyword of the FIELD macro that defines the key field on the root segment.

ROOT SEG KEY LENGTH-1
The value coded on the BYTES= keyword (of the FIELD macro that defines the key field on the root segment) minus 1.

START POSITION OF KEY-1
The value coded on the START= keyword (of the FIELD macro that defines the key field on the root segment) minus 1.

SC
The segment code (in hexadecimal) of this segment.

SEGNAME
The segment name, as coded on the SEGM macro in the DBD, of this segment.

OCCUR
The number of occurrences of this segment in the data set group.

OVERFLW
The number of occurrences of this segment in the overflow (ESDS or OSAM) part of the data set group.

PL-DLTS
The number of occurrences of this segment that is flagged as both physically and logically deleted (bits 5 and 6 of the delete byte are on).

P-DLETS
The number of occurrences of this segment that is flagged as physically and not logically deleted (bit 5 of the delete byte is on, and bit 6 is off).

L-DLETS
The number of occurrences of this segment that is flagged as logically and not physically deleted (bit 6 of the delete byte is on, and bit 5 is off).

PFX LNG
The number of bytes (in decimal) of the prefix part of this segment.

DAT LNG
The number of bytes (in decimal) of the data part of this segment. If this number is followed by a **V**, then it indicates that this is a variable-length segment and that this is the average number of bytes in the data part of this segment. If DAT LNG is an average value, then so is SEG LEN.

SEG LNG
The total number of bytes (in decimal) of this segment. For a variable-length segment, this is the average segment length.

RULES
The rules used for insertion, deletion, and replacement of occurrences of this segment type:

I The path type that must be used to insert occurrences of this type (for segments participating in a logical relationship)

D The path type that must be used to delete occurrences of this type (for segments participating in a logical relationship)

R The path type that must be used to replace occurrences of this type (for segments participating in a logical relationship)

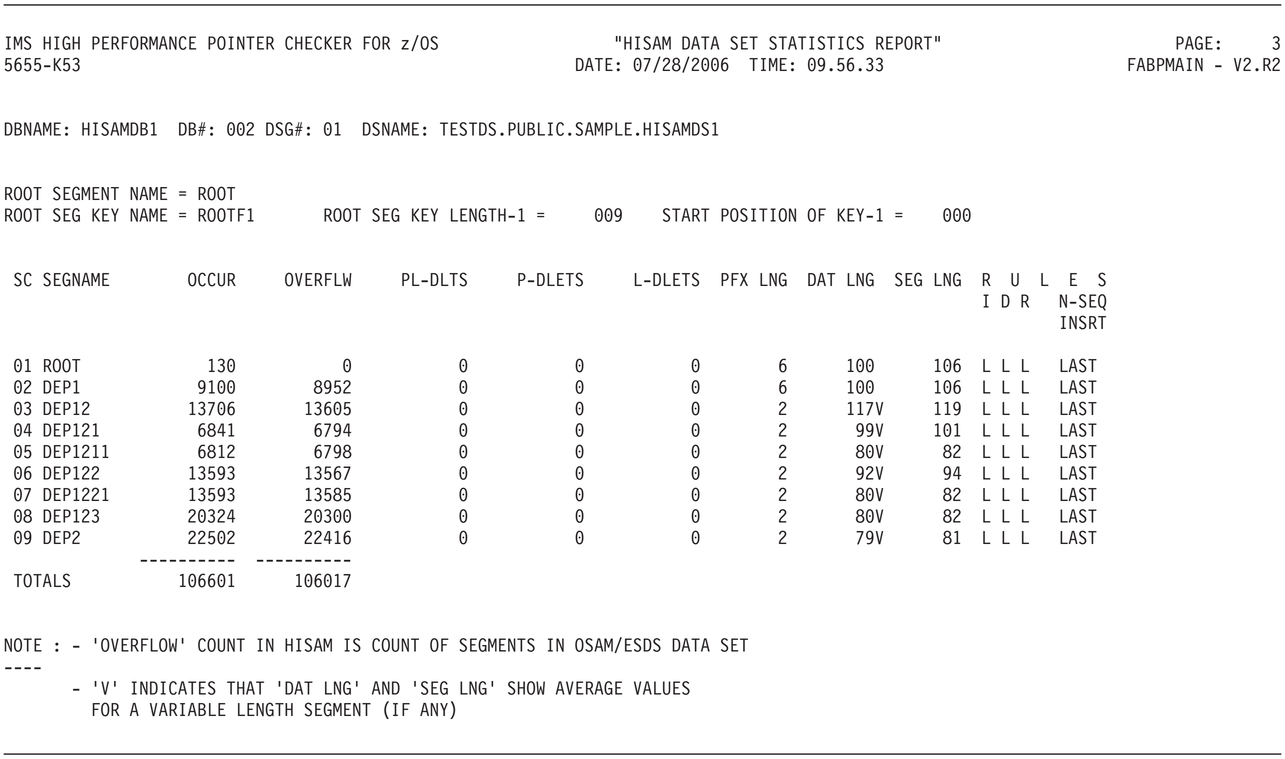
N-SEQ INSRT
Where new occurrences of this segment type are inserted into their physical database (for segments with no sequence field or a nonunique sequence field).

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS              "HISAM DATA SET STATISTICS REPORT"                    PAGE:     1
5655-K53                                               DATE: 07/28/2006  TIME: 09.56.33                        FABPMAIN - V2.R2

DBNAME: HISAMDB1  DB#: 002 DSG#: 01

DATA SET SUMMARY(PRIMARY)
-------------------------
  DBD NAME                         = HISAMDB1
  DB NUMBER                        = 002
  DATABASE ORGANIZATION            = HISAM
  ACCESS METHOD                    = VSAM KSDS
  DDNAME OF DATA SET               = HISAMDS1
  DSNAME OF DATA SET               = TESTDS.PUBLIC.SAMPLE.HISAMDS1
  BLOCK SIZE                       =  8,192
  RECORD SIZE                      =    510
  DATA SET CREATION DATE           = 07/06/2006
                  TIME             = N / A

DATA SET SUMMARY(OVERFLOW)
--------------------------
  ACCESS METHOD                    = VSAM ESDS
  DDNAME OF DATA SET               = HISAMDS2
  DSNAME OF DATA SET               = TESTDS.PUBLIC.SAMPLE.HISAMDS2
  BLOCK SIZE                       =  8,192
  RECORD SIZE                      =    512
  DATA SET CREATION DATE           = 07/06/2006
                  TIME             = 17:35:40
```

Figure 28. STATIPRT—HISAM Data Set Statistics report (Part 1 of 3)

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS              "HISAM DATA SET STATISTICS REPORT"                    PAGE:     2
5655-K53                                               DATE: 07/28/2006  TIME: 09.56.33                        FABPMAIN - V2.R2

DBNAME: HISAMDB1  DB#: 002 DSG#: 01

SPACE USE ANALYSIS(PRIMARY)
---------------------------
  TOTAL NUMBER OF LOGICAL RECORDS              =         130
  NUMBER OF LOGICAL RECORDS WITH FREE SPACE    =           5
  SEGMENT SIZE RANGE FOR THIS DATA SET(FROM DBD) =       102 TO    177
  TOTAL BYTES OF SPACE                         =        73,728  100.0 %  0.000 GB (   0.0 % OF 4 GB LIMIT )
  SEGMENT PREFIX                               =         2,280    3.1 %
  SEGMENT DATA                                 =        57,377   77.8 %
  SEGMENT PAD                                  =             0    0.0 %
  FREE SPACE (USABLE)                          =         7,747   10.5 %
  FREE SPACE (NOT USABLE)                      =         5,584    7.6 %
  UNKNOWN DATA                                 =             0    0.0 %
  DL/I OVERHEAD                                =           650    0.9 %
  VSAM CI OVERHEAD                             =            90    0.1 %
  NUMBER OF LOGICAL RECORDS ERROR DETECTED     =             0

SPACE USE ANALYSIS(OVERFLOW)
----------------------------
  TOTAL NUMBER OF LOGICAL RECORDS              =      21,260
  NUMBER OF LOGICAL RECORDS WITH FREE SPACE    =       1,158
  SEGMENT SIZE RANGE FOR THIS DATA SET(FROM DBD) =       102 TO    177
  TOTAL BYTES OF SPACE                         =    11,624,448  100.0 %  0.011 GB (   0.3 % OF 4 GB LIMIT )
  SEGMENT PREFIX                               =       247,842    2.1 %
  SEGMENT DATA                                 =     9,474,932   81.5 %
  SEGMENT PAD                                  =             0    0.0 %
  FREE SPACE (USABLE)                          =       162,847    1.4 %
  FREE SPACE (NOT USABLE)                      =     1,610,155   13.9 %
  UNKNOWN DATA                                 =             0    0.0 %
  DL/I OVERHEAD                                =       114,492    1.0 %
  VSAM CI OVERHEAD                             =        14,180    0.1 %
  NUMBER OF LOGICAL RECORDS ERROR DETECTED     =             0
```

Figure 28. STATIPRT—HISAM Data Set Statistics report (Part 2 of 3)



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS              "HISAM DATA SET STATISTICS REPORT"                  PAGE:     3
5655-K53                                               DATE: 07/28/2006  TIME: 09.56.33                     FABPMAIN - V2.R2


DBNAME: HISAMDB1  DB#: 002 DSG#: 01  DSNAME: TESTDS.PUBLIC.SAMPLE.HISAMDS1


ROOT SEGMENT NAME = ROOT
ROOT SEG KEY NAME = ROOTF1       ROOT SEG KEY LENGTH-1 =     009    START POSITION OF KEY-1 =    000


 SC SEGNAME        OCCUR     OVERFLW     PL-DLTS     P-DLETS     L-DLETS  PFX LNG  DAT LNG  SEG LNG  R  U  L  E  S
                                                                                                     I D R   N-SEQ
                                                                                                             INSRT

 01 ROOT             130           0           0           0           0        6      100      106  L L L   LAST
 02 DEP1            9100        8952           0           0           0        6      100      106  L L L   LAST
 03 DEP12          13706       13605           0           0           0        2      117V     119  L L L   LAST
 04 DEP121          6841        6794           0           0           0        2       99V     101  L L L   LAST
 05 DEP1211         6812        6798           0           0           0        2       80V      82  L L L   LAST
 06 DEP122         13593       13567           0           0           0        2       92V      94  L L L   LAST
 07 DEP1221        13593       13585           0           0           0        2       80V      82  L L L   LAST
 08 DEP123         20324       20300           0           0           0        2       80V      82  L L L   LAST
 09 DEP2           22502       22416           0           0           0        2       79V      81  L L L   LAST
               ----------  ----------
 TOTALS           106601      106017


NOTE : - 'OVERFLOW' COUNT IN HISAM IS COUNT OF SEGMENTS IN OSAM/ESDS DATA SET
----
       - 'V' INDICATES THAT 'DAT LNG' AND 'SEG LNG' SHOW AVERAGE VALUES
         FOR A VARIABLE LENGTH SEGMENT (IF ANY)
```

Figure 28. STATIPRT—HISAM Data Set Statistics report (Part 3 of 3)

HISAM Segment Level Statistics report

This report (see Figure 29 on page 132) contains various totals for each segment type in the HISAM. (including SHISAM) data set group. This report is produced for each data set specified on the DATABASE statement in the PROCCTL data set.

The report fields are as follows:

DBNAME DB# DSG# DSNAME
The name of the DBD, the database number (in hexadecimal), the data set group number (in hexadecimal), and the name of the data set.

SOURCE
The segment that contains the pointer (also called the source of the pointer). The following four fields all pertain to it:

DB
The database number (in hexadecimal) that identifies the database containing the segment that contains the pointer

DG
The data set group number (in hexadecimal) that identifies the database containing the segment that contains the pointer

SC
The segment code (in hexadecimal) of the segment that contains the pointer

SEGNAME
The segment name, as coded in the SEGM macro in the DBD, of the segment that is the target of a pointer.

PTR
The type of pointer such as LP, LTF, LTB, LCF, and LCL.

TARGET
The target of a pointer. The following four fields all pertain to it:

DB
The database number (in hexadecimal) that identifies the database containing the target of a pointer

DG
The data set group number (in hexadecimal) that identifies the database containing the target of a pointer

SC
The segment code (in hexadecimal) of the target of a pointer

SEGNAME
The segment name, as coded in the SEGM macro in the DBD, of the segment that is the target of a pointer.

EXTERNAL
The number of pointers that point to another data set.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS               "HISAM SEGMENT LEVEL STATISTICS REPORT"                       PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                         FABPMAIN - V2.R2


DBNAME: HISAMDB1  DB#: 002 DSG#: 01  DSNAME:  TESTDS.PUBLIC.SAMPLE.HISAMDS1


<---- SOURCE ---->     <---- TARGET ---->
DB  DG SC SEGNAME  PTR DB  DG SC SEGNAME  EXTERNAL


002 01 01 ROOT
                   CTR                        9100
                   *LC 001 01 02 DEP1

002 01 02 DEP1
   (PHYS. PAIRED)   LP 001 01 01 ROOT         9100

002 01 03 DEP12        NO POINTERS

002 01 04 DEP121       NO POINTERS

002 01 05 DEP1211      NO POINTERS

002 01 06 DEP122       NO POINTERS

002 01 07 DEP1221      NO POINTERS

002 01 08 DEP123       NO POINTERS

002 01 09 DEP2         NO POINTERS

THE NUMBER OF DIRECT-ADDRESS POINTERS POINTING TO LOGICAL RECORDS IN THE OVERFLOW DATA SET
  IN PRIMARY DATA SET  =         130
  IN OVERFLOW DATA SET =      21,130



NOTE : THE *LP AND *LC ENTRIES (IF ANY) COMPLETE THE DISPLAY OF ALL LOGICAL RELATIONSHIPS
----
           *LP SHOWS THAT THE SOURCE (LCHILD) HAS A SYMB LP PTR TO ITS TARGET (LPARENT)
           *LC SHOWS THAT THE SOURCE (LPARENT) HAS A CTR FIELD (THE NUMBER OF LOGICAL CHILDREN)
```

Figure 29. STATIPRT—HISAM Segment Level Statistics report

Interval Statistics report

This report (see Figure 30 on page 134) contains various totals for each segment type in the HIDAM or HDAM data set group. The report is produced each time an interval (defined by INTERVAL= keyword on the OPTION statement) is processed. Each value in the report is the cumulative value of each interval that has been reported before, and will be accumulated to the next report. It is produced unless INTST= NO is specified on the REPORT statement as input for the PROCCTL data set. This report is available only for an HD database.

The report fields are as follows:

DBNAME DB# PARTNAME PART ID REORG# DSG# DDNAME DSNAME DBORG

The name of the DBD, the database number (in hexadecimal), the name of the partition, partition id, partition reorg number, the data set group number (in hexadecimal) or the data set group ID (in an alphabetical character), the ddname of this data set group, the name of the data set, and the organization type of the data set.

INTERVAL: START BLOCK, END BLOCK

The beginning number and the ending number of the interval block to be reported.

SC

The segment code (in hexadecimal) of this segment.

SEGNAME

The segment name, as coded on the SEGM macro in the DBD, of this segment.

BASE

The information for the occurrences, the prefix, and the data part of this segment in the base part of the data set group.

OVERFLOW

The information for the occurrences, the prefix, and the data part of this segment in the overflow (ESDS or OSAM) part of the data set group is described. For HDAM, overflow blocks are those whose addresses are larger than the maximum that the randomizing routine can return. For HIDAM, overflow blocks are those whose RBAs are greater than that of the block that contains the root segment whose sequence field is all high values.

The number of occurrences in OVERFLOW is applicable to a primary data set group only.

The following three fields pertain to each of BASE and OVERFLOW:

OCCURRENCES

The number of occurrences of this segment in the data set group.

SPLIT-PRFX

The number of prefix parts of this segment whose prefix and data parts are separated.

SPLIT-DATA

The number of data parts of this segment whose prefix and data parts are separated.

PL-DLETS

The number of occurrences of this segment that are flagged as both physically and logically deleted (bits 5 and 6 of the delete byte are on).

P-DLETS
The number of occurrences of this segment that are flagged as physically and not logically deleted (bit 5 of the delete byte is on, and bit 6 is off).

L-DLETS
The number of occurrences of this segment that are flagged as logically and not physically deleted (bit 6 of the delete byte is on, and bit 5 is off).

RULES
The rules used for insertion, deletion, and replacement of occurrences of this segment type:

I The path type that must be used to insert occurrences of this type (for segments participating in a logical relationship)

D The path type that must be used to delete occurrences of this type (for segments participating in a logical relationship)

R The path type that must be used to replace occurrences of this type (for segments participating in a logical relationship)

INSERT
Where new occurrences of this segment type are inserted into their physical database (for segments with no sequence field or a nonunique sequence field).

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                     "INTERVAL STATISTICS REPORT"                           PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                         FABPMAIN - V2.R2


DBNAME: HDAMDB2   DB#: 001 DSG#: 01  DDNAME: HDAMDS4   DSNAME: TESTDS.PUBLIC.SAMPLE.HDAMDS4                  DBORG: HDAM

INTERVAL: START BLOCK         1
          END   BLOCK       2474


             -------------- BASE --------------  ------------ OVERFLOW ------------                                 ---- RULES ----
SC SEGNAME   OCCURRENCES SPLIT-PRFX  SPLIT-DATA  OCCURRENCES SPLIT-PRFX  SPLIT-DATA  PL-DLETS   P-DLETS   L-DLETS   I  D  R  INSERT

01 ROOT              70           0           0           0           0           0         0         0         0   L  L  L   LAST
02 DEP1             120           0           0        8980           0           0         0         0         0   L  L  L   LAST
03 DEP2              90           0           0        8827           0           0         0         0         0   L  L  L   LAST
             ----------  ----------  ----------  ----------  ----------  ----------
TOTALS              280           0           0       17807           0           0


NOTE :  - VIRTUAL SEGMENTS ARE NOT SHOWN
----
        - 'OVERFLOW' COUNT IN HDAM/PHDAM IS COUNT OF SEGMENTS BEYOND ROOT ADDRESSABLE AREA

        - 'OVERFLOW' COUNT IN HIDAM/PHIDAM IS COUNT OF SEGMENTS BEYOND HIGH RBA AT LAST REORGANIZATION OR INITIAL LOAD
```

Figure 30. STATIPRT—Interval Statistics report

Bit Map Display report

This report (see Figure 31 on page 136) contains the entire bit map (printed in binary) of all bit-map blocks for the database data set group. This report is produced unless BITMAP=NO is specified on the REPORT statement. This report is only available for an HD database.

The report fields are as follow:

DBNAME DB# PARTNAME PART ID REORG# DSG# DDNAME DSNAME DBORG

The name of the DBD, the database number (in hexadecimal), the name of the partition, partition id, partition reorg number, the data set group number (in hexadecimal) or the data set group ID (in an alphabetical character), the ddname of this data set group, the name of the data set, and the organization type of the data set.

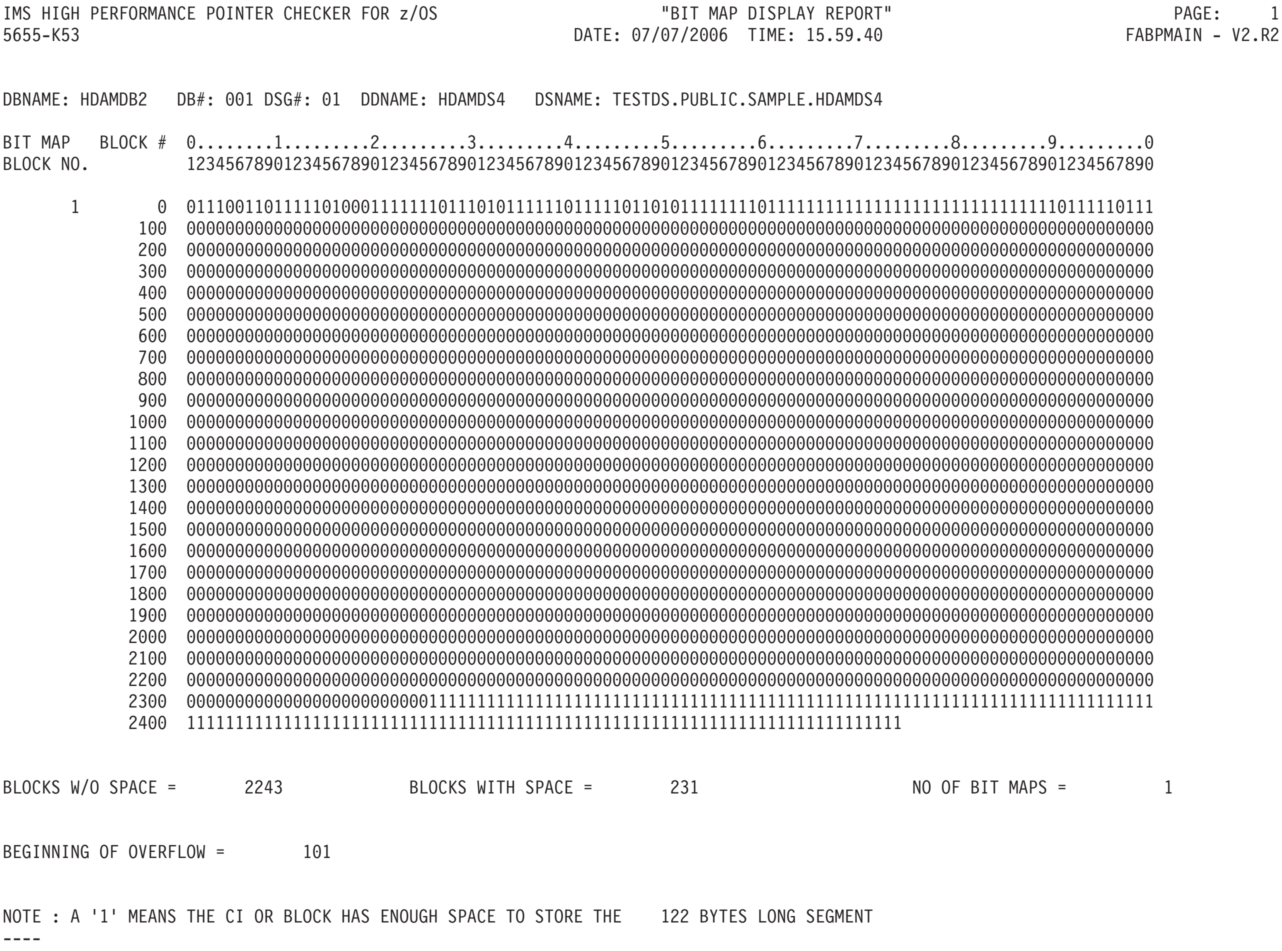
BIT MAP BLOCK NO.

The block number which contains a bit map.

The rest of the report is self-explanatory. The following field may be of special interest:

BEGINNING OF OVERFLOW

For a primary data set group, this field shows the block number where overflow blocks begin for HDAM databases. For secondary data set groups, this field shows the next block number of the current maximum block number.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                 "BIT MAP DISPLAY REPORT"                       PAGE:     1
5655-K53                                                DATE: 07/07/2006  TIME: 15.59.40                FABPMAIN - V2.R2


DBNAME: HDAMDB2   DB#: 001 DSG#: 01  DDNAME: HDAMDS4   DSNAME: TESTDS.PUBLIC.SAMPLE.HDAMDS4

BIT MAP   BLOCK #  0........1.........2.........3.........4.........5.........6.........7.........8.........9.........0
BLOCK NO.          1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

       1        0  0111001101111101000111111101110101111110111110110101111111101111111111111111111111111111111101111101111
              100  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
              200  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
              300  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
              400  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
              500  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
              600  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
              700  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
              800  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
              900  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
             1000  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
             1100  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
             1200  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
             1300  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
             1400  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
             1500  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
             1600  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
             1700  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
             1800  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
             1900  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
             2000  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
             2100  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
             2200  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
             2300  0000000000000000000000000111111111111111111111111111111111111111111111111111111111111111111111111111
             2400  11111111111111111111111111111111111111111111111111111111111111111111111111


BLOCKS W/O SPACE =       2243             BLOCKS WITH SPACE =       231                     NO OF BIT MAPS =          1


BEGINNING OF OVERFLOW =        101


NOTE : A '1' MEANS THE CI OR BLOCK HAS ENOUGH SPACE TO STORE THE    122 BYTES LONG SEGMENT
----
```

Figure 31. STATIPRT—Bit Map Display report

Free Space Map report

This report (see Figure 32 on page 138) contains information about the percentage of free space in each block or control interval (CI) in the database. This report is produced once per each data set group that is processed, unless FSEMAP=NO is specified on the REPORT statement as input for the PROCCTL data set. This report is only available for an HD database.

The report fields are as follow:

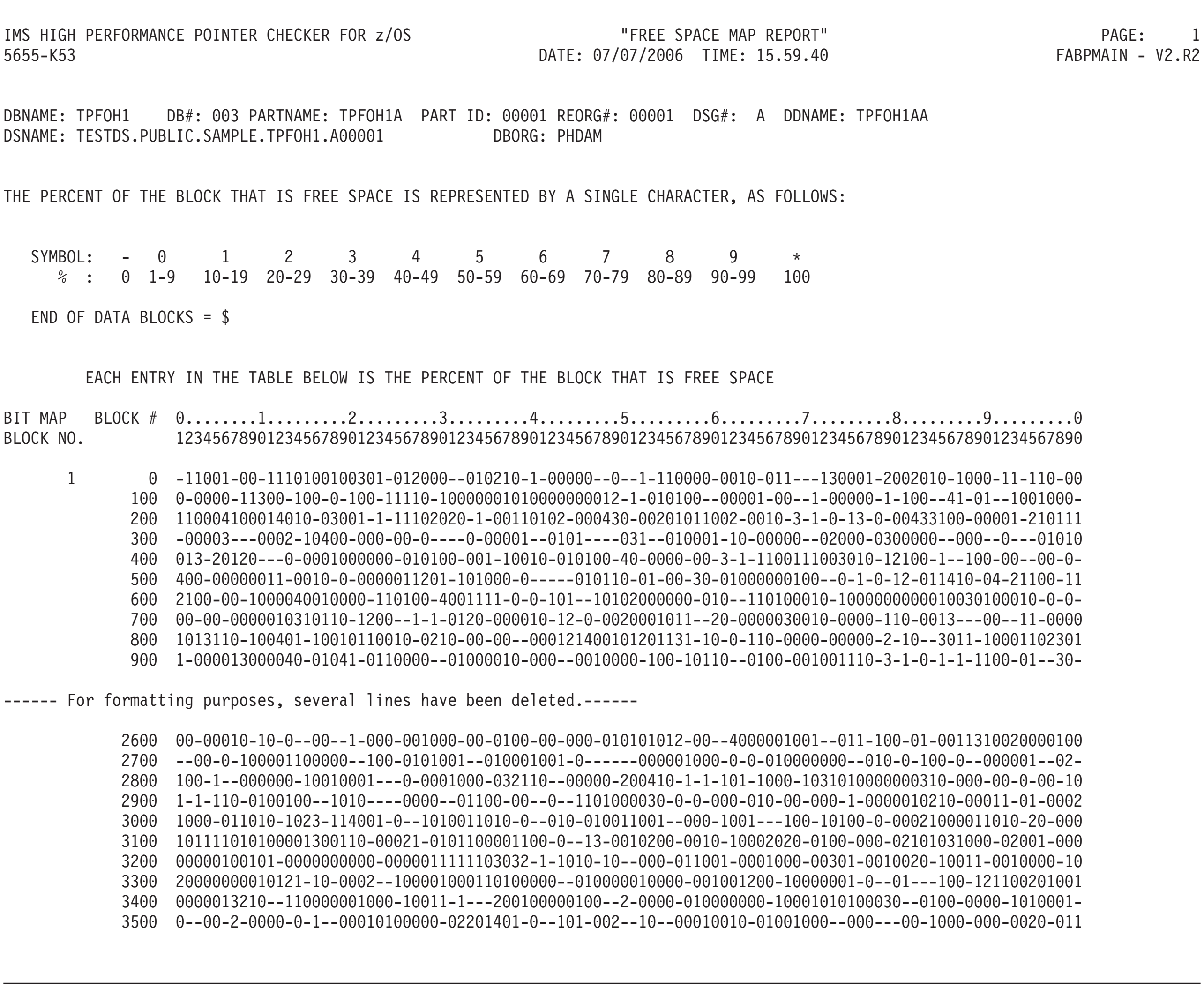
DBNAME DB# PARTNAME PART ID REORG# DSG# DDNAME DSNAME DBORG

The name of the DBD, the database number (in hexadecimal), the name of the partition, partition id, partition reorg number, the data set group number (in hexadecimal) or the data set group ID (in an alphabetical character), the ddname of this data set group, the name of the data set, and the organization type of the data set.

BIT MAP BLOCK NO.

The block number which contains a bit map.

The rest of the report is self-explanatory.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                       "FREE SPACE MAP REPORT"                            PAGE:      1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                        FABPMAIN - V2.R2


DBNAME: TPFOH1    DB#: 003 PARTNAME: TPFOH1A  PART ID: 00001 REORG#: 00001  DSG#:  A  DDNAME: TPFOH1AA
DSNAME: TESTDS.PUBLIC.SAMPLE.TPFOH1.A00001            DBORG: PHDAM


THE PERCENT OF THE BLOCK THAT IS FREE SPACE IS REPRESENTED BY A SINGLE CHARACTER, AS FOLLOWS:


   SYMBOL:   -   0      1      2      3      4      5      6      7      8      9      *
      %  :   0  1-9   10-19  20-29  30-39  40-49  50-59  60-69  70-79  80-89  90-99   100

   END OF DATA BLOCKS = $


         EACH ENTRY IN THE TABLE BELOW IS THE PERCENT OF THE BLOCK THAT IS FREE SPACE

BIT MAP   BLOCK #  0........1.........2.........3.........4.........5.........6.........7.........8.........9.........0
BLOCK NO.          1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

       1        0  -11001-00-1110100100301-012000--010210-1-00000--0--1-110000-0010-011---130001-2002010-1000-11-110-00
              100  0-0000-11300-100-0-100-11110-100000010100000000012-1-010100--00001-00--1-00000-1-100--41-01--1001000-
              200  110004100014010-03001-1-11102020-1-00110102-000430-00201011002-0010-3-1-0-13-0-00433100-00001-210111
              300  -00003---0002-10400-000-00-0----0-00001--0101----031--010001-10-00000--02000-0300000--000--0---01010
              400  013-20120---0-0001000000-010100-001-10010-010100-40-0000-00-3-1-1100111003010-12100-1--100-00--00-0-
              500  400-00000011-0010-0-0000011201-101000-0-----010110-01-00-30-01000000100--0-1-0-12-011410-04-21100-11
              600  2100-00-1000040010000-110100-4001111-0-0-101--10102000000-010--110100010-100000000010030100010-0-0-
              700  00-00-0000010310110-1200--1-1-0120-000010-12-0-0020001011--20-0000030010-0000-110-0013---00--11-0000
              800  1013110-100401-10010110010-0210-00-00--0001214001012011131-10-0-110-0000-00000-2-10--3011-10001102301
              900  1-000013000040-01041-0110000--01000010-000--0010000-100-10110--0100-001001110-3-1-0-1-1-1100-01--30-

------ For formatting purposes, several lines have been deleted.------

             2600  00-00010-10-0--00--1-000-001000-00-0100-00-000-010101012-00--4000001001--011-100-01-0011310020000100
             2700  --00-0-100001100000--100-0101001--010001001-0------000001000-0-0-010000000--010-0-100-0--000001--02-
             2800  100-1--000000-10010001---0-0001000-032110--00000-200410-1-1-101-1000-1031010000000310-000-00-0-00-10
             2900  1-1-110-0100100--1010----0000--01100-00--0--1101000030-0-0-000-010-00-000-1-0000010210-00011-01-0002
             3000  1000-011010-1023-114001-0--1010011010-0--010-010011001--000-1001---100-10100-0-00021000011010-20-000
             3100  1011110101000013001100-00021-0101100001100-0--13-0010200-0010-10002020-0100-000-02101031000-02001-000
             3200  00000100101-0000000000-0000011111103032-1-1010-10--000-011001-0001000-00301-0010020-10011-0010000-10
             3300  20000000010121-10-0002--1000010001101000000--010000010000-001001200-10000001-0--01---100-121100201001
             3400  0000013210--110000001000-10011-1---200100000100--2-0000-010000000-10001010100030--0100-0000-1010001-
             3500  0--00-2-0000-0-1--00010100000-02201401-0--101-002--10--00010010-01001000--000---00-1000-000-0020-011
```

Figure 32. STATIPRT—Free Space Map report (Part 1 of 2)

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                   "FREE SPACE MAP REPORT"                                  PAGE:     5
5655-K53                                                  DATE: 07/07/2006  TIME: 15.59.40                          FABPMAIN - V2.R2


DBNAME: TPFOH1    DB#: 003 PARTNAME: TPFOH1A  PART ID: 00001 REORG#: 00001  DSG#:  A DDNAME: TPFOH1AA
DSNAME: TESTDS.PUBLIC.SAMPLE.TPFOH1.A00001

BIT MAP   BLOCK #  0........1.........2.........3.........4.........5.........6.........7.........8.........9.........0
BLOCK NO.          1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

            18000  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
            18100  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
            18200  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
            18300  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
            18400  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
            18500  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
            18600  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
            18700  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
            18800  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
            18900  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
            19000  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
            19100  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
            19200  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
            19300  0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
            19400  0000000000000000000000111111111111111111111111111111111111111111111111111111111111111111111111111111
            19500  11111111111111111111111111111111111111111111111115**************************************************
            19600  ****************************************************************************************************
            19700  ****************************************************************************************************
            19800  ********************************************$
```

Figure 32. STATIPRT—Free Space Map report (Part 2 of 2)

Maximum Free Space Distribution report

This report (see Figure 33) contains the following information for each block or CI in the database:

- The length of the longest free space element in the block or CI
- The number of free space elements in the block or CI.

This report is produced unless MAXFSD= NO is specified on the REPORT statement as input for the PROCCTL data set. This report is only available for an HD database.

The report fields are as follows:

DBNAME DB# PARTNAME PART ID REORG# DSG# DDNAME DSNAME DBORG

The name of the DBD, the database number (in hexadecimal), the name of the partition, partition id, partition reorg number, the data set group number (in hexadecimal) or the data set group ID (in an alphabetical character), the ddname of this data set group, the name of the data set, and the organization type of the data set.

- Each entry in the map is six digits: a 5-digit FSE length followed by a 1-digit FSE count symbol.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS             "MAXIMUM FREE SPACE DISTRIBUTION REPORT"                 PAGE:     1
5655-K53                                                 DATE: 07/07/2006  TIME: 15.59.40                        FABPMAIN - V2.R2

DBNAME: HDAMDB2   DB#: 001 DSG#: 01  DDNAME: HDAMDS4   DSNAME: TESTDS.PUBLIC.SAMPLE.HDAMDS4

THE NUMBER OF FREE SPACE ELEMENTS (FSE'S) IN THE BLOCK IS REPRESENTED BY A SINGLE CHARACTER, AS FOLLOWS:


   SYMBOL:  BLANK  A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z  *
   COUNT :   0/1   2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28+


            EACH ENTRY IN THE TABLE BELOW IS THE LENGTH OF THE LONGEST FSE & THE NUMBER OF FSE'S IN THE BLOCK

 BLOCK #     1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17    18    19    20

       0     0  1008  1008  1008    44    80  1008  1008    68  1008   532  1008   544  1008    56  1008    68    68    56  1008
      20   544  1008  1008   532   532  1008    56   544   544   532    68  1008    56  1008   532   520   544  1008  1008    56
      40  1008   544   532   544   544    68   532  1008    44  1008    56   544  1008   532  1008  1008  1008  1008  1008    68
      60   544   532   544   532  1008   532  1008   532  1008   520  1008  1008   544   520  1008  1008  1008   544  1008   544
      80  1008  1008  1008  1008   544   544   532   532  1008  1008    56   544  1008   520  1008  1008    56   544  1008  1008
     100    84    84    84    72    84    84    84    96    84   108    84    96    84    84    72    84    96    84    84    96
     120    96    84    72    84    96    72    96    84    84    72    84    96    96    72    84    84    72    60    60    96
     140    72    60    84    84    84   108    72    84    96    72    84    84    72    84    60    96    96    72    96    84
     160    96    84    84    84    96    96    84    96    72    84    84    72   108    96    84    96    84    96    96    84
     180    84    72    72    60    96    96    84    84    60    96    96    84    72    96    84    96    72    96    84    96

------ For formatting purposes, several lines have been deleted.------

     600    72   108    84    84    60    84    72    96    72    72    84    84    96    96    60    72    72    96    84    84
     620    72    84    60   108    84    84    96    72    96    72    84    72    96    72    84    84    72    84    84    84
     640    84    72    84    72    84    96    72    96    96    96    84    84    96    72    72    96    96    60    84    84
     660    72    84    96    96    96    84    84    72    84    84    96    84    96    84    96    72    96    96    72    72
     680    96    72    84    84    84    72    84    48    96    84    96    84    96    96    96    84   108    84    84    72
     700    84    96    72    84    84    96    72    96    96    60    96    72    84    72    72    96    84    96    72    84
     720    60    84    60    72    84    84    84    96    72    84    96    84    84    84    84    84    96    96    60    84
     740    96    84    84    84    72    84    72    96    96    60    96    72    96    96    72    96    84    84    84    84
     760    84    84    72    60    96    72    96    84    96    84    96    96    60    84    84    84   108    72    72    84
     780    84    84    72    84    84    96    84    96    84    72    72    84    84    96    72    72    96    72    84    84
     800    84    72    84    72    84    96    96    96    84    96    96    60    96    72    84    96    84    84    96    96
```

Figure 33. STATIPRT—Maximum Free Space Distribution report (Part 1 of 2)

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS              "MAXIMUM FREE SPACE DISTRIBUTION REPORT"                PAGE:     3
5655-K53                                                  DATE: 07/07/2006  TIME: 15.59.40                      FABPMAIN - V2.R2


DBNAME: HDAMDB2   DB#: 001 DSG#: 01  DDNAME: HDAMDS4   DSNAME: TESTDS.PUBLIC.SAMPLE.HDAMDS4

 BLOCK #     1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17    18    19    20

    1820    60    84    84    60    84    96    60    96    96    72    96    72    96    72    96    72    84    72    84    72
    1840    72    60    84    36    84    96    84    72    72    84    84    72    84    84    72    96    72    84    72    84
    1860    84    96    84    96    84    84    84    84    84    84    72    72    84    84    96    72    96    84    84    96
    1880    84    84    84    72    84    84    96    96    96    84    84    84   108    84    96    84    84    84    84   108
    1900    72    84    96    84    96    96    84    84    84    96    84    72    84    84    72    96    84    96    84    60
    1920    96    96    84    96    72    96    84    72    84    84    96    72    96    84    72    84    96    84    96    84
    1940    96    72    96    72    84    84    96    96    96    72    96    72    96    72    84    84    96    72    84    84
    1960    60    96    96    96    72    96    72    84    96    96    84    60    84    96    72    84    84    96    96    72
    1980    96    96    72    84    72    84    84    84    96    84    84    72    84    72    72    96    84    84    84    84
    2000    96    84    84    72    84    84    84    84    72    84    72    72    96    84    84    84    84    84    96    84

------ For formatting purposes, several lines have been deleted.------

    2280    72    84    96    84    72    96    84    72    84    72    96    84    84    72    72    84    96    84    96    84
    2300    84    60    84    84    84    84    84    84    72    84    84    60    96    72    96    84    84    72    84    96
    2320    84    84    60    84    96   206  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012
    2340  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012
    2360  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012
    2380  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012
    2400  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012
    2420  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012
    2440  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012
    2460  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012  1012
```

Figure 33. STATIPRT—Maximum Free Space Distribution report (Part 2 of 2)

Interval Free Space Summary report

This report (see Figure 34 on page 143) contains a summary of the free space for each processing interval defined by INTERVAL= keyword on the OPTION statement in histogram format. The report is produced each time an interval (defined by INTERVAL= keyword on the OPTION statement) is processed. Each value in the report is the cumulative value of each interval that has been reported before, and will be accumulated to the next report. This report is produced unless INTFS= NO is specified on the REPORT statement as input for the PROCCTL data set. This report is separated into two pages. This report is only available for an HD database.

The report fields are as follows:

DBNAME DB# PARTNAME PART ID REORG# DSG# DDNAME DSNAME DBORG

The name of the DBD, the database number (in hexadecimal), the name of the partition, partition id, partition reorg number, the data set group number (in hexadecimal) or the data set group ID (in an alphabetical character), the ddname of this data set group, the name of the data set, and the organization type of the data set.

INTERVAL: START BLOCK, END BLOCK

The beginning number and the ending number of the interval block to be reported.

The following six fields are the column headings of the histograms:

FREE SPACE (BYTES)

This column defines the histogram classes. Each class is 20 bytes wide.

# OF FSE

The number of free space elements (whose lengths are in the current histogram class) that were found up to the end of this processing interval

Note: The sum of the entries in this column is the total number of free space elements detected by the SCAN processor.

# OF BLOCKS

The number of database blocks or CIs that contain free space elements (whose lengths are in the current histogram class) that were found up to the end of this processing interval

Note: Some blocks may contain more than one free space element. Since this may cause some blocks to be counted in more than one "# OF BLOCKS" entry, the sum of all the entries in this column may be greater than the total number of blocks processed.

TOTAL BYTES

The total number of bytes of free space from all free space elements (whose lengths are in the current histogram class) that were found up to the end of this processing interval

Note: The sum of the entries in this column is the total number of bytes of free space detected by the SCAN processor.

AVG FSE LEN

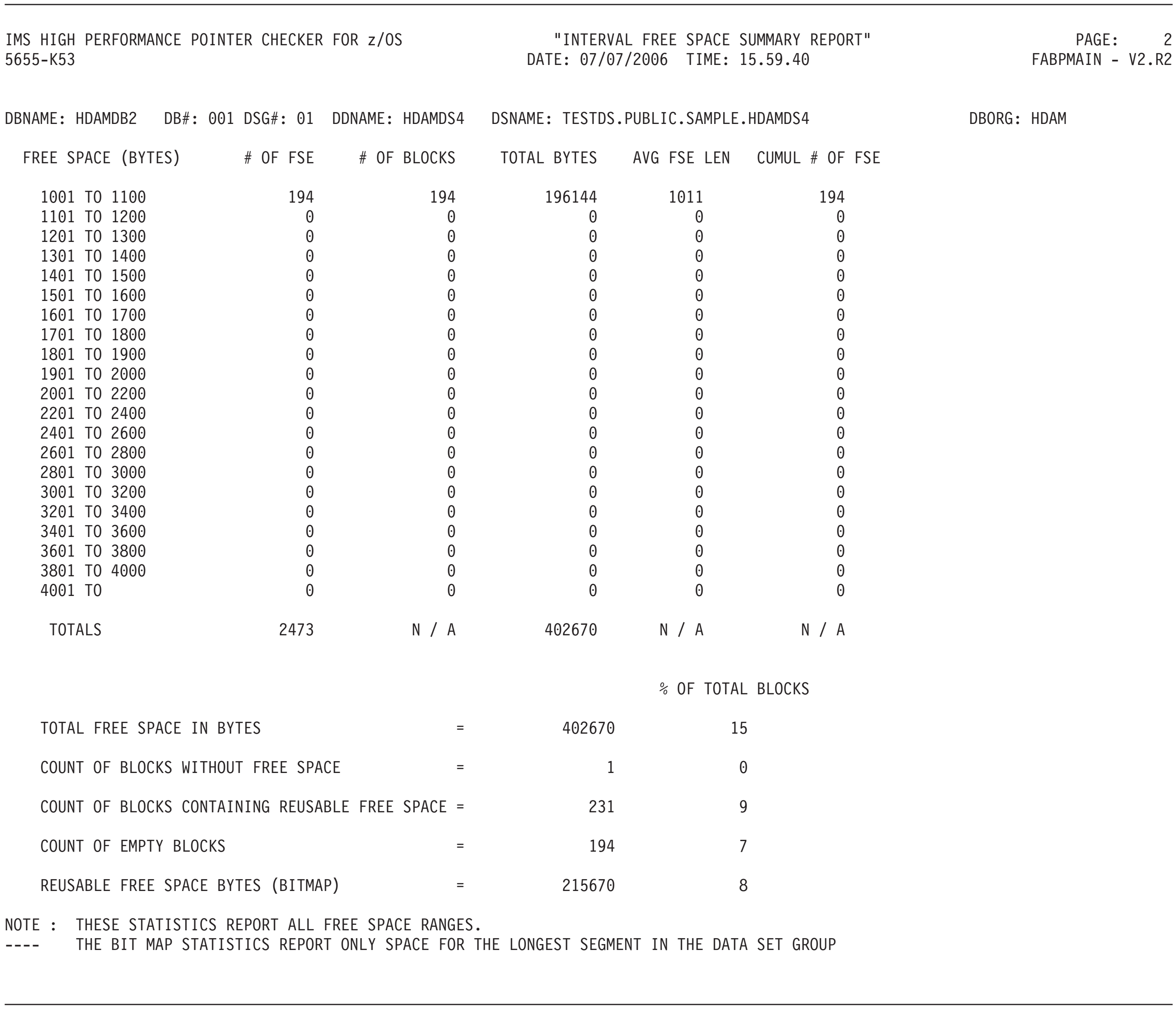
The average length of all free space elements (whose lengths are in the current histogram class) that were found up to the end of this processing interval

CUMUL # OF FSE
The number of free space elements (whose lengths are in the current histogram class or in a larger histogram class) that were found up to the end of this processing interval.

The remaining fields in this report are self-explanatory.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                 "INTERVAL FREE SPACE SUMMARY REPORT"                    PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                        FABPMAIN - V2.R2


DBNAME: HDAMDB2   DB#: 001 DSG#: 01  DDNAME: HDAMDS4   DSNAME: TESTDS.PUBLIC.SAMPLE.HDAMDS4                  DBORG: HDAM


INTERVAL: START BLOCK            1
          END   BLOCK         2474


  FREE SPACE (BYTES)       # OF FSE     # OF BLOCKS     TOTAL BYTES    AVG FSE LEN   CUMUL # OF FSE


     NO FREE SPACE                                1

       1 TO   20                  0               0               0           0            2473
      21 TO   40                  1               1              36          36            2473
      41 TO   60                130             130            7628          58            2472
      61 TO   80                510             510           36704          71            2342
      81 TO  100               1563            1563          138528          88            1832
     101 TO  120                 38              38            4104         108             269
     121 TO  140                  0               0               0           0             231
     141 TO  160                  0               0               0           0             231
     161 TO  180                  0               0               0           0             231
     181 TO  200                  0               0               0           0             231

------ For formatting purposes, several lines have been deleted.------

     501 TO  550                 36              36           19320         536             230
     551 TO  600                  0               0               0           0             194
     601 TO  650                  0               0               0           0             194
     651 TO  700                  0               0               0           0             194
     701 TO  750                  0               0               0           0             194
     751 TO  800                  0               0               0           0             194
     801 TO  850                  0               0               0           0             194
     851 TO  900                  0               0               0           0             194
     901 TO  950                  0               0               0           0             194
     951 TO 1000                  0               0               0           0             194
```

Figure 34. STATIPRT—Interval Free Space Summary report (Part 1 of 2)



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS              "INTERVAL FREE SPACE SUMMARY REPORT"                   PAGE:     2
5655-K53                                                DATE: 07/07/2006  TIME: 15.59.40                      FABPMAIN - V2.R2

DBNAME: HDAMDB2   DB#: 001 DSG#: 01  DDNAME: HDAMDS4   DSNAME: TESTDS.PUBLIC.SAMPLE.HDAMDS4                  DBORG: HDAM

  FREE SPACE (BYTES)       # OF FSE     # OF BLOCKS     TOTAL BYTES    AVG FSE LEN   CUMUL # OF FSE

    1001 TO 1100               194             194          196144        1011             194
    1101 TO 1200                 0               0               0           0               0
    1201 TO 1300                 0               0               0           0               0
    1301 TO 1400                 0               0               0           0               0
    1401 TO 1500                 0               0               0           0               0
    1501 TO 1600                 0               0               0           0               0
    1601 TO 1700                 0               0               0           0               0
    1701 TO 1800                 0               0               0           0               0
    1801 TO 1900                 0               0               0           0               0
    1901 TO 2000                 0               0               0           0               0
    2001 TO 2200                 0               0               0           0               0
    2201 TO 2400                 0               0               0           0               0
    2401 TO 2600                 0               0               0           0               0
    2601 TO 2800                 0               0               0           0               0
    2801 TO 3000                 0               0               0           0               0
    3001 TO 3200                 0               0               0           0               0
    3201 TO 3400                 0               0               0           0               0
    3401 TO 3600                 0               0               0           0               0
    3601 TO 3800                 0               0               0           0               0
    3801 TO 4000                 0               0               0           0               0
    4001 TO                      0               0               0           0               0

     TOTALS                   2473           N / A          402670       N / A           N / A

                                                                          % OF TOTAL BLOCKS

    TOTAL FREE SPACE IN BYTES                     =          402670           15

    COUNT OF BLOCKS WITHOUT FREE SPACE            =               1            0

    COUNT OF BLOCKS CONTAINING REUSABLE FREE SPACE =            231            9

    COUNT OF EMPTY BLOCKS                         =             194            7

    REUSABLE FREE SPACE BYTES (BITMAP)            =          215670            8

NOTE :  THESE STATISTICS REPORT ALL FREE SPACE RANGES.
----    THE BIT MAP STATISTICS REPORT ONLY SPACE FOR THE LONGEST SEGMENT IN THE DATA SET GROUP
```

Figure 34. STATIPRT—Interval Free Space Summary report (Part 2 of 2)

HD Data Set Statistics report

This report (see Figure 35 on page 150) contains the following parts:

- Data Set Group Summary: This part shows information about the database data set and the root segments. If the data set is an image copy, DSNAME OF DATA SET and DATA SET CREATION DATE TIME show the information about the image copy data set.
- Block/CI Summary: This part shows the status of the use of a block or CI.
- Space Use Analysis: This part shows the status of the use of the data set.
- Segment Occurrences/Split: This shows various totals for each segment type.

This report is only available for an HD database. This report provides information about the data set. If the database is HALDB, this report provides information about each partition.

The report fields are as follows:

DBNAME DB# PARTNAME PART ID REORG# DSG# DDNAME DSNAME DBORG
The name of the DBD, the database number (in hexadecimal), the name of the partition, partition id, partition reorg number, the data set group number (in hexadecimal) or the data set group ID (in an alphabetical character), the ddname of this data set group, the name of the data set, and the organization type of the data set.

DATA SET GROUP SUMMARY: This part shows information about the database data set. If the data set is an image copy, DSAME and DATA SET CREATION DATE TIME show the information about the image copy data set.

DBD NAME
The name of the DBD as coded on the NAME= keyword of the DBD macro.

DB NUMBER
The database number (in hexadecimal) used to identify the database throughout the HD Pointer Checker run.

DATABASE ORGANIZATION
The IMS organization used for this database (HDAM, HIDAM, PHDAM, or PHIDAM).

ACCESS METHOD
The access method used for this database: VSAM or OSAM.

PARTITION NAME
The name of this partition. It is printed for HALDB.

PARTITION ID
The ID of partition (in decimal) assigned by IMS. It is printed for HALDB.

NUMBER OF PARTITIONS IN DATABASE
The number of partitions (in decimal) defined in this HALDB.

PARTITION SELECTION EXIT NAME
The name of the user-supplied partition selection module, as coded on the PSNAME= keyword of the DBD macro, of this HALDB.

PARTITION SELECTION STRING
The partition selection string of this partition in this HALDB.

PARTITION HIGH KEY
The partition high key of this partition in the HALDB.

DDNAME OF DATA SET
The DD name of this database data set.

DATA SET GROUP NUMBER
This shows data set group number (in hexadecimal). It is only for non-HALDB.

DATA SET GROUP ID
This shows and the data set group ID (in an alphabetical character). It is only for HALDB.

NUMBER OF DATA SET GROUPS
This shows the total number of data set groups.

DSNAME OF DATA SET
The name of database data set, or if the processed data set is an image copy, the name of the image copy data set.

BLOCK SIZE
The block size for OSAM or CI size for VSAM.

RECORD SIZE
The data set record size.

DATA SET CREATION DATE TIME
The date and time when this data set was created. If the data set is an image copy, the date and time the image copy data set was made are given. If the data set is a real database of OSAM or a Fast Recovery image copy, the time shows "N/A".

The following fields are printed only for data set that contains root segments:

ROOT SEGMENT NAME
The value coded on the NAME= keyword of the SEGM macro that defines the root segment.

ROOT SEGMENT KEY NAME
The key name of the root segment.

ROOT SEGMENT KEY LENGTH-1
The value coded on the BYTES= keyword (of the FIELD macro that defines the key field on the root segment) minus 1.

ROOT SEGMENT KEY OFFSET
The offset from the segment start position to the key start position. The value coded on the START= keyword (of the FIELD macro that defines the key field on the root segment) minus 1.

DIRECT ALGORITHM NAME
The name of the user-supplied randomizing module, as coded on the RMNAME= keyword of the DBD macro. It is only for HDAM or PHDAM database.

HIGH BLOCK NUMBER
The maximum relative block number that the randomizing module is permitted to produce, as coded on the RMNAME= keyword of the DBD macro. It is only for HDAM or PHDAM database.

RAPS PER BLOCK
The number of root anchor points in each CI or each block in the root addressable area, as coded on the RMNAME= keyword of the DBD macro. It is only for HDAM or PHDAM database.

TOTAL RAPS
The number of RAPS in this database data set or partition. It is only for HDAM or PHDAM database.

BYTE LIMIT COUNT
The maximum number of bytes of database record that can be stored into the root addressable area in a series of inserts unbroken by a call to another database record, as coded on the RMNAME= keyword of the DBD macro. It is only for HDAM or PHDAM database.

The following two fields reflect the results of the SCAN process:

NUMBER OF KEY RECORDS WRITTEN
The number of records written by the SCAN processor to the KEYSIN data set (for use by the HD Tuning Aid utility).

COUNT OF RAPS NOT USED IN ALLOCATED RAA
The number of root anchor points in this data set group or partition that do not point to any root segment. It is only for HDAM or PHDAM database.

BLOCK/CI SUMMARY: This part shows the status of the use of a block or CI for each data set.

TOTAL NUMBER OF BLOCKS
The number of blocks or CIs read in this data set group or partition by the SCAN processor.

COMPLETELY FULL (NO FSE)
The number of blocks or CIs that contain no free space element, and what percentage it makes of the total.

PARTIALLY FULL (1 OR MORE SEGMENTS)
The number of blocks or CIs that contain both segment data and free space elements, and what percentage it makes of the total.

EMPTY (FORMATTED BUT NO SEGMENTS)
The number of blocks or CIs that are formatted and contain no segment data, and what percentage it makes of the total.

UNUSED (NOT FORMATTED)
The number of blocks or CIs that are not used by DL/I, and what percentage it makes of the total.

VSAM BLOCK 0
The first block of a VSAM data set. It is only for VSAM.

BLOCKS POINTER OR T2 ERROR DETECTED
The number of blocks or CIs that contain pointer errors or T2 errors that were detected by the scan processor.

BLOCKS WITH SLACK BYTES
The number of blocks or CIs containing slack bytes that were detected by the SCAN processor. Slack bytes represent an area in the data part of a segment that could not be classified as either segments or free space. The length of slack bytes is equal to or less than the T2len specification in OPTION T2CHK=.

NUMBER OF SLACK BYTES
The number of blocks with slack bytes whose length is less than that specified by OPTION T2CHK=.

BLOCKS ADDED SINCE LAST LOAD/RELOAD
The number of blocks or CIs added after the initial load or the last

reorganization. They are in a block or a CI higher than the high key root segment. It is only for HIDAM or PHIDAM.

SPACE USE ANALYSIS: This part shows the status of the space use for each data set.

TOTAL NUMBER OF BLOCKS
The number of blocks or CIs scanned by the SCAN processor in this data set group or partition.

NUMBER OF BLOCKS WITH FREE SPACE
The number of blocks in this data set group or partition that contain at least one free space element each.

NUMBER OF FREE SPACE ELEMENTS
The number of free space elements in this data set group or partition.

NUMBER OF FSE THAT WILL HOLD LARGEST SEGMENT
The number of free space elements (FSEs) that can hold the largest segment in this data set group or partition. The largest segment indicates the segment that has the longest value in SEGMENT SIZE RANGE FOR THIS DSG (FROM DBD).

NUMBER OF FSE TOO SMALL FOR SMALLEST SEGMENT
The number of FSEs whose lengths are less than the smallest segment in this data set group or partition. These spaces will not be used by DL/I until they are reclaimed. See “How IMS reclaims space” on page 284. The smallest segment is the segment that has the shortest value in SEGMENT SIZE RANGE FOR THIS DSG (FROM DBD).

SEGMENT SIZE RANGE FOR THIS DSG (FROM DBD)
The maximum and minimum segment lengths of the segments in this data set group.

The segment length is the prefix length plus the data length. The data length is a numeric value specified in the BYTES= keyword in the SEGM statement in the DBD. For a variable-length segment, the maximum segment size that is specified in the BYTES= keyword in the SEGM statement is taken as the data length.

If a segment edit/compression exit is defined, by the COMPRTN= keyword, to a segment, the actual segment length can be longer or shorter than the length in the BYTES= keyword. However, HD Pointer Checker uses the length in the BYTES= keyword, as previously described, to calculate the maximum and minimum segment lengths in this report. For example, even if the segment edit/compression routine is defined to a fixed-length segment, and it makes segments with segment length that is longer than the segment length in the BYTES= keyword, HD Pointer Checker uses the length in the BYTES= keyword.

FREE SPACE BLOCK EVERY N BLOCKS (FROM DBD)
The "free block frequency factor", as coded on the FRSPC= keyword of the DATASET macro in the DBD. It specifies that every N-th CI or block in this data set group is left as free space during the load or reorganization of a database.

% FREE SPACE WITHIN EACH BLOCK (FROM DBD)
The free space percentage factor. It is specified by FRSPC= in the DBD DATASET statement. It shows the percentage of free space size in each block or CI.

FREE SPACE SCAN CYLINDERS (FROM DBD)
The number of direct access device cylinders to be scanned when searching for

available storage space during segment insertions, as coded in the SCAN= keyword of the DATASET macro in the DBD.

TOTAL BYTES OF SPACE

The following items are shown:

- The database data set size in bytes
- The percentage of the database data set size to the total size (always 100.0%)
- The database data set size in giga bytes
- The percentage of database data set size to the maximum size
- The database data set size limit

SEGMENT PREFIX

The number of bytes of segment prefix and what percentage it makes of the total bytes.

SEGMENT DATA

The number of bytes of segment data and what percentage it makes of the total bytes.

SEGMENT PAD

The number of bytes of segment pad and what percentage it makes of the total bytes. Within a block or a CI, the starting point of a segment or a free space element is always at an even boundary. Therefore, if a segment length is odd, one byte is added before the next segment or free space element. This one byte is called the *segment pad*. If the variable segment length is less than the MIN value in the DBD, the difference is also counted as a segment pad.

FREE SPACE (USABLE)

The number of bytes of usable free space and what percentage it makes of the total bytes. The free space usable indicates that the length of the free space is sufficient to store the minimum length segment in this data set group or partition. The minimum segment length is the shortest value in "SEGMENT SIZE RANGE FOR THIS DSG (FROM DBD)".

FREE SPACE (NOT USABLE)

The number of bytes of free space not usable and what percentage it makes of the total bytes. Free space is not usable if its length is less than the minimum length segment in this data set group or partition.

SLACK

The number of bytes of slack and what percentage it makes of the total. The slack bytes make up an area in the data part of a segment that can be classified as neither segments nor free space. The length of the slack bytes is equal to or less than the T2len specification in OPTION T2CHK=. Additionally, for the VSAM, one bytes precedes each RDF or CIDF fields that is at the bottom of each CI. This one byte is counted with the slack bytes.

UNKNOWN DATA

The number of bytes of unknown data, and what percentage it makes of the total. The unknown data are contiguous bytes that cannot be classified as either segments or free space and that are longer than the T2len specification in OPTION T2CHK=.

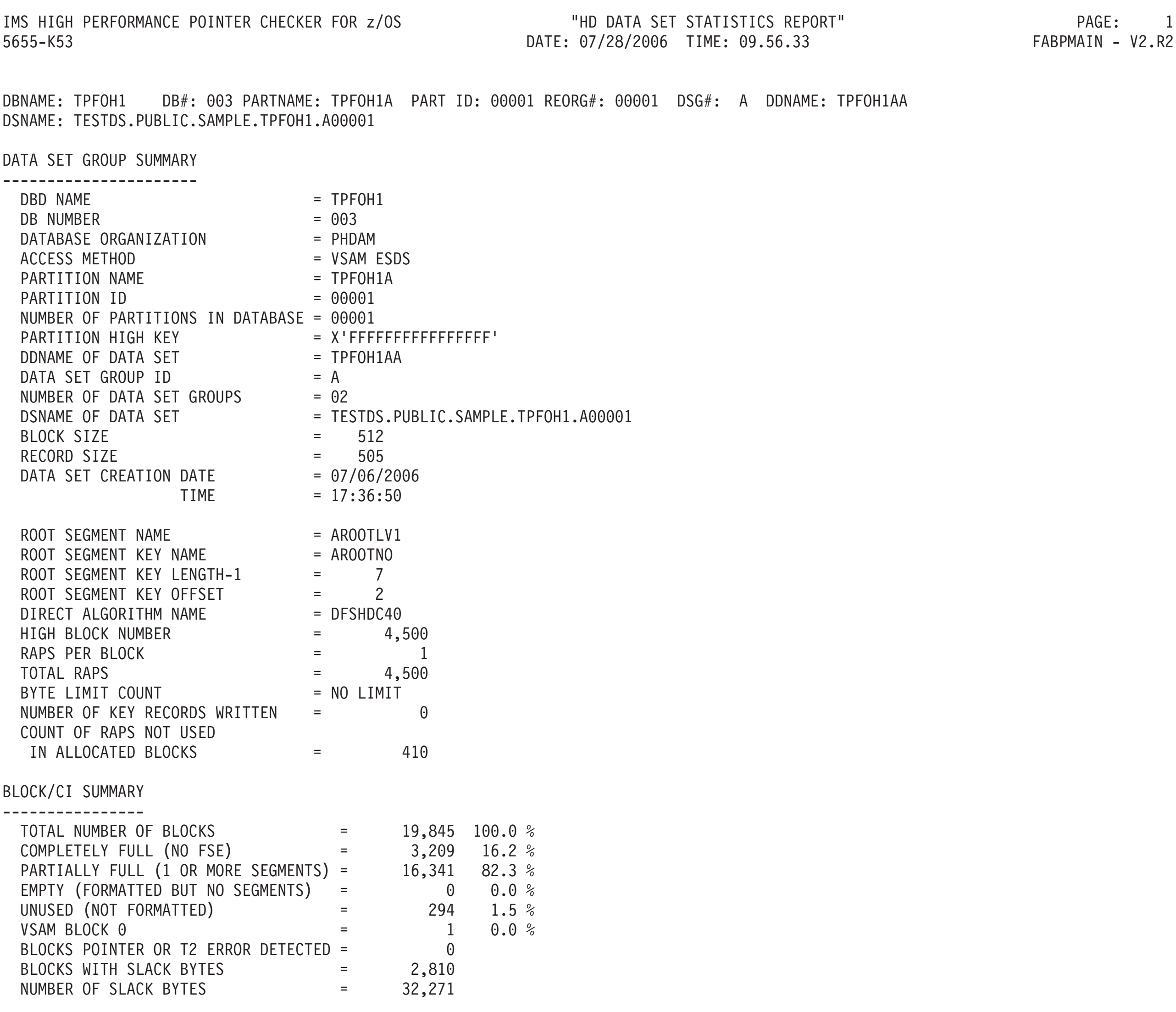
DL/I OVERHEAD

Total sum of the following bytes and what percentage it makes of the total:

- Bit map block
- FSEAP: Four bytes for each block or CI
- RAP: Four bytes for each RAP

- Pointer from the prefix of the split segment to the data: four bytes for each split segment
- Segment code and delete byte in the data of a split segment: two bytes for each split segment
- The size of the first CI that is reserved for VSAM (only for VSAM)

VSAM CI OVERHEAD (Only for VSAM)
The sum of the bytes in the RDF and CIDF fields which are the last seven length fields in each CI.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                   "HD DATA SET STATISTICS REPORT"                          PAGE:     1
5655-K53                                                   DATE: 07/28/2006  TIME: 09.56.33                         FABPMAIN - V2.R2


DBNAME: TPFOH1    DB#: 003 PARTNAME: TPFOH1A  PART ID: 00001 REORG#: 00001  DSG#:  A  DDNAME: TPFOH1AA
DSNAME: TESTDS.PUBLIC.SAMPLE.TPFOH1.A00001

DATA SET GROUP SUMMARY
----------------------
  DBD NAME                         = TPFOH1
  DB NUMBER                        = 003
  DATABASE ORGANIZATION            = PHDAM
  ACCESS METHOD                    = VSAM ESDS
  PARTITION NAME                   = TPFOH1A
  PARTITION ID                     = 00001
  NUMBER OF PARTITIONS IN DATABASE = 00001
  PARTITION HIGH KEY               = X'FFFFFFFFFFFFFFFF'
  DDNAME OF DATA SET               = TPFOH1AA
  DATA SET GROUP ID                = A
  NUMBER OF DATA SET GROUPS        = 02
  DSNAME OF DATA SET               = TESTDS.PUBLIC.SAMPLE.TPFOH1.A00001
  BLOCK SIZE                       =    512
  RECORD SIZE                      =    505
  DATA SET CREATION DATE           = 07/06/2006
                    TIME           = 17:36:50

  ROOT SEGMENT NAME                = AROOTLV1
  ROOT SEGMENT KEY NAME            = AROOTNO
  ROOT SEGMENT KEY LENGTH-1        =      7
  ROOT SEGMENT KEY OFFSET          =      2
  DIRECT ALGORITHM NAME            = DFSHDC40
  HIGH BLOCK NUMBER                =       4,500
  RAPS PER BLOCK                   =           1
  TOTAL RAPS                       =       4,500
  BYTE LIMIT COUNT                 = NO LIMIT
  NUMBER OF KEY RECORDS WRITTEN    =           0
  COUNT OF RAPS NOT USED
   IN ALLOCATED BLOCKS             =         410

BLOCK/CI SUMMARY
----------------
  TOTAL NUMBER OF BLOCKS              =      19,845  100.0 %
  COMPLETELY FULL (NO FSE)            =       3,209   16.2 %
  PARTIALLY FULL (1 OR MORE SEGMENTS) =      16,341   82.3 %
  EMPTY (FORMATTED BUT NO SEGMENTS)   =           0    0.0 %
  UNUSED (NOT FORMATTED)              =         294    1.5 %
  VSAM BLOCK 0                        =           1    0.0 %
  BLOCKS POINTER OR T2 ERROR DETECTED =           0
  BLOCKS WITH SLACK BYTES             =       2,810
  NUMBER OF SLACK BYTES               =      32,271
```

Figure 35. STATIPRT—HD Data Set Statistics report (Part 1 of 3)

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS               "HD DATA SET STATISTICS REPORT"                      PAGE:     2
5655-K53                                               DATE: 07/28/2006  TIME: 09.56.33                     FABPMAIN - V2.R2


DBNAME: TPFOH1    DB#: 003 PARTNAME: TPFOH1A  PART ID: 00001 REORG#: 00001  DSG#:  A  DDNAME: TPFOH1AA
DSNAME: TESTDS.PUBLIC.SAMPLE.TPFOH1.A00001

SPACE USE ANALYSIS
------------------
  TOTAL NUMBER OF BLOCKS                          =      19,845
  NUMBER OF BLOCKS WITH FREE SPACE                =      16,635
  NUMBER OF FREE SPACE ELEMENTS                   =      16,763
  NUMBER OF FSE THAT WILL HOLD LARGEST SEGMENT    =         295
  NUMBER OF FSE TOO SMALL FOR SMALLEST SEGMENT    =      15,018
  SEGMENT SIZE RANGE FOR THIS DSG (FROM DBD)      =         130 TO    246
  FREE SPACE BLOCK EVERY N BLOCKS (FROM DBD)      =           0
  % FSPC WITHIN EACH BLOCK (FROM DBD)             =           0
  FREE SPACE SCAN CYLINDERS (FROM DBD)            = NO SCAN
  TOTAL BYTES OF SPACE                            =     10,160,640  100.0 %  0.009 GB (   0.2 % OF 4 GB LIMIT )
  SEGMENT PREFIX                                  =      2,279,194   22.4 %
  SEGMENT DATA                                    =      6,442,650   63.4 %
  SEGMENT PAD                                     =        257,778    2.5 %
  FREE SPACE (USABLE)                             =        418,212    4.1 %
  FREE SPACE (NOT USABLE)                         =        489,678    4.8 %
  SLACK                                           =         32,271    0.3 %
  UNKNOWN DATA                                    =              0    0.0 %
  DL/I OVERHEAD                                   =        101,949    1.0 %
  VSAM CI OVERHEAD                                =        138,908    1.4 %
```

Figure 35. STATIPRT—HD Data Set Statistics report (Part 2 of 3)

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS               "HD DATA SET STATISTICS REPORT"                      PAGE:     3
5655-K53                                               DATE: 07/28/2006  TIME: 09.56.33                     FABPMAIN - V2.R2


DBNAME: TPFOH1    DB#: 003 PARTNAME: TPFOH1A  PART ID: 00001 REORG#: 00001  DSG#:  A  DDNAME: TPFOH1AA
DSNAME: TESTDS.PUBLIC.SAMPLE.TPFOH1.A00001


             -------------- BASE --------------  ------------ OVERFLOW ------------                                ---- RULES ----
SC SEGNAME   OCCURRENCES SPLIT-PRFX  SPLIT-DATA  OCCURRENCES SPLIT-PRFX  SPLIT-DATA  PL-DLETS    P-DLETS   L-DLETS   I  D  R  INSERT

01 AROOTLV1        4270          0           0        6730          0           0          0          0         0  L  L  L   LAST
02 ADEP1LV2           5          0           0         215          0           0          0          0         0  L  L  L   LAST
04 ADEP3LV2        4485          0           0       12055          0           0          0          0         0  L  L  L   LAST
05 ADEP4LV3        4418          0           0       12099          0           0          0          0         0  L  L  L   LAST
06 ADEP5LV3        2250          0           0        6045          0           0          0          0         0  L  L  L   LAST
07 ADEP6LV4        5492        161           2       15081         99         258          0          0         0  L  L  L   LAST
08 ADEP7LV5           0          0           0        6892          0           0          0          0         0  L  L  L   LAST
             ----------  ----------  ----------  ----------  ----------  ----------
TOTALS            20920        161           2       59117         99         258


NOTE : - VIRTUAL SEGMENTS ARE NOT SHOWN
----
       - 'OVERFLOW' COUNT IN HDAM/PHDAM IS COUNT OF SEGMENTS BEYOND ROOT ADDRESSABLE AREA

       - 'OVERFLOW' COUNT IN HIDAM/PHIDAM IS COUNT OF SEGMENTS BEYOND HIGH RBA AT LAST REORGANIZATION OR INITIAL LOAD
```

Figure 35. STATIPRT—HD Data Set Statistics report (Part 3 of 3)

DB Record Distribution Statistics report

This report (see Figure 36 on page 154) contains the following information:

- Statistics about the locations of HDAM or PHDAM root segment
- How long and how the HDAM or PHDAM RAP chains are distributed
- The number of dependents stored in the same block or CI as their root segment
- The number of dependents stored in the same block or CI as their segment code, and the percentage of each that is included in the root block.

This report is produced unless you specify DBDIST= NO on the REPORT statement, or INCORE=NO on the OPTION statement. This report is only available for an HD database. This report provides information about the data set. If the database is HALDB, this report provides information about each partition.

The report fields are as follows:

DBNAME DB# PARTNAME PART ID REORG# DSG# DDNAME DSNAME DBORG

The name of the DBD, the database number (in hexadecimal), the name of the partition, partition id, partition reorg number, the data set group number (in hexadecimal) or the data set group ID (in an alphabetical character), the ddname of this data set group, the name of the data set, and the organization type of the data set.

MAXIMUM ROOTS PER BLOCK

The maximum number of root segments contained in one Block (OSAM DB) or CI (VSAM DB).

DISTRIBUTION OF ROOT SEGMENTS (HDAM/PHDAM ONLY): This part shows how the HDAM or PHDAM roots were distributed relative to their home block. If the distribution of the root segment part is requested with the OPTION HOMECHK=(YES,-nnn,+mmm) option in the PROCCTL statement, HD Pointer Checker generates it. If the database is PHDAM, HD Pointer Checker generates the part for each partition.

Note: When the prefix and data parts of an HDAM or a PHDAM root segment are separated, HD Pointer Checker uses the block that contains the data part as its location.

LOCATION

The block location, relative to the home block, where the root segment was found. The locations, within -200 to +200 block to the home block, are indicated by every one block. The locations, relatively lower than -200 and upper than +200 to the home block, are indicated by every 100 blocks.

NUMBER OF ROOTS

The number of roots that were placed within the associated relative block.

PERCENTAGE

The percentage of roots that were placed within the associated relative block.

DISTRIBUTION OF RAP CHAIN LENGTHS (HDAM/PHDAM ONLY): This part shows how long and how distributed the HDAM or PHDAM RAP chains are. HD Pointer Checker generates this part if so requested by the REPORT CHAINDIST option in the PROCCTL statement.

CHAIN LENGTH

The number of roots that is randomized to a particular RAP. A chain length of 1 means the number of roots that has no synonym root.

NUMBER OF RAPS
The number of RAPs that have this chain length.

NUMBER OF ROOTS
The total number of roots involved in this chain length.

PERCENTAGE OF ROOTS
The percentage of the total number of roots involved in this chain length.

CUMULATIVE OF PERCENTAGE
The cumulated value of PERCENTAGE OF ROOTS.

RAPS USED (ACTIVE)
The number of RAPS one or more of whose roots are randomized.

RAPS NOT USED
The number of RAPS whose roots are not randomized.

TOTAL RAPS
The total number of roots in this HDAM database or PHIDAM partition.

MAXIMUM ROOTS PER RAP
The maximum number of the chain length. If the number exceeds 254, it is displayed as 255+.

AVERAGE ROOTS PER ACTIVE RAP
The average number of roots per RAP that is active.

AVERAGE ROOT PER TOTAL RAP
The average number of roots per total RAPs.

NUMBER OF SYNONYM CHAINS
The total number of synonym chains. The number of chains whose length is one is not included.

AVERAGE ROOTS PER SYNONYM CHAINS
The total number of synonym roots for synonym chains divided by the total number of synonym chains. Neither the number of roots nor the number of chains whose length is one is included.

DISTRIBUTION OF DEPENDENT SEGMENTS IN ROOT BLOCK
This part shows the number of root segments that have exactly 0, 1, ..., 23, or 24+ dependent segments in the same block. It also shows the total number of root segments in this data set group or partition.

DISTRIBUTION OF DEPENDENT SEGMENTS BY SEGMENT CODE
This part shows the number of dependent segments with segment code 2, 3, ..., 23, or 24+ that are stored in the same block as their root segment. It also shows the number of dependent segments with segment code 2, 3, ..., 23, or 24+ in this data set group or partition.

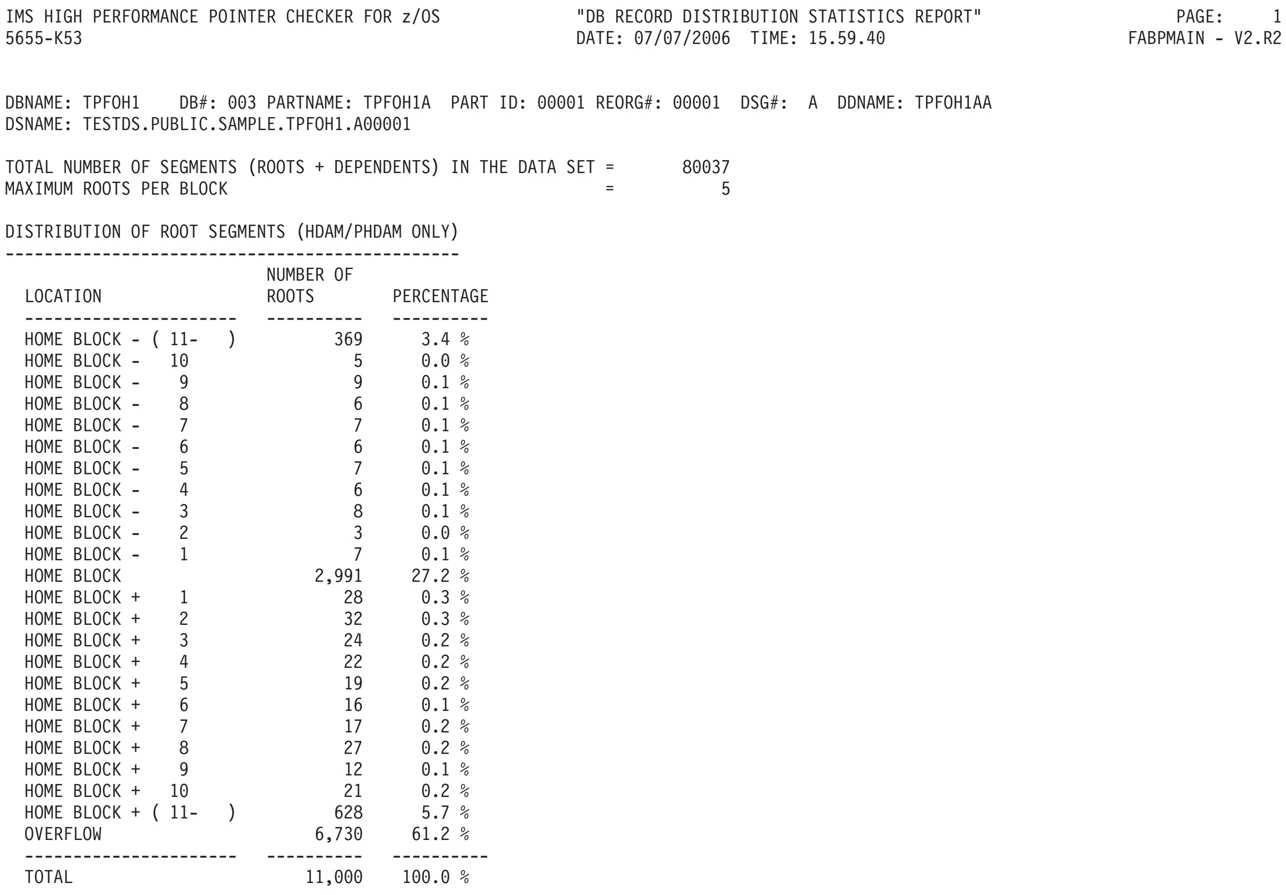
The following three fields are headings for DISTRIBUTION OF DEPENDENT SEGMENTS BY SEGMENT CODE:

#DEPS IN ROOT BLOCK
The number of dependent segments in the root block in which the segments exist.

#DEPS IN ALL BLOCKS
The number of dependent segments in all blocks.

PERCENTAGE (ROOT/ALL)
The percentage of the above values; #DEPS IN ROOT BLOCK to #DEPS IN ALL BLOCKS.

This part also shows the total number of dependent segments in this data set group.

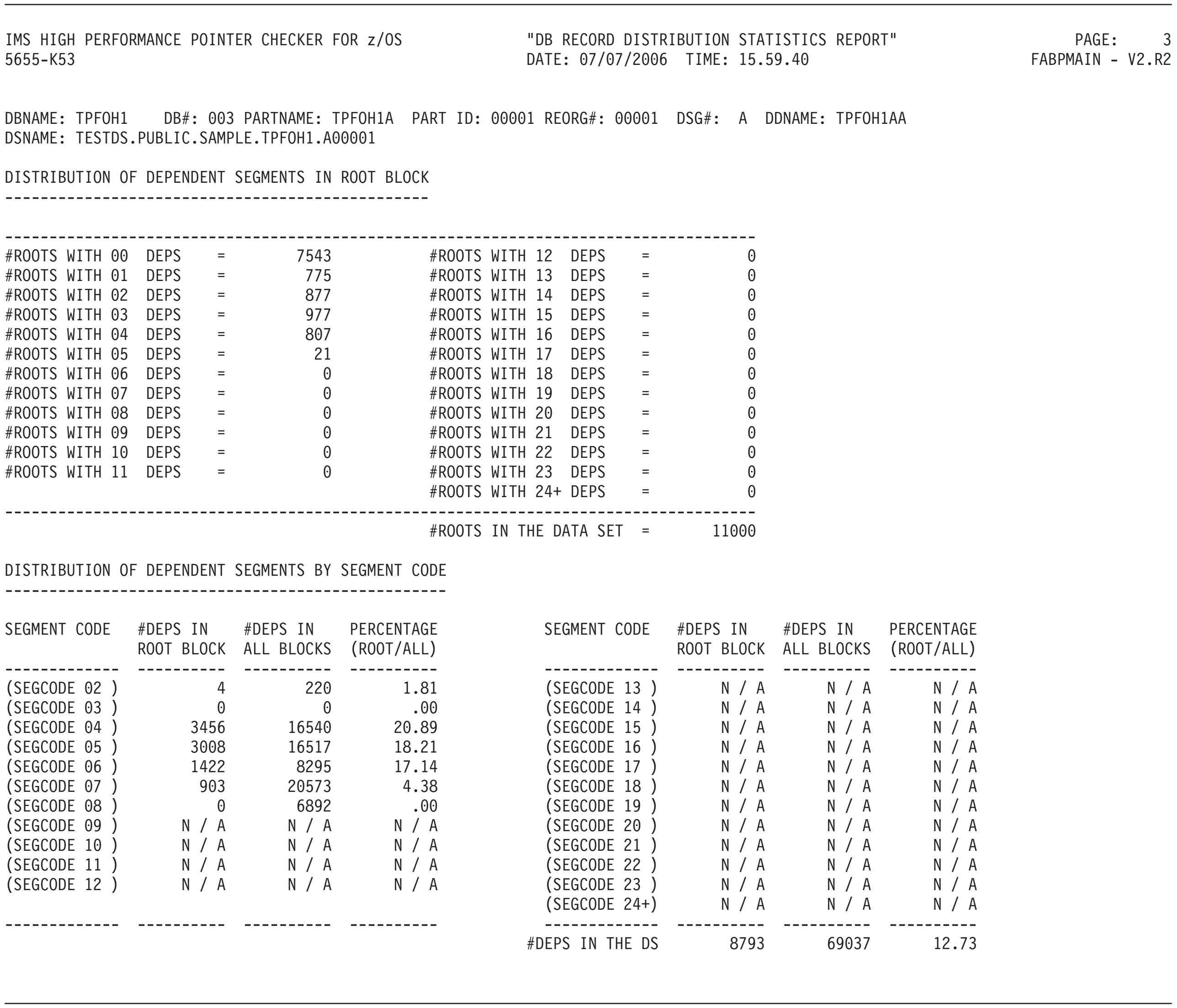


```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS           "DB RECORD DISTRIBUTION STATISTICS REPORT"                PAGE:     1
5655-K53                                                DATE: 07/07/2006  TIME: 15.59.40                        FABPMAIN - V2.R2


DBNAME: TPFOH1    DB#: 003 PARTNAME: TPFOH1A  PART ID: 00001 REORG#: 00001  DSG#:  A  DDNAME: TPFOH1AA
DSNAME: TESTDS.PUBLIC.SAMPLE.TPFOH1.A00001

TOTAL NUMBER OF SEGMENTS (ROOTS + DEPENDENTS) IN THE DATA SET =       80037
MAXIMUM ROOTS PER BLOCK                                        =           5

DISTRIBUTION OF ROOT SEGMENTS (HDAM/PHDAM ONLY)
-----------------------------------------------
                           NUMBER OF
  LOCATION                 ROOTS        PERCENTAGE
  ----------------------   ----------   ----------
  HOME BLOCK - ( 11-   )          369      3.4 %
  HOME BLOCK -   10                 5      0.0 %
  HOME BLOCK -    9                 9      0.1 %
  HOME BLOCK -    8                 6      0.1 %
  HOME BLOCK -    7                 7      0.1 %
  HOME BLOCK -    6                 6      0.1 %
  HOME BLOCK -    5                 7      0.1 %
  HOME BLOCK -    4                 6      0.1 %
  HOME BLOCK -    3                 8      0.1 %
  HOME BLOCK -    2                 3      0.0 %
  HOME BLOCK -    1                 7      0.1 %
  HOME BLOCK                    2,991     27.2 %
  HOME BLOCK +    1                28      0.3 %
  HOME BLOCK +    2                32      0.3 %
  HOME BLOCK +    3                24      0.2 %
  HOME BLOCK +    4                22      0.2 %
  HOME BLOCK +    5                19      0.2 %
  HOME BLOCK +    6                16      0.1 %
  HOME BLOCK +    7                17      0.2 %
  HOME BLOCK +    8                27      0.2 %
  HOME BLOCK +    9                12      0.1 %
  HOME BLOCK +   10                21      0.2 %
  HOME BLOCK + ( 11-   )          628      5.7 %
  OVERFLOW                      6,730     61.2 %
  ----------------------   ----------   ----------
  TOTAL                        11,000    100.0 %
```

Figure 36. STATIPRT—DB Record Distribution Statistics report (Part 1 of 3)

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS           "DB RECORD DISTRIBUTION STATISTICS REPORT"               PAGE:     2
5655-K53                                                DATE: 07/07/2006  TIME: 15.59.40                     FABPMAIN - V2.R2

DBNAME: TPFOH1    DB#: 003 PARTNAME: TPFOH1A  PART ID: 00001 REORG#: 00001  DSG#:  A  DDNAME: TPFOH1AA
DSNAME: TESTDS.PUBLIC.SAMPLE.TPFOH1.A00001

DISTRIBUTION OF RAP CHAIN LENGTHS (HDAM/PHDAM ONLY)
---------------------------------------------------
  CHAIN    NUMBER      NUMBER      PERCENTAGE   CUMULATIVE
  LENGTH   OF RAPS     OF ROOTS    OF ROOTS     PERCENTAGE
  ------   ---------   ---------   ----------   ----------
     1           978         978      8.9 %         8.9 %
     2         1,148       2,296     20.9 %        29.8 %
     3           899       2,697     24.5 %        54.3 %
     4           575       2,300     20.9 %        75.2 %
     5           298       1,490     13.5 %        88.7 %
     6           134         804      7.3 %        96.0 %
     7            36         252      2.3 %        98.3 %
     8            15         120      1.1 %        99.4 %
     9             7          63      0.6 %       100.0 %
    10             0           0      0.0 %       100.0 %
    11             0           0      0.0 %       100.0 %
    12             0           0      0.0 %       100.0 %
    13             0           0      0.0 %       100.0 %
    14             0           0      0.0 %       100.0 %
    15+            0           0      0.0 %       100.0 %
  ------   ---------   ---------   ----------   ----------
  TOTAL        4,090      11,000    100.0 %

  RAPS USED (ACTIVE)             =       4,090   90.9 %
  RAPS NOT USED                  =         410    9.1 %
  TOTAL RAPS                     =       4,500  100.0 %

  MAXIMUM ROOTS PER RAP          =           9
  AVERAGE ROOTS PER ACTIVE RAP   =         2.7
  AVERAGE ROOTS PER TOTAL RAP    =         2.4

  NUMBER OF SYNONYM CHAINS       =       3,112
  AVERAGE ROOTS PER SYNONYM CHAIN =        3.2
```

Figure 36. STATIPRT—DB Record Distribution Statistics report (Part 2 of 3)



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS              "DB RECORD DISTRIBUTION STATISTICS REPORT"                    PAGE:     3
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                         FABPMAIN - V2.R2

DBNAME: TPFOH1    DB#: 003 PARTNAME: TPFOH1A  PART ID: 00001 REORG#: 00001  DSG#:  A  DDNAME: TPFOH1AA
DSNAME: TESTDS.PUBLIC.SAMPLE.TPFOH1.A00001

DISTRIBUTION OF DEPENDENT SEGMENTS IN ROOT BLOCK
------------------------------------------------

-------------------------------------------------------------------------------------
#ROOTS WITH 00  DEPS    =        7543           #ROOTS WITH 12  DEPS    =           0
#ROOTS WITH 01  DEPS    =         775           #ROOTS WITH 13  DEPS    =           0
#ROOTS WITH 02  DEPS    =         877           #ROOTS WITH 14  DEPS    =           0
#ROOTS WITH 03  DEPS    =         977           #ROOTS WITH 15  DEPS    =           0
#ROOTS WITH 04  DEPS    =         807           #ROOTS WITH 16  DEPS    =           0
#ROOTS WITH 05  DEPS    =          21           #ROOTS WITH 17  DEPS    =           0
#ROOTS WITH 06  DEPS    =           0           #ROOTS WITH 18  DEPS    =           0
#ROOTS WITH 07  DEPS    =           0           #ROOTS WITH 19  DEPS    =           0
#ROOTS WITH 08  DEPS    =           0           #ROOTS WITH 20  DEPS    =           0
#ROOTS WITH 09  DEPS    =           0           #ROOTS WITH 21  DEPS    =           0
#ROOTS WITH 10  DEPS    =           0           #ROOTS WITH 22  DEPS    =           0
#ROOTS WITH 11  DEPS    =           0           #ROOTS WITH 23  DEPS    =           0
                                                #ROOTS WITH 24+ DEPS    =           0
-------------------------------------------------------------------------------------
                                                #ROOTS IN THE DATA SET  =       11000

DISTRIBUTION OF DEPENDENT SEGMENTS BY SEGMENT CODE
--------------------------------------------------

SEGMENT CODE   #DEPS IN    #DEPS IN    PERCENTAGE            SEGMENT CODE   #DEPS IN    #DEPS IN    PERCENTAGE
               ROOT BLOCK  ALL BLOCKS  (ROOT/ALL)                           ROOT BLOCK  ALL BLOCKS  (ROOT/ALL)
-------------  ----------  ----------  ----------            -------------  ----------  ----------  ----------
(SEGCODE 02 )           4         220        1.81            (SEGCODE 13 )       N / A       N / A       N / A
(SEGCODE 03 )           0           0         .00            (SEGCODE 14 )       N / A       N / A       N / A
(SEGCODE 04 )        3456       16540       20.89            (SEGCODE 15 )       N / A       N / A       N / A
(SEGCODE 05 )        3008       16517       18.21            (SEGCODE 16 )       N / A       N / A       N / A
(SEGCODE 06 )        1422        8295       17.14            (SEGCODE 17 )       N / A       N / A       N / A
(SEGCODE 07 )         903       20573        4.38            (SEGCODE 18 )       N / A       N / A       N / A
(SEGCODE 08 )           0        6892         .00            (SEGCODE 19 )       N / A       N / A       N / A
(SEGCODE 09 )       N / A       N / A       N / A            (SEGCODE 20 )       N / A       N / A       N / A
(SEGCODE 10 )       N / A       N / A       N / A            (SEGCODE 21 )       N / A       N / A       N / A
(SEGCODE 11 )       N / A       N / A       N / A            (SEGCODE 22 )       N / A       N / A       N / A
(SEGCODE 12 )       N / A       N / A       N / A            (SEGCODE 23 )       N / A       N / A       N / A
                                                             (SEGCODE 24+)       N / A       N / A       N / A
-------------  ----------  ----------  ----------            -------------  ----------  ----------  ----------
                                                           #DEPS IN THE DS        8793       69037       12.73
```

Figure 36. STATIPRT—DB Record Distribution Statistics report (Part 3 of 3)

Separator page for Partition Statistics reports

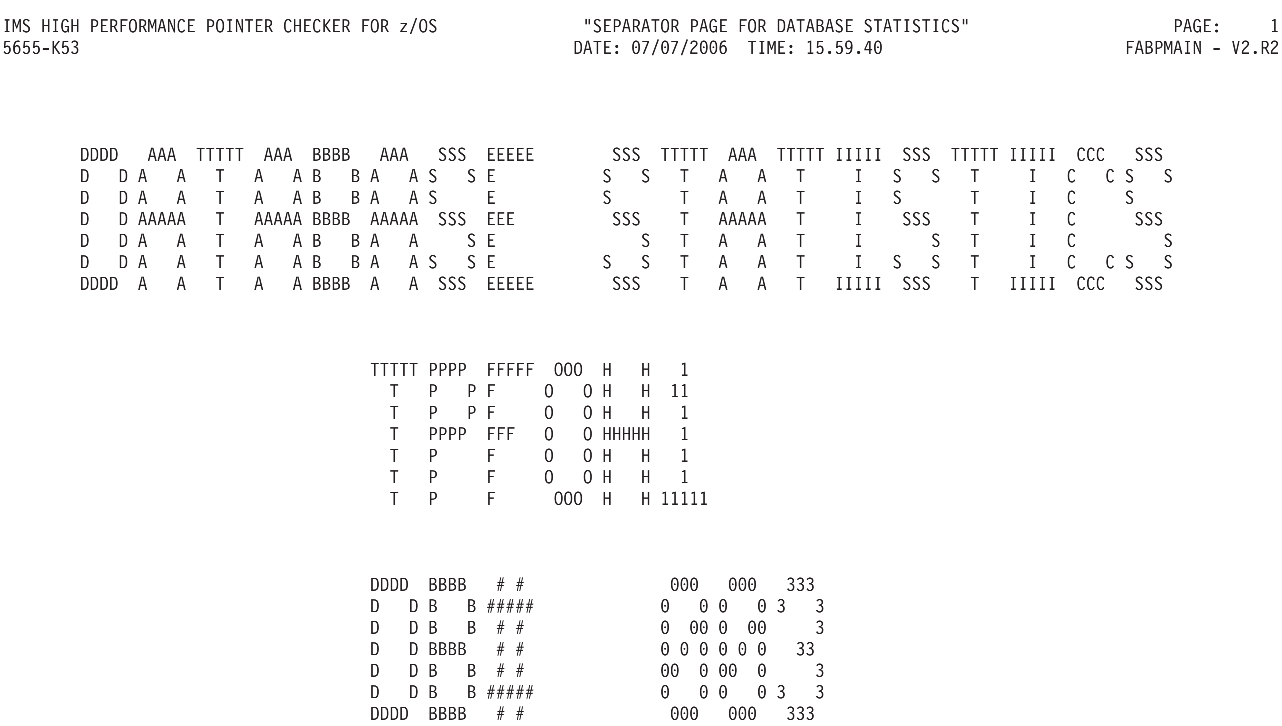
This separator page (see Figure 37) contains the title “Partition Statistics,” the DBD name, the partition name, the DB number, and the partition ID. It signifies that the Partition statistics reports will follow. It is produced unless SEP=NO is specified on the PROC statement as input for the PROCCTL data set.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS              "SEPARATOR PAGE FOR PARTITION STATISTICS"                     PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                         FABPMAIN - V2.R2


  PPPP   AAA  RRRR  TTTTT IIIII TTTTT IIIII  OOO  N   N        SSS  TTTTT  AAA  TTTTT IIIII  SSS  TTTTT IIIII  CCC   SSS
  P   P A   A R   R   T     I     T     I   O   O N   N       S   S   T   A   A   T     I   S   S   T     I   C   C S   S
  P   P A   A R   R   T     I     T     I   O   O NN  N       S       T   A   A   T     I   S       T     I   C     S
  PPPP  AAAAA RRRR    T     I     T     I   O   O N N N        SSS    T   AAAAA   T     I    SSS    T     I   C      SSS
  P     A   A R R     T     I     T     I   O   O N  NN           S   T   A   A   T     I       S   T     I   C         S
  P     A   A R  R    T     I     T     I   O   O N   N       S   S   T   A   A   T     I   S   S   T     I   C   C S   S
  P     A   A R   R   T   IIIII   T   IIIII  OOO  N   N        SSS    T   A   A   T   IIIII  SSS    T   IIIII  CCC   SSS


        TTTTT PPPP  FFFFF  OOO  H   H   1                           TTTTT PPPP  FFFFF  OOO  H   H   1    AAA
          T   P   P F     O   O H   H  11                             T   P   P F     O   O H   H  11   A   A
          T   P   P F     O   O H   H   1                             T   P   P F     O   O H   H   1   A   A
          T   PPPP  FFF   O   O HHHHH   1                             T   PPPP  FFF   O   O HHHHH   1   AAAAA
          T   P     F     O   O H   H   1                             T   P     F     O   O H   H   1   A   A
          T   P     F     O   O H   H   1                             T   P     F     O   O H   H   1   A   A
          T   P     F      OOO  H   H 11111                           T   P     F      OOO  H   H 11111 A   A


        DDDD  BBBB   # #               000   000   333              PPPP  IIIII DDDD          000   000   000   000     1
        D   D B   B #####             0   0 0   0 3   3             P   P   I   D   D       0   0 0   0 0   0 0   0  11
        D   D B   B  # #              0  00 0  00     3             P   P   I   D   D       0  00 0  00 0  00 0  00   1
        D   D BBBB   # #              0 0 0 0 0 0   33              PPPP    I   D   D       0 0 0 0 0 0 0 0 0 0 0 0   1
        D   D B   B  # #              00  0 00  0     3             P       I   D   D       00  0 00  0 00  0 00  0   1
        D   D B   B #####             0   0 0   0 3   3             P       I   D   D       0   0 0   0 0   0 0   0   1
        DDDD  BBBB   # #               000   000   333              P     IIIII DDDD          000   000   000   000  11111
```

Figure 37. STATIPRT—Separator page for Partition Statistics reports

Separator page for Database Statistics reports

This separator page (see Figure 38) contains the title “Database Statistics,” the DBD name, and the DB number. It signifies that the Database Statistics reports will follow. It is produced unless SEP=NO is specified on the PROC statement as input for the PROCCTL data set.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS               "SEPARATOR PAGE FOR DATABASE STATISTICS"                     PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                         FABPMAIN - V2.R2


        DDDD   AAA  TTTTT  AAA  BBBB   AAA   SSS  EEEEE        SSS  TTTTT  AAA  TTTTT IIIII  SSS  TTTTT IIIII  CCC   SSS
        D   D A   A   T   A   A B   B A   A S   S E           S   S   T   A   A   T     I   S   S   T     I   C   C S   S
        D   D A   A   T   A   A B   B A   A S     E           S       T   A   A   T     I   S       T     I   C     S
        D   D AAAAA   T   AAAAA BBBB  AAAAA  SSS  EEE          SSS    T   AAAAA   T     I    SSS    T     I   C      SSS
        D   D A   A   T   A   A B   B A   A     S E               S   T   A   A   T     I       S   T     I   C         S
        D   D A   A   T   A   A B   B A   A S   S E           S   S   T   A   A   T     I   S   S   T     I   C   C S   S
        DDDD  A   A   T   A   A BBBB  A   A  SSS  EEEEE        SSS    T   A   A   T   IIIII  SSS    T   IIIII  CCC   SSS


                                      TTTTT PPPP  FFFFF  OOO  H   H   1
                                        T   P   P F     O   O H   H  11
                                        T   P   P F     O   O H   H   1
                                        T   PPPP  FFF   O   O HHHHH   1
                                        T   P     F     O   O H   H   1
                                        T   P     F     O   O H   H   1
                                        T   P     F      OOO  H   H 11111


                                      DDDD  BBBB   # #               000   000   333
                                      D   D B   B #####             0   0 0   0 3   3
                                      D   D B   B  # #              0  00 0  00     3
                                      D   D BBBB   # #              0 0 0 0 0 0   33
                                      D   D B   B  # #              00  0 00  0     3
                                      D   D B   B #####             0   0 0   0 3   3
                                      DDDD  BBBB   # #               000   000   333
```

Figure 38. STATIPRT—Separator page for Database Statistics reports

Partition Statistics report and Database Statistics report

The Partition Statistics report that is shown in Figure 39 on page 166 is printed only for HALDB, and it is printed for each partition. The Database Statistics report is printed for each database. Each of these reports contains the following parts:

- Database Record Statistics and Database Record Statistics by data set group: Various average values of database record.
- Segment and Pointer Statistics: Various information about segments and pointers.
- Total Pointer Statistics: Summary of pointers.
- Rate of Segment I/O Occurrence: The probability of doing I/O when getting access to a target segment from a source segment.
- VL Segment Length Statistics: Various information about variable-length segments.
- VL Segment Split Statistics: Various information about split segments.
- Summary of VL Segment Sizes: Summary of the variable-length segments.

DATABASE RECORD STATISTICS: This part shows the average values of database record in the partition or database.

SC
Segment code (in hexadecimal).

LV Segment Level (in decimal)

DG
The data set group number (in hexadecimal) for non-HALDB, or the data set group ID (in an alphabetical character) for HALDB, which identifies the database data set containing the segment.

SEGNAME
The name of segment coded in SEGM= in DBD.

OCCURRENCES
The number of occurrences of this segment.

SEGMENT LENGTH
The following information about the length of this segment:

PRFX
The length of the prefix part of this segment.

DATA
The data part length of this segment. "V" after the numeric means that this segment is of either variable length or fixed length with a segment edit/compression facility, and the size is the average length of the data part.

TOTAL
The sum of the prefix part and the data part of this segment.

AVERAGE OCCURRENCES
The average number of occurrences of the following information:

PER ROOT
The number of occurrences of the segments in each database record.

PER PARENT
The number of occurrences of the segments in each parent.

AVERAGE LENGTH / DB RECORD
The sum of the bytes of the segments in each database record.

CUMULATIVE LENGTH
The cumulative value of AVERAGE LENGTH / DB RECORD.

AVERAGE DB RECORD LENGTH
The average length of database records. Both prefix parts and data parts of all segment types are included.

AVERAGE DB RECORD PREFIX LENGTH
The average lengths of prefixes in database records.

DATABASE RECORD STATISTICS BY DATA SET GROUP: This part shows the average values of database record in the partition or database in order of the data set group. This part is printed only for HDAM, HIDAM, PHDAM and PHIDAM. All fields in this part except the following field are the same as the DATABASE RECORD STATISTICS part.

CUMULATIVE LENGTH
The cumulative value of AVERAGE LENGTH/DB RECORD for each data set group.

SEGMENT AND POINTER STATISTICS: This part contains the following information:

- A map of the prefix of each segment type that is defined for the partition or database.
- A map of the logical relationships for each segment type that is defined for the partition or database.
- The total number of pointer types for each segment type in the partition or database.

SOURCE
The segment that contains the pointer (also called the source of the pointer). The following four fields all pertain to it:

DB
The database number (in hexadecimal) that identifies the database containing the source of the pointer.

DG
The data set group number (in hexadecimal) or the data set group ID (in an alphabetical character) that identifies the database containing the segment that contains the pointer.

SC
The segment code (in hexadecimal) of the segment that contains the pointer

SEGNAME
The segment name, as coded on the SEGM macro in the DBD, of the segment that contains the pointer.

PTR TYP
The type of pointer such as RAP, CTR, PTF, PTB, PP, PCF, PCL, LTF, LTB, LP, LCF, LCL, *LP, *LC, HF, HB, and VLS. The following two values show the existence of a logical relationship when no direct pointer exists:

***LP**
The source segment is a logical child, and the target segment is its logical parent. There is no direct logical parent pointer; instead, the source segment has a symbolic pointer (the logical parent concatenated key) to its logical parent.

***LC**
The source segment is a logical parent, and the target segment is its logical child. There is no logical child pointer.

The VLS pointer points from the prefix to the data in a split segment. The count of VLS pointers in SAME BLOCK is the number of split segments that have prefix and data in the same block. The number of split segments that have prefix and data in different blocks is the count in TOTAL of NO IN SAME BLOCK.

TARGET
The target of a pointer. The following four fields all pertain to it:

DB
The database number (in hexadecimal) that identifies the database containing the target of a pointer.

DG
The data set group number (in hexadecimal) or the data set group ID (in an alphabetical character) that identifies the database containing the target of a pointer.

SC
The segment code (in hexadecimal) of the target of a pointer.

SEGNAME
The segment name, as coded on the SEGM macro in the DBD, of the target of a pointer.

COUNT OF POINTERS
Each kind of pointer is totaled according to the following classes:

NOT USED (ZERO POINTER)
The number of unused pointers that do not point to any target. Unused pointers contain the value zero.

USED
The information of used pointers that point to targets. Used pointers contain non-zero values.

TOTAL
The number of pointers used. It contains pointers to target segments outside this data set.

IN SAME BLOCK
The number of pointers that point to the block containing the pointer.

Note: This does not include the number of dependent segments that are in the same block as their root segment.

NOT IN SAME BLOCK
Information about pointers that point to different blocks within the same data set.

TOTAL
The number of pointers not in the same block.

BLOCK-1
The number of pointers that point to the block immediately before the block containing the pointer.

BLOCK+1
The number of pointers that point to the block immediately after the block containing the pointer.

BEYOND
The number of pointers that point to blocks (in the same data set) that is not adjacent to or the same as the block containing the pointer.

The following field can be used to indicate whether the database needs reorganization:

PROB OF IO
The probability of doing I/O when accessing a target segment via the indicated pointer. This is computed by the following formula: If the target and the pointer are not in the same data set:

$$P=1$$

otherwise:

$$P = 1 - (S/T)$$

Where: S = the number of pointers in the same block as their target
T = the total number of pointers.

TOTAL POINTER STATISTICS: This part presents the number of pointers for each pointer type.

For description of each field, see "SEGMENT AND POINTER STATISTICS" on page 160.

NOTE: This part is the notifications for Segment and Pointer Statistics and Total Pointer Statistics. This part is printed only once, after Total Pointer Statistics.

RATE OF SEGMENT I/O OCCURRENCE: This part shows the probability of doing I/O when getting access to a target segment from a source segment.

RAP TO ROOT
The probability of doing I/O when getting access to a root segment from RAP.

The probabilities are shown in a hierarchical diagram of the database record structure.

- The number below the segment name shows the probability of doing I/O when getting access to a target segment from a source segment of the same segment type. If NT is specified for the segment, *.*** is shown instead of the number.
- The number in the arrow shows the probability of doing I/O when getting access to a target segment from a source segment of a different segment type.
- The number in parentheses below the segment shows the probability of doing I/O when getting access to a target segment from a source segment that is related to by a hierarchical pointer. The probability is shown only for a physical path. It is not shown for a logical relationship.

This part is generated when SEGIO=YES is specified in the REPORT statement of the PROCCTL data set.

VL SEGMENT LENGTH STATISTICS: This part presents information on the number of occurrences and segment length of variable-length segments. It also contains fixed-length segments compressed by the segment edit/compression exit routine. If no variable-length segment or compressed fixed-length segment is defined in the database, this part is not printed.

SC
Segment code (in hexadecimal).

SEGMENT NAME
The segment name, as coded on the SEGM macro in the DBD.

TYPE
This field shows the segment that is defined as fixed length or variable length, and a segment edit/compression exit routine is specified in the DBD.

FC
The segment is defined as fixed length, and a segment edit/compression exit routine is defined. The actual segment length is variable because the segment length can be changed by the segment edit/compression exit routine.

V The segment is variable length and a segment edit/compression exit routine is not defined.

VC
The segment is variable length and a segment edit/compression exit routine is defined.

OCCURRENCES
The number of occurrences of the following segments.

PREFIX LENGTH
Prefix length of this segment.

DATA LENGTH
The following information about the data length of this segment:

MIN
The minimum length of the data that can be stored in the database, including the segment length and the padded area. For a fixed-length segment, if you have specified that a segment edit/compress facility is to be used, this field can contain different values, depending on the values in COMPRTN=.

- When COMPRTN=(,,) or COMPRTN=(,,,*max*) is coded, the MIN field is always 4 bytes.
- When COMPRTN=(,,,*pad size*,PAD) is coded, the MIN field is *pad size*.

MAX
The maximum length of the data that can be stored in the database, including the segment length and the padded area. For a variable-length segment, the value in this field is the same as the maximum coded on BYTES= in the DBD. For a fixed-length segment, if you have specified that a segment edit/compress facility is to be used, this field can contain different values, depending on the values in COMPRTN=.

If COMPRTN=(,,) or COMPRTN=(,,,*max*) is coded, one of the following values is used:

- If *max* is 10 or greater, the MAX field is the value of BYTES= plus *max*.
- If *max* is less than 10 or is not coded, the MAX field is the value of BYTES= plus 10, because IMS allows the segment edit/compression routine to increase the length of the segment by up to 10 bytes.

When COMRTN=(,,,*pad size*,PAD) is coded, one of the following values is used:

- If *pad size* adds 10 or more bytes to the value specified in BYTES=, the value in the MAX field is *pad size*.
- If *pad size* is less than 10 length additional to the value specified in BYTES=, the MAX is the value of BYTES= plus 10.

AVERAGE
The average length of occurrences of the segment in the database, partition, or database. It contains only the segment length. If the segment is edited by the segment edit/compression exit, the segment length that has been edited is used.

OCCURRENCES
The number of occurrences of the following segments. For a compressed segment, the length of data that has been compressed is used for comparison.

LENGTH < MIN
The number of occurrences of segments, whose data lengths are less than the minimum.

LENGTH >= MIN
The number of occurrences of segments, whose data lengths are equal to or greater than the minimum.

COMPRESSION EXIT NAME
The name of the segment edit/compression exit routine, as coded on COMPRTN= in the DBD.

COMPRESSION FACTOR
If a segment edit/compression exit routine is specified for this segment, the data compression factor is shown in the column.

The compression factor is calculated by the following formula:

```
Bytes before compression - Bytes after compression
--------------------------------------------------
               Bytes before compression
```

Bytes before compression

- If the segment is fixed length and compressed, the value shows the specified length in BYTES= operand of SEGM statement in DBDGEN.
- If the segment is variable length and compressed, the value shows the average length of the data portion of all segments before compression.

Bytes after compression
It is the average length of the data portion of all segments after compression. This length is the same as the length when the segments are stored in the databases.

Notes:

1. If the bytes after compression is greater than before compression, a negative number is shown.
2. When segment is variable length with a segment/edit compression exit (TYPE column is VC), HD Pointer Checker calls the segment edit/compression exit routine to obtain the length of the data portion before compression. It is required that you specify the load module library that contains the user exit for the IMS2 DD or the STEPLIB DD in the JCL.

The compression factor is shown when COMPFACT=YES is specified in the REPORT statement of the PROCCTL data set. When COMPFACT=NO is specified, N/AVAIL is shown in the column.

If a segment edit/compression routine is not defined in the segment, blank is set in this column.

VL SEGMENT SPLIT STATISTICS: This part presents information on the split variable length segment for each segment type. It also contains fixed-length segment compressed by the segment edit/compress facility. If no variable-length segment or compressed fixed-length segment is defined in the database, this part is not printed.

SC
Segment code (in hexadecimal).

SEGMENT NAME
The segment name, as coded on the SEGM macro in the DBD.

TYPE
This field shows the segment that is defined as fixed length or variable length, and a segment edit/compression exit routine that is specified in the DBD.

FC
The segment is defined as fixed length, and a segment edit/compression exit routine is defined. The actual segment length is variable because the segment length can be changed by the segment edit/compression exit routine.

V The segment is variable length and a segment edit/compression exit routine is not defined.

VC
The segment is variable length and a segment edit/compression exit routine is defined.

OCCURRENCES
The number of occurrences of this segment.

NOT SPLIT
The number of occurrences that are not split, and what percentage it makes of the total.

SPLIT
The number of occurrences that are split, and what percentage it makes of the total.

PREFIX AND DATA
The following information about the prefix and data of split segments:

IN SAME BLOCK
The number of occurrences of both prefix part and data part that exist in the same block, and what percentage it makes of the number of split segment.

NOT IN SAME BLOCK
The number of occurrences of data part that exist in a different block from the prefix part, and what percentage it makes of the number of split.

SUMMARY OF VL SEGMENT SIZES: This part presents the distribution of the variable segment length for each segment type. It also contains fixed-length segment compressed by the segment edit/compress facility. If no variable-length segment or compressed fixed-length segment is defined in the database, this part is not printed. This part is printed when requested by OPTION VLSSUMM=YES in the PROCCTL data set.

SC
Segment code (in hexadecimal).

SEGMENT NAME
The segment name, as coded on the SEGM macro in the DBD.

SIZE RANGE
The size (in bytes) range of the part of the variable-length segment to be reported. For compressed segment, the length after compression is used for the size. The shortest and the longest sizes are actual values detected in the database, not extracted from the DBD. This report divides the size range into 20.

Note: In the following cases, however, the size range might become less than 20 or the size range might not be fixed.

- The maximum segment length specified in the SEGM statement for the DBD exceeds a certain value.
- The segment length is distributed within a certain narrow range.

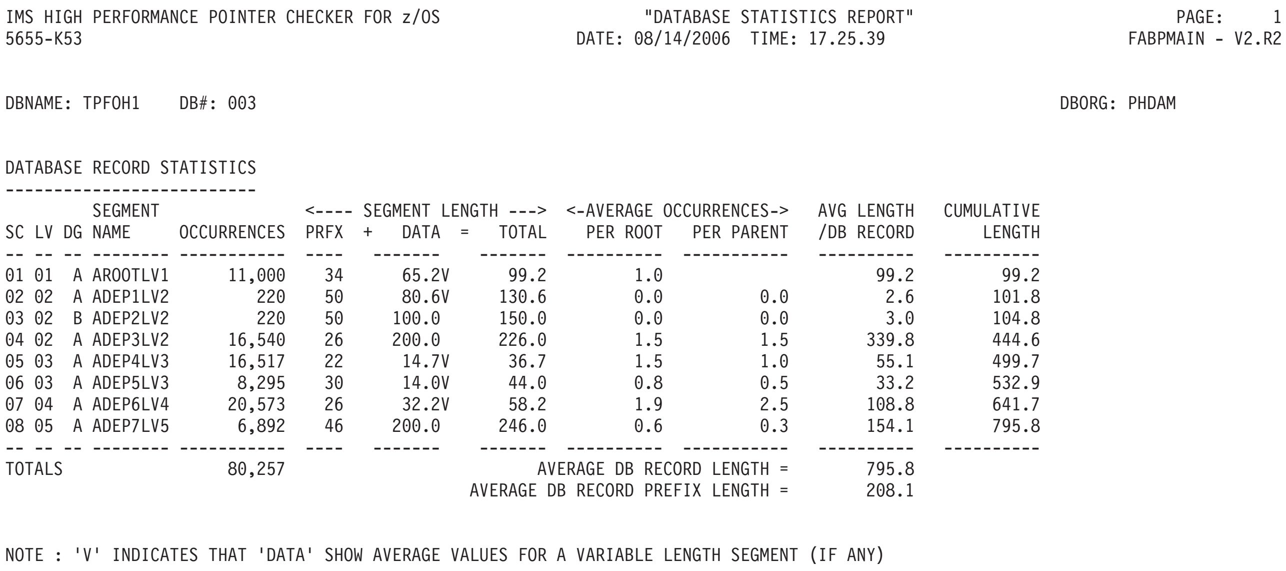
OCCURRENCES
The number of occurrences that fall into this range.

PERCENTAGE
The percentage of occurrences of this range, and what percentage it makes of the total.

CUMULATIVE PERCENTAGE
The cumulative value of PERCENTAGE.

AVERAGE SEGMENT LENGTH
The average length of the data part in segments of this type in this partition or database.

The summary of VL segment sizes is shown when VLSSUMM=YES is specified in the PROC statement of the PROCCTL data set.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                     "DATABASE STATISTICS REPORT"                           PAGE:     1
5655-K53                                                   DATE: 08/14/2006  TIME: 17.25.39                         FABPMAIN - V2.R2


DBNAME: TPFOH1    DB#: 003                                                                                           DBORG: PHDAM


DATABASE RECORD STATISTICS
--------------------------
         SEGMENT               <---- SEGMENT LENGTH ---> <-AVERAGE OCCURRENCES->   AVG LENGTH   CUMULATIVE
SC LV DG NAME     OCCURRENCES  PRFX  +   DATA  =   TOTAL    PER ROOT   PER PARENT   /DB RECORD       LENGTH
-- -- -- -------- -----------  ----   -------    -------  ----------  -----------   ----------   ----------
01 01  A AROOTLV1      11,000    34      65.2V      99.2         1.0                      99.2         99.2
02 02  A ADEP1LV2         220    50      80.6V     130.6         0.0          0.0          2.6        101.8
03 02  B ADEP2LV2         220    50     100.0      150.0         0.0          0.0          3.0        104.8
04 02  A ADEP3LV2      16,540    26     200.0      226.0         1.5          1.5        339.8        444.6
05 03  A ADEP4LV3      16,517    22      14.7V      36.7         1.5          1.0         55.1        499.7
06 03  A ADEP5LV3       8,295    30      14.0V      44.0         0.8          0.5         33.2        532.9
07 04  A ADEP6LV4      20,573    26      32.2V      58.2         1.9          2.5        108.8        641.7
08 05  A ADEP7LV5       6,892    46     200.0      246.0         0.6          0.3        154.1        795.8
-- -- -- -------- -----------  ----   -------    -------  ----------  -----------   ----------   ----------
TOTALS                 80,257                          AVERAGE DB RECORD LENGTH =        795.8
                                                AVERAGE DB RECORD PREFIX LENGTH =        208.1


NOTE : 'V' INDICATES THAT 'DATA' SHOW AVERAGE VALUES FOR A VARIABLE LENGTH SEGMENT (IF ANY)
```

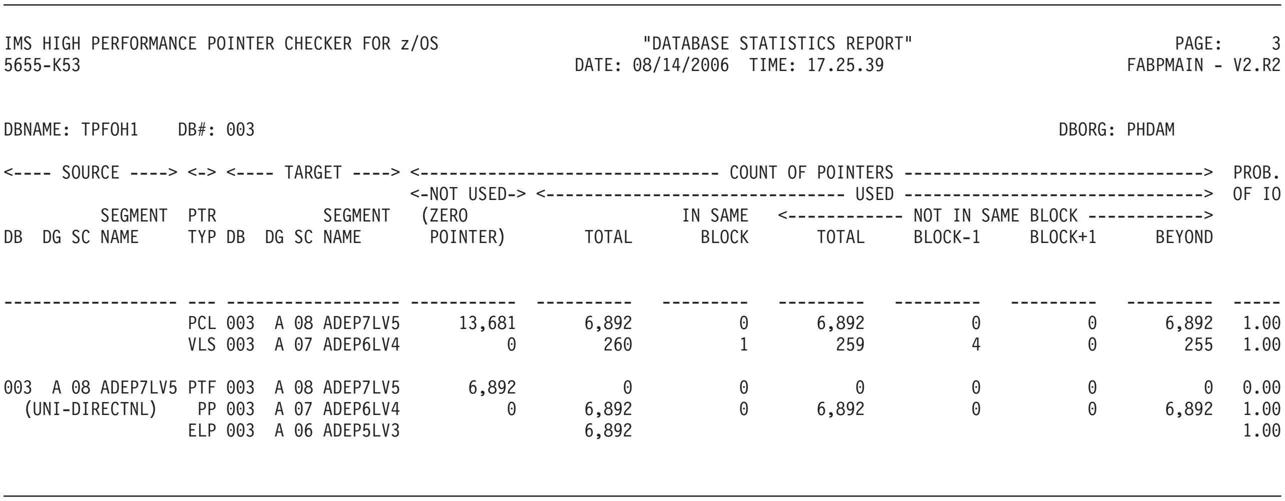
Figure 39. STATIPRT—Database Statistics report (Part 1 of 9)

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS              "DATABASE STATISTICS REPORT"                  PAGE:     1
5655-K53                                            DATE: 08/14/2006  TIME: 17.25.39                FABPMAIN - V2.R2

DBNAME: TPFOH1    DB#: 003                                                                       DBORG: PHDAM

DATABASE RECORD STATISTICS BY DATA SET GROUP
--------------------------------------------
         SEGMENT                 <---- SEGMENT LENGTH --->  <-AVERAGE OCCURRENCES->   AVG LENGTH   CUMULATIVE
DG SC LV NAME     OCCURRENCES  PRFX  +   DATA  =   TOTAL    PER ROOT   PER PARENT   /DB RECORD       LENGTH
-- -- -- -------- -----------  ----  -------    -------  ----------  -----------   ----------   ----------
 A 01 01 AROOTLV1      11,000    34     65.2V      99.2         1.0                        99.2         99.2
   02 02 ADEP1LV2         220    50     80.6V     130.6         0.0          0.0            2.6        101.8
   04 02 ADEP3LV2      16,540    26    200.0      226.0         1.5          1.5          339.8        441.6
   05 03 ADEP4LV3      16,517    22     14.7V      36.7         1.5          1.0           55.1        496.7
   06 03 ADEP5LV3       8,295    30     14.0V      44.0         0.8          0.5           33.2        529.9
   07 04 ADEP6LV4      20,573    26     32.2V      58.2         1.9          2.5          108.8        638.7
   08 05 ADEP7LV5       6,892    46    200.0      246.0         0.6          0.3          154.1        792.8

 B 03 02 ADEP2LV2         220    50    100.0      150.0         0.0          0.0            3.0          3.0
-- -- -- -------- -----------  ----  -------    -------  ----------  -----------   ----------   ----------
TOTALS                 80,257                          AVERAGE DB RECORD LENGTH =        795.8
                                                AVERAGE DB RECORD PREFIX LENGTH =        208.1

NOTE : 'V' INDICATES THAT 'DATA' SHOW AVERAGE VALUES FOR A VARIABLE LENGTH SEGMENT (IF ANY)
```

Figure 39. STATIPRT—Database Statistics report (Part 2 of 9)

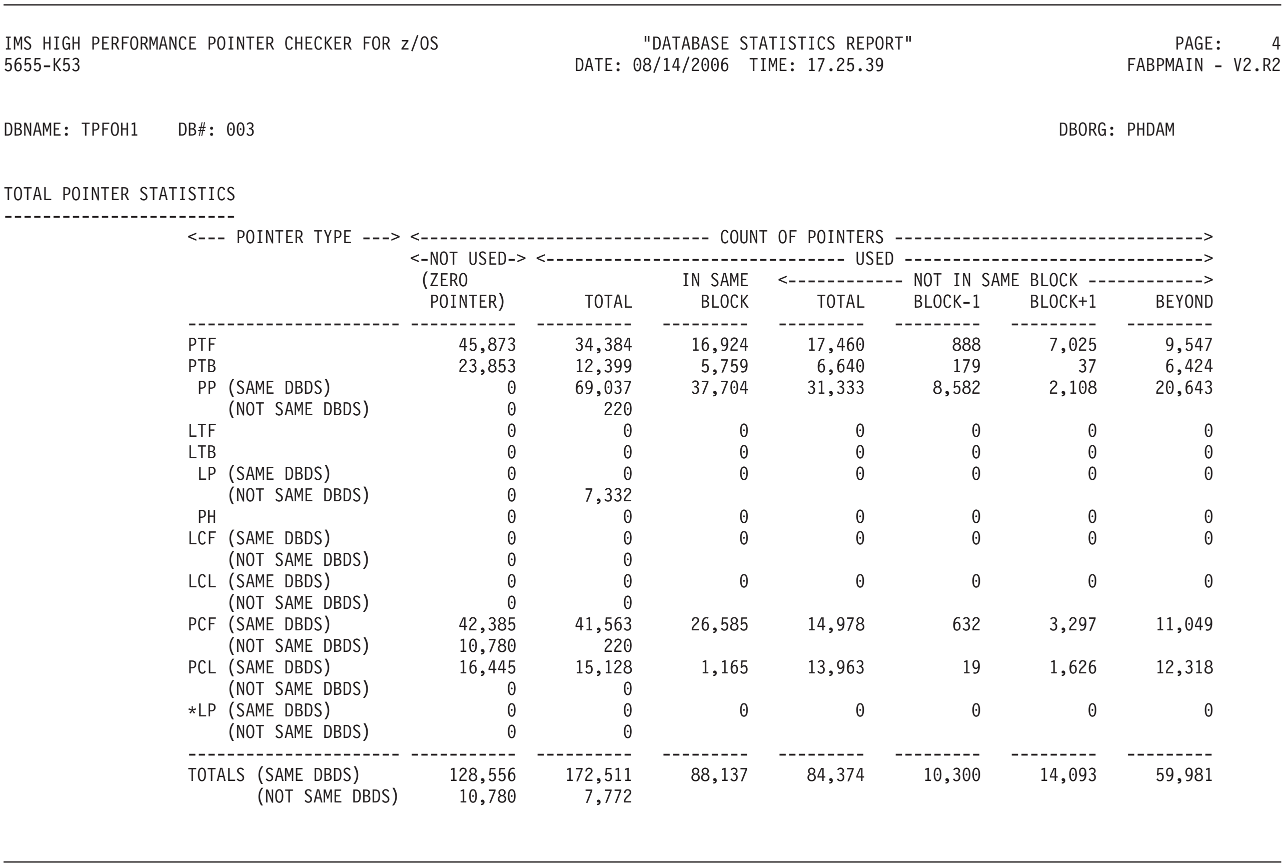
```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                     "DATABASE STATISTICS REPORT"                        PAGE:        2
5655-K53                                                   DATE: 08/14/2006  TIME: 17.25.39                         FABPMAIN - V2.R2

DBNAME: TPFOH1    DB#: 003                                                                                   DBORG: PHDAM

SEGMENT AND POINTER STATISTICS
------------------------------
<---- SOURCE ----> <-> <---- TARGET ----> <------------------------------- COUNT OF POINTERS ------------------------------->  PROB.
                                          <-NOT USED-> <------------------------------- USED ------------------------------->  OF IO
          SEGMENT  PTR          SEGMENT   (ZERO                       IN SAME   <------------ NOT IN SAME BLOCK ------------>
DB  DG SC NAME     TYP DB  DG SC NAME       POINTER)        TOTAL       BLOCK       TOTAL     BLOCK-1     BLOCK+1      BEYOND
------------------ --- ------------------ -----------  ----------   ---------   ---------   ---------   ---------   ---------  -----
                   RAP 003  A 01 AROOTLV1         410       4,090       2,343       1,747           0          16       1,731   0.43

003  A 01 AROOTLV1 PTF 003  A 01 AROOTLV1       4,090       6,910         700       6,210           6          12       6,192   0.90
                   PTB 003  A 01 AROOTLV1       4,090       6,910         700       6,210          12           6       6,192   0.90
                   PCF 003  A 02 ADEP1LV2      10,780         220           4         216           0           0         216   0.98
                   PCF 003  B 03 ADEP2LV2      10,780         220                                                               1.00
                   PCF 003  A 04 ADEP3LV2       2,764       8,236       3,456       4,780           5       2,133       2,642   0.58
                   PCL 003  A 04 ADEP3LV2       2,764       8,236       1,165       7,071          19       1,626       5,426   0.86
                   VLS 003  A 01 AROOTLV1           0           0           0           0           0           0           0   0.00
                   *LC 004  A 02 CDEP1LV2

003  A 02 ADEP1LV2 PTF 003  A 02 ADEP1LV2         220           0           0           0           0           0           0   0.00
  (UNI-DIRECTNL)   PTB 003  A 02 ADEP1LV2         220           0           0           0           0           0           0   0.00
                    PP 003  A 01 AROOTLV1           0         220           4         216           0           0         216   0.98
                   VLS 003  A 02 ADEP1LV2           0           0           0           0           0           0           0   0.00
                   ELP 005  A 01 BROOTLV1                     220                                                               1.00

003  B 03 ADEP2LV2 PTF 003  B 03 ADEP2LV2         220           0           0           0           0           0           0   0.00
  (PHYS. PAIRED)   PTB 003  B 03 ADEP2LV2         220           0           0           0           0           0           0   0.00
                    PP 003  A 01 AROOTLV1           0         220                                                               1.00
                   ELP 004  A 01 CROOTLV1                     220                                                               1.00
                   ELC 004  A 02 CDEP1LV2                     220                                                               1.00

003  A 04 ADEP3LV2 PTF 003  A 04 ADEP3LV2       8,236       8,304         932       7,372          16       6,103       1,253   0.89
                    PP 003  A 01 AROOTLV1           0      16,540       3,456      13,084       3,876          25       9,183   0.79
                   PCF 003  A 05 ADEP4LV3       5,512      11,028       9,750       1,278         353         384         541   0.12
                   PCF 003  A 06 ADEP5LV3       8,245       8,295       7,792         503          64         255         184   0.06

003  A 05 ADEP4LV3 PTF 003  A 05 ADEP4LV3      11,028       5,489       5,059         430          31         167         232   0.08
                   PTB 003  A 05 ADEP4LV3      11,028       5,489       5,059         430         167          31         232   0.08
                    PP 003  A 04 ADEP3LV2           0      16,517      14,370       2,147         820         521         806   0.13
                   VLS 003  A 05 ADEP4LV3           0           0           0           0           0           0           0   0.00

003  A 06 ADEP5LV3 CTR                                      6,892
                   PTF 003  A 06 ADEP5LV3       8,295           0           0           0           0           0           0   0.00
                   PTB 003  A 06 ADEP5LV3       8,295           0           0           0           0           0           0   0.00
                    PP 003  A 04 ADEP3LV2           0       8,295       7,792         503         255          64         184   0.06
                   PCF 003  A 07 ADEP6LV4       1,403       6,892       5,583       1,309         210         525         574   0.19
                   VLS 003  A 06 ADEP5LV3           0           0           0           0           0           0           0   0.00
                   *LC 003  A 08 ADEP7LV5

003  A 07 ADEP6LV4 PTF 003  A 07 ADEP6LV4       6,892      13,681      10,233       3,448         835         743       1,870   0.25
                    PP 003  A 06 ADEP5LV3           0      20,573      12,082       8,491       3,631       1,498       3,362   0.41
                   PCF 003  A 08 ADEP7LV5      13,681       6,892           0       6,892           0           0       6,892   1.00
```

Figure 39. STATIPRT—Database Statistics report (Part 3 of 9)



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                     "DATABASE STATISTICS REPORT"                           PAGE:     3
5655-K53                                                   DATE: 08/14/2006  TIME: 17.25.39                         FABPMAIN - V2.R2

DBNAME: TPFOH1    DB#: 003                                                                                   DBORG: PHDAM

<---- SOURCE ----> <-> <---- TARGET ----> <------------------------------- COUNT OF POINTERS --------------------------------> PROB.
                                          <-NOT USED-> <------------------------------- USED --------------------------------> OF IO
          SEGMENT  PTR           SEGMENT  (ZERO                       IN SAME   <------------ NOT IN SAME BLOCK ------------>
DB  DG SC NAME     TYP DB  DG SC NAME       POINTER)        TOTAL       BLOCK       TOTAL     BLOCK-1     BLOCK+1      BEYOND


------------------ --- ------------------ -----------   ----------   ---------   ---------   ---------   ---------   --------- -----
                   PCL 003  A 08 ADEP7LV5      13,681        6,892           0       6,892           0           0       6,892  1.00
                   VLS 003  A 07 ADEP6LV4           0          260           1         259           4           0         255  1.00

003  A 08 ADEP7LV5 PTF 003  A 08 ADEP7LV5       6,892            0           0           0           0           0           0  0.00
  (UNI-DIRECTNL)    PP 003  A 07 ADEP6LV4           0        6,892           0       6,892           0           0       6,892  1.00
                   ELP 003  A 06 ADEP5LV3                    6,892                                                              1.00
```

Figure 39. STATIPRT—Database Statistics report (Part 4 of 9)



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                     "DATABASE STATISTICS REPORT"                           PAGE:     4
5655-K53                                                   DATE: 08/14/2006  TIME: 17.25.39                         FABPMAIN - V2.R2

DBNAME: TPFOH1    DB#: 003                                                                                   DBORG: PHDAM


TOTAL POINTER STATISTICS
------------------------
                   <--- POINTER TYPE ---> <------------------------------ COUNT OF POINTERS --------------------------------->
                                          <-NOT USED-> <------------------------------- USED -------------------------------->
                                          (ZERO                       IN SAME   <------------ NOT IN SAME BLOCK ------------>
                                            POINTER)        TOTAL       BLOCK       TOTAL     BLOCK-1     BLOCK+1      BEYOND
                   ---------------------- -----------   ----------   ---------   ---------   ---------   ---------   ---------
                   PTF                         45,873       34,384      16,924      17,460         888       7,025       9,547
                   PTB                         23,853       12,399       5,759       6,640         179          37       6,424
                    PP (SAME DBDS)                  0       69,037      37,704      31,333       8,582       2,108      20,643
                       (NOT SAME DBDS)              0          220
                   LTF                              0            0           0           0           0           0           0
                   LTB                              0            0           0           0           0           0           0
                    LP (SAME DBDS)                  0            0           0           0           0           0           0
                       (NOT SAME DBDS)              0        7,332
                    PH                              0            0           0           0           0           0           0
                   LCF (SAME DBDS)                  0            0           0           0           0           0           0
                       (NOT SAME DBDS)              0            0
                   LCL (SAME DBDS)                  0            0           0           0           0           0           0
                       (NOT SAME DBDS)              0            0
                   PCF (SAME DBDS)             42,385       41,563      26,585      14,978         632       3,297      11,049
                       (NOT SAME DBDS)         10,780          220
                   PCL (SAME DBDS)             16,445       15,128       1,165      13,963          19       1,626      12,318
                       (NOT SAME DBDS)              0            0
                   *LP (SAME DBDS)                  0            0           0           0           0           0           0
                       (NOT SAME DBDS)              0            0
                   ---------------------- -----------   ----------   ---------   ---------   ---------   ---------   ---------
                   TOTALS (SAME DBDS)         128,556      172,511      88,137      84,374      10,300      14,093      59,981
                          (NOT SAME DBDS)      10,780        7,772
```

Figure 39. STATIPRT—Database Statistics report (Part 5 of 9)

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                     "DATABASE STATISTICS REPORT"                           PAGE:     5
5655-K53                                                   DATE: 08/14/2006  TIME: 17.25.39                         FABPMAIN - V2.R2

DBNAME: TPFOH1    DB#: 003                                                                                 DBORG: PHDAM

RATE OF SEGMENT I/O OCCURRENCE
------------------------------

  RAP TO ROOT : 0.427

  AROOTLV1 -+-0.982--> ADEP1LV2
   0.899    |           0.000
            |
            +-1.000--> ADEP2LV2
            |           0.000
            |
            +-0.719--> ADEP3LV2 -+-0.116--> ADEP4LV3
                        0.888    |           0.078
                                 |
                                 +-0.061--> ADEP5LV3 ---0.190--> ADEP6LV4 ---1.000--> ADEP7LV5
                                             0.000                0.252                0.000

NOTE :  THE NUMBERS IN THE HIERARCHICAL DIAGRAM SHOW THE PROBABILITY OF DOING I/O WHEN ACCESSING A TARGET SEGMENT FROM A SOURCE
----    SEGMENT.
          - THE NUMBER BELOW THE SEGMENT NAME SHOWS THE PROBABILITY OF DOING I/O WHEN ACCESSING A TARGET SEGMENT FROM A SOURCE
            SEGMENT OF THE SAME SEGMENT TYPE.
          - THE NUMBER IN THE ARROW SHOWS THE PROBABILITY OF DOING I/O WHEN ACCESSING A TARGET SEGMENT FROM A SOURCE SEGMENT
            OF A DIFFERENT SEGMENT TYPE.
          - THE NUMBER IN PARENTHESES BELOW THE SEGMENT SHOWS THE PROBABILITY OF DOING I/O WHEN ACCESSING A TARGET SEGMENT FROM A
            SOURCE SEGMENT THAT IS RELATED TO BY A HIERARCHICAL POINTER.
        THE NUMBER *XX SHOWS CONTINUATION TO THE NEXT FIGURE OF TREE.
```

Figure 39. STATIPRT—Database Statistics report (Part 6 of 9)

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                     "DATABASE STATISTICS REPORT"                           PAGE:     6
5655-K53                                                   DATE: 08/14/2006  TIME: 17.25.39                         FABPMAIN - V2.R2


DBNAME: TPFOH1    DB#: 003                                                                                 DBORG: PHDAM


VL SEGMENT LENGTH STATISTICS
----------------------------
   SEGMENT                      PREFIX  <--- DATA LENGTH  --->  <---- OCCURRENCES ---->  <- COMPRESSION  ->
SC NAME      TYPE  OCCURRENCES  LENGTH  MIN    MAX     AVERAGE  LENGTH<MIN  LENGTH>=MIN  EXIT NAME  FACTOR
-- --------  ----  -----------  ------  -----  -----   -------  ----------  -----------  ---------  -------
01 AROOTLV1   V         11,000      34     30    100      65.2           0       11,000
02 ADEP1LV2   V            220      50     40    120      80.6           0          220
05 ADEP4LV3   VC        16,517      22     30    200      14.7      16,517            0  DFSCMPX0   N/AVAIL
06 ADEP5LV3   FC         8,295      30      4    110      14.0           0        8,295  DFSCMPX0   N/AVAIL
07 ADEP6LV4   V         20,573      26     30    200      32.2           0       20,573
-- --------  ----  -----------  ------  -----  -----   -------  ----------  -----------  ---------  -------
TOTALS                  56,605
```

Figure 39. STATIPRT—Database Statistics report (Part 7 of 9)

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                     "DATABASE STATISTICS REPORT"                           PAGE:     7
5655-K53                                                   DATE: 08/14/2006  TIME: 17.25.39                         FABPMAIN - V2.R2

DBNAME: TPFOH1    DB#: 003                                                                                 DBORG: PHDAM

VL SEGMENT SPLIT STATISTICS
---------------------------
   SEGMENT                                                                  <----------- PREFIX AND DATA  ----------->
SC NAME      TYPE  OCCURRENCES  NOT SPLIT            SPLIT                IN SAME BLOCK        NOT IN SAME BLOCK
-- --------  ----  -----------  --------------------  --------------------  --------------------  --------------------
01 AROOTLV1   V         11,000       11,000  100.0 %            0    0.0 %            0    0.0 %            0    0.0 %
02 ADEP1LV2   V            220          220  100.0 %            0    0.0 %            0    0.0 %            0    0.0 %
05 ADEP4LV3   VC        16,517       16,517  100.0 %            0    0.0 %            0    0.0 %            0    0.0 %
06 ADEP5LV3   FC         8,295        8,295  100.0 %            0    0.0 %            0    0.0 %            0    0.0 %
07 ADEP6LV4   V         20,573       20,313   98.7 %          260    1.3 %            1    0.4 %          259   99.6 %
-- --------  ----  -----------  --------------------  --------------------  --------------------  --------------------
TOTALS                  56,605       56,345   99.5 %          260    0.5 %            1    0.4 %          259   99.6 %
```

Figure 39. STATIPRT—Database Statistics report (Part 8 of 9)

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                     "DATABASE STATISTICS REPORT"                           PAGE:     8
5655-K53                                                   DATE: 08/14/2006  TIME: 17.25.39                         FABPMAIN - V2.R2

DBNAME: TPFOH1    DB#: 003                                                                                 DBORG: PHDAM

SUMMARY OF VL SEGMENT SIZES
---------------------------
   SEGMENT                                            CUMULATIVE
SC NAME         SIZE RANGE  OCCURRENCES  PERCENTAGE  PERCENTAGE
-- -------- ---------------  -----------  ----------  ----------
01 AROOTLV1     30 -     32          465      4.2 %      4.2 %
                33 -     36          608      5.5 %      9.8 %
                37 -     39          482      4.4 %     14.1 %
                40 -     43          607      5.5 %     19.7 %
                44 -     46          424      3.9 %     23.5 %
                47 -     50          611      5.6 %     29.1 %
                51 -     53          478      4.3 %     33.4 %
                54 -     57          598      5.4 %     38.8 %
                58 -     60          458      4.2 %     43.0 %
                61 -     64          658      6.0 %     49.0 %
                65 -     68          628      5.7 %     54.7 %
                69 -     71          441      4.0 %     58.7 %
                72 -     75          625      5.7 %     64.4 %
                76 -     78          480      4.4 %     68.8 %
                79 -     82          657      6.0 %     74.7 %
                83 -     85          439      4.0 %     78.7 %
                86 -     89          623      5.7 %     84.4 %
                90 -     92          444      4.0 %     88.4 %
                93 -     96          624      5.7 %     94.1 %
                97 -    100          650      5.9 %    100.0 %
-- -------- ---------------  -----------  ----------  ----------
TOTALS          30 -    100       11,000    100.0 %

AVERAGE SEGMENT LENGTH=      65.2
```

Figure 39. STATIPRT—Database Statistics report (Part 9 of 9)

VALIDPRT data set

This section explains the VALIDPRT data set.

Function

The VALIDPRT data set contains the following reports produced by the HD Pointer Checker processor (FABPMAIN):

- Separator page for validation reports
- Scan of HISAM database report
- Scan of index database report
- Validation of a pointer to a target at SCAN (HDAM/HIDAM) report
- Legend for scan and validation report
- Description of all scanned database report
- Validation of a pointer to a target at CHECK report
- Legend for check process validation report

Separator page for validation reports

This separator page (see Figure 40) contains the title of “Validation Report,” and indicates that validation reports will follow this page. It is produced unless SEP=NO is specified on the PROC statement as input for the PROCCTL data set.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                    "SEPARATOR PAGE FOR VALIDATION"                          PAGE:      1
5655-K53                                                    DATE: 07/07/2006  TIME: 15.59.40                          FABPMAIN - V2.R2



                                V   V  AAA  L     IIIII DDDD   AAA  TTTTT IIIII  OOO  N   N
                                V   V A   A L       I   D   D A   A   T     I   O   O N   N
                                V   V A   A L       I   D   D A   A   T     I   O   O NN  N
                                V   V AAAAA L       I   D   D AAAAA   T     I   O   O N N N
                                V   V A   A L       I   D   D A   A   T     I   O   O N  NN
                                 V V  A   A L       I   D   D A   A   T     I   O   O N   N
                                  V   A   A LLLLL IIIII DDDD  A   A   T   IIIII  OOO  N   N



                                            RRRR  EEEEE PPPP   OOO  RRRR  TTTTT
                                            R   R E     P   P O   O R   R   T
                                            R   R E     P   P O   O R   R   T
                                            RRRR  EEE   PPPP  O   O RRRR    T
                                            R R   E     P     O   O R R     T
                                            R  R  E     P     O   O R  R    T
                                            R   R EEEEE P      OOO  R   R   T
```

Figure 40. VALIDPRT—Separator page for validation reports

Scan of HISAM Database report

This report (see Figure 41 on page 176 and Figure 42 on page 177) contains the following kinds of information:

- A summary description of a primary data set (VSAM KSDS) and an overflow data set (VSAM ESDS) of the HISAM (including SHISAM) database
- A list of errors that were detected by the SCAN processor in those data sets
- The total number of records (of various types) that were involved in the processing of the HISAM (including SHISAM) database

This report is produced for each data set specified on the DATABASE statement in the PROCCTL data set.

The report fields are as follows:

DBNAME DB# DSG# DSNAME
The name of the DBD, the database number (in hexadecimal), the data set group number (in hexadecimal), and the name of the data set.

DATABASE ORGANIZATION
This field indicates that the database is an index database. For VSAM databases, it also indicates whether it contains unique keys or duplicate keys.

ACCESS METHOD
This field indicates how many data sets comprise this database and what their access methods are (VSAM KSDS or ESDS).

PRIME or OVERFLOW DDNAME
The ddname as coded on the DD1= keyword or OVFLW= keyword of the DATASET macro in the DBD.

DB BLOCKSIZE
The block size or CI size of the database data set.

DB LRECL
The logical record length of the database data set.

DB DEVICE TYPE
The device type on which the database data set is located.

If the specified data set is an image copy database data set, the following four fields are not applicable:

CYLS/DEVICE
The number of cylinders on the device on which the database data set is located

TRKS/CYL
The number of tracks on one cylinder on the device on which the database data set is located

MAXIMUM BLOCKSIZE
The largest block size allowed on the device on which the database data set is located

MAXIMUM TRACK LENGTH
The length of one track on the device on which the database data set is located.

DB PHYS.BLKS/TRACK
The number of database blocks or CIs that are on one track on the device on which the database data set is located.

DB INDEX KEYLENGTH-1
The executable length of the key field (that is, key length - 1).

TP
The type of record. The HD Pointer Checker classifies its work records into types (T1, T2, and so on).

TARGET
The target of a pointer. The following four fields all pertain to it:

DB
The database number (in hexadecimal) that identifies the database containing the target of a pointer.

DG
The data set group number (in hexadecimal) that identifies the database containing the target of a pointer.

RBA
The relative byte address (in hexadecimal) of the target of a pointer. This is the actual value of the pointer itself.

SC
The segment code (in hexadecimal) of the target of a pointer.

SOURCE
The segment that contains the pointer (also called the source of the pointer). The following four fields all pertain to it:

DB
The database number (in hexadecimal) that identifies the database containing the segment that contains the pointer

DG
The data set group number (in hexadecimal) that identifies the database containing the segment that contains the pointer

RBA
The relative byte address (in hexadecimal) of the segment that contains the pointer

SC
The segment code (in hexadecimal) of the segment that contains the pointer.

PTR
The type of pointer such as LP, LTF, LTB, LCF, and LCL.

ERROR MESSAGES
The following two fields pertain to error messages:

NUMBER
The message number. You can find information about each message in Appendix A, “Messages and codes,” on page 595.

DESCRIPTION
The message text.

TOTAL
The following totals are contained in this report:

T1 The number of valid database segments that were detected by the SCAN processor in this HISAM data set group

T2 The number of areas containing slack bytes that were detected by the SCAN processor in this HISAM data set group

T5 The number of pointers whose target is missing that were detected by the SCAN in this data set group

T8 The number of pointers that were detected by the SCAN processor in the primary (KSDS) part of this HISAM data set group

T9 The number of pointers that were detected by the SCAN processor in the overflow (ESDS or OSAM) part of this HISAM data set group

TA
The number of keys of index source segment occurrences that are detected by the SCAN processor.

TC
The number of index source segment (ISS) occurrences, logical children, and the sum of CTR values that are detected by the SCAN processor.

TT The number of segment occurrences that are pointed to by symbolic pointers that are detected by the SCAN processor.

TU
The number of logical parent's concatenated keys (LPCK) that are detected by the SCAN processor.

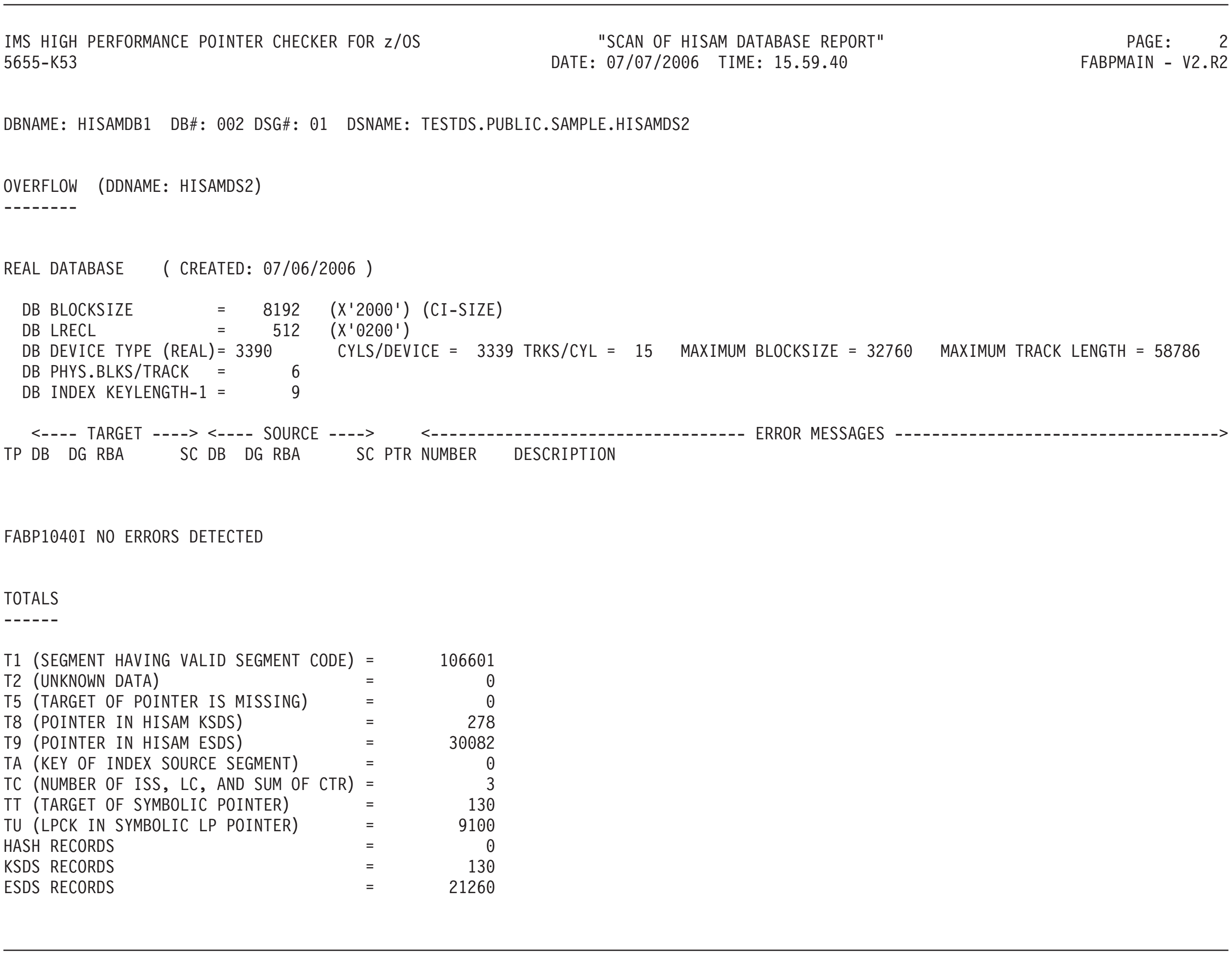
HASH RECORDS
The number of HASH records that were scanned by the SCAN processor

KSDS RECORDS
The number of logical records that were detected by the SCAN processor in the primary (KSDS) part of this HISAM (including SHISAM) data set group

ESDS RECORDS
The number of logical records that were detected by the SCAN processor in the overflow (ESDS) part of this HISAM data set group.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                   "SCAN OF HISAM DATABASE REPORT"                           PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                          FABPMAIN - V2.R2


DBNAME: HISAMDB1  DB#: 002 DSG#: 01  DSNAME: TESTDS.PUBLIC.SAMPLE.HISAMDS1


DATABASE ORGANIZATION  = HISAM
ACCESS METHOD          = VSAM KSDS/ESDS


PRIME  (DDNAME: HISAMDS1)
-----


REAL DATABASE    ( CREATED: 07/06/2006 )

  DB BLOCKSIZE         =    8192   (X'2000') (CI-SIZE)
  DB LRECL             =     510   (X'01FE')
  DB DEVICE TYPE (REAL)= 3390       CYLS/DEVICE =  3339 TRKS/CYL =  15   MAXIMUM BLOCKSIZE = 32760   MAXIMUM TRACK LENGTH = 58786
  DB PHYS.BLKS/TRACK   =       6
  DB INDEX KEYLENGTH-1 =       9

   <---- TARGET ----> <---- SOURCE ---->     <---------------------------------- ERROR MESSAGES ----------------------------------->
TP DB  DG RBA      SC DB  DG RBA      SC PTR NUMBER    DESCRIPTION



FABP1040I NO ERRORS DETECTED
```

Figure 41. VALIDPRT—Scan of HISAM Database report (Primary part)



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS               "SCAN OF HISAM DATABASE REPORT"                         PAGE:     2
5655-K53                                                DATE: 07/07/2006  TIME: 15.59.40                        FABPMAIN - V2.R2


DBNAME: HISAMDB1  DB#: 002 DSG#: 01  DSNAME: TESTDS.PUBLIC.SAMPLE.HISAMDS2


OVERFLOW  (DDNAME: HISAMDS2)
--------


REAL DATABASE    ( CREATED: 07/06/2006 )

  DB BLOCKSIZE         =    8192   (X'2000') (CI-SIZE)
  DB LRECL             =     512   (X'0200')
  DB DEVICE TYPE (REAL)= 3390       CYLS/DEVICE =  3339 TRKS/CYL =  15   MAXIMUM BLOCKSIZE = 32760   MAXIMUM TRACK LENGTH = 58786
  DB PHYS.BLKS/TRACK   =       6
  DB INDEX KEYLENGTH-1 =       9

   <---- TARGET ----> <---- SOURCE ---->     <---------------------------------- ERROR MESSAGES ----------------------------------->
TP DB  DG RBA      SC DB  DG RBA      SC PTR NUMBER    DESCRIPTION



FABP1040I NO ERRORS DETECTED


TOTALS
------

T1 (SEGMENT HAVING VALID SEGMENT CODE) =       106601
T2 (UNKNOWN DATA)                      =            0
T5 (TARGET OF POINTER IS MISSING)      =            0
T8 (POINTER IN HISAM KSDS)             =          278
T9 (POINTER IN HISAM ESDS)             =        30082
TA (KEY OF INDEX SOURCE SEGMENT)       =            0
TC (NUMBER OF ISS, LC, AND SUM OF CTR) =            3
TT (TARGET OF SYMBOLIC POINTER)        =          130
TU (LPCK IN SYMBOLIC LP POINTER)       =         9100
HASH RECORDS                           =            0
KSDS RECORDS                           =          130
ESDS RECORDS                           =        21260
```

Figure 42. VALIDPRT—Scan of HISAM Database report (Overflow part and totals)

Scan of Index Database report

This report (see Figure 43 on page 181 and Figure 44 on page 182) contains the following kinds of information:

- A summary description of data sets (VSAM KSDS and ESDS) of the index database.

  Note: The VSAM ESDS is used only when a secondary index database has duplicate keys.
- A list of errors that were detected by the SCAN processor in those data sets.
- The total number of records (of various types) that were involved in the processing of the index database.

This report is produced for each data set specified on the DATABASE statement in the PROCCTL data set. It contains at least one page for each of its data sets.

The report fields are as follows:

DBNAME DB# PARTNAME PART ID REORG# DSG# DSNAME
The name of the DBD, the database number (in hexadecimal), the name of the partition, partition id, partition reorg number, the data set group number (in hexadecimal) or the data set group ID (in an alphabetical character) of the index database, and the name of the data set. The reorg number is always shown as N/A (Not Applicable).

DATABASE ORGANIZATION
This field indicates that the database is an index database. For VSAM databases, it also indicates whether it contains unique keys or duplicate keys.

ACCESS METHOD
This field indicates how many data sets comprise this database and what their access methods are (VSAM KSDS or ESDS).

INDEXED DATABASE
The name of the DBD, as coded on the NAME= keyword of the DATASET macro in the DBD, of the *primary* database that is being indexed.

SECONDARY INDEX DEFINITION
The following information about the secondary index definitions is in this report:

SOURCE SEGMENT NAME
The name of the index source segment.

TARGET SEGMENT NAME
The name of the index target segment.

SPARSE INDEX
Whether sparse indexing is defined or not (YES or NO). If sparse indexing is defined, the following information is given:

NULL VALUE
The value of a user-supplied NULLVAL=.

SECONDARY INDEX MAINTENANCE EXIT
The name of a user-supplied secondary index maintenance exit routine.

INDEX or OVERFLOW DDNAME
The ddname, as coded on the DD1= keyword or OVFLW= keyword of the DATASET macro in the DBD, of the *index* database.

DB BLOCKSIZE
The block size or CI size of the database data set.

DB LRECL
The logical record length of the database data set.

DB DEVICE TYPE
The device type on which the database data set is located.

If the specified data set is an image copy database data set, the following four fields are not applicable:

CYLS/DEVICE
The number of cylinders on the device on which the database data set is located

TRKS/CYL
The number of tracks on one cylinder on the device on which the database data set is located

MAXIMUM BLOCKSIZE
The largest block size allowed on the device on which the database data set is located

MAXIMUM TRACK LENGTH
The length of one track on the device on which the database data set is located.

DB PHYS.BLKS/TRACK
The number of database blocks or CIs that are on one track on the device on which the database data set is located.

DB INDEX KEYLENGTH-1
The executable length of the key field (that is, key length - 1).

TP
The type of record. The HD Pointer Checker classifies its work records into types (T6, T7, and so on).

TARGET
The target of a pointer. The following four fields all pertain to it:

DB
The database number (in hexadecimal) that identifies the database containing the target of a pointer.

PID
The partition ID (in decimal) that identifies the partition containing the target of a pointer.

DG
The data set group number (in hexadecimal) or the data set group ID (in an alphabetical character) that identifies the database containing the target of a pointer.

RBA
The relative byte address (in hexadecimal) of the target of a pointer. This is the actual value of the pointer itself.

SC
The segment code (in hexadecimal) of the target of a pointer.

SOURCE
The segment that contains the pointer (also called the source of the pointer). The following eight fields all pertain to it:

DB
The database number (in hexadecimal) that identifies the database containing the segment that contains the pointer.

DG
The data set group number (in hexadecimal) or the data set group ID (in an alphabetical character) that identifies the segment that contains the pointer.

RBA
The relative byte address (in hexadecimal) of the segment that contains the pointer.

SC
The segment code (in hexadecimal) of the segment that contains the pointer. This will always be ‘01’ for an index database.

PTR
The type of pointer.

IN:	Index
OF:	Index, Overflow
SX:	Secondary Index
SXO:	Secondary Index, Overflow

RRN
The relative-record number (in hexadecimal) of the segment that contains the pointer.

CHAIN
The pointer (in hexadecimal) to the next index or overflow record.

KEY
The key field (in character format) in the index segment. If the key length is longer than 30 bytes, only the first 30 bytes are printed.

MESSAGES
The following two fields pertain to messages:

NUMBER
The message number. You can find information about each message in Appendix A, “Messages and codes,” on page 595.

DESCRIPTION
The message text.

TOTALS
The following totals are contained in this report:

INDEX: TOTAL VALID SEGMENTS
The number of valid database segments that were detected by the SCAN processor in the VSAM KSDS part of this index database

OVERFLOW: TOTAL VALID SEGMENTS
The number of valid database segments that were detected by the SCAN processor in the overflow (VSAM ESDS) part of this index database

TOTAL DELETED SEGMENT
The number of deleted database segments that were detected by the SCAN processor in the primary or overflow part of this index database

T6 (POINTER RECORD IN INDEX DATABASE)
The number of pointers that were detected by the SCAN processor in the VSAM KSDS part of this index database

T7 (POINTER RECORD IN INDEX OVERFLOW DATABASE)
The number of pointers that were detected by the SCAN processor in the overflow (VSAM ESDS) part of this index database

HASH RECORDS
The number of HASH records that were created by the SCAN processor in the primary or overflow part of this index database.

TC (NUMBER OF INDEX POINTER SEGMENT TYPES)
The number of index pointer segment types that the SCAN processor detected in this data set.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                  "SCAN OF INDEX DATABASE REPORT"                          PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                          FABPMAIN - V2.R2


DBNAME: TPFOX1    DB#: 006 PARTNAME: TPFOX1A  PART ID: 00001 REORG#: N / A  DSG#:  A
DSNAME: TESTDS.PUBLIC.SAMPLE.TPFOX1.A00001


DATABASE ORGANIZATION  = INDEX (UNIQUE KEYS)
ACCESS METHOD          = VSAM KSDS


INDEXED DATABASE       = TPFOH2


SECONDARY INDEX DEFINITION
--------------------------

  SOURCE SEGMENT NAME              =  BDEP1LV2
  TARGET SEGMENT NAME              =  BROOTLV1
  SPARSE INDEX                     =        NO


INDEX  (DDNAME: TPFOX1AA)
-----


REAL DATABASE    ( CREATED: 07/06/2006 )

  DB BLOCKSIZE         =     512   (X'0200') (CI-SIZE)
  DB LRECL             =      54   (X'0036')
  DB DEVICE TYPE (REAL)= 3390       CYLS/DEVICE =  3339 TRKS/CYL =  15   MAXIMUM BLOCKSIZE = 32760   MAXIMUM TRACK LENGTH = 58786
  DB PHYS.BLKS/TRACK   =      49
  DB INDEX KEYLENGTH-1 =      15

   <------- TARGET -------> <-------------------------- SOURCE -------------------------> <----------- MESSAGES ------------------->
TP DB  PID   DG RBA      SC DB  DG RBA      SC PTR RRN      CHAIN  KEY                    NUMBER    DESCRIPTION




FABP1180I SCAN COMPLETED
```

Figure 43. VALIDPRT—Scan of Index Database report (Index)

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                   "SCAN OF INDEX DATABASE REPORT"                          PAGE:     2
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                         FABPMAIN - V2.R2


DBNAME: TPFOX1    DB#: 006 PARTNAME: TPFOX1A  PART ID: 00001 REORG#: N / A  DSG#:  A
DSNAME: TESTDS.PUBLIC.SAMPLE.TPFOX1.A00001

   <------- TARGET -------> <-------------------------- SOURCE --------------------------> <----------- MESSAGES ------------------->
TP DB  PID   DG RBA      SC DB  DG RBA      SC PTR RRN      CHAIN  KEY                    NUMBER    DESCRIPTION




TOTALS
------

INDEX    :  TOTAL VALID SEGMENTS                            =          9178
            TOTAL DELETED SEGMENTS                          =             0
            T6 (POINTER RECORD IN INDEX DATABASE)           =          9178
            HASH RECORDS                                    =             0

OVERFLOW :  TOTAL VALID SEGMENTS                            =             0
            TOTAL DELETED SEGMENTS                          =             0
            T7 (POINTER RECORD IN INDEX OVERFLOW DATABASE)  =             0
            HASH RECORDS                                    =             0

            TU (SYMBOLIC POINTER RECORD IN INDEX AND OVERFLOW)
                                                            =             0
            TC (NUMBER OF INDEX POINTER SEGMENT TYPES)      =             1
```

Figure 44. VALIDPRT—Scan of Index Database report (Overflow)

Validation of a Pointer to a Target at SCAN (HDAM/HIDAM/ PHDAM/PHIDAM) report

This report (see Figure 45 on page 187) contains the following kinds of information:

- A summary description of the database data set group
- A list of errors that were detected by the SCAN processor in that data set group
- The total number of records (of various types) that were involved in the processing of the data set group.

This report is produced for each database data set group specified on the DATABASE statement in the PROCCTL data set for an HD database. It contains at least one page for each of its data sets. It is not produced when HASH=YES is specified on the PROC statement.

The record fields are as follows:

DBNAME DB# PARTNAME PART ID REORG# DSG# DDNAME DSNAME
The name of the DBD, the database number (in hexadecimal), the name of the partition, partition id, partition reorg number, the data set group number (in hexadecimal) or the data set group ID (in an alphabetical character), the ddname of the primary database, and the name of the data set.

DATABASE ORGANIZATION
This field indicates whether the database is HIDAM or HDAM.

ACCESS METHOD
This field indicates whether the database is VSAM/ESDS or OSAM.

DB BLOCKSIZE
The block size or CI size of the database data set.

DB LRECL
The logical record length of the database data set.

DB DEVICE TYPE
The device type on which the database data set is located.

If the specified data set is an image copy database data set, the following four fields are not applicable:

CYLS/DEVICE
The number of cylinders on the device on which the database data set is located

TRKS/CYL
The number of tracks on one cylinder on the device on which the database data set is located

MAXIMUM BLOCKSIZE
The largest block size allowed on the device on which the database data set is located

MAXIMUM TRACK LENGTH
The length of one track on the device on which the database data set is located.

DB PHYS.BLKS/TRACK
The number of database blocks or CIs that are on one track on the device on which the database data set is located.

TP
The type of record. The HD Pointer Checker classifies its work records into types (T0, T1, and so on).

DUMP NO.
The relative block number of the block or CI that contains the invalid pointer or error. It is the same number as the dump number in the Block Map/ Block Dump report.

TARGET
The target of a pointer. The following four fields all pertain to it:

DB
The database number (in hexadecimal) that identifies the database containing the target of a pointer.

PID
The partition ID (in decimal) that identifies the partition containing the target of a pointer.

DG
The data set group number (in hexadecimal) that identifies the database containing the target of a pointer.

RBA
The relative byte address (in hexadecimal) of the target of a pointer. This is the actual value of the pointer itself.

SC
The segment code (in hexadecimal) of the target of a pointer.

SOURCE
The segment containing the pointer (also called the source of the pointer). The following four fields all pertain to it:

DB
The database number (in hexadecimal) that identifies the database containing the segment that contains the pointer

DG
The data set group number (in hexadecimal) or the data set group ID (in an alphabetical character) that identifies the database containing the target of a pointer.

RBA
The relative byte address (in hexadecimal) of the segment that contains the pointer

SC
The segment code (in hexadecimal) of the segment that contains the pointer.

ADDRESS OF POINTER
The following six fields all pertain to the pointer:

BLOCK#
The relative block number of the block or CI that contains the pointer.

Note: The first bit map is always in block 1, regardless of whether the database is OSAM or VSAM/ESDS; therefore, block 0 does not exist for OSAM databases.

VOLUME
The volume serial number of the device that contains the pointer.

CCCCHHHHRR
The actual direct access address of the record that contains the pointer. This is a 10-digit hexadecimal number. CCCC is the cylinder, HHHH is the track, and RR is the record number.

OFST
The hexadecimal displacement of the pointer within its record.

PTR
The type of pointer such as PTF, PTB, PP, PCF, PCL, LTF, LTB, LP, LCF, LCL, HF, and HB.

RBA
The relative byte address (in hexadecimal) of the pointer.

ERROR MESSAGES
The following two fields pertain to error messages:

NUMBER
The message number. You can find information about each message in Appendix A, “Messages and codes,” on page 595.

DESCRIPTION
The message text.

TOTALS
The following totals are contained in this report:

T0 The number of valid free space elements that were detected by the SCAN processor in this data set group.

T1 The number of valid database segments that were detected by the SCAN processor in this data set group.

T2 The number of areas containing slack bytes that were detected by the SCAN processor in this data set group. Slack bytes represent an area in the data part of a segment that could not be classified as either segments or free space.

T3 The number of pointers that were detected by the SCAN processor in this data set group and that will be validated in the CHECK process.

T4 The number of pointers that were detected by the SCAN processor in this data set group whose target has an unexpected segment code.

T5 The number of pointers that were detected by the SCAN processor in this data set group whose target is missing.

TA
The number of keys of index source segment which were detected by the SCAN processor.

TC
The number of index source segment types that the SCAN processor detected in this data set group.

HASH RECORDS
The number of HASH records that were scanned by the SCAN processor.

TS TOTAL POINTERS CHECKED (TARGET IN SAME BLOCK)
The number of pointers that were detected by the SCAN processor in this data set group that were verified and whose target was in the same block as the pointer.

TX TOTAL POINTERS CHECKED (TARGET IN ADJACENT BLOCK)
The number of pointers that were detected by the SCAN processor in this data set group that were verified and whose target was in a block adjacent to the block containing the pointer.

POINTER ERRORS FROM IN-CORE CHECK
The number of pointer error messages that were printed in this report.

HIGHEST RBA
The relative byte address of the last segment in the data set group.

TOTAL BLOCKS
The number of database blocks or CIs that were processed by the SCAN processor.

% POINTERS VALIDATED FROM IN-CORE CHECK
The percentage of all pointers in this data set group that were checked by the SCAN processor.

T2LEN
The maximum length of T2 record which is not considered to be error. This length has been specified with T2CHK= parameter on the OPTION statement (the default value is 7).

T2NUM
The maximum number of the T2 records which are not considered to be error. This number has been specified with T2CHK= parameter on the OPTION statement (the default value is 0).

# OF RECS LESS THAN EQUAL T2LEN
The number of T2 records whose length is less than or equal to the specified T2LEN.

# OF RECS BETWEEN 8 AND T2LEN
The number of T2 records whose length is between 8 and the specified T2LEN.

% T2 LENGTH IN ALL BLOCKS
The percentage of the total length of T2 records in this data set group.

TOTAL T2 LENGTH
The total length of T2 records.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS           "VALIDATION OF A POINTER TO A TARGET AT SCAN REPORT"             PAGE:     1
5655-K53                                                   DATE: 07/13/2006  TIME: 10.59.50                          FABPMAIN - V2.R2


DBNAME: HDAMDB2   DB#: 001 DSG#: 01  DDNAME: HDAMDS4           DSNAME: TESTDS.PUBLIC.SAMPLE.HDAMDS4


DATABASE ORGANIZATION  = HDAM
ACCESS METHOD          = VSAM ESDS


REAL DATABASE    ( CREATED: 07/06/2006 )

  DB BLOCKSIZE         =    1024   (X'0400') (CI-SIZE)
  DB LRECL             =    1017   (X'03F9')
  DB DEVICE TYPE (REAL)= 3390       CYLS/DEVICE =  3339 TRKS/CYL =  15   MAXIMUM BLOCKSIZE = 32760   MAXIMUM TRACK LENGTH = 58786
  DB PHYS.BLKS/TRACK   =      33


   DUMP<---- TARGET ----> <---- SOURCE ----> <----------- ADDRESS OF POINTER ----------->
TP NO. DB  DG RBA      SC DB  DG RBA      SC   BLOCK# VOLUME CCCCHHHHRR OFST PTR RBA
                                                                                       <------------- ERROR MESSAGES -------------->
                                                                                       NUMBER    DESCRIPTION

T0 001 001 01 00015008                             84 PMR004 02BE000213 0008           FABP0030E BAD FREE SPACE ELEMENT
T2 001 001 01 00015008 00                          84 PMR004 02BE000213 0008           FABP0410E 03F0 (HEX) BYTES OF UNKNOWN DATA
T5 OUT 001 01 0001FF6A 03 001 01 000158FC 03      126 PMR004 02BE000215 00FE PTF 000158FE
                                                                                       FABP0960E TARGET IS NOT A VALID SEGMENT
T5 OUT 001 01 00027808 01 001 01 00007804 00      157 PMR004 02BE00001F 0004 RAP 00007804
                                                                                       FABP0960E TARGET IS NOT A VALID SEGMENT
T5 002 001 01 0003D17E 02 001 01 0003D004 02      244 PMR004 02BE00070E 0006 PTF 0003D006
                                                                                       FABP0960E TARGET IS NOT A VALID SEGMENT


TOTALS
------

T0 RECORD FOR EACH FREE SPACE RANGE                  =            2473           T2LEN                            =               7
T1 TARGET HAVING VALID SEGMENT CODE                  =           18087           T2NUM                            =               0
T2 UNKNOWN DATA                                      =               1           # OF RECS LESS THAN EQUAL T2LEN  =               0
T3 POINTER RECORD TO BE VALIDATED IN CHECK PROCESS   =            9234           # OF RECS BETWEEN 8 AND T2LEN    =           N / A
T4 TARGET POINTER HAS UNEXPECTED SEGMENT CODE        =               0           % T2 LENGTH IN ALL BLOCKS        =             0.0
T5 TARGET OF POINTER IS MISSING                      =               3           TOTAL T2 LENGTH                  =            1008
TA KEY OF INDEX SOURCE SEGMENT                       =               0
TC NUMBER OF ISS, LC, AND SUM OF CTR FIELD VALUE     =               3
TT TARGET OF SYMBOLIC POINTER                        =              70
TU LPCK IN SYMBOLIC LP POINTER                       =            9100
HASH RECORDS                                         =               0
TS TOTAL POINTERS CHECKED (TARGET IN SAME BLOCK)     =           29041
TX TOTAL POINTERS CHECKED (TARGET IN ADJACENT BLOCK) =            6943
POINTER ERRORS FROM IN-CORE CHECK                    =               5
HIGHEST RBA                                          =        00245ABC
TOTAL BLOCKS                                         =            2474
% POINTERS VALIDATED FROM IN-CORE CHECK              =            79.5
```

Figure 45. VALIDPRT—Validation of a Pointer to a Target at SCAN (HDAM/HIDAM/PHDAM/PHIDAM) report

Legend for Scan and Validation report

This report (see Figure 46) contains a brief description of the headings and abbreviations used in the Scan of HISAM Database report, the Scan of Index Database report, and the Validation of a Pointer to a Target at SCAN (HDAM/HIDAM) report.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS             "LEGEND FOR SCAN / VALIDATION REPORT"                        PAGE:     1
5655-K53                                                DATE: 07/07/2006  TIME: 15.59.40                          FABPMAIN - V2.R2


TP ..... RECORD TYPE (T1, T2, ETC.)

  T0 ..... RECORD FOR EACH FREE SPACE RANGE
  T1 ..... SEGMENT HAVING VALID SEGMENT CODE
  T2 ..... SLACK BYTES MORE THAN SPECIFIED T2LEN (UNKNOWN DATA)
  T3 ..... POINTER RECORD TO BE VALIDATED IN CHECK PROCESS
  T4 ..... TARGET POINTER HAS UNEXPECTED SEGMENT CODE
  T5 ..... TARGET OF POINTER IS MISSING
  T6 ..... POINTER IN INDEX DATABASE
  T7 ..... POINTER IN INDEX OVERFLOW DATABASE
  T8 ..... POINTER IN HISAM KSDS
  T9 ..... POINTER IN HISAM ESDS
  TA ..... KEY OF INDEX SOURCE SEGMENT
  TC ..... NUMBER OF ISS, IPS, LC, AND SUM OF COUNTER VALUE
  TT ..... TARGET OF SYMBOLIC POINTER
  TU ..... SYMBOLIC POINTER

DUMP NO. . BLK MAP/DUMP NUMBER FOR PTR/TGT IN SAME BLK, ELSE "OUT"

TARGET OF POINTER:

  DB ..... DATABASE NUMBER OF TARGET SEGMENT
  DG ..... DATA SET GROUP OF TARGET SEGMENT
  RBA .... RELATIVE BYTE ADDRESS OF TARGET SEGMENT
  SC ..... SEGMENT CODE OF TARGET SEGMENT
     ..... ACTUAL (IF TP=T1) OR EXPECTED TARGET SEGMENT CODE

SOURCE OF POINTER:

  DB ..... DATABASE NUMBER OF SEGMENT CONTAINING POINTER
  DG ..... DATA SET GROUP OF SEGMENT CONTAINING POINTER
  RBA .... RELATIVE BYTE ADDRESS OF SEGMENT CONTAINING POINTER
  SC ..... SEGMENT CODE OF SEGMENT CONTAINING POINTER
  PTR..... POINTER TYPE
  RRN..... RELATIVE RECORD NUMBER OF INDEX OR OVERFLOW POINTER
  CHAIN .. POINTER TO NEXT INDEX OR OVERFLOW RECORD
  KEY .... FIRST 30 BYTES OF KEY

DISK ADDRESS OF POINTER:

  BLOCK# . BLOCK CONTAINING POINTER
  VOLUME . DIRECT ACCESS VOLUME ID
  CCCCHHHHRR OFST .. DASD (AMASPZAP) ADDRESS OF POINTER
  PTR .... POINTER TYPE
  RBA .... RBA OF POINTER IN ERROR

ERROR MESSAGES:

  NUMBER . ERROR MESSAGE NUMBER
  DESCRIPTION .. SHORT DESCRIPTION OF ERROR
```

Figure 46. VALIDPRT—Legend for Scan and Validation report

Description of All Scanned Databases report

This report (see Figure 47 on page 190) contains a list of all the databases that were scanned during your HD Pointer Checker run.

The report fields are as follows:

DBNAME
The name of the DBD as coded on the NAME= keyword of the DBD macro

PARTITION NAME
The name of the partition.

PARTITION ID
The partition ID (in decimal) that identifies the partition

DDNAME
The ddname as coded on the DD1= keyword or OVFLW= keyword of the DATASET macro in the DBD

DB#
The database number (in hexadecimal) that identifies the database

DSG#
The data set group number (in hexadecimal) or the data set group ID (in an alphabetical character) that identifies the database data set group.

DBLG#
The database logical group number which indicates that physical databases with the same database logical group number are logically related to each other

DB-ORGANIZATION
The type of database organization (HISAM, HIDAM, HDAM, INDEX, and so on)

ACCESS
The access method used by the database (VSAM, OSAM, or ISAM, and so on)

BLKSIZ
The block size or CI size of the database data set

LRECL
The logical record length of the database data set

DBTYPE
Indicates whether the database data set is a real database or an image copy database

CRT-DATE
The date on which the data set was created (if available)

CRT-TIME
The time at which the data set was created (if available)

DVCTYPE
The device type on which the database data set is located.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS             "DESCRIPTION OF ALL SCANNED DATABASE REPORT"                 PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                         FABPMAIN - V2.R2


DBNAME   PARTITION PARTITION DDNAME   DB# DSG# DBLG# DB-ORGANIZATION   ACCESS    BLKSIZ LRECL DBTYPE CRT-DATE   CRT-TIME DVCTYPE
         NAME      ID

HDAMDB2                      HDAMDS4  001  01   001  HDAM              VSAM ESDS   1024  1017  REAL  2006.07.06 17.36.26 3390
HISAMDB1                     HISAMDS1 002  01   001  HISAM             VSAM KSDS   8192   510  REAL  2006.07.06  N/AVAIL 3390
HISAMDB1                     HISAMDS2 002  01   001  HISAM       OFLW  VSAM KSDS   8192   512  REAL  2006.07.06 17.35.40 3390
TPFOH1   TPFOH1A   00001     TPFOH1AA 003   A   002  PHDAM             VSAM ESDS    512   505  REAL  2006.07.06 17.36.50 3390
TPFOH1   TPFOH1A   00001     TPFOH1AB 003   B   002  PHDAM             VSAM ESDS    512   505  REAL  2006.07.06 17.36.50 3390
TPFOH3   TPFOH3A   00001     TPFOH3AA 004   A   002  PHDAM             VSAM ESDS    512   505  REAL  2006.07.06 17.38.30 3390
TPFOH2   TPFOH2A   00001     TPFOH2AA 005   A   002  PHIDAM            VSAM ESDS    512   505  REAL  2006.07.06 17.38.16 3390
TPFOH2   TPFOH2A   00001     TPFOH2AX 005   X   002  PHIDAM IDX        VSAM KSDS    512    14  REAL  2006.07.06  N/AVAIL 3390
TPFOX1   TPFOX1A   00001     TPFOX1AA 006   A   002  PSINDEX           VSAM KSDS    512    54  REAL  2006.07.06  N/AVAIL 3390
```

Figure 47. VALIDPRT—Description of All Scanned Database report (Part 1 of 2)

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS             "DESCRIPTION OF ALL SCANNED DATABASE REPORT"                 PAGE:     2
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                         FABPMAIN - V2.R2


PROGRAM FUNCTIONS
-----------------

CHECK PROCESS PERFORMS THE FOLLOWING POINTER CHECKS:

BASIC: (VALIDATION OF A POINTER TO A TARGET)
-----
       - VALIDATE POINTERS TO TARGET
       - DETECT MISSING TARGETS
       - DETECT TARGET IN FREE SPACE
       - DETECT TARGET SEGMENT CODE INVALID
       - DETECT DUPLICATE POINTERS TO TARGET
         (EXCEPT LP AND PP)
       - DETECT INVALID COMBINATIONS OF POINTERS TO TARGET

COMPREHENSIVE: (EVALUATION OF ALL PTRS TO SAME TARGET)
-------------
       - DETECT UNREFERENCED SEGMENTS
       - VERIFY INDEX TO HIDAM/PHIDAM ROOT PRESENT
       - VERIFY REQUIRED PHYSICAL POINTERS TO TARGET
       - VERIFY REQUIRED LOGICAL  POINTERS TO TARGET
       - CHECK COUNTER FIELD AGAINST ACTUAL LCHILD COUNT
       - VERIFY CORRESPONDING PHYSICALLY PAIRED SEGS PRESENT

          *****   W A R N I N G   *****

       COMPREHENSIVE CHECKS CAN ONLY BE MADE CORRECTLY
       IF ALL RELATED DATABASES, DATA SET GROUPS AND
       REQUIRED POINTERS HAVE BEEN SCANNED.

       IF ALL THE REQUIRED DATABASES, HAVE NOT BEEN SCANNED
       SPURIOUS MESSAGES WILL BE PRODUCED UNLESS
       INFO IS SUPPLED TO TURN OFF THE CHECKS NOT DESIRED
       CHECK=(CHK,NNNNNN)
          POSITION 1: 1 IF INDEX TO HIDAM/PHIDAM ROOT CHECK  (DEFAULT)
                      0 IF NO INDEX TO HIDAM/PHIDAM ROOT CHECK
          POSITION 2: 1 CHK FOR REQUIRED PHYS PTRS    (DEFAULT)
                      0 IF NO PHYSICAL PTR CHECK
          POSITION 3: 1 CHK FOR REQUIRED LOG. PTRS    (DEFAULT)
                      0 IF NO LOGICAL PTR CHECK
          POSITION 4: 1 CHECK CTR VERSUS #LCHILD(S)   (DEFAULT)
                      0 IF NO CTR/LCHILD COUNT CHECK
          POSITION 5: 1 VERIFY PHYS PAIRED SEGS       (DEFAULT)
                      0 IF NO PHYSICALLY PAIRED SEG CHECK
          POSITION 6: 1 DO NOT PRINT HASH FORMULAS    (DEFAULT)
                      0 PRINT HASH FORMULAS
```

Figure 47. VALIDPRT—Description of All Scanned Database report (Part 2 of 2)

Validation of a Pointer to a Target at CHECK report

This report (see Figure 48 on page 192) contains the following kinds of information:

- A list of errors that were detected by the CHECK processor in each data set group
- Various totals.

This report is produced for each primary database that was scanned by the SCAN processor. This report is not produced when HASH=YES is specified on the PROC statement in the PROCCTL data set.

The report fields are as follows:

DBNAME DB# PARTNAME PART ID REORG# DSG# DDNAME
The name of the DBD, the database number (in hexadecimal), the name of the partition, partition id, partition reorg number, the data set group number (in hexadecimal) or the data set group ID (in an alphabetical character) and the ddname.

TP
The type of record that is printed on this line. HD Pointer Checker classifies its work records into types (T0, T1, and so on).

TARGET
The target of a pointer. The following four fields all pertain to it:

DB
The database number (in hexadecimal) that identifies the database containing the target of a pointer.

PID
The partition ID (in decimal) that identifies the partition containing the segment that contains the pointer

DG
The data set group number (in hexadecimal) that identifies the database containing the target of a pointer.

RBA
The relative byte address (in hexadecimal) of the target of a pointer. This is the actual value of the pointer itself.

SC
The segment code (in hexadecimal) of the target of a pointer. This can also be the expected segment code, if the target is in error.

SOURCE
The segment that contains the pointer (also called the source of the pointer). The following four fields all pertain to it:

DB
The database number (in hexadecimal) that identifies the database containing the segment that contains the pointer

DG
The data set group number (in hexadecimal) that identifies the database containing the segment that contains the pointer

RBA
The relative byte address (in hexadecimal) of the segment that contains the pointer

SC
The segment code (in hexadecimal) of the segment that contains the pointer.

ADDRESS OF POINTER
The following six fields all pertain to the pointer itself:

BLOCK#
The relative block number of the block or CI that contains the pointer.

Note: The first bit map is always in block 1, regardless of whether the database is OSAM or VSAM/ESDS. So block 0 does not exist for OSAM databases.

VOLUME
The volume serial number of the device that contains the pointer.

CCCCHHHHRR
The actual direct access address of the record that contains the pointer. This is a 10-digit hexadecimal number. CCCC is the cylinder, HHHH is the track, and RR is the record number.

OFST
The hexadecimal displacement of the pointer within its record.

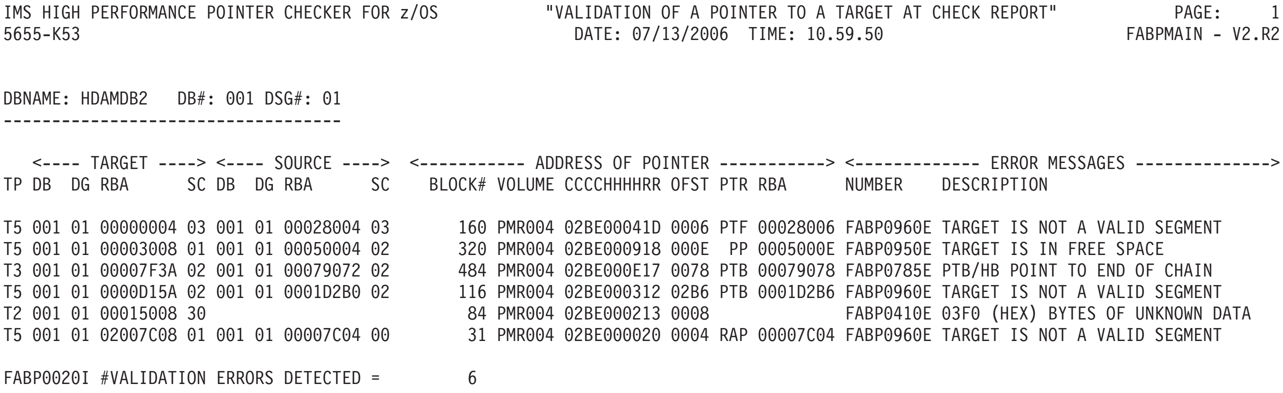
PTR
The type of pointer such as PTF, PTB, PP, PCF, PCL, LTF, LTB, LP, LCF, LCL, HF, HB, IN, INO, SX, and SXO

RBA
The relative byte address (in hexadecimal) of the pointer.

ERROR MESSAGES
The following two fields pertain to error messages:

NUMBER
The message number. You can find information about each message in Appendix A, “Messages and codes,” on page 595.

DESCRIPTION
The message text.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS          "VALIDATION OF A POINTER TO A TARGET AT CHECK REPORT"          PAGE:     1
5655-K53                                                  DATE: 07/13/2006  TIME: 10.59.50                      FABPMAIN - V2.R2


DBNAME: HDAMDB2   DB#: 001 DSG#: 01
-----------------------------------

   <---- TARGET ----> <---- SOURCE ---->  <----------- ADDRESS OF POINTER -----------> <------------- ERROR MESSAGES -------------->
TP DB  DG RBA      SC DB  DG RBA      SC    BLOCK# VOLUME CCCCHHHHRR OFST PTR RBA      NUMBER    DESCRIPTION

T5 001 01 00000004 03 001 01 00028004 03       160 PMR004 02BE00041D 0006 PTF 00028006 FABP0960E TARGET IS NOT A VALID SEGMENT
T5 001 01 00003008 01 001 01 00050004 02       320 PMR004 02BE000918 000E  PP 0005000E FABP0950E TARGET IS IN FREE SPACE
T3 001 01 00007F3A 02 001 01 00079072 02       484 PMR004 02BE000E17 0078 PTB 00079078 FABP0785E PTB/HB POINT TO END OF CHAIN
T5 001 01 0000D15A 02 001 01 0001D2B0 02       116 PMR004 02BE000312 02B6 PTB 0001D2B6 FABP0960E TARGET IS NOT A VALID SEGMENT
T2 001 01 00015008 30                           84 PMR004 02BE000213 0008              FABP0410E 03F0 (HEX) BYTES OF UNKNOWN DATA
T5 001 01 02007C08 01 001 01 00007C04 00        31 PMR004 02BE000020 0004 RAP 00007C04 FABP0960E TARGET IS NOT A VALID SEGMENT

FABP0020I #VALIDATION ERRORS DETECTED =         6
```

Figure 48. VALIDPRT—Validation of a Pointer to a Target at CHECK report

Legend for Check Process Validation report

This report (see Figure 49) contains a brief description of the headings and abbreviations used in the validation of a pointer to a target at CHECK report.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS          "LEGEND FOR CHECK PROCESS VALIDATION REPORT"               PAGE:     1
5655-K53                                                DATE: 07/07/2006  TIME: 15.59.40                       FABPMAIN - V2.R2


TP ...... RECORD TYPE (T1, T2, ETC.)

TARGET OF POINTER:

DB ...... DATABASE NUMBER OF TARGET SEGMENT
DG ...... DATA SET GROUP NUMBER OF TARGET SEGMENT
RBA ..... RELATIVE BYTE ADDRESS OF TARGET SEGMENT
SC ...... ACTUAL SEGCODE OF TGT (IF TP=T1), ELSE EXPECTED TGT SC

SOURCE OF POINTER:

DB ...... DATABASE NUMBER OF SEGMENT CONTAINING POINTER
DG ...... DATA SET GROUP NUMBER OF SEGMENT CONTAINING POINTER
RBA ..... RELATIVE BYTE ADDRESS OF SEGMENT CONTAINING POINTER
SC ...... SEGMENT CODE OF SEGMENT CONTAINING POINTER

DISK ADDRESS OF POINTER:

BLOCK# .. BLOCK CONTAINING POINTER
VOLUME .. DIRECT ACCESS VOLUME ID
CCCCHHHHRR OFST .. ADDR (AMASPZAP) OF PTR IN ERROR

PTR ..... POINTER TYPE
RBA ..... RBA OF POINTER IN ERROR

ERROR MESSAGES:

NUMBER ... ERROR MESSAGE NUMBER
DESCRIPTION .. SHORT DESCRIPTION OF ERROR

***** SNAP OF BLOCKS CONTAINING ERRORS, WHERE TGT AND PTR TO TGT ARE NOT IN THE SAME BLOCK

      1. USE BLOCK DUMP OPTION BY SPECIFYING BLOCKDUMP PARAMETER IN DATABASE STATEMENT
      2. OS UTILITIES
      3. VSAM ACCESS METHOD SERVICES

***** CTL RECORDS ARE WRITTEN TO BLKMAP PROCESS FOR:

      1. ALL POINTERS POINTING TO A SEG CONTAINING BAD POINTERS
      2. ALL POINTERS POINTING TO THE SEG WHICH IS POINTED AT BY A BAD POINTER
      3. ALL POINTERS POINTING TO A SEG WITH BAD CTR VALUES

***** PSEUDO ERROR MESSAGES MAY BE GENERATED WHEN INDEX OVERFLOW DATA SET PTR'S ARE VALIDATED/EVALUATED

CAUSE: DL/I DOES NOT SET THE DELETE FLAG IN INDEX OVERFLOW RECORD AFTER ROOT DELETION
       THESE RECORDS ARE LOCATED BY THE PTR CHECKER AS 'ACTIVE', ALTHOUGH THEY ARE INACTIVE TO DL/I
```

Figure 49. VALIDPRT—Legend for Check Process Validation report

EVALUPRT data set

This section explains the EVALUPRT data set.

Function

The EVALUPRT data set contains the following reports produced by the HD Pointer Checker processor (FABPMAIN):

- Separator page for evaluation reports
- Evaluation of All Pointers to the Same Target report
- Legend for Check Process Evaluation report
- Check Process Total report
- HASH Evaluation report
- Evaluation of Index Pointers and Keys report
- Database Repair Guidelines report
- Separator page for reconstruction reports
- DMB Directory and Control Card Format report
- Pointer Chain Reconstruction report
- Legend for Reconstruction report

Separator page for evaluation reports

This separator page (see Figure 50) contains the title of “Evaluation Report”, and indicates that evaluation reports will follow the page. It is produced unless SEP=NO is specified on the PROC statement as input for the PROCCTL data set.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                   "SEPARATOR PAGE FOR EVALUATION"                           PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                          FABPMAIN - V2.R2



                                EEEEE V   V  AAA  L     U   U  AAA  TTTTT IIIII  OOO  N   N
                                E     V   V A   A L     U   U A   A   T     I   O   O N   N
                                E     V   V A   A L     U   U A   A   T     I   O   O NN  N
                                EEE   V   V AAAAA L     U   U AAAAA   T     I   O   O N N N
                                E     V   V A   A L     U   U A   A   T     I   O   O N  NN
                                E      V V  A   A L     U   U A   A   T     I   O   O N   N
                                EEEEE   V   A   A LLLLL  UUU  A   A   T   IIIII  OOO  N   N



                                            RRRR  EEEEE PPPP   OOO  RRRR  TTTTT
                                            R   R E     P   P O   O R   R   T
                                            R   R E     P   P O   O R   R   T
                                            RRRR  EEE   PPPP  O   O RRRR    T
                                            R R   E     P     O   O R R     T
                                            R  R  E     P     O   O R  R    T
                                            R   R EEEEE P      OOO  R   R   T
```

Figure 50. EVALUPRT—Separator page for evaluation reports

Evaluation of All Pointers to the Same Target report

This report (see Figure 51 on page 198) contains the following kinds of information:

- A list of errors that were detected by the CHECK processor in each data set group
- Various totals.

This report is produced for each primary database that was scanned by the SCAN processor. This report is not produced when HASH=YES is specified on the PROC statement as input for the PROCCTL data set.

The report fields are as follows:

DBNAME DB# PARTNAME PART ID REORG# DSG# DDNAME
The name of the DBD, the database number (in hexadecimal), the name of the partition, partition id, partition reorg number, the data set group number (in hexadecimal) or the data set group ID (in an alphabetical character) and the ddname.

TP
The type of record that is printed on this line. The HD Pointer Checker classifies its work records into types (T0, T1, and so on).

TARGET
The target of a pointer. The following four fields all pertain to it:

DB
The database number (in hexadecimal) that identifies the database containing the target of a pointer.

DG
The data set group number (in hexadecimal) or the data set group ID (in an alphabetical character) that identifies the database containing the target of a pointer.

RBA
The relative byte address (in hexadecimal) of the target of a pointer. This is the actual value of the pointer itself.

SC
The segment code (in hexadecimal) of the target of a pointer. This can also be the expected segment code, if the target is in error.

The following six fields show various totals and flags that pertain to the segment:

SEGM PTRS
A flag (from the DBD) showing the pointers contained in the segment. Bit settings are as follows:

Bit	Pointer
0 (X'80')	CTR — Logical child counter
1 (X'40')	PTF — Physical twin forward
2 (X'20')	PTB — Physical twin backward
3 (X'10')	PP — Physical parent
4 (X'08')	LTF — Logical twin forward
5 (X'04')	LTB — Logical twin backward

6 (X'02')	LP — Logical parent
7 (X'01')	H — Hierarchical.

PHYS PT2TGT
A flag indicating the physical pointers to the segment that were detected by the SCAN and CHECK processes. Bit settings are as follows:

Bit	Pointer
0 (X'80')	HF — Hierarchical forward
1 (X'40')	HB — Hierarchical backward
2 (X'20')	PTF — Physical twin forward
3 (X'10')	PTB — Physical twin backward
4 (X'08')	PCF — Physical child first
5 (X'04')	PCL — Physical child last
6 (X'02')	RAP — Root anchor point
7 (X'01')	VLS — Pointer to data part of a split (variable—length) segment.

LOGICAL PT2TGT
A flag indicating the logical pointers to the segment that were detected by the SCAN and CHECK processes. Bit settings are as follows:

Bit	Pointer
0 (X'80')	LTF — Logical twin forward
1 (X'40')	LTB — Logical twin backward
2 (X'20')	LCF — Logical child first
3 (X'10')	LCL — Logical child last
4 (X'08')	LP — Logical parent
5 (X'04')	PP — Physical parent
6 (X'02')	IN/SX — HIDAM primary index pointer or secondary index pointer
7 (X'01')	SXO — Secondary index pointer from overflow

COUNTER VALUE
The value of the logical child counter in the segment

LCHILD COUNT
The number of logical children (from unidirectional and physically paired logical children) detected by the SCAN processor for the segment

#PHY PAIRED SEG POINTING TO TGT

LP= The number of logical parent pointers (from physically paired logical children) detected by the SCAN and CHECK processors to the segment

PP= The number of physical parent pointers (from physically paired logical children) detected by the SCAN and CHECK processors to the segment.

ERROR MESSAGES
The following two fields pertain to error messages:

NUMBER
The message number. You can find information about each message in Appendix A, “Messages and codes,” on page 595.

DESCRIPTION
The message text.

INDEX SEGMENT COUNTS STATISTICS
This report contains the following information about index counts and indexed databases:

INDEX DATABASE NAME
The name of the INDEX DBD.

INDEXED DATABASE NAME
The name of the INDEXed DBD.

INDEX SOURCE SEGMENT NAME
The name of the index source segment.

INDEX TARGET SEGMENT NAME
The name of the index target segment.

SPARSE INDEX
Whether sparse indexing is defined or not (YES or NO).

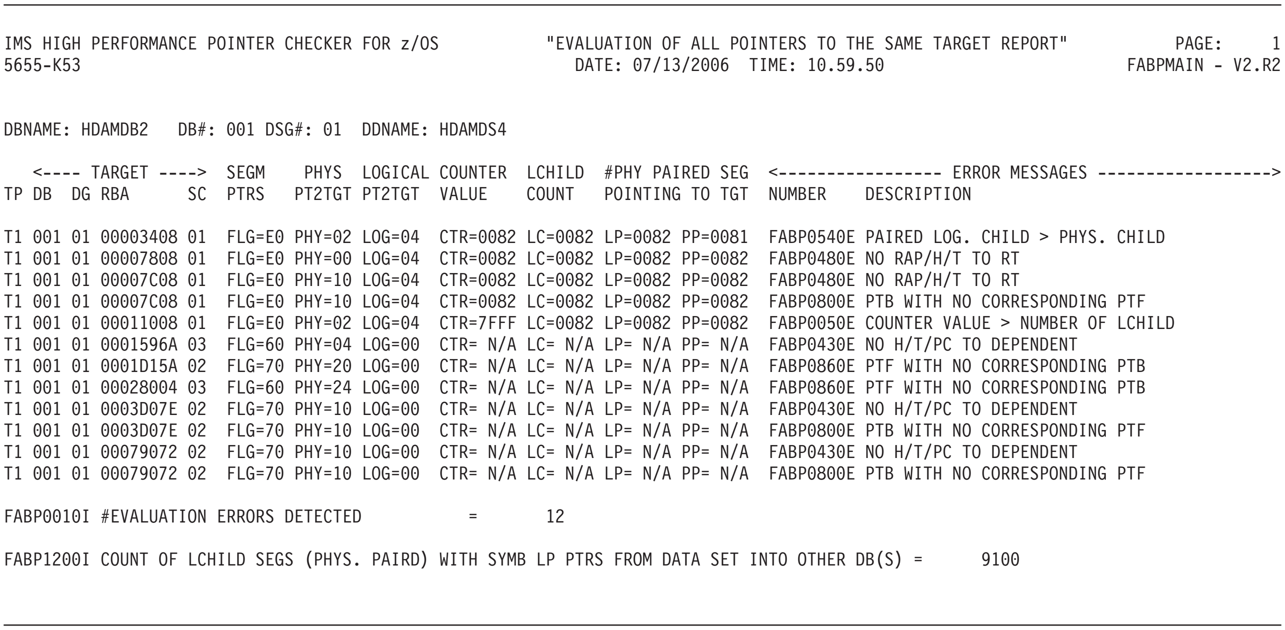
NUMBER OF INDEX POINTER SEGMENT
The number of index pointer segment.

NUMBER OF INDEX SOURCE SEGMENT
The number of index source segment.

NUMBER OF SUPPRESSED SEGMENT (NULLVAL)
The number of index pointer segments that are suppressed by NULLVAL=.

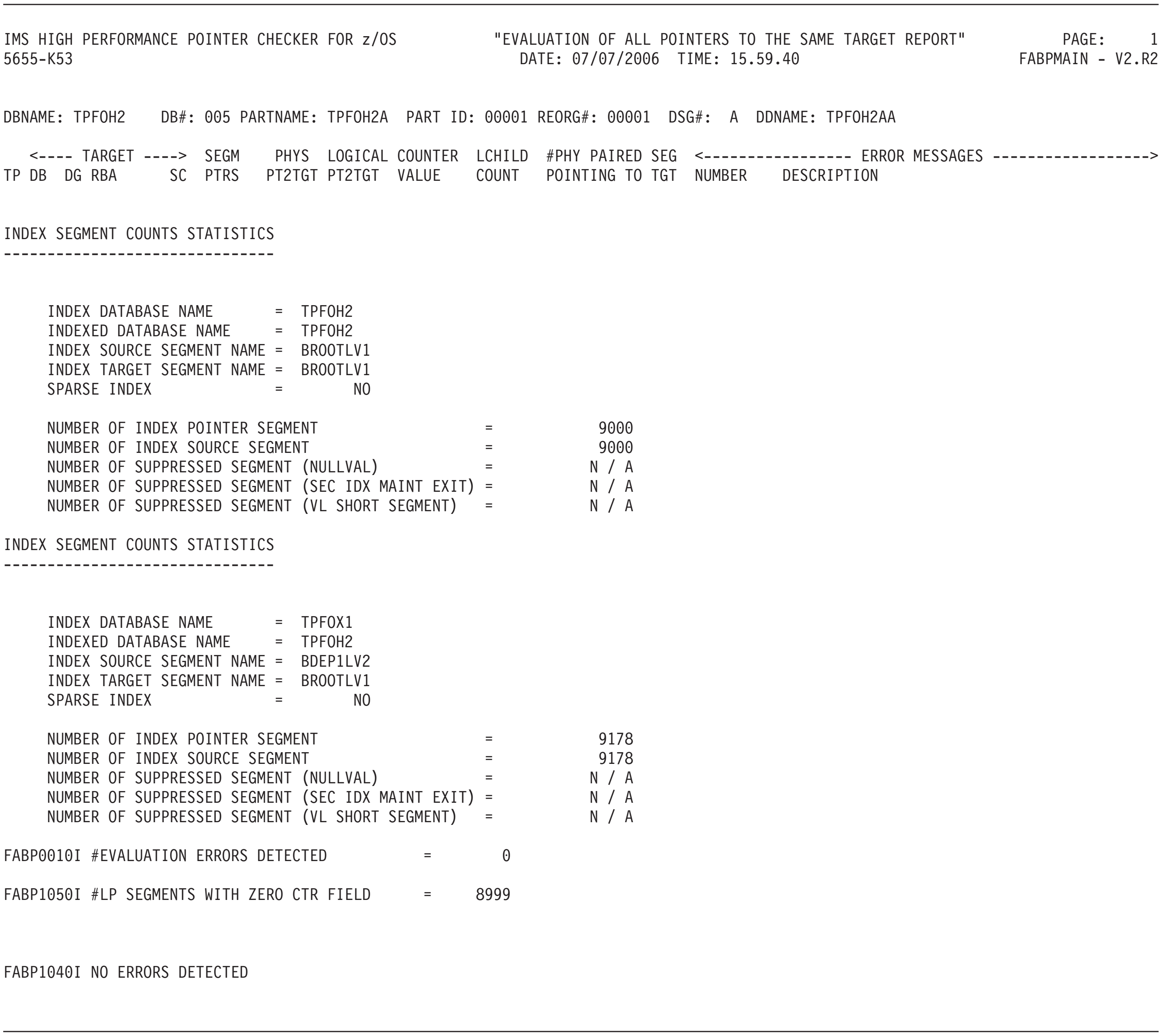
NUMBER OF SUPPRESSED SEGMENT (SEC IDX MAINT EXIT)
The number of index pointer segments that are suppressed by the Secondary Index Maintenance Exit routine.

NUMBER OF SUPPRESSED SEGMENT (VL SHORT SEGMENT)
The number of index pointer segments that are suppressed by the missing index source segment data of the search field for the index pointer segment.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS          "EVALUATION OF ALL POINTERS TO THE SAME TARGET REPORT"          PAGE:     1
5655-K53                                                  DATE: 07/13/2006  TIME: 10.59.50                         FABPMAIN - V2.R2


DBNAME: HDAMDB2   DB#: 001 DSG#: 01  DDNAME: HDAMDS4

   <---- TARGET ---->  SEGM    PHYS  LOGICAL COUNTER  LCHILD  #PHY PAIRED SEG  <----------------- ERROR MESSAGES ------------------>
TP DB  DG RBA      SC  PTRS   PT2TGT PT2TGT  VALUE    COUNT   POINTING TO TGT  NUMBER    DESCRIPTION

T1 001 01 00003408 01  FLG=E0 PHY=02 LOG=04  CTR=0082 LC=0082 LP=0082 PP=0081  FABP0540E PAIRED LOG. CHILD > PHYS. CHILD
T1 001 01 00007808 01  FLG=E0 PHY=00 LOG=04  CTR=0082 LC=0082 LP=0082 PP=0082  FABP0480E NO RAP/H/T TO RT
T1 001 01 00007C08 01  FLG=E0 PHY=10 LOG=04  CTR=0082 LC=0082 LP=0082 PP=0082  FABP0480E NO RAP/H/T TO RT
T1 001 01 00007C08 01  FLG=E0 PHY=10 LOG=04  CTR=0082 LC=0082 LP=0082 PP=0082  FABP0800E PTB WITH NO CORRESPONDING PTF
T1 001 01 00011008 01  FLG=E0 PHY=02 LOG=04  CTR=7FFF LC=0082 LP=0082 PP=0082  FABP0050E COUNTER VALUE > NUMBER OF LCHILD
T1 001 01 0001596A 03  FLG=60 PHY=04 LOG=00  CTR= N/A LC= N/A LP= N/A PP= N/A  FABP0430E NO H/T/PC TO DEPENDENT
T1 001 01 0001D15A 02  FLG=70 PHY=20 LOG=00  CTR= N/A LC= N/A LP= N/A PP= N/A  FABP0860E PTF WITH NO CORRESPONDING PTB
T1 001 01 00028004 03  FLG=60 PHY=24 LOG=00  CTR= N/A LC= N/A LP= N/A PP= N/A  FABP0860E PTF WITH NO CORRESPONDING PTB
T1 001 01 0003D07E 02  FLG=70 PHY=10 LOG=00  CTR= N/A LC= N/A LP= N/A PP= N/A  FABP0430E NO H/T/PC TO DEPENDENT
T1 001 01 0003D07E 02  FLG=70 PHY=10 LOG=00  CTR= N/A LC= N/A LP= N/A PP= N/A  FABP0800E PTB WITH NO CORRESPONDING PTF
T1 001 01 00079072 02  FLG=70 PHY=10 LOG=00  CTR= N/A LC= N/A LP= N/A PP= N/A  FABP0430E NO H/T/PC TO DEPENDENT
T1 001 01 00079072 02  FLG=70 PHY=10 LOG=00  CTR= N/A LC= N/A LP= N/A PP= N/A  FABP0800E PTB WITH NO CORRESPONDING PTF

FABP0010I #EVALUATION ERRORS DETECTED          =       12

FABP1200I COUNT OF LCHILD SEGS (PHYS. PAIRD) WITH SYMB LP PTRS FROM DATA SET INTO OTHER DB(S) =      9100
```

Figure 51. EVALUPRT—Evaluation of All Pointers to the Same Target report (Part 1 of 2)



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS           "EVALUATION OF ALL POINTERS TO THE SAME TARGET REPORT"          PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                           FABPMAIN - V2.R2


DBNAME: TPFOH2    DB#: 005 PARTNAME: TPFOH2A  PART ID: 00001 REORG#: 00001  DSG#:  A  DDNAME: TPFOH2AA

   <---- TARGET ---->  SEGM    PHYS  LOGICAL COUNTER  LCHILD  #PHY PAIRED SEG  <----------------- ERROR MESSAGES ------------------>
TP DB  DG RBA      SC  PTRS   PT2TGT PT2TGT  VALUE    COUNT   POINTING TO TGT  NUMBER    DESCRIPTION


INDEX SEGMENT COUNTS STATISTICS
-------------------------------


     INDEX DATABASE NAME       =  TPFOH2
     INDEXED DATABASE NAME     =  TPFOH2
     INDEX SOURCE SEGMENT NAME =  BROOTLV1
     INDEX TARGET SEGMENT NAME =  BROOTLV1
     SPARSE INDEX              =        NO

     NUMBER OF INDEX POINTER SEGMENT                   =           9000
     NUMBER OF INDEX SOURCE SEGMENT                    =           9000
     NUMBER OF SUPPRESSED SEGMENT (NULLVAL)            =          N / A
     NUMBER OF SUPPRESSED SEGMENT (SEC IDX MAINT EXIT) =          N / A
     NUMBER OF SUPPRESSED SEGMENT (VL SHORT SEGMENT)   =          N / A

INDEX SEGMENT COUNTS STATISTICS
-------------------------------


     INDEX DATABASE NAME       =  TPFOX1
     INDEXED DATABASE NAME     =  TPFOH2
     INDEX SOURCE SEGMENT NAME =  BDEP1LV2
     INDEX TARGET SEGMENT NAME =  BROOTLV1
     SPARSE INDEX              =        NO

     NUMBER OF INDEX POINTER SEGMENT                   =           9178
     NUMBER OF INDEX SOURCE SEGMENT                    =           9178
     NUMBER OF SUPPRESSED SEGMENT (NULLVAL)            =          N / A
     NUMBER OF SUPPRESSED SEGMENT (SEC IDX MAINT EXIT) =          N / A
     NUMBER OF SUPPRESSED SEGMENT (VL SHORT SEGMENT)   =          N / A

FABP0010I #EVALUATION ERRORS DETECTED          =       0

FABP1050I #LP SEGMENTS WITH ZERO CTR FIELD      =    8999


FABP1040I NO ERRORS DETECTED
```

Figure 51. EVALUPRT—Evaluation of All Pointers to the Same Target report (Part 2 of 2)

Legend for Check Process Evaluation report

This report (see Figure 52) contains a brief description of the headings and abbreviations used in the Evaluation of all pointers to the same target report.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS            "LEGEND FOR CHECK PROCESS EVALUATION REPORT"                PAGE:     1
5655-K53                                                  DATE: 07/07/2006  TIME: 15.59.40                         FABPMAIN - V2.R2


EVALUATION OF POINTERS TO SAME TARGET
-------------------------------------

TP ....... RECORD TYPE

TARGET SEGMENT:

DB ....... TARGET DATABASE NUMBER
DG ....... TARGET DATA SET GROUP NUMBER
RBA ...... RBA OF TARGET SEGMENT
SC ....... SEGMENT CODE OF TARGET SEGMENT

SEGM ..... (FLG) FLAG FROM DBD SHOWING POINTERS IN THIS SEG (COMBINATIONS OF POINTERS WILL SHOW DIFFERENT HEX PRINTOUTS)
PTRS       BIT  PTR   PRINTS AS (HEX)        BIT  PTR   PRINTS AS (HEX)
            0 = CTR      80                   4 = LTF      08
            1 = PTF      40                   5 = LTB      04
            2 = PTB      20                   6 = LP       02
            3 = PP       10                   7 = H        01

PHYS ..... (PHY) PHYSICAL POINTERS DETECTED TO THIS SEG
PT2TGT     BIT  PTR   PRINTS AS (HEX)        BIT  PTR   PRINTS AS (HEX)
            0 = HF       80                   4 = PCF      08
            1 = HB       40                   5 = PCL      04
            2 = PTF      20                   6 = RAP      02
            3 = PTB      10                   7 = VLS      01

LOGICAL .. (LOG) LOGICAL POINTERS DETECTED TO THIS SEG
PT2TGT     BIT  PTR   PRINTS AS (HEX)        BIT  PTR   PRINTS AS (HEX)
            0 = LTF      80                   4 = LP       08
            1 = LTB      40                   5 = PP       04
            2 = LCF      20                   6 = IN/SX    02
            3 = LCL      10                   7 = OF/SXO   01

COUNTER VALUE ... (CTR) VALUE IN SEGMENT SHOWING NUMBER OF
           LOGICAL CHILDREN POINTING TO THIS SEGMENT

LCHILD COUNT .... (LC) COUNT OF LOGICAL CHILDREN POINTING TO THIS SEGMENT
           IN A UNI-DIR. OR PHYS. PAIRED LOGICAL RELATIONSHIP

#PHY PAIRED SEG POINTING TO TARGET ... (LP) COUNT OF LP POINTERS
           FROM PHYSICALLY PAIRED SEGMENTS POINTING TO THIS SEGMENT
           (PP) COUNT OF PP POINTERS FROM PHYSICALLY PAIRED SEGMENTS
           TO THIS SEGMENT

ERROR MESSAGES:

NUMBER ... ERROR MESSAGE NUMBER
DESCRIPTION .. SHORT DESCRIPTION OF ERROR
```

Figure 52. EVALUPRT—Legend for Check Process Evaluation report

Check Process Total report

This report (see Figure 53 on page 202) contains the counts of the various work records that were generated during this HD Pointer Checker run.

The report fields are as follows:

T0 The number of valid free space elements detected by the SCAN processor.

T1 The number of valid database segments detected by the SCAN processor.

T2 The number of areas containing slack bytes detected by the SCAN processor. Slack bytes represent an area in the data part of a segment that could not be classified as either segments or free space.

T3 The number of pointers detected and not validated by the SCAN processor.

T4 The number of pointers detected whose target has an unexpected segment code.

T5 The number of pointers detected whose target is missing.

T6-T9
The total of T6, T7, T8, and T9 pointer types.

T6 The pointers detected by the SCAN processor in the VSAM KSDS part of a HIDAM index or secondary index database.

T7 The pointers detected by the SCAN processor in the overflow (VSAM ESDS) part of a HIDAM or secondary index database.

T8 The pointers detected by the SCAN processor in the primary (VSAM KSDS) part of a HISAM database.

T9 The pointers detected by the SCAN processor in the overflow (VSAM ESDS) part of a HISAM database.

TC
The number of index source segment occurrences, the number of index pointer occurrences, the number of logical children, and the sum of counter field value in logical parents that are detected by the SCAN processor.

HASH RECORDS
The number of HASH records created by the HASH Check option.

INPUT COUNT
The number of work records read by the CHECK processor.

PTRS POINTING TO OTHER DATABASES
The number of the pointers pointing to other databases and detected by the SCAN processor.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                "CHECK PROCESS TOTAL REPORT"                   PAGE:     1
5655-K53                                              DATE: 07/07/2006  TIME: 15.59.40                     FABPMAIN - V2.R2


POINTER ERRORS WHERE TARGET IS NOT IN SAME BLOCK    =               0

T0 (FREE SPACE ELEMENT)                             =           25844
T1 (SEGMENT HAVING VALID SEGMENT CODE)              =          228604
T2 (UNKNOWN DATA)                                   =               0
T3 (POINTER WHOSE TARGET WAS NOT VALIDATED IN SCAN) =           72508
T4 (TARGET OF POINTER HAS UNEXPECTED SEGMENT CODE)  =               0
T5 (TARGET OF POINTER IS MISSING)                   =               0
T6-T9 (CUMULATED TOTAL OF POINTER TYPES BELOW)      =           48539
T6 (POINTER IN HIDAM/PHIDAM OR SEC. INDEX)
T7 (POINTER IN HIDAM/PHIDAM OR SEC. INDEX OVERFLOW)
T8 (POINTER IN HISAM KSDS)
T9 (POINTER IN HISAM ESDS)
TC (NUMBER OF ISS, IPS, LC, AND SUM OF CTR FIELD)   =               7
HASH RECORDS                                        =               0

INPUT COUNT                                         =          375495

PTRS POINTING TO OTHER DATABASES                    =               0


*** IF THE COUNT OF PTRS POINTING TO OTHER DB(S) NOT ZERO,
    EITHER ALL RELATED DATABASES WERE NOT SCANNED,

             - OR -

    ALL PTR/TGT RECS WERE NOT CREATED FOR/PASSED TO CHECK PROCESS
```

Figure 53. EVALUPRT—Check Process Total report

HASH Evaluation report

This report (see Figure 54) contains messages issued by the HD Pointer Checker processor (FABPMAIN) through the HASH check. The messages are described in Appendix A, “Messages and codes,” on page 595.

The record fields are as follows:

ERRORS
The report heading for the following kinds of errors:

TOTAL
The total number of errors

SEVERE
The number of errors in physical child/twin pointer chain

PHYSICAL
The number of errors in physical hierarchical path

LOGICAL
The number of errors in logical relationship pointers.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                       "HASH EVALUATION REPORT"                             PAGE:     1
5655-K53                                                   DATE: 07/13/2006  TIME: 11.07.51                                      FABPMAIN - V2.R2

FABP1959E MISMATCH BETWEEN RAPS & PTF POINTERS TO ROOT SEGMENT: ROOT     AND TARGET RBA VALUES
FABP1996E COUNTER VALUE IN LP (SEGMENT: ROOT    ) IS NOT EQUAL TO THE NUMBER OF LOGICAL CHILDREN
FABP1997E MISMATCH BETWEEN LP POINTERS AND LOGICAL PARENT RBA VALUES (DB: HDAMDB2  SEGMENT: ROOT    )
FABP1960E THE NUMBER OF SEGMENTS: DEP1     IS NOT EQUAL TO THE NUMBER OF THE PTF POINTERS
FABP1961E MISMATCH BETWEEN PTF POINTERS IN SEGMENT: DEP1     AND TARGET RBA VALUES
FABP1962E THE NUMBER OF SEGMENTS: DEP1     IS NOT EQUAL TO THE NUMBER OF THE PTB POINTERS
FABP1963E MISMATCH BETWEEN PTB POINTERS IN SEGMENT: DEP1     AND TARGET RBA VALUES
FABP1966E THE NUMBER OF SEGMENTS: DEP1     IS NOT EQUAL TO THE NUMBER OF ITS PTF POINTERS & PCF POINTERS IN THE PARENT: ROOT
FABP1967E MISMATCH BETWEEN PTF POINTERS IN SEGMENT: DEP1     & PCF POINTERS IN PARENT: ROOT     AND TARGET RBA VALUES
FABP1969E MISMATCH BETWEEN PTB POINTERS IN SEGMENT: DEP1     & PCL POINTERS IN PARENT: ROOT     AND TARGET RBA VALUES
FABP1970E MISMATCH BETWEEN PP POINTERS IN THE LAST PHYSICAL CHILD: DEP1     AND RBA VALUES OF PARENT: ROOT     WITH NON-ZERO PCF
FABP1974E THE NO. OF THE LAST SEGMENTS IN TWIN CHAINS: DEP1     IS NOT EQUAL TO THE NO. OF THE PCL POINTERS IN PARENT: ROOT
FABP1975E MISMATCH BETWEEN PCL POINTERS IN PARENT: ROOT     AND RBA VALUES OF THE LAST SEGMENT IN TWIN CHAINS: DEP1
FABP1982E MISMATCH BETWEEN PP POINTERS IN THE LAST PHYSICAL CHILD: DEP1     AND RBA VALUES OF PARENT: ROOT     WITH NON-ZERO PCL
FABP1960E THE NUMBER OF SEGMENTS: DEP2     IS NOT EQUAL TO THE NUMBER OF THE PTF POINTERS
FABP1961E MISMATCH BETWEEN PTF POINTERS IN SEGMENT: DEP2     AND TARGET RBA VALUES
FABP1962E THE NUMBER OF SEGMENTS: DEP2     IS NOT EQUAL TO THE NUMBER OF THE PTB POINTERS
FABP1963E MISMATCH BETWEEN PTB POINTERS IN SEGMENT: DEP2     AND TARGET RBA VALUES
FABP1966E THE NUMBER OF SEGMENTS: DEP2     IS NOT EQUAL TO THE NUMBER OF ITS PTF POINTERS & PCF POINTERS IN THE PARENT: DEP1
FABP1967E MISMATCH BETWEEN PTF POINTERS IN SEGMENT: DEP2     & PCF POINTERS IN PARENT: DEP1     AND TARGET RBA VALUES
FABP1974E THE NO. OF THE LAST SEGMENTS IN TWIN CHAINS: DEP2     IS NOT EQUAL TO THE NO. OF THE PCL POINTERS IN PARENT: DEP1
FABP1975E MISMATCH BETWEEN PCL POINTERS IN PARENT: DEP1     AND RBA VALUES OF THE LAST SEGMENT IN TWIN CHAINS: DEP2
FABP2001I EVAL OF DB: HDAMDB2  DB#: 001            DSG#: 01 COMPLETED  ERRORS:     22 TOTAL (      5 SEV.     15 PHY.      2 LOG.)

FABP2001I EVAL OF DB: HISAMDB1 DB#: 002            DSG#: 01 COMPLETED  ERRORS:      0 TOTAL (      0 SEV.      0 PHY.      0 LOG.)

FABP2002I                                               RUN COMPLETED  ERRORS:     22 TOTAL

FABP2008E ERRORS WERE DETECTED ON RAP, PHYSICAL CHILD/TWIN POINTER, OR HIERARCHICAL POINTER CHAIN

FABP2004E ERRORS WERE DETECTED IN LOGICAL RELATIONSHIP POINTERS

FABP2005E ERRORS WERE DETECTED ON PHYSICAL CHILD/TWIN POINTER OR HIERARCHICAL POINTER CHAIN
```

Figure 54. EVALUPRT—HASH Evaluation report

Evaluation of Index Pointers and Keys report

This report (see Figure 55 on page 206) contains the errors for index pointers and keys detected through the Index Key Check process. It is produced when IXKEYCHK=YES and HASH=NO are specified in the PROC statement, and all index databases to be checked are specified in the DATABASE statements.

The following errors can be detected and reported with the messages in this report, when the index database is either of the following:

- A primary index database of HIDAM.
- A secondary index database using a direct pointing, and the segment type of an index source segment (ISS) is same as the one of index target segments (ITS).
- Missing index pointer

  If there is no index pointer segment whose pointer value is equal to the RBA of the index target segment, this segment is missing an index pointer.

  This segment is reported by the message FABP2020E with its key value.
- Invalid index key

  If the pointer value of the index pointer segment is equal to the RBA of the index target segment, and the key value of the index pointer segment is not equal to the key value of source segment or the root key value of HIDAM root segment, this index pointer segment contains an invalid index key.

  This segment and index pointer segment are reported by the message FABP2021E with its key value.
- Invalid index pointer

  If there is no index target segment pointed by the index pointer segment, this index pointer segment contains an invalid index pointer.

  This segment is reported by the message FABP2022E with its key value.
- Invalid index key length

  If the length of the index key from DBD (SRCH field and SUBSEQ field) is unmatched with the length of index key in T6, T7, or TA record, this record is reported by the message FABP2025E, and index key checking is skipped for the remaining sort records within the same database data set group with the message FABP2027E.

The following errors can be detected and reported with the messages in this report, when the index database is either of the following:

- A secondary index database that uses a symbolic pointing.
- A secondary index database that uses a direct pointing, and the segment type of an ISS is not the same as the one of ITS.
- Invalid index key, Missing index pointer, or Invalid index pointer

  If the number of index source segments is not equal to one of index pointer segments which contain the same key value, there is a missing index pointer, an invalid index key, or an invalid index pointer in the database.

  These segments which contain the same index key are reported by the message FABP2023E or FABP2024E.
- Invalid index key length

  If the length of the index key from DBD (SRCH filed and SUBSEQ field) is unmatched with the length of index key in T6, T7, or TA record, this record is reported by the message FABP2025E, and index key checking is skipped for the remaining sort records within the same database data set group with the message FABP2027E.

The report fields are as follows:

DBNAME DB# PARTNAME PART ID REORG# DSG# DSNAME
The name of the DBD, the database number (in hexadecimal), the name of the partition, partition id, partition reorg number, the data set group number (in hexadecimal) or the data set group ID (in an alphabetical character) and the name of the data set name.

ERROR MESSAGE
Error message detected through the Index Key Check process. If no message is placed in this field, this line is the part of the preceding message.

TP
The type of record that is written on work data sets (T1, T6, T7, or TA is reported).

KL
The index key length.

TARGET
The target of an index pointer. The following five fields all pertain to it:

DB
The database number (in hexadecimal) that identifies the database containing the target of an index pointer.

PID
The partition ID (in decimal) that identifies the partition containing the segment that contains the pointer

DG
The data set group number (in hexadecimal) or the data set group ID (in an alphabetical character) that identifies the database containing the target of a pointer.

SC
The segment code (in hexadecimal) of the target of an index pointer.

XD
First 1 byte is the database number (in hexadecimal) of index database. The last 1 byte is a constant value specified by CONST= parameter on XDFLD statement of DBDGEN. If CONST= parameter is not specified, the last 1 byte is not reported.

RBA
The relative byte address (in hexadecimal) of the target of an index pointer.

SOURCE
The segment that contains index key. The following four fields all pertain to it:

DB
The database number (in hexadecimal) that identifies the database containing the target of an index pointer

PID
The partition ID (in decimal) that identifies the partition containing the segment that contains the pointer

DG
The data set group number (in hexadecimal) or the data set group ID (in an alphabetical character) that identifies the database containing the target of a pointer.

RBA

The relative byte address (in hexadecimal) of the segment that contains index key

SC

The segment code (in hexadecimal) of the target of an index pointer

DBD KL

The index key field length specified by DBD

KEY VALUE (HEX)

The key value (in hexadecimal) extracted from database

SRC=

The key value extracted from index target database. It is contained in type A (TA) record.

IDX=

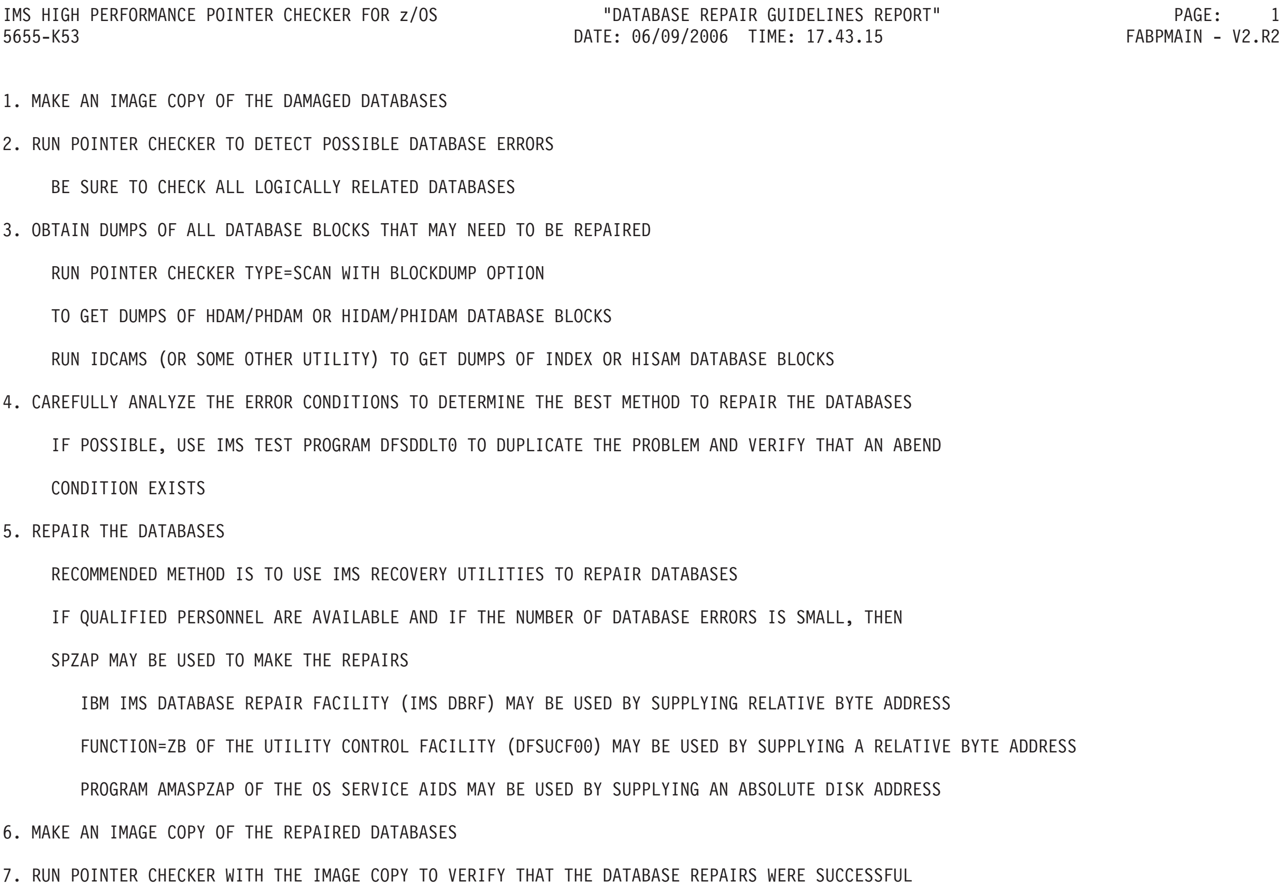
The key value extracted from index database. It is contained in type 6 or 7 (T6 or T7) record.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS            "EVALUATION OF INDEX POINTERS AND KEYS REPORT"                  PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                         FABPMAIN - V2.R2


DBNAME: TPFOH2    DB#: 005 PARTNAME: TPFOH2A  PART ID: 00001 REORG#: 00001 DSG#:  A
DSNAME: TESTDS.PUBLIC.SAMPLE.TPFOH2.A00001           DBORG: PHIDAM
-----------------------------------------------------------------------------------

< ERROR > <> <> <------ TARGET ------> <------- SOURCE -------> DBD<----------------------- KEY VALUE (HEX) ----------------------->
 MESSAGE  TP KL DB  DG SC XD  RBA      DB  PID   DG RBA      SC KL   SRC= : KEY FROM SOURCE SEGMENT,  IDX= : KEY FROM INDEX SEGMENT


TOTALS
------
MESSAGE   DESCRIPTION                                                     NUMBER OF ERRORS
--------- --------------------------------------------------------------  ----------------


FABP1040I NO ERRORS DETECTED
```

Figure 55. EVALUPRT—Evaluation of Index Pointers and Keys report

Database Repair Guidelines report

This report (see Figure 56) contains a brief summary of suggested steps to take when you have to repair a broken database.

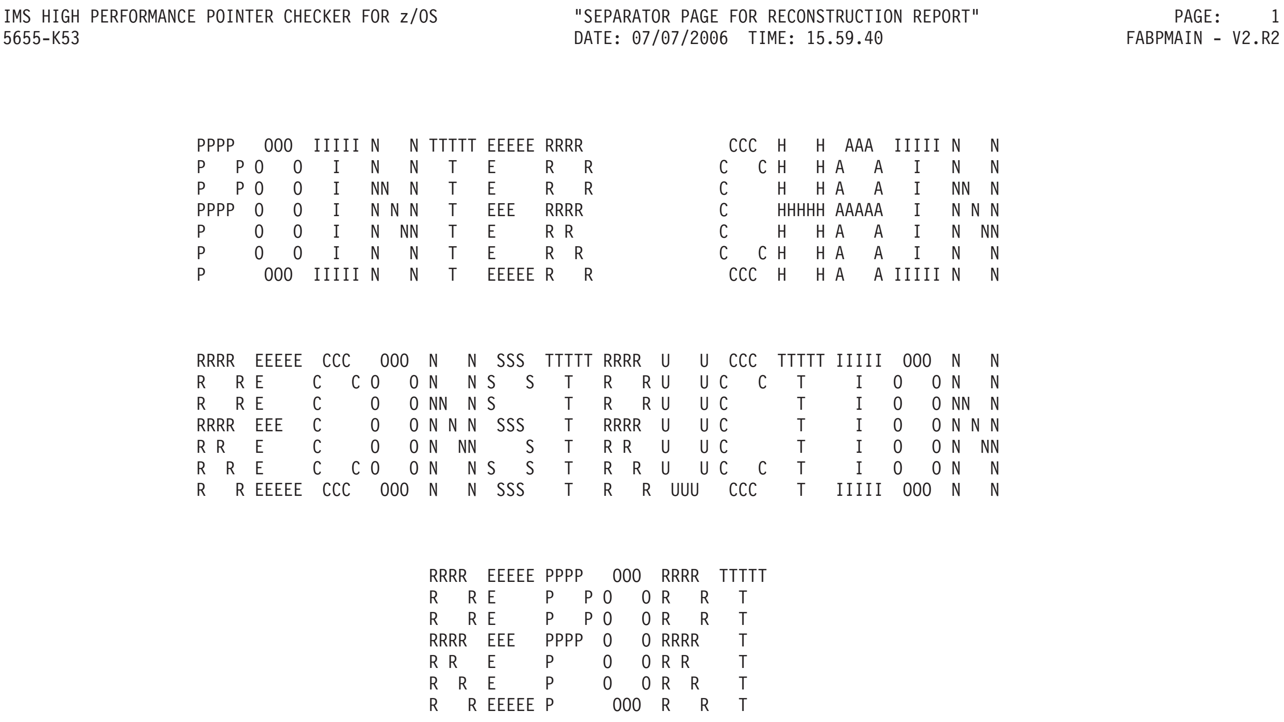


```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS              "DATABASE REPAIR GUIDELINES REPORT"                     PAGE:     1
5655-K53                                                DATE: 06/09/2006  TIME: 17.43.15                      FABPMAIN - V2.R2

1. MAKE AN IMAGE COPY OF THE DAMAGED DATABASES

2. RUN POINTER CHECKER TO DETECT POSSIBLE DATABASE ERRORS

     BE SURE TO CHECK ALL LOGICALLY RELATED DATABASES

3. OBTAIN DUMPS OF ALL DATABASE BLOCKS THAT MAY NEED TO BE REPAIRED

     RUN POINTER CHECKER TYPE=SCAN WITH BLOCKDUMP OPTION

     TO GET DUMPS OF HDAM/PHDAM OR HIDAM/PHIDAM DATABASE BLOCKS

     RUN IDCAMS (OR SOME OTHER UTILITY) TO GET DUMPS OF INDEX OR HISAM DATABASE BLOCKS

4. CAREFULLY ANALYZE THE ERROR CONDITIONS TO DETERMINE THE BEST METHOD TO REPAIR THE DATABASES

     IF POSSIBLE, USE IMS TEST PROGRAM DFSDDLT0 TO DUPLICATE THE PROBLEM AND VERIFY THAT AN ABEND

     CONDITION EXISTS

5. REPAIR THE DATABASES

     RECOMMENDED METHOD IS TO USE IMS RECOVERY UTILITIES TO REPAIR DATABASES

     IF QUALIFIED PERSONNEL ARE AVAILABLE AND IF THE NUMBER OF DATABASE ERRORS IS SMALL, THEN

     SPZAP MAY BE USED TO MAKE THE REPAIRS

        IBM IMS DATABASE REPAIR FACILITY (IMS DBRF) MAY BE USED BY SUPPLYING RELATIVE BYTE ADDRESS

        FUNCTION=ZB OF THE UTILITY CONTROL FACILITY (DFSUCF00) MAY BE USED BY SUPPLYING A RELATIVE BYTE ADDRESS

        PROGRAM AMASPZAP OF THE OS SERVICE AIDS MAY BE USED BY SUPPLYING AN ABSOLUTE DISK ADDRESS

6. MAKE AN IMAGE COPY OF THE REPAIRED DATABASES

7. RUN POINTER CHECKER WITH THE IMAGE COPY TO VERIFY THAT THE DATABASE REPAIRS WERE SUCCESSFUL
```

Figure 56. EVALUPRT—Database Repair Guidelines report

Separator page for reconstruction reports

This separator page (see Figure 57) contains the title of “Reconstruction Report,” and indicates reports for reconstruction will follow the page. It is produced unless SEP=NO is specified on the PROC statement as input for the PROCCTL data set.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS              "SEPARATOR PAGE FOR RECONSTRUCTION REPORT"                    PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                          FABPMAIN - V2.R2



                    PPPP   OOO  IIIII N   N TTTTT EEEEE RRRR               CCC  H   H  AAA  IIIII N   N
                    P   P O   O   I   N   N   T   E     R   R             C   C H   H A   A   I   N   N
                    P   P O   O   I   NN  N   T   E     R   R             C     H   H A   A   I   NN  N
                    PPPP  O   O   I   N N N   T   EEE   RRRR              C     HHHHH AAAAA   I   N N N
                    P     O   O   I   N  NN   T   E     R R               C     H   H A   A   I   N  NN
                    P     O   O   I   N   N   T   E     R  R              C   C H   H A   A   I   N   N
                    P      OOO  IIIII N   N   T   EEEEE R   R              CCC  H   H A   A IIIII N   N


                    RRRR  EEEEE  CCC   OOO  N   N  SSS  TTTTT RRRR  U   U  CCC  TTTTT IIIII  OOO  N   N
                    R   R E     C   C O   O N   N S   S   T   R   R U   U C   C   T     I   O   O N   N
                    R   R E     C     O   O NN  N S       T   R   R U   U C       T     I   O   O NN  N
                    RRRR  EEE   C     O   O N N N  SSS    T   RRRR  U   U C       T     I   O   O N N N
                    R R   E     C     O   O N  NN     S   T   R R   U   U C       T     I   O   O N  NN
                    R  R  E     C   C O   O N   N S   S   T   R  R  U   U C   C   T     I   O   O N   N
                    R   R EEEEE  CCC   OOO  N   N  SSS    T   R   R  UUU   CCC    T   IIIII  OOO  N   N


                                            RRRR  EEEEE PPPP   OOO  RRRR  TTTTT
                                            R   R E     P   P O   O R   R   T
                                            R   R E     P   P O   O R   R   T
                                            RRRR  EEE   PPPP  O   O RRRR    T
                                            R R   E     P     O   O R R     T
                                            R  R  E     P     O   O R  R    T
                                            R   R EEEEE P      OOO  R   R   T
```

Figure 57. EVALUPRT—Separator page for reconstruction reports

DMB Directory and Control Card Format report

This report (see Figure 58 on page 210) contains a list of all the databases that were defined (either explicitly or implicitly) by your PSB. It is only produced when TYPE=BLKMAP is specified on the PROC statement.

The report fields are as follows:

DB NAME
The name of a DBD load module

DB NUMBER
The database number (in hexadecimal) used to identify the database throughout the HD Pointer Checker run

PARTITION ID
The partition ID (in decimal) that identifies the partition

DSG NUMBER
The data set group number (in hexadecimal) that identifies the database data set group throughout the HD Pointer Checker run.

DBLG NUMBER
The database logical group number (in hexadecimal) that identifies the database logical group throughout the HD Pointer Checker run.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS             "DMB DIRECTORY / CONTROL CARD FORMAT REPORT"                 PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 16.10.58                       FABPMAIN - V2.R2


DB NAME            DB NUMBER         PARTITION ID     DSG NUMBER       DBLG NUMBER

HDAMDB2               001                                01                001
HISAMDB1              002                                01                001


CONTROL CARD FORMAT
-------------------

COLUMNS  1-18: XXXNNNNNYYZZZZZZZZ

               XXX = DATABASE NUMBER
             NNNNN = PARTITION ID
                      IF NON-HALDB, FILL WITH BLANK
                      IF HALDB, DECIMAL DIGITS
                YY = DATA SET GROUP NUMBER
                      IF NON-HALDB, HEXADECIMAL DIGITS
                      IF HALDB, 'A-J' FOLLOWING ONE BLANK
          ZZZZZZZZ = RBN IN PRINTABLE HEX OF
                     A SEGMENT FOR WHICH ALL
                     POINTERS TO THE SAME SEGMENT
                     ARE TO BE PRINTED


COMMENTS TO FOLLOWING PRINT OUTPUT
----------------------------------

WITH CONTROL CARD   : PRINTS ALL POINTERS TO
                      SPECIFIED RBA

WITHOUT CONTROL CARD: PRINTS ALL POINTERS TO A
                      POINTER WHOSE TGT WAS BAD

               ***  W A R N I N G  ***

      IF IN-CORE POINTER CHECKING WAS DONE, ALL
      POINTERS VALIDATED IN MEMORY HAVE BEEN
      DISCARDED.

      IF HASH CHECKING WAS DONE, NO POINTER
      INFORMATION HAS BEEN CREATED.

      TO SEE ANY POINTER, RERUN POINTER CHECKER
      WITH NEITHER THE HASH NOR THE IN-CORE
      POINTER VALIDATION.
```

Figure 58. EVALUPRT—DMB Directory and Control Card Format report

Pointer Chain Reconstruction report

This report (see Figure 59 on page 212) contains the following kinds of information:

- A list of all pointers that point to a segment containing an invalid pointer.
- When the BLOCKMAP processor runs as a separate job (or a job step) using the BLKMAPIN data set and the CHECKREC data set as input, this report contains a list of the pointers that point to each target RBA specified in the BLKMAPIN data set. The CHECKREC data set created with 'INCORE=NO' at SCAN process is required as the input to list all the pointers.

The report fields are as follows:

TP
The type of record. The HD Pointer Checker classifies its work records into types (T1, T2, and so on).

DUMP NO.
The relative block number of the block or CI that contains the invalid pointer or error. It is the same number as the dump number in the Block Map/ Block Dump reports.

TARGET
The target of a pointer. The following four fields all pertain to it:

DB
The database number (in hexadecimal) that identifies the database containing the target of a pointer.

PID
The partition ID (in decimal) that identifies the partition containing the target of a pointer.

DG
The data set group number (in hexadecimal) or the data set group ID (in an alphabetical character) that identifies the database containing the target of a pointer.

RBA
The relative byte address (in hexadecimal) of the target of a pointer. This is the actual value of the pointer itself.

SC
The segment code (in hexadecimal) of the target of a pointer.

SOURCE
The segment that contains the pointer (also called the source of the pointer). The following six fields all pertain to it:

DB
The database number (in hexadecimal) that identifies the database containing the segment that contains the pointer

PID
The partition ID (in decimal) that identifies the partition containing the target of a pointer.

DG
The data set group number (in hexadecimal) or the data set group ID (in an alphabetical character) that identifies the database containing the target of a pointer.

RBA

The relative byte address (in hexadecimal) of the segment that contains the pointer.

SC

The segment code (in hexadecimal) of the segment that contains the pointer.

PTR

The type of pointer.

RRN

The relative-record number (for an index or overflow pointer).

MESSAGES

The following two fields pertain to error messages:

NUMBER

The message number. You can find information about each message in Appendix A, “Messages and codes,” on page 595.

DESCRIPTION

The message text.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                    "POINTER CHAIN RECONSTRUCTION"                          PAGE:     1
5655-K53                                                   DATE: 07/13/2006  TIME: 10.59.50                         FABPMAIN - V2.R2


   DUMP <------- TARGET -------> <------------- SOURCE -------------->
TP NO   DB  PID   DG RBA      SC DB  PID   DG RBA      SC PTR RRN
                                                         <------------------------------- MESSAGES -------------------------------->
                                                         NUMBER    DESCRIPTION


   0003 001       01 00002D76                            FABP1320I INPUT RECORD FROM JRM FILE
T1      001       01 00002D76 03                         FABP1330I ADDRESS FOUND IN WORK DATA SET
T3      001       01 00002D76 03 001       01 001DDF36 03 PTB
                                                         FABP1340I SEGMENT POINTS TO ABOVE INPUT ADDRESS

                                                         FABP1280I NO MORE RECORDS FOR SPECIFIED RBA

   0004 001       01 00003408                            FABP1320I INPUT RECORD FROM JRM FILE
T1      001       01 00003408 01                         FABP1330I ADDRESS FOUND IN WORK DATA SET
T3      001       01 00003408 01 001       01 000500EC 02  PP
                                                         FABP1340I SEGMENT POINTS TO ABOVE INPUT ADDRESS
T3      001       01 00003408 01 001       01 00050166 02  PP
                                                         FABP1340I SEGMENT POINTS TO ABOVE INPUT ADDRESS

------ For formatting purposes, several lines have been deleted.------

   0013 001       01 00079072                            FABP1320I INPUT RECORD FROM JRM FILE
T1      001       01 00079072 02                         FABP1330I ADDRESS FOUND IN WORK DATA SET

                                                         FABP1280I NO MORE RECORDS FOR SPECIFIED RBA

   0014 001       01 00245ABC                            FABP1320I INPUT RECORD FROM JRM FILE
T1      001       01 00245ABC 03                         FABP1330I ADDRESS FOUND IN WORK DATA SET

                                                         FABP1280I NO MORE RECORDS FOR SPECIFIED RBA

FABP1290I END OF FILE ON CONTROL DATA SET
```

Figure 59. EVALUPRT—Pointer Chain Reconstruction report

Legend for Reconstruction report

This report (see Figure 60) contains a brief description of the headings and abbreviations used in the Pointer Chain Reconstruction report.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                  "LEGEND FOR RECONSTRUCTION REPORT"                        PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                         FABPMAIN - V2.R2


TP ...... RECORD TYPE (T1, T2, ETC.)
DUMP NO.. DUMP NUMBER IN BLOCK MAP/BLOCK DUMP REPORT

TARGET OF POINTER:

DB ...... DATABASE NUMBER OF TARGET SEGMENT
PID...... PARTITION ID OF TARGET SEGMENT
DG ...... DATA SET GROUP OF TARGET SEGMENT
RBA ..... RELATIVE BYTE ADDRESS OF TARGET SEGMENT
SC ...... SEGMENT CODE OF TARGET SEGMENT

SOURCE OF POINTER:

DB ...... DATABASE NUMBER OF SEGMENT CONTAINING POINTER
PID...... PARTITION ID OF SEGMENT CONTAINING POINTER
DG ...... DATA SET GROUP OF SEGMENT CONTAINING POINTER
RBA ..... RBN OF SEGMENT CONTAINING POINTER
SC ...... SEGMENT CODE OF SEGMENT CONTAINING POINTER
PTR ..... POINTER TYPE
RRN ..... RELATIVE RECORD NUMBER OF INDEX OR OVERFLOW POINTER

MESSAGES:

NUMBER .. MESSAGE NUMBER
DESCRIPTION .. SHORT DESCRIPTION OF MESSAGE
```

Figure 60. EVALUPRT—Legend for Reconstruction report

EVALUPR2 data set

This section explains the EVALUPR2 data set.

Function

The EVALUPRT2 data set contains the Evaluation of Symbolic Pointers produced by the HD Pointer Checker processor (FABPMAIN).

Evaluation of Symbolic Pointers report

The report shown in Figure 61 on page 215 contains the evaluation result of symbolic pointer and its target segments. It is produced when SYMLPCHK=YES and/or SYMIXCHK=YES is specified on the PROC statement and when the databases have symbolic pointers to be checked.

DBNAME DB# DDNAME DSG# DSNAME DBORG
The name of the DBD, the database number (in hexadecimal), the data set group number (in hexadecimal), the name of data set, and the database organization.

ERROR MESSAGE
The error message that is generated and detected during the symbolic pointer evaluation process. If no message number is printed in the field, this line is part of the preceding message.

TP
The type of record that is printed on this line. The HD Pointer Checker classifies its work records into types (T1, T3, and so on).

TARGET
This is information about the pointer target of the symbolic pointer. It consists of four items:

DB
The database number (in hexadecimal) that identifies the database that contains the target segment.

DG
The data set group number (in hexadecimal) that identifies the database that contains the target segment.

SC
The segment code (in hexadecimal) of the target segment.

SEGNAME
The segment name of the target segment.

SOURCE
The segment that contains the symbolic pointer (also called the source of the symbolic pointer). It consists of four items:

DB
The database number (in hexadecimal) that identifies the database that contains the source segment.

DG
The data set group number (in hexadecimal) that identifies the database that contains the source segment.

SC
The segment code (in hexadecimal) of the source segment.

SEGNAME
The segment name of the source segment.

PTR TYPE
The type of pointer.

SYMBOLIC POINTER VALUE
The symbolic pointer value (in hexadecimal or character).

TOTALS
The error or information message issued during the symbolic pointer evaluation process. The number of messages issued is also shown.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS               "EVALUATION OF SYMBOLIC POINTERS REPORT"                     PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                         FABPMAIN - V2.R2


DBNAME: HDAMDB2   DB#: 001  DDNAME: HDAMDS4   DSG#: 01  DSNAME: TESTDS.PUBLIC.SAMPLE.HDAMDS4                  DBORG: HDAM

ERROR        <---- TARGET ----> <---- SOURCE ----> PTR
MESSAGE   TP DB  DG SC SEGNAME  DB  DG SC SEGNAME  TYP SYMBOLIC POINTER VALUE
--------- -- ------------------ ------------------ --- ----------------------------------------------------------------

TOTALS
------
MESSAGE   DESCRIPTION                                                     NUMBER OF ERRORS
--------- --------------------------------------------------------------  ----------------
FABP1040I NO ERRORS DETECTED
```

Figure 61. EVALUPR2—Evaluation of Symbolic Pointers report

EVALIPRT data set

This section explains the EVALIPRT data set.

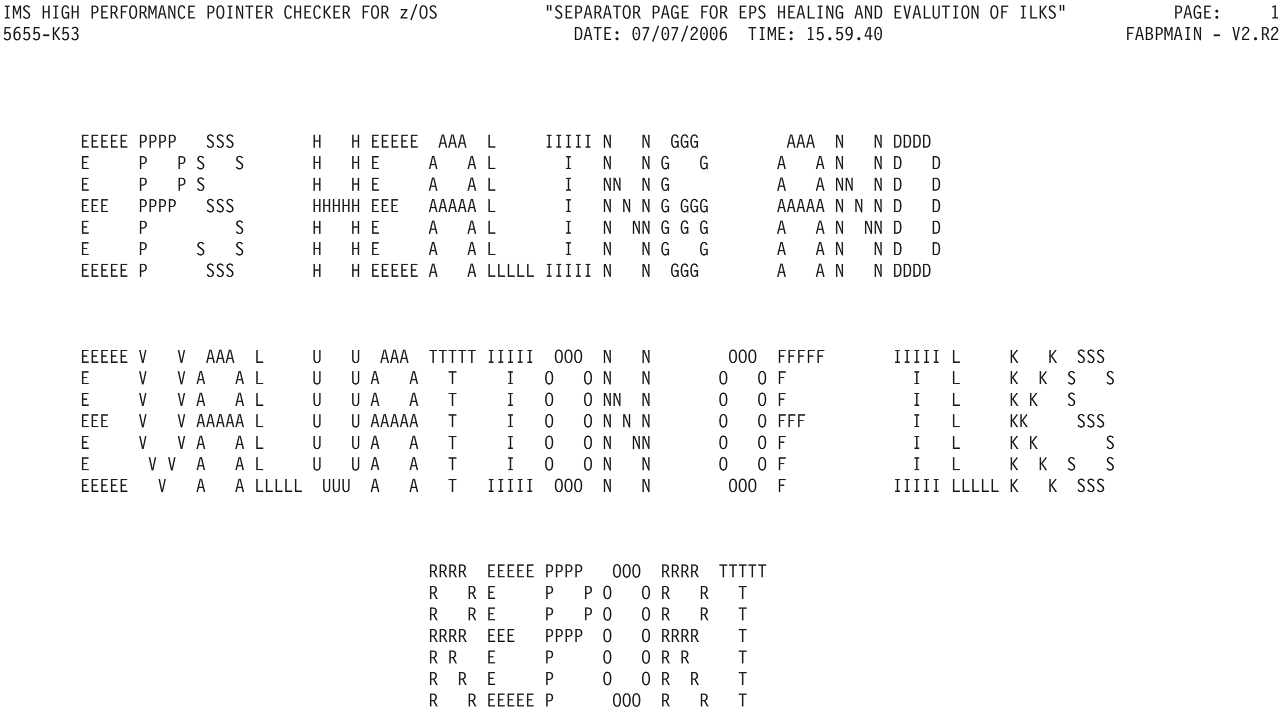
Function

The EVALIPRT data set contains the following reports produced by the HD Pointer Checker processor (FABPMAIN):

- Separator page for EPS Healing reports and Evaluation of ILKS reports
- EPS Healing report
- Evaluation of ILKS report

Separator page for EPS Healing reports and Evaluation of ILKs reports

This separator page (see Figure 62) contains the title of “Separator page for EPS Healing reports and Evaluation of ILKs reports” and indicates reports for EPS Healing and Evaluation of ILKs will follow the page. It is produced unless SEP=NO is specified on the PROC statement as input for the PROCCTL data set.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS           "SEPARATOR PAGE FOR EPS HEALING AND EVALUTION OF ILKS"          PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                         FABPMAIN - V2.R2



        EEEEE PPPP   SSS        H   H EEEEE  AAA  L     IIIII N   N  GGG         AAA  N   N DDDD
        E     P   P S   S       H   H E     A   A L       I   N   N G   G       A   A N   N D   D
        E     P   P S           H   H E     A   A L       I   NN  N G           A   A NN  N D   D
        EEE   PPPP   SSS        HHHHH EEE   AAAAA L       I   N N N G GGG       AAAAA N N N D   D
        E     P         S       H   H E     A   A L       I   N  NN G G G       A   A N  NN D   D
        E     P     S   S       H   H E     A   A L       I   N   N G   G       A   A N   N D   D
        EEEEE P      SSS        H   H EEEEE A   A LLLLL IIIII N   N  GGG        A   A N   N DDDD


        EEEEE V   V  AAA  L     U   U  AAA  TTTTT IIIII  OOO  N   N        OOO  FFFFF       IIIII L     K   K  SSS
        E     V   V A   A L     U   U A   A   T     I   O   O N   N       O   O F             I   L     K  K  S   S
        E     V   V A   A L     U   U A   A   T     I   O   O NN  N       O   O F             I   L     K K   S
        EEE   V   V AAAAA L     U   U AAAAA   T     I   O   O N N N       O   O FFF           I   L     KK     SSS
        E     V   V A   A L     U   U A   A   T     I   O   O N  NN       O   O F             I   L     K K        S
        E      V V  A   A L     U   U A   A   T     I   O   O N   N       O   O F             I   L     K  K  S   S
        EEEEE   V   A   A LLLLL  UUU  A   A   T   IIIII  OOO  N   N        OOO  F           IIIII LLLLL K   K  SSS


                                            RRRR  EEEEE PPPP   OOO  RRRR  TTTTT
                                            R   R E     P   P O   O R   R   T
                                            R   R E     P   P O   O R   R   T
                                            RRRR  EEE   PPPP  O   O RRRR    T
                                            R R   E     P     O   O R R     T
                                            R  R  E     P     O   O R  R    T
                                            R   R EEEEE P      OOO  R   R   T
```

Figure 62. EVALIPRT—Separator page for EPS Healing reports and Evaluation of ILKS reports

EPS Healing report

This report (see Figure 63 on page 219) contains the errors for ILEs against each EPS through the EPS healing process. It is produced when EPSCHK=YES is specified on the PROC statement and the database having EPS are checked.

If there is no ILE whose ILK is equal to the ILK embedded in the EPS, the ILE is missing. The EPS whose ILK does not exist in ILDS is reported by message FABP2100E.

DBNAME, DB#, PART NAME, PART ID, REORG#
The name of the DBD, the database number (in hexadecimal), the name of the partition, partition ID, and partition reorg number.

ERROR MESSAGE
Error messages detected through the EPS healing process.

TP
The type of record that is written on work data sets (TH or TJ is reported).

TARGET
This is information about the pointer target of EPS. It consist of four items:

DB
The database number (in hexadecimal) that identifies the database containing the target of EPS.

DG
The data set group ID (in an alphabetical character) that identifies the database containing the target of EPS.

SC
The segment code (in hexadecimal) of the target of EPS.

RBA
The relative byte address (in hexadecimal) of the target of EPS.

SOURCE
The segment that contains the EPS (also called the source of the EPS). It consist of five items:

DB
The database number (in hexadecimal) that identifies the database containing the segment that contains the EPS.

PID
The partition ID (in decimal) that identifies the partition containing the segment that contains the EPS.

DG
The data set group ID (in an alphabetical character) that identifies the database containing the segment that contains the EPS.

RBA
The relative byte address (in hexadecimal) of the segment that contains the EPS.

SC
The segment code (in hexadecimal) of the segment that contains the EPS.

PTR
The type of pointer that resides in the EPS.

ILK VALUE(HEX)
The ILK of that target segment that resides in the EPS of source segment is printed.

TOTALS
The error message issued during the EPS healing process. The number of the messages is also given.

INPUT RECORDS SUMMARY
This is the statistics information about input records of EPS Healing process of this target partition. The input record is one of the following pointers:

- The LP pointer or the paired LC pointer that points to this partition
- The pointer in secondary indexes (PSINDEXs) that points to this partition

These pointers may refer to the ILK (indirect list key) on the ILDS data set.

RECORD TYPE
This is the record type of the input records.

DIRECT
This is the number of pointer records each of which has the latest RBA of its target segment. HD Pointer Checker does not refer to the Indirect List Data Set (ILDS).

INDIRECT
This is the number of pointer records that do not have the latest RBAs. To get the latest RBAs HD Pointer Checker refers to the Indirect List Data Set (ILDS).

TOTAL
This is the total number of input pointer records.

OUTPUT RECORDS SUMMARY
This is the statistics information about output records for the EPS Healing Process of this target partition.

RECORD TYPE
The record type of the output records.

RECORDS
This is the number of output pointer records.

ILDS ACCESS
This is the number of Indirect List Data Set records referred to in the EPS Healing Process of this target partition.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                         "EPS HEALING REPORT"                               PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                         FABPMAIN - V2.R2

DBNAME: TPFOH1    DB#: 003 PARTNAME: TPFOH1A  PART ID: 00001 REORG#: 00001
---------------------------------------------------------------------------

< ERROR > <> <---- TARGET ----> <------- SOURCE -------> <PTR> <- ILK VALUE(HEX) ->
 MESSAGE  TP DB  DG SC RBA      DB  PID   DG RBA      SC
FABP1040I NO ERRORS DETECTED

INPUT RECORDS SUMMARY
---------------------
  RECORD TYPE                    DIRECT        INDIRECT           TOTAL
  ---------------------- --------------- --------------- ---------------
  TH (LP/PAIRED LC PTR)           7,332               0           7,332
  TJ (PSINDEX POINTER)                0               0               0
  ---------------------- --------------- --------------- ---------------
  TOTAL                           7,332               0           7,332

  NOTE : DIRECT:   POINTER RECORD DIRECTLY POINTS TO ITS TARGET.
  ----   INDIRECT: POINTER RECORD POINTS TO ITS TARGET WITH REFERENCE TO AN ILDS(INDIRECT LIST DATA SET).

OUTPUT RECORDS SUMMARY
----------------------
  RECORD TYPE             NO. OF RECORDS
  ---------------------- ---------------
  T3 (LP POINTER)                  7,112
  T3 (PAIRED LC POINTER)             220
  T6 (PSINDEX POINTER)                 0
  ---------------------- ---------------
  TOTAL                            7,332

ILDS ACCESS
-----------
  READ RECORDS         =               0
```

Figure 63. EVALIPRT—EPS Healing report

Evaluation of ILKS report

This report (see Figure 64 on page 221) contains the errors for ILKs that reside in the EPS and its target segment. It is produced when EPSCHK=YES is specified on the PROC statement and the databases having EPS are checked.

If the ILK in the EPS is different from the ILK of its target segment, message FABP2101E is generated.

DBNAME DB# PARTITION NAME PARTITION ID DSG#
The name of the DBD, the database number (in hexadecimal), the name of the partition, the partition ID, and the data set group ID (in an alphabetical character).

ERROR MESSAGE
The error message is generated and detected during the ILK evaluation process. If no message number is printed in the field, this line is a part of the preceding message.

TP
The type of record that is printed on this line. The HD Pointer Checker classifies its work records into types (T1, T3 and so on).

TARGET
This is information about the pointer target of EPS. It consist of four items:

DB
The database number (in hexadecimal) that identifies the database containing the target of EPS.

DG
The data set group ID (in an alphabetical character) that identifies the database containing the target of EPS.

SC
The segment code (in hexadecimal) of the target of EPS.

RBA
The relative byte address (in hexadecimal) of the segment that contains the EPS.

SOURCE
The segment that contains the EPS (also called the source of the EPS). It consist of five items:

DB
The database number (in hexadecimal) that identifies the database containing the segment that contains the EPS.

PID
The partition ID (in decimal) that identifies the partition containing the segment that contains the EPS.

DG
The data set group ID (in an alphabetical character) that identifies the database containing the segment that contains the EPS.

RBA
The relative byte address (in hexadecimal) of the segment that contains the EPS.

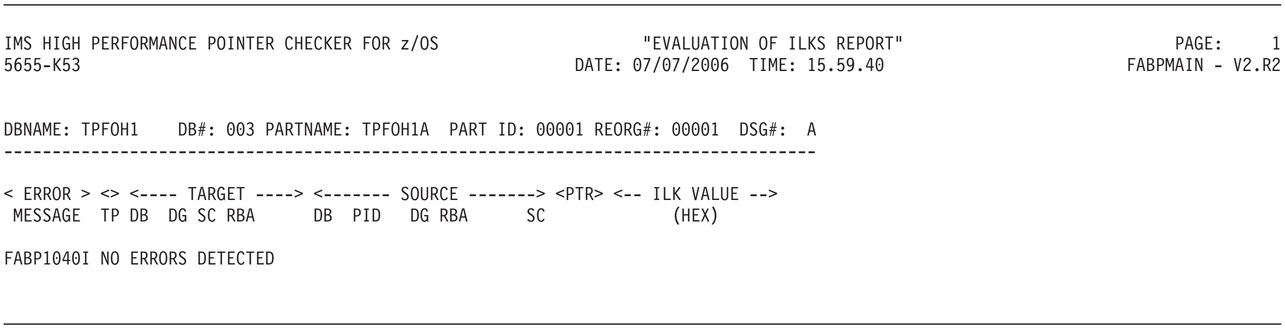
SC
The segment code (in hexadecimal) of the segment that contains the EPS.

PTR
The type of pointer that resides in the EPS.

ILK VALUE(HEX)
ILK value (hex).

ILK=
ILK in the prefix of the target segment or in the EPS of the source segment.

TOTALS
The error or information message issued during the ILK evaluation process. The number of messages is also given.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                     "EVALUATION OF ILKS REPORT"                             PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                         FABPMAIN - V2.R2

DBNAME: TPFOH1    DB#: 003 PARTNAME: TPFOH1A  PART ID: 00001 REORG#: 00001  DSG#:  A
------------------------------------------------------------------------------------

< ERROR > <> <---- TARGET ----> <------- SOURCE -------> <PTR> <-- ILK VALUE -->
 MESSAGE  TP DB  DG SC RBA      DB  PID   DG RBA      SC             (HEX)

FABP1040I NO ERRORS DETECTED
```

Figure 64. EVALIPRT—Evaluation of ILKS report

SNAPPIT data set

This section explains the SNAPPIT data set.

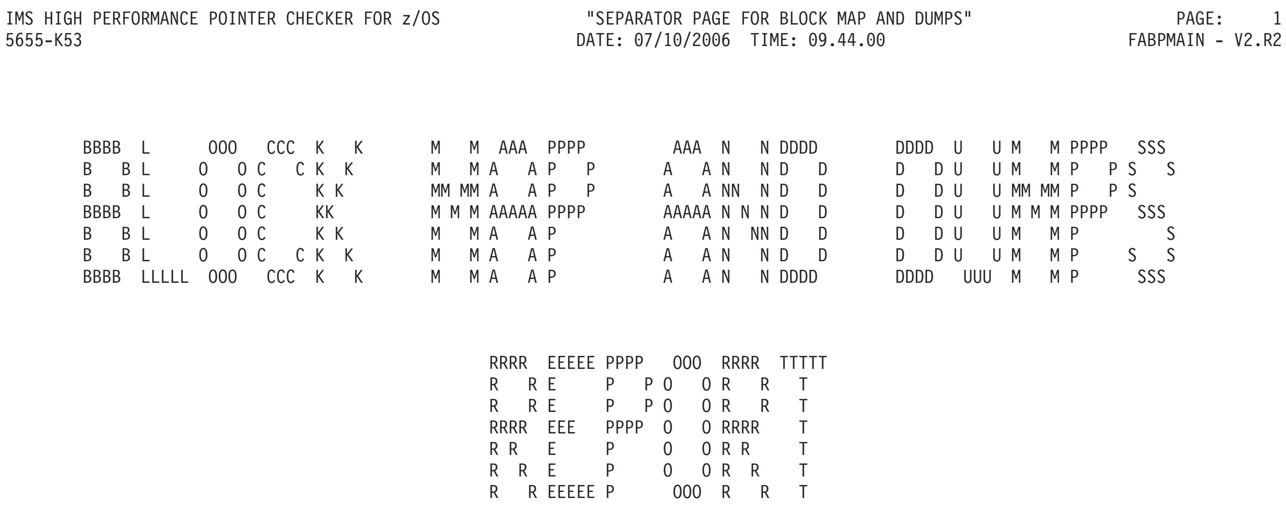
Function

The SNAPPIT data set contains the following reports produced by the HD Pointer Checker processor (FABPMAIN):

- Separator page for Block Map and Dumps reports
- Block Map and Block Dump report

Separator page for Block Map and Dumps reports

This separator page (see Figure 65) contains the title of “Block Map and Dumps Report,” and indicates that block maps and dumps reports will follow the page. It is produced unless SEP=NO is specified on the PROC statement as input for the PROCCTL data set.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS               "SEPARATOR PAGE FOR BLOCK MAP AND DUMPS"                     PAGE:     1
5655-K53                                                   DATE: 07/10/2006  TIME: 09.44.00                         FABPMAIN - V2.R2


        BBBB  L      OOO   CCC  K   K       M   M  AAA  PPPP         AAA  N   N DDDD        DDDD  U   U M   M PPPP   SSS
        B   B L     O   O C   C K  K        M   M A   A P   P        A   A N   N D   D        D   D U   U M   M P   P S   S
        B   B L     O   O C     K K         MM MM A   A P   P        A   A NN  N D   D        D   D U   U MM MM P   P S
        BBBB  L     O   O C     KK          M M M AAAAA PPPP         AAAAA N N N D   D        D   D U   U M M M PPPP   SSS
        B   B L     O   O C     K K         M   M A   A P            A   A N  NN D   D        D   D U   U M   M P         S
        B   B L     O   O C   C K  K        M   M A   A P            A   A N   N D   D        D   D U   U M   M P     S   S
        BBBB  LLLLL  OOO   CCC  K   K       M   M A   A P            A   A N   N DDDD         DDDD   UUU  M   M P      SSS


                                                  RRRR  EEEEE PPPP   OOO  RRRR  TTTTT
                                                  R   R E     P   P O   O R   R   T
                                                  R   R E     P   P O   O R   R   T
                                                  RRRR  EEE   PPPP  O   O RRRR    T
                                                  R R   E     P     O   O R R     T
                                                  R  R  E     P     O   O R  R    T
                                                  R   R EEEEE P      OOO  R   R   T
```

Figure 65. SNAPPIT—Separator page for Block Map and Dumps reports

Block Map and Block Dump report

These reports (see Figure 66 on page 226, Figure 67 on page 227, and Figure 68 on page 228) are used to analyze database blocks or CIs in order to determine the best way to repair them.

These reports are produced in the following case of HD Pointer Checker run:

- BLOCKDUMP=(rba,nnn) is specified on DATABASE statement. This option is used as a stand-alone program to print the specified blocks of the block maps and block dumps from HDAM or HIDAM database in SCAN process.
- DIAGDUMP=FIRST100 is specified on OPTION statement with TYPE=ALL, or SCAN. Reports are produced for the block maps and block dumps of the first 100 blocks except first IMS control block.
- DIAGDUMP=ERROR is specified on OPTION statement with TYPE=ALL, SCAN CHECK, BLKMAP.

  Reports are produced when the invalid pointer or error is detected during the validation of a pointer to a target at SCAN process. Reports are also produced when the control data set that contains pointer chaining information exists during the pointer chain reconstruction at BLOCKMAP process.

There are three formats for this report, one for the block map, and two for the block dump. Figure 66 on page 226 shows the report for the block map. Figure 67 on page 227 and Figure 68 on page 228 show the reports for the block dump. When DUMPFORM=FORMAT (default) of OPTION statement is specified in the PROCCTL data set, the report for the block dump is printed in the “logical formatted block dump format” (Figure 67 on page 227). If DUMPFORM=UNFORMAT is specified, it is printed in the “unformatted dump format” (Figure 68 on page 228).

The report fields are as follows:

DBNAME DB# PARTNAME PART ID REORG# DSG# DDNAME DSNAME
The name of the DBD, the database number (in hexadecimal), the name of the partition, partition id, partition reorg number, the data set group number (in hexadecimal) or the data set group ID (in an alphabetical character), ddname and the name of the data set name.

DUMP NO.
The dump number to be reported.

BLOCK#
The relative block number of the block or CI that contains the pointer.

Note: The first bit map is always in block 1, regardless of whether the database is OSAM or VSAM/ESDS; thus, block 0 does not exist for OSAM databases.

BLKRBA
The relative byte address of the first byte in the block or CI.

DATABASE ORGANIZATION
The organization type of the database.

The block map is a list of all the segments and free space elements that the HD Pointer Checker found in the block. Each entry is 32 bytes long. This report contains a hexadecimal and character dump of the block map. The record fields pertaining only to the format of the block map are as follows:

STORAGE
The memory address of the first byte in the line of the block map.

TYPE(HEX)
The segment type of the area defined by this block map entry. If the entry describes a free space element, it contains "FREE SPC".

RBA
The relative byte address (RBA) of the area defined by this block map entry.

ADDRESS
The address in memory of the area defined by this block map entry.

FLAGS
The flags for the pointers in the segments defined by this block map.

First Byte
A flag showing the physical pointers to the segment defined by this block map entry that were detected by the SCAN processor. Bit settings are as follows:

Bit	Pointer
0 (X'80')	HF — Hierarchical forward
1 (X'40')	HB — Hierarchical backward
2 (X'20')	PTF — Physical twin forward
3 (X'10')	PTB — Physical twin backward
4 (X'08')	PCF — Physical child first
5 (X'04')	PCL — Physical child last
6 (X'02')	RAP — Root anchor point
7 (X'01')	VLS — Pointer to data part of a split (variable—length) segment.

Second Byte
A flag showing the logical pointers to the segment defined by this block map entry that were detected by the SCAN processor. Bit settings are as follows:

Bit	Pointer
0 (X'80')	LTF — Logical twin forward
1 (X'40')	LTB — Logical twin backward
2 (X'20')	LCF — Logical child first
3 (X'10')	LCL — Logical child last
4 (X'08')	LP — Logical parent
5 (X'04')	PP — Physical parent
6 (X'02')	IN — HIDAM index pointer

Third Byte
A flag showing the pointers contained in the segment defined by this block map entry. Bit settings are as follows:

Bit	Pointer
0 (X'80')	CTR — Logical child counter
1 (X'40')	PTF — Physical twin forward
2 (X'20')	PTB — Physical twin backward

3 (X'10')	PP — Physical parent
4 (X'08')	LTF — Logical twin forward
5 (X'04')	LTB — Logical twin backward
6 (X'02')	LP — Logical parent
7 (X'01')	HF — Hierarchical forward.

SC

The segment code of the segment defined by this block map entry

COUNTERS

The two-byte field counters for the segments defined by this block map entry

Position 0-1

The value of the logical child counter in the segment defined by this block map entry

Position 2-3

The number of logical children (from unidirectional and physically paired logical children) detected by the SCAN processor for the segment defined by this block map entry

Position 4-5

The number of logical parent pointers (from physically paired logical children) detected by the SCAN processor to the segment defined by this block map entry

Position 6-7

The number of physical parent pointers (from physically paired logical children) detected by the SCAN processor to the segment defined by this block map entry

LENGTH

The length of the segment, free space element, short free space and unknown defined by this block map entry

ROOT#

The root number (in packed decimal) in this block that owns the segment defined by this block map entry

TYPE(CHAR)

The description of this mapped item in character image

segment name

The segment name

FSAP

Free space anchor point

RAP

Root anchor point

SHORT FS

Free space that is not formatted into a free space element

UNKNOWN

Data that cannot be classified as segment data or a free space element, and that is too long to be considered as a short free space element

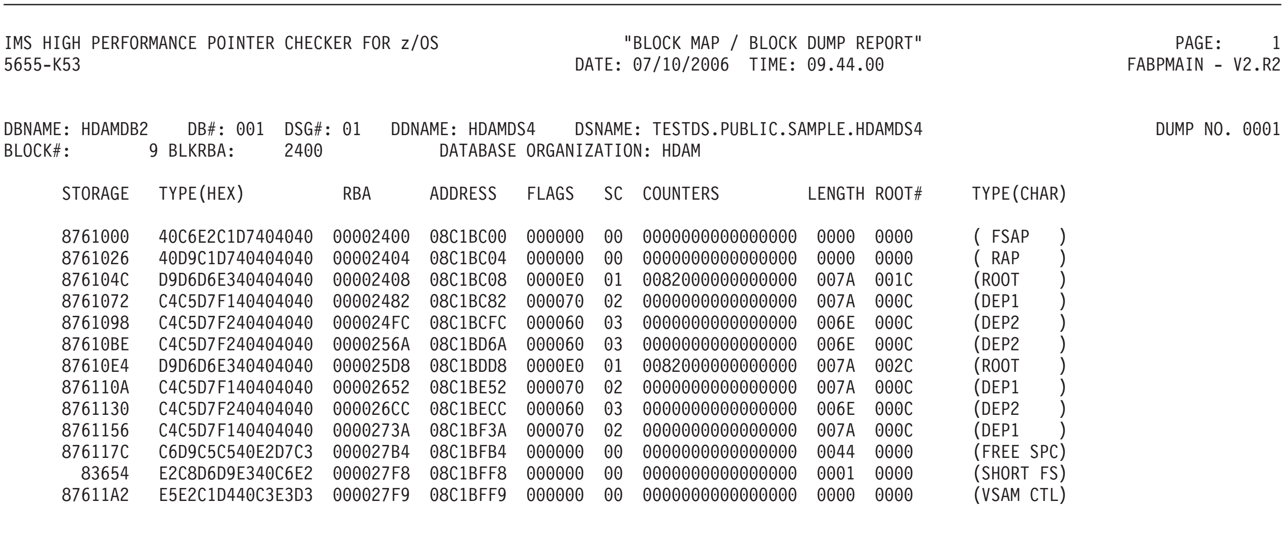
VSAM CTL

VSAM control area.

The block dump is a hexadecimal and character dump of one database or CI. The record fields pertaining only to the format of the block dump are as follows:

RBA
The relative byte address of the first byte of the physical block dump in the same line

OFFSET
The hexadecimal displacement of the first byte in the same line.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                "BLOCK MAP / BLOCK DUMP REPORT"                        PAGE:     1
5655-K53                                                 DATE: 07/10/2006  TIME: 09.44.00                        FABPMAIN - V2.R2


DBNAME: HDAMDB2    DB#: 001  DSG#: 01   DDNAME: HDAMDS4    DSNAME: TESTDS.PUBLIC.SAMPLE.HDAMDS4                        DUMP NO. 0001
BLOCK#:        9 BLKRBA:     2400           DATABASE ORGANIZATION: HDAM

      STORAGE   TYPE(HEX)          RBA      ADDRESS   FLAGS   SC  COUNTERS         LENGTH ROOT#     TYPE(CHAR)

      8761000   40C6E2C1D7404040  00002400  08C1BC00  000000  00  0000000000000000  0000  0000      ( FSAP   )
      8761026   40D9C1D740404040  00002404  08C1BC04  000000  00  0000000000000000  0000  0000      ( RAP    )
      876104C   D9D6D6E340404040  00002408  08C1BC08  0000E0  01  0082000000000000  007A  001C      (ROOT    )
      8761072   C4C5D7F140404040  00002482  08C1BC82  000070  02  0000000000000000  007A  000C      (DEP1    )
      8761098   C4C5D7F240404040  000024FC  08C1BCFC  000060  03  0000000000000000  006E  000C      (DEP2    )
      87610BE   C4C5D7F240404040  0000256A  08C1BD6A  000060  03  0000000000000000  006E  000C      (DEP2    )
      87610E4   D9D6D6E340404040  000025D8  08C1BDD8  0000E0  01  0082000000000000  007A  002C      (ROOT    )
      876110A   C4C5D7F140404040  00002652  08C1BE52  000070  02  0000000000000000  007A  000C      (DEP1    )
      8761130   C4C5D7F240404040  000026CC  08C1BECC  000060  03  0000000000000000  006E  000C      (DEP2    )
      8761156   C4C5D7F140404040  0000273A  08C1BF3A  000070  02  0000000000000000  007A  000C      (DEP1    )
      876117C   C6D9C5C540E2D7C3  000027B4  08C1BFB4  000000  00  0000000000000000  0044  0000      (FREE SPC)
        83654   E2C8D6D9E340C6E2  000027F8  08C1BFF8  000000  00  0000000000000000  0001  0000      (SHORT FS)
      87611A2   E5E2C1D440C3E3D3  000027F9  08C1BFF9  000000  00  0000000000000000  0000  0000      (VSAM CTL)
```

Figure 66. SNAPPIT—Block Map and Dumps report

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS              "BLOCK MAP / BLOCK DUMP REPORT"                             PAGE:     2
5655-K53                                              DATE: 07/10/2006  TIME: 09.44.00                           FABPMAIN - V2.R2


DBNAME: HDAMDB2    DB#: 001  DSG#: 01   DDNAME: HDAMDS4    DSNAME: TESTDS.PUBLIC.SAMPLE.HDAMDS4                         DUMP NO. 0001
BLOCK#:        9 BLKRBA:     2400            DATABASE ORGANIZATION: HDAM

    RBA  OFFSET

    2400 0000  03B40000                                                                    *....                             *
    2404 0004  00002408                                                                    *....                             *
    2408 0008  01000000 00820000 25D80000 00000000 2482000C 755AF0F0 F0F0F0F0 F0F0F2F2   *.........Q............0000000022*
         0028  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *                                *
     LINE   00002448   SAME AS ABOVE
         0068  40404040 40404040 40404040 40404040 40404040 40404040 4040                *                                *
    2482 0082  0200000B F55A0000 00000000 24080000 24FC0000 256AF0F0 F0F0F0F0 F0F0F0F1   *....5.................0000000001*
         00A2  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *                                *
     LINE   000024C2   SAME AS ABOVE
         00E2  40404040 40404040 40404040 40404040 40404040 40404040 0000                *                        ..      *
    24FC 00FC  03000000 256A0000 0000F0F0 F0F0F0F0 F2F6F4F9 40404040 40404040 40404040   *..........0000002649            *
         011C  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *                                *
     LINE   0000253C   SAME AS ABOVE
         015C  40404040 40404040 40404040 0000                                           *            ..                  *
    256A 016A  03000000 00000000 24FCF0F0 F0F0F0F0 F2F6F5F0 40404040 40404040 40404040   *..........0000002650            *
         018A  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *                                *
     LINE   000025AA   SAME AS ABOVE
         01CA  40404040 40404040 40404040 0000                                           *            ..                  *
    25D8 01D8  01000000 00820000 C5F00000 24080000 26520011 E8ECF0F0 F0F0F0F0 F0F0F3F3   *........E0..........Y.0000000033*
         01F8  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *                                *
     LINE   00002618   SAME AS ABOVE
         0238  40404040 40404040 40404040 40404040 40404040 40404040 4040                *                                *
    2652 0252  02000000 273A0000 00000000 25D80000 26CC0000 26CCF0F0 F0F0F0F0 F0F0F0F1   *.............Q........0000000001*
         0272  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *                                *
     LINE   00002692   SAME AS ABOVE
         02B2  40404040 40404040 40404040 40404040 40404040 40404040 0000                *                        ..      *
    26CC 02CC  03000000 00000000 0000F0F0 F0F0F0F0 F4F0F2F8 40404040 40404040 40404040   *..........0000004028            *
         02EC  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *                                *
     LINE   0000270C   SAME AS ABOVE
         032C  40404040 40404040 40404040 0000                                           *            ..                  *
    273A 033A  02000011 615A0000 26520000 25D80000 00000000 0000F0F0 F0F0F0F0 F0F0F0F2   *..../........Q........0000000002*
         035A  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040   *                                *
     LINE   0000277A   SAME AS ABOVE
         039A  40404040 40404040 40404040 40404040 40404040 40404040 0000                *                        ..      *
    27B4 03B4  00000044 00000000 00000000 00000000 00000000 00000000 00000000 00000000   *................................*
         03D4  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   *................................*
         03F4  00000000                                                                    *....                            *
    27F8 03F8  00                                                                          *.                               *
    27F9 03F9  0003F903 F90000                                                             *..9.9..                         *
```

Figure 67. SNAPPIT—Block Map and Block Dump report (Block Dump - Logical formatted)

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                   "BLOCK MAP / BLOCK DUMP REPORT"                          PAGE:     2
5655-K53                                                   DATE: 07/10/2006  TIME: 09.44.08                         FABPMAIN - V2.R2


DBNAME: HDAMDB2    DB#: 001  DSG#: 01   DDNAME: HDAMDS4    DSNAME: TESTDS.PUBLIC.SAMPLE.HDAMDS4                        DUMP NO. 0001
BLOCK#:        9 BLKRBA:     2400           DATABASE ORGANIZATION: HDAM

    RBA  OFFSET

    2400 0000  03B40000 00002408 01000000 00820000   25D80000 00000000 2482000C 755AF0F0 *.................Q............00*
    2420 0020  F0F0F0F0 F0F0F2F2 40404040 40404040   40404040 40404040 40404040 40404040 *00000022                        *
    2440 0040  40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040 *                                *
     LINE   00002460   SAME AS ABOVE
    2480 0080  40400200 000BF55A 00000000 00002408   000024FC 0000256A F0F0F0F0 F0F0F0F0 *  ....5.................00000000*
    24A0 00A0  F0F14040 40404040 40404040 40404040   40404040 40404040 40404040 40404040 *01                              *
    24C0 00C0  40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040 *                                *
    24E0 00E0  40404040 40404040 40404040 40404040   40404040 40404040 40400000 03000000 *                          ......*
    2500 0100  256A0000 0000F0F0 F0F0F0F0 F2F6F4F9   40404040 40404040 40404040 40404040 *......0000002649                *
    2520 0120  40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040 *                                *
     LINE   00002540   SAME AS ABOVE
    2560 0160  40404040 40404040 00000300 00000000   000024FC F0F0F0F0 F0F0F2F6 F5F04040 *        ............0000002650  *
    2580 0180  40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040 *                                *
     LINE   000025A0   SAME AS ABOVE
    25C0 01C0  40404040 40404040 40404040 40404040   40404040 40400000 01000000 00820000 *                      ..........*
    25E0 01E0  C5F00000 24080000 26520011 E8ECF0F0   F0F0F0F0 F0F0F3F3 40404040 40404040 *E0..........Y.0000000033        *
    2600 0200  40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040 *                                *
     LINE   00002620   SAME AS ABOVE
    2640 0240  40404040 40404040 40404040 40404040   40400200 0000273A 00000000 000025D8 *                  .............Q*
    2660 0260  000026CC 000026CC F0F0F0F0 F0F0F0F0   F0F14040 40404040 40404040 40404040 *........0000000001              *
    2680 0280  40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040 *                                *
     LINE   000026A0   SAME AS ABOVE
    26C0 02C0  40404040 40404040 40400000 03000000   00000000 0000F0F0 F0F0F0F0 F4F0F2F8 *          ............0000004028*
    26E0 02E0  40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040 *                                *
     LINE   00002700   SAME AS ABOVE
    2720 0320  40404040 40404040 40404040 40404040   40404040 40404040 00000200 0011615A *                        ....../.*
    2740 0340  00002652 000025D8 00000000 00000000   F0F0F0F0 F0F0F0F0 F0F24040 40404040 *.......Q........0000000002      *
    2760 0360  40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040 *                                *
     LINE   00002780   SAME AS ABOVE
    27A0 03A0  40404040 40404040 40404040 40404040   40400000 00000044 00000000 00000000 *                  ..............*
    27C0 03C0  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000 *................................*
    27E0 03E0  00000000 00000000 00000000 00000000   00000000 00000000 000003F9 03F90000 *...........................9.9..*
```

Figure 68. SNAPPIT—Block Map and Block Dump report (Block Dump - Unformatted)

SUMMARY data set

This section explains the SUMMARY data set.

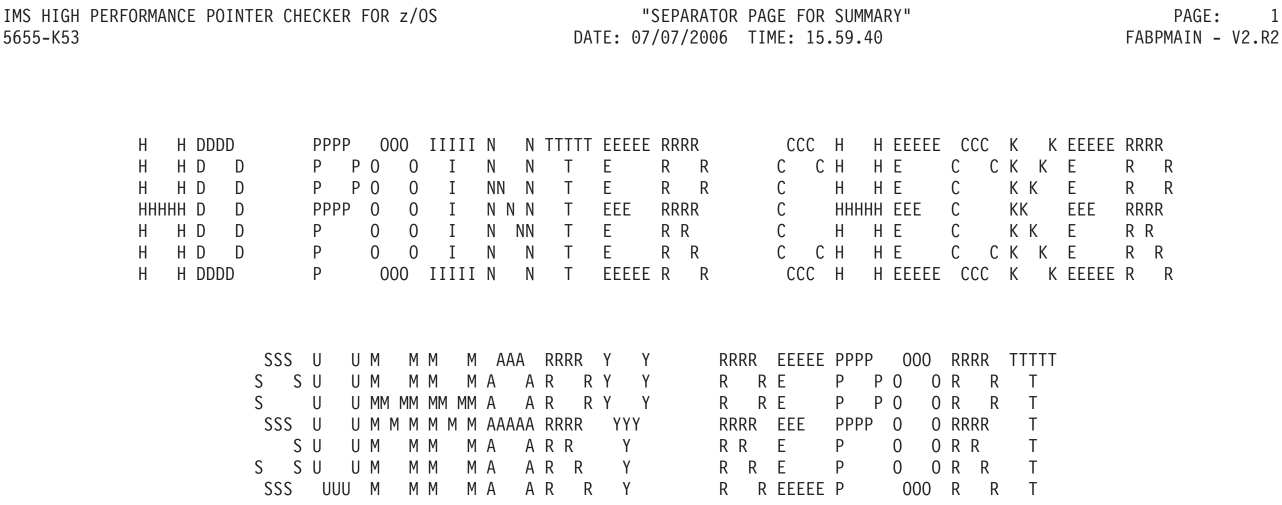
Function

The SUMMARY data set contains the following reports produced by the HD Pointer Checker processor (FABPMAIN):

- Separator page for HD Pointer Checker Summary reports
- HD Pointer Checker Summary report

Separator page for HD Pointer Checker Summary reports

This separator page (see Figure 69) contains the title of “HD Pointer Checker Summary Report,” and indicates the HD Pointer Checker Summary report will follow the page. It is produced unless SEP=NO is specified on the PROC statement as input for the PROCCTL data set.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                     "SEPARATOR PAGE FOR SUMMARY"                           PAGE:     1
5655-K53                                                   DATE: 07/07/2006  TIME: 15.59.40                         FABPMAIN - V2.R2



              H   H DDDD        PPPP   OOO  IIIII N   N TTTTT EEEEE RRRR         CCC  H   H EEEEE  CCC  K   K EEEEE RRRR
              H   H D   D       P   P O   O   I   N   N   T   E     R   R       C   C H   H E     C   C K  K  E     R   R
              H   H D   D       P   P O   O   I   NN  N   T   E     R   R       C     H   H E     C     K K   E     R   R
              HHHHH D   D       PPPP  O   O   I   N N N   T   EEE   RRRR        C     HHHHH EEE   C     KK    EEE   RRRR
              H   H D   D       P     O   O   I   N  NN   T   E     R R         C     H   H E     C     K K   E     R R
              H   H D   D       P     O   O   I   N   N   T   E     R  R        C   C H   H E     C   C K  K  E     R  R
              H   H DDDD        P      OOO  IIIII N   N   T   EEEEE R   R        CCC  H   H EEEEE  CCC  K   K EEEEE R   R


                           SSS  U   U M   M M   M  AAA  RRRR  Y   Y       RRRR  EEEEE PPPP   OOO  RRRR  TTTTT
                          S   S U   U M   M M   M A   A R   R Y   Y       R   R E     P   P O   O R   R   T
                          S     U   U MM MM MM MM A   A R   R Y   Y       R   R E     P   P O   O R   R   T
                           SSS  U   U M M M M M M AAAAA RRRR   YYY        RRRR  EEE   PPPP  O   O RRRR    T
                              S U   U M   M M   M A   A R R     Y         R R   E     P     O   O R R     T
                          S   S U   U M   M M   M A   A R  R    Y         R  R  E     P     O   O R  R    T
                           SSS   UUU  M   M M   M A   A R   R   Y         R   R EEEEE P      OOO  R   R   T
```

Figure 69. SUMMARY—Separator page for HD Pointer Checker Summary reports

HD Pointer Checker Summary report

This report (see Figure 70 on page 232) contains the summary of the HD Pointer Checker run. It is produced at the end of the HD Pointer Checker run if TYPE=ALL, SCAN, or CHECK is specified on the PROC statement as input for the PROCCTL data set.

The reports fields are as follows:

DBNAME
The name of the DBD as coded on the NAME= keyword of the DBD macro.

DB#
The database number (in hexadecimal) that identifies the database processed throughout the HD Pointer Checker run.

PID
The partition ID (in decimal) that identifies the partition containing the segment that contains the pointer

PARTNAME
The name of the partition.

DSG#
The data set group number (in hexadecimal) that identifies the database processed throughout the HD Pointer Checker run.

DBLG#
The database logical group number which indicates that physical databases with the same database logical group number are logically related to each other.

DDNAME
The ddname as coded on the DD1= keyword or OVFLW= keyword of the DATASET macro in the DBD.

DB-ORGANIZATION
The type of database organization (HISAM, HIDAM, HDAM, INDEX, LOGICAL, and so on).

C-DATE
The date when the database data set was actually created. If the specified data set is an image copy database data set, this is the date when the image copy database data set was created.

ACCM
The type of access method.

BLKSZ
The block size.

D-DATE
The date when the SCAN process was performed if the specified data set is a real database data set or a Fast Recovery image copy data set. If the specified data set is an image copy database data set, this is the date when the image copy database data set was created.

D-TIME
The time when the SCAN process was performed if the specified data set is a real database data set. If the specified data set is an image copy database data set, this is the time when the image copy database data set was created.

LRECL
The logical record length of the database data set.

DBTYPE
Indicates whether the database data set is a real database or an image copy data set. If an image copy data set that has been created more than 5 days older than the date when the SCAN process was done, the indicator "*" follows the image copy indicator "IMGCPY".

DEVICE
The device type of a real database or the type of the image copy database data set (that is, TAPE or DASD) if an image copy database data set is used.

CHK-DATE
The date when the CHECK process was done.

CHK-TIME
The time when the CHECK process was done.

%SEGMS IN OFLW
The percentage of segments in the overflow (ESDS or OSAM) part of the data set group.

DATA-SET SIZE
The data set size that is used by IMS.

CYL'S
The data set size in cylinders. This is the round up value of the number of scanned CIs/blocks divided by the number of physical blocks per cylinder for the particular DASD.

BYTES
The data set size in bytes. The number of scanned CIs/blocks multiplied by the length of the CI/block.

F-SPACE
The reusable free space in the database as indicated by FSEs that can hold the longest segment in the data set group.

% The percentage of the reusable free space in the data set that is the percentage of F-SPACE in bytes against DATA-SET SIZE in bytes.

BYTES
The F-SPACE in bytes. (See Figure 34 on page 143.)

DETECTED ERRORS
The number of errors detected throughout the HD Pointer Checker run is shown by the following fields:

TOTAL
The total number of errors which include unknown areas (that is, T2 records)

UNKNOWN
The total number of unknown areas (T2 records).

When DETECTED ERRORS TOTAL and UNKNOWN overflow, values 9999 are shown with an asterisk in the fields.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                     "HD POINTER CHECKER SUMMARY"                           PAGE:     1
5655-K53                                                   DATE: 07/28/2006  TIME: 09.56.33                         FABPMAIN - V2.R2

DBNAME/        DDNAME/          C-DATE/    D-DATE/    D-TIME    CHK-DATE/  CHK-TIME/ DATA-SET SIZE     F-SPACE %/  DETECTED ERRORS
DB# DSG# DBLG# DB-ORGANIZATION  ACCM BLKSZ LRECL DBTYPE DEVICE  %SEGMS IN OFLW       CYL'S     BYTES     BYTES     TOTAL  UNKNOWN
--- ---- ----- ---------------  ---- ----- ----- ------ ------  -------------------  ----   ---------  ----------  -----  -------

HDAMDB2        HDAMDS4          07/06/2006 07/28/2006 09.56.33  07/28/2006 09.56.33                           8 %
001 01   001   HDAM             ESDS  1024  1017   REAL  3390    98                     5     2534400      215670      0        0

HISAMDB1       HISAMDS1         07/06/2006 07/28/2006 09.56.33  07/28/2006 09.56.33
002 01   001   HISAM            KSDS  8192   510   REAL  3390    99                   N/A                               0        0

HISAMDB1       HISAMDS2         07/06/2006 07/28/2006 09.56.33  07/28/2006 09.56.33
002 01   001   HISAM       OFLW ESDS  8192   512   REAL  3390   N/A                   N/A                               0        0
```

Figure 70. SUMMARY—HD Pointer Checker Summary report (Part 1 of 2)

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                 "HD POINTER CHECKER SUMMARY (HALDB)"                       PAGE:     1
5655-K53                                                   DATE: 07/28/2006  TIME: 09.56.33                         FABPMAIN - V2.R2

DBNAME/   PARTNAME/ DDNAME/     C-DATE/    D-DATE/    D-TIME    CHK-DATE/  CHK-TIME/ DATA-SET SIZE     F-SPACE %/  DETECTED ERRORS
DB# PID  DSG# DBLG# DB-ORG      ACCM BLKSZ LRECL DBTYPE DEVICE  %SEGMS IN OFLW       CYL'S     BYTES     BYTES     TOTAL  UNKNOWN
--- ---- ---- ----- ----------- ---- ----- ----- ------ ------  -------------------  ----   ---------  ----------  -----  -------

TPFOH1    TPFOH1A   TPFOH1AA    07/06/2006 07/28/2006 09.56.33  07/28/2006 09.56.33                           1 %
003 00001 A   002   PHDAM       ESDS   512   505   REAL  3390    73                    27    10160640      147298      0        0

TPFOH1    TPFOH1A   TPFOH1AB    07/06/2006 07/28/2006 09.56.33  07/28/2006 09.56.33                          87 %
003 00001 B   002   PHDAM       ESDS   512   505   REAL  3390     0                     1      376320      329850      0        0

TPFOH3    TPFOH3A   TPFOH3AA    07/06/2006 07/28/2006 09.56.33  07/28/2006 09.56.33                          37 %
004 00001 A   002   PHDAM       ESDS   512   505   REAL  3390    65                     2      752640      280350      0        0

TPFOH2    TPFOH2A   TPFOH2AA    07/06/2006 07/28/2006 09.56.33  07/28/2006 09.56.33                          10 %
005 00001 A   002   PHIDAM      ESDS   512   505   REAL  3390     0                     6     2257920      241226      0        0

TPFOH2    TPFOH2A   TPFOH2AX    07/06/2006 07/28/2006 09.56.33  07/28/2006 09.56.33
005 00001 X   002   PHIDAM IDX  KSDS   512    14   REAL  3390     0                   N/A                               0        0

TPFOX1    TPFOX1A   TPFOX1AA    07/06/2006 07/28/2006 09.56.33  07/28/2006 09.56.33
006 00001 A   002   PSINDEX     KSDS   512    54   REAL  3390     0                   N/A                               0        0
```

Figure 70. SUMMARY—HD Pointer Checker Summary report (Part 2 of 2)

DBSRCPRT data set

The DBSRCPRT data set contains the report produced when REPORT DECODEDBD=YES is specified. This report has the following two parts:

- Messages

  This part contains the FABN*nnnn* messages issued by IMS Library Integrity Utilities' DBD reversal function. Figure 71 shows an example of this part.

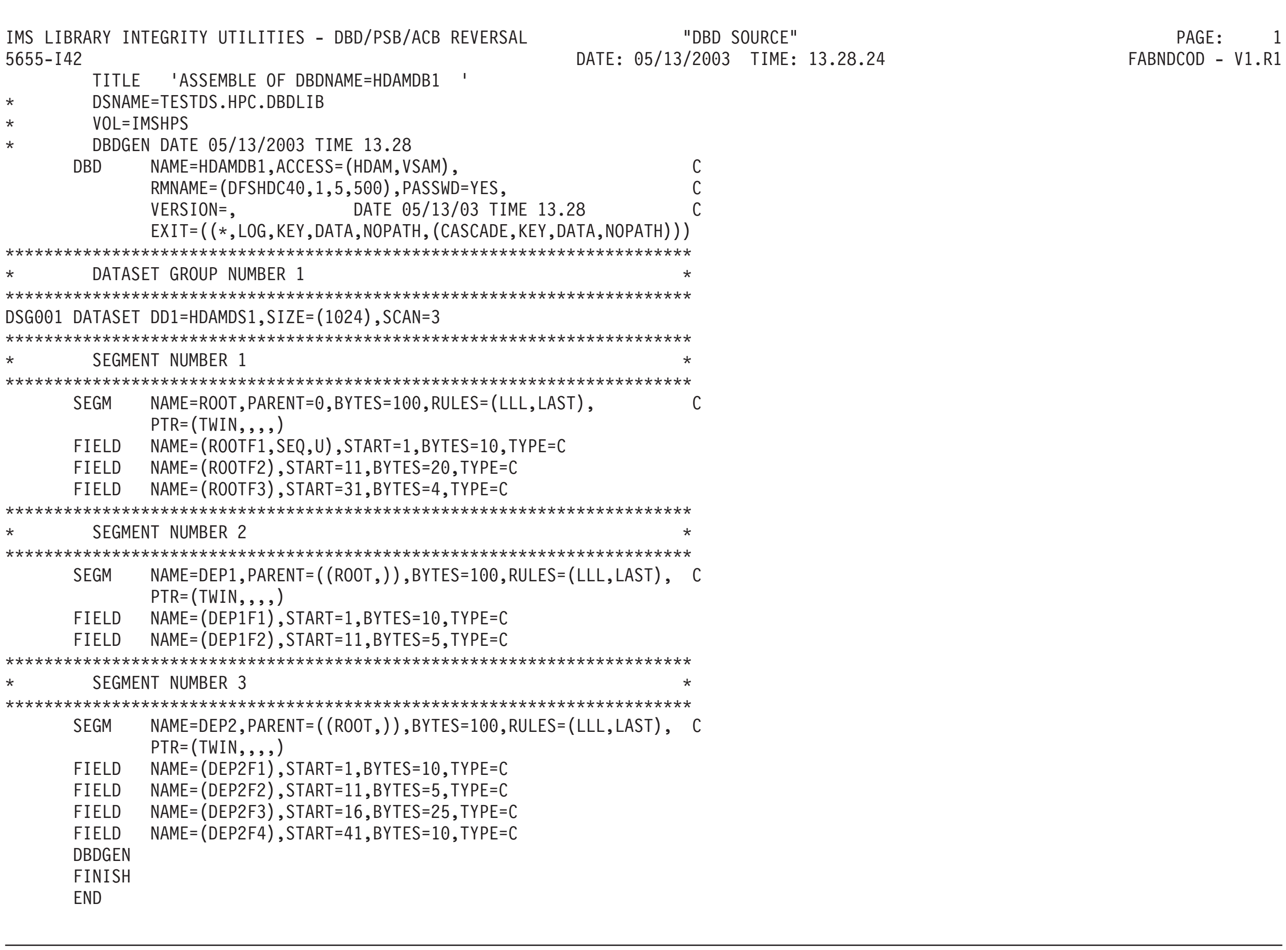
- DBD Source

  This part contains the DBD sources decoded by IMS Library Integrity Utilities' DBD reversal function. Because the logical record length is 133 and the report header is added per page, it cannot be used as input of the IMS DBDGEN utility. Figure 72 on page 234 shows an example of this part.

For the details of the FABN*nnnn* messages and the DBD sources, read about DBD/PSB/ACB Reversal in *IMS Library Integrity Utilities for z/OS User's Guide* (SC18-7025).

```
IMS LIBRARY INTEGRITY UTILITIES - DBD/PSB/ACB REVERSAL                  "MESSAGES"                                       PAGE:     1
5655-I42                                                   DATE: 05/13/2003  TIME: 13.28.24                         FABNDCOD - V1.R1

FABN0032I MEMBER HDAMDB1  PROCESSED
```

Figure 71. DBSRCPRT—Messages report



```
IMS LIBRARY INTEGRITY UTILITIES - DBD/PSB/ACB REVERSAL                "DBD SOURCE"                                        PAGE:     1
5655-I42                                                   DATE: 05/13/2003  TIME: 13.28.24                          FABNDCOD - V1.R1
         TITLE   'ASSEMBLE OF DBDNAME=HDAMDB1  '
*        DSNAME=TESTDS.HPC.DBDLIB
*        VOL=IMSHPS
*        DBDGEN DATE 05/13/2003 TIME 13.28
       DBD     NAME=HDAMDB1,ACCESS=(HDAM,VSAM),                        C
               RMNAME=(DFSHDC40,1,5,500),PASSWD=YES,                   C
               VERSION=,            DATE 05/13/03 TIME 13.28           C
               EXIT=((*,LOG,KEY,DATA,NOPATH,(CASCADE,KEY,DATA,NOPATH)))
***********************************************************************
*        DATASET GROUP NUMBER 1                                       *
***********************************************************************
DSG001 DATASET DD1=HDAMDS1,SIZE=(1024),SCAN=3
***********************************************************************
*        SEGMENT NUMBER 1                                             *
***********************************************************************
       SEGM    NAME=ROOT,PARENT=0,BYTES=100,RULES=(LLL,LAST),          C
               PTR=(TWIN,,,,)
       FIELD   NAME=(ROOTF1,SEQ,U),START=1,BYTES=10,TYPE=C
       FIELD   NAME=(ROOTF2),START=11,BYTES=20,TYPE=C
       FIELD   NAME=(ROOTF3),START=31,BYTES=4,TYPE=C
***********************************************************************
*        SEGMENT NUMBER 2                                             *
***********************************************************************
       SEGM    NAME=DEP1,PARENT=((ROOT,)),BYTES=100,RULES=(LLL,LAST),  C
               PTR=(TWIN,,,,)
       FIELD   NAME=(DEP1F1),START=1,BYTES=10,TYPE=C
       FIELD   NAME=(DEP1F2),START=11,BYTES=5,TYPE=C
***********************************************************************
*        SEGMENT NUMBER 3                                             *
***********************************************************************
       SEGM    NAME=DEP2,PARENT=((ROOT,)),BYTES=100,RULES=(LLL,LAST),  C
               PTR=(TWIN,,,,)
       FIELD   NAME=(DEP2F1),START=1,BYTES=10,TYPE=C
       FIELD   NAME=(DEP2F2),START=11,BYTES=5,TYPE=C
       FIELD   NAME=(DEP2F3),START=16,BYTES=25,TYPE=C
       FIELD   NAME=(DEP2F4),START=41,BYTES=10,TYPE=C
       DBDGEN
       FINISH
       END
```

Figure 72. DBSRCPRT—DBD source report

DBMAPPRT data set

The DBMAPPRT data set contains the report produced when REPORT MAPDBD=YES is specified. This report has the following two parts:

- Messages

  This part contains the FABM*nnnn* messages issued by IMS Library Integrity Utilities' DBD map function. Figure 73 shows an example of this part.
- DBD Map

  This part contains the DBD maps created by IMS Library Integrity Utilities' DBD map function. The maps depict the hierarchical structure of databases as described in the DBDs. Figure 74 shows an example of this part.

For the details of the FABM*nnnn* messages and the DBD maps, read about DBD/PSB/ACB Mapper in *IMS Library Integrity Utilities for z/OS User's Guide* (SC18-7025).

```
IMS LIBRARY INTEGRITY UTILITIES - DBD/PSB/ACB MAPPER                    "MESSAGES"                                       PAGE:     1
5655-I42                                                   DATE: 03/19/2003  TIME: 17.13.05                            FABMDMAP - V1.R1
FABM0034I MEMBER HDAMDB1  PROCESSED
```

Figure 73. DBMAPPRT—Messages report

```
IMS LIBRARY INTEGRITY UTILITIES - DBD/PSB/ACB MAPPER                  "DBD MAP"                                          PAGE:   A
5655-I42                                                   DATE: 03/19/2003  TIME: 17.13.05                            FABMDMAP - V1.R1
  DBDNAME=HDAMDB1   VOLUME=IMSHPS  DSNAME=TESTDS.HPC.DBDLIB
  DBDMAP OF HDAMDB1                                                           ACCESS=HDAM VSAM


                                                          ***********
                                                          *   ROOT  *
                                                          *******001*
                                                         ._____|______.
                                                         |            |
                                                    ***********  ***********
                                                    *   DEP1  *  *   DEP2  *
                                                    *******002*  *******003*
```

Figure 74. DBMAPPRT—DBD Map report

Chapter 4. Operating instructions for Disk Address Analyzer

The FABPCHRO program is a single job step program which prints the absolute disk address of user-specified relative byte address. It runs as a batch job.

This chapter describes the JCL and all other input you must specify to run FABPCHRO. It also describes the output data set and reports produced by FABPCHRO.

Topics:

- "Job control language"
- "Input" on page 238
- "Output" on page 239

Job control language

This section explains the JCL statements of the FABPCHRO program.

FABPCHRO JCL

To run FABPCHRO, supply the appropriate DD statements. Table 24 summarizes the DD statements. Actual JCL requirements are as follows:

Table 24. FABPCHRO DD statements

DDNAME	Use	Format	Need
CTL	Input	LRECL=80	Required
ddname	Input		Required
PRT	Output	LRECL=133	Required
SYSUDUMP	Output	SYSOUT	Optional

EXEC

This statement must be in the following format:

```
//      EXEC PGM=FABPCHRO
```

PRT DD

This required output data set contains the report produced by FABPCHRO. If BLKSIZE is coded on the DD statement, it must a multiple of 133.

CTL DD

This required input data set contains your description of the processing to be done by module FABPCHRO. It describes the databases that will be processed, and it contains the RBAs for which absolute disk addresses are needed.

ddname DD

This input data set is an IMS database data set and a DD statement exists for every database data set that you want to process. Use the ddname that is specified in the DBD. The actual data set must be a real database. Image copy is not a valid input for FABPCHRO. Do not provide a DD statement for any image copy data set.

For a VSAM database data set, the AMP parameter can be used to specify the number of VSAM I/O buffers. This specification overrides the VSAMBF parameter of the OPTION statement.

For an OSAM database data set and an image copy, the DCB=BUFNO= parameter can be used to specify the number of I/O buffers.

You should be careful when specifying this parameter since:

- The region size will also increase when buffer space is specified
- The excessive number of buffers may cause an open error for the database data set.

SYSUDUMP DD (or SYSABEND)
This defines output from a system ABEND dump routine. It is used only when a dump is required.

No cataloged procedure is provided for printing absolute disk addresses (module FABPCHRO) because the JCL for that program is very simple.

Input

This section explains the input for the FABPCHRO program.

FABPCHRO CTL data set

This section explains the FABPCHRO CTL data set.

Function

The CTL data set contains your description of the processing to be done by module FABPCHRO. It contains target relative byte addresses (RBAs). Module FABPCHRO prints the absolute disk address for each input target RBA.

Format

This control data set usually resides in the input stream. However, it can also be defined as a sequential data set or as a member of a partitioned data set. It must contain one 80-byte, fixed-length record for each target RBA to be processed. BLKSIZE, if coded, must be a multiple of 80. The CTL data set can be coded as shown in Figure 75.

```
//CTL       DD *
DSFACHO0  0000069A
DSFACHO0  0CCCCC9A
DSFACHO0  0000079A
DSSCHHV0  00000900            VSAM
DSSCHHV0  00001000            VSAM
DSSCHXI0  00000000
DSSCHXI0  00000053
DSSCHXI0  000000CF
DSSCHXI0  000018CF
DSSCHXI0  00002000
/*
```

Figure 75. Example of the FABPCHRO CTRL data set

There is only one record type in the CTL data set. Figure 76 on page 239 shows the format of the CTL data set.

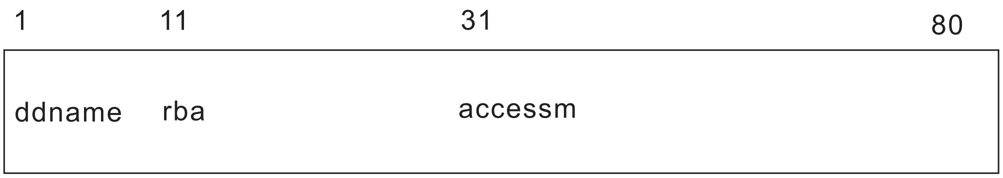


Figure 76. Format of the FABPCHRO CTL data set

Position	Description
1	This 8-digit field contains the (left-justified) DDNAME of the database that contains the target RBA.
11	This 8-digit field contains the target RBA. This field is hexadecimal, with leading zeros (if needed).
31	This 4-digit field indicates the access method used for the database that contains the target RBA. Use one of the following codes: **VSAM** This is a VSAM data set. **Blank** This is an OSAM data set.

Note: If you code this field incorrectly, results are unpredictable.

Output

This section explains the output of the FABPCHRO program.

FABPCHRO PRT data set

This section explains the FABPCHRO PRT data set.

Function

The PRT data set contains all of the reports produced by the FABPCHRO program. These include the following:

- Control Card Format report
- Control Card and Pointer Information report

Control Card Format report

This report (see Figure 77) contains a brief description of the format of the FABPCHRO control statement.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS              "CONTROL CARD FORMAT REPORT"                      PAGE:     1
5655-K53                                            DATE: 07/10/2006  TIME: 10.54.21                   FABPCHRO - V2.R2


COLUMNS  1- 8: DDNAME

COLUMNS 11-18: RBA IN PRINTABLE HEX

COLUMNS 31-34: 'VSAM' IF VSAM DATABASE
               '    ' IF OSAM DATABASE
```

Figure 77. FABPCHRO—Control Card Format report

Control Card and Pointer Information report

This report (see Figure 78 on page 241) contains the following information:

- The user’s input control statements

- The resulting disk addresses
- A summary of data set extents.

The report fields are as follows:

DDNAME
The ddname used on the input database data set.

RBA
The target relative byte address for which an absolute disk address is desired.

For an RBA beyond 4 GB, it is a 32-bit odd value based on the over 4-GB RBA rule.

For example, the hexadecimal value x’10000F000’ is shown as 0000F001.

BLKSIZE
The block size or CI size of the input database data set.

BLOCK#
The relative physical record number of the block or CI that contains the target relative byte address.

Notes:

1. The first block in an OSAM database is BLOCK#=1.
2. The first block in a VSAM database is BLOCK#=0.
3. For VSAM databases with CI size larger than 4096, BLOCK# is not the same as the CI number.
4. This field is *not* the same as the BLOCK# fields in the pointer error messages (see this field under “Validation of a Pointer to a Target at SCAN (HDAM/HIDAM/PHDAM/PHIDAM) report” on page 183 and “Validation of a Pointer to a Target at CHECK report” on page 191).

VOLUME
The volume serial number of the device that contains the target relative byte address.

CCCCHHHHRR
The actual direct access address of the record that contains the target relative byte address. This is a 10-digit hexadecimal number. CCCC is the cylinder, HHHH is the track, and RR is the record number.

OFST
The hexadecimal displacement of the target relative byte address within its physical record.

Note: This is not necessarily the offset from the beginning of a VSAM CI. It is the offset from the beginning of the VSAM physical record. If your CI size is greater than 4096, these are not the same.

The following fields pertain to the summary of data set extents:

VOLSER
The volume serial number of the device that contains an extent of the input database data set.

LOW-CCHH
The lowest cylinder and track on the extent described by this report line.

HIGH-CCHH
The highest cylinder and track on the extent described by this report line.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS              "CONTROL CARD/POINTER INFORMATION REPORT"                     PAGE:     1
5655-K53                                                   DATE: 07/10/2006  TIME: 10.54.21                         FABPCHRO - V2.R2


     C O N T R O L   C A R D                       P O I N T E R   I N F O R M A T I O N

0        1         2         3           DDNAME    RBA     BLKSIZE    BLOCK#  VOLUME CCCCHHHHRR OFST
1234567890123456789012345678901234      --------  --------  -------  --------  ------ ---------- ----

HISAMDS1 00000001             VSAM      HISAMDS1  0000001   X'2000'         0  PMR004 0259000001 0010

HISAMDS1 00000002             VSAM      HISAMDS1  0000002   X'2000'         0  PMR004 0259000001 0020

HISAMDS1 00000003             VSAM      HISAMDS1  0000003   X'2000'         0  PMR004 0259000001 0030

HISAMDS1 00000004             VSAM      HISAMDS1  0000004   X'2000'         0  PMR004 0259000001 0040

HISAMDS1 00000005             VSAM      HISAMDS1  0000005   X'2000'         0  PMR004 0259000001 0050


SUMMARY OF EXTENTS FOR HISAMDS1

VOLSER  LOW-CCHH  HIGH-CCHH
-------------------------------
PMR004  02590000  028A000E
```

Figure 78. FABPCHRO—Control Card and Pointer Information report

Chapter 5. Operating instructions for the HD Pointer Checker Site Default Generation utility

The HD Pointer Checker Site Default Generation utility (HDPC Site Default Generation utility) enables you to use your own default value for the PROCCTL control statements. It runs as a batch job.

This chapter describes how to generate and use the PROCCTL site default table.

Topics:

Functions

The HDPC Site Default Generation utility has two functions; generating and reporting the PROCCTL site default table.

Generating the PROCCTL site default table

The HDPC Site Default Generation utility analyzes the PROCCTL control statements and generates a source code for the PROCCTL site default table. If you use the site default, the library for the PROCCTL site default table module (FABPCTL0) must be concatenated to the STEPLIB DD of FABPMAIN runtime JCL.

When FABPMAIN finds the name FABPCTL0 in the STEPLIB libraries, HD Pointer Checker loads it and uses it as the default value of the PROCCTL statement.

If you specify a value in the PROCCTL control statement in the HD Pointer Checker FABPMAIN JCL, you can override the site default value at run time.

To use the PROCCTL site default, do as follows:

1. Run the HDPC Site Default Generation utility (FABPTGEN) job step to create a source code of the PROCCTL site default table (FABPCTL0)
2. Assemble and link the FABPCTL0 source code
3. Concatenate the load module library in which FABPCTL0 resides to the STEPLIB of the HD Pointer Checker FABPMAIN JCL

Note: The HD Pointer Checker PROCCTL site default table is available only when HD Pointer Checker runs as a stand alone utility. It is not available when the HASH Check option is called in the IMS HP Image Copy job, the IMS Parallel Reorganization job, or the IMS Database Recovery Facility job.

IMS HP Image Copy has its own site default table. To set the site default in the IMS HP Image Copy job, use the IMS HP Image Copy site default generation utility. For more details, see *IMS High Performance Image Copy User's Guide for Versions 3* (SC18-7617) and *IMS High Performance Image Copy User's Guide for Versions 4* (SC18-9409).

Figure 79 on page 244 shows the process flow of how to use the PROCCTL site default table.

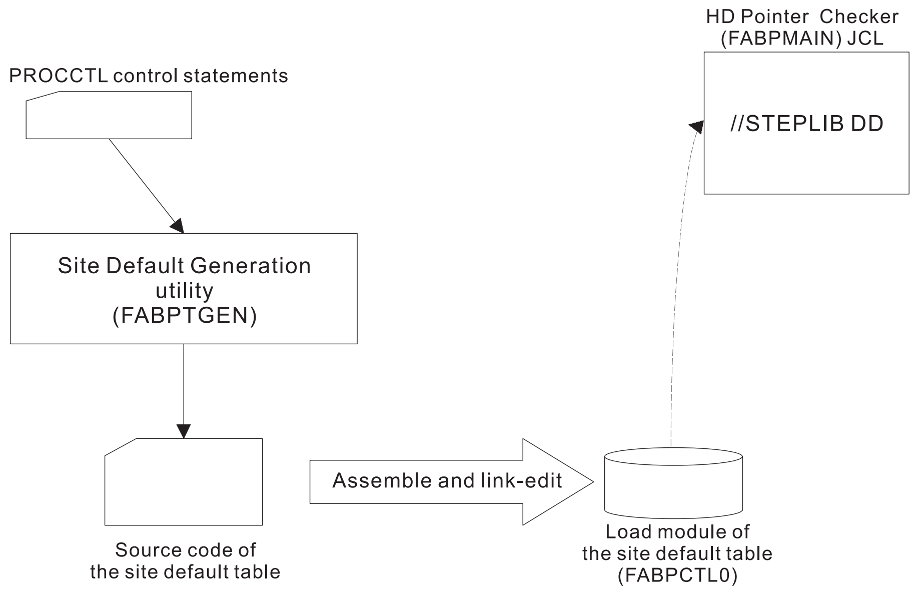

HD Pointer Checker (FABPMAIN) JCL
PROCCTL control statements
//STEPLIB DD
Site Default Generation utility (FABPTGEN)
Assemble and link-edit
Source code of the site default table
Load module of the site default table (FABPCTL0)

Figure 79. Process flow of PROCCTL site default table creation

Reporting the PROCCTL site default table

The HDPC Site Default Generation utility reads the PROCCTL site default table and prints the site default values that are set in the reports.

Job control language

This section describes the requirements for using the HDPC Site Default Generation utility (FABPTGEN).

FABPTGEN JCL

To run the HDPC Site Default Generation utility (FABPTGEN), supply an EXEC statement with the PARM parameters and appropriate DD statements as shown in Table 25.

Table 25 summarizes the DD statements.

Table 25. FABPTGEN DD statements

DDNAME	Use: Input (In), Output (Out)	Format	EXEC PARM= Required (R) or optional (O) Header	
			PARM='GEN'	PARM='REPORT'
STEPLIB	In	PDS	R	R
PROCCTL	In	LRECL=80	R	
SYSPUNCH	Out	LRECL=80	R	
SYSPRINT	Out	LRECL=133	R	R

Table 25. FABPTGEN DD statements (continued)

DDNAME	Use: Input (In), Output (Out)	Format	EXEC PARM= Required (R) or optional (O) Header	
			PARM='GEN'	PARM='REPORT'
SYSABEND or SYSUDUMP	Out	LRECL=133	O	O

EXEC

This statement must be in the following format:

```
    //    EXEC PGM=FABPTGEN,PARM='parm'
```

Specify GEN or REPORT for *parm*.

GEN

Specifies to generate a site default table. This is the default.

REPORT

Specifies to print the site default values stored in the site default table.

Sample JCLs that run the FABPTGEN program with PARM='GEN' and PARM='REPORT' are provided in the SHPSSAMP data set that is provided by the product. The member names are FABPDFL1 and FABPDFL2.

STEPLIB DD

This required input data set contains the IMS HP Pointer Checker load module library. When PARM='REPORT' is specified in the EXEC statement, you must also specify the data set that includes the site default table module member (FABPCTL0).

PROCCTL DD

This is a required data set when PARM='GEN' is specified in the EXEC statement. The format is the same as the FABPMAIN PROCCTL statement. Specify this input data set to include your own default values for the PROCCTL control statements.

SYSPUNCH DD

This is a required output data set when PARM='GEN' is specified in the EXEC statement.

An assembler source code of the site default table will be produced in this data set. The following DCB parameters must be specified:

RECFM=F or FB

LRECL=80

BLKSIZE=80 or multiple of 80

SYSPRINT DD

This is a required output data set. The PROCCTL statements report and the site default values report will be printed in this data set. You can specify SYSOUT=* (or JES output class name) instead of a data set name.

- PROCCTL statements report
  This report contains exactly the same control statements that you specified in the PROCCTL data set. It is generated when FABPTGEN generates the PROCCTL site default table.

- Site default values report
  This report prints the site default values that is stored in the PROCCTL site default table.

SYSUDUMP DD (or SYSABEND)
This defines the output for a system ABEND dump routine. It is used only when a dump is required.

FABPTGEN PROCCTL data set

The PROCCTL control statements are required to generate the PROCCTL site default table.

Specify keywords of which you want to change the default values from the HD Pointer Checker's system default value to your own suitable value. The HDPC Site Default Generation utility analyzes PROC, OPTION, REPORT, and DATABASE statements, and sets the site default values.

Table 26 through Table 29 on page 248 shows the keywords that are available for the HDPC Site Default Generation utility. You can specify some of the keywords. If keywords are omitted, HD Pointer Checker system default values will be used.

For description of each keyword, see "FABPMAIN PROCCTL data set" on page 71.

PROC control statement
This control statement is required. Table 26 shows the keywords that are available for this control statement.

Table 26. The keywords available for the site default in the PROC control statement

Keyword	System default value of HD Pointer Checker	Can specify site default value?
TYPE	(None)	Yes. This keyword is required. (Only ALL or SCAN)
DBORG	ALL	Yes
HASH	NO	Yes
EPSCHK	YES	Yes
IXKEYCHK	NO	Yes
SEP	YES	Yes
VLSSUMM	NO	Yes
CHECK	(CHK,111111)	No
CHECKREC	NO	No
SYMIXCHK	NO	Yes
SYMLPCHK	NO	Yes
ICRG#CHK	NO	Yes
RETCDDSN	*NO	Yes
USER	*NO	Yes
ITKBSRVR	*NO	Yes
ITKBLOAD	*NO	Yes

OPTION control statement
This control statement is optional. The OPTION statement can be specified before and after the DATABASE statement, however, the HDPC Site Default

Generation utility analyzes only the OPTION statement before the DATABASE statement, and ignores the rest.

Table 27. The keywords available for the site default in the OPTION control statement

Keyword	System default value of HD Pointer Checker	Can specify site default value?
ERRLIMIT	YES	No
DIAGDUMP	ERROR	Yes
DSSIZE	(None)	No
DUMPFORM	FORMAT	Yes
HISTORY	NO	Yes
HOMECHK	YES	Yes
ICUNIT	(None)	Yes
INCORE	YES	Yes
INTERVAL	DATASET	Yes
VSAMBF	(None)	Yes
IBUFF	10 TRK	Yes
KEYSIN	NO	Yes
T2CHK	(0,7)	Yes
ZEROCTR	NO	Yes
DIAG	NO	No
PRINTDATA	NO	No
NOCHKP	(None)	No
SPIXCHK	YES	Yes
PTRCHK	YES	Yes
SPMN	NO	Yes
THRESHOLDS	See “OPTION statement” on page 87 for the system default value of each keyword.	Yes

REPORT control statement

This control statement is optional. The REPORT statement can be specified before and after the DATABASE statement, however, the HDPC Site Default Generation utility analyzes only the REPORT statement before the DATABASE statement and ignores the rest.

Table 28. The keywords available for the site default in the REPORT control statement

Keyword	System default value of HD Pointer Checker	Can specify site default value?
RUNTM	YES	Yes
INTST	YES	
BITMAP	YES	
FSEMAP	YES	
MAXFSD	YES	
INTFS	YES	
DBDIST	YES	
CHAINDIST	YES	
DECODEDBD	NO	
MAPDBD	NO	
COMPFACT	NO	
SEGIO	NO	

DATABASE control statement

This control statement is optional. More than one DATABASE control statements can specified, however, HDPC Site Default Generation utility will analyze only the first statement and ignores the rest of the statements.

Table 29. The keywords available for the site default in the DATABASE control statement

Keyword	System default value of HD Pointer Checker	Can specify site default value?
DB	(None)	No
PART	*ALL	No
NUM	(None)	No
DD	*ALL	No
OVERFLOW	(None)	No
PRIMEDB	(None)	No
DATASET	REAL	Yes
SCANGROUP	1	No
BLOCKDUMP	(None)	No
DBALL	No	Yes

The keywords indicated with No in column marked ″Can specify site default value?″ cannot be specified in the PROCCTL data set. The HDPC Site Default Generation utility will ignore such keywords.

The statements underlined in Figure 80 on page 249 show the effective statements and keyword for setting the site defaults.

```
PROC TYPE=ALL,HASH=YES          <= Site default
  OPTION HISTORY=YES            <= Site default
  REPORT BITMAP=NO              <= Site default
DATABASE DB=dbdname1,DATASET=IC  <= Site default can only be specified for
                                    DATASET= in the DATABASE statement
  OPTION HISTORY=YES
  REPORT BITMAP=YES
DATABASE DB=dbdname1,DATASET=IC
  OPTION HISTORY=YES
  REPORT DBDIST=NO
END
```

Figure 80. Example of the FABPTGEN PROCCTL control statement

Assembling and link-editing FABPCTL0

To create the site default table module FABPCTL0, assemble and link the SYSPUNCH that is generated by FABPTGEN.

For SYSIN of the assemble job step, specify the SYSPUNCH data set that is generated in the FABPTGEN. In the link-edit job step, it is recommended that you use AMODE=31, RMODE=ANY instead of the default AMODE=24, RMODE=24 by adding MODE=31 and RMDE=ANY to the EXEC statement PARM list.

Figure 81 on page 250 shows a sample for creating the site default table module FABPCTL0.

```
//FABPDFL1 JOB ...........
//*********************************************************************
//*   Licensed Materials - Property of IBM                            *
//*                                                                   *
//*   5655-K53                                                        *
//*                                                                   *
//*   (c) Copyright IBM Corp. 2006 All Rights Reserved.               *
//*                                                                   *
//*   US Government Users Restricted Rights - Use,                    *
//*   duplication or disclosure restricted by GSA ADP                 *
//*   Schedule Contract with IBM Corp.                                *
//*                                                                   *
//*********************************************************************
//* FABPDFL1:                                                         *
//*   HD Pointer Checker Site Default Generation Utility (PARM='GEN')*
//*   Sample JCL                                                      *
//*                                                                   *
//*   This is a sample JCL for running Site Default Generation        *
//*   Utility. It generates the site default table of HD Pointer     *
//*   Checker.                                                        *
//*                                                                   *
//*   This JCL consists of the following steps:                       *
//*     1) Generate the site default table source code                *
//*     2) Assemble the site default table                            *
//*     3) Link-Edit the site default table module                    *
//*********************************************************************
//*
//**********************************************************************
//*  FABPTGEN - HDPC SITE DEFAULT GENARATION UTILITY
//*  ( PARM='GEN' SAMPLE PROCEDURE )
//**********************************************************************
//HDPCTGEN PROC HLQ='HPS'
//*---------------------------------------------------------------------
//*  CREATE SOURCE CODE OF SITE DEFAULT TABLE
//*---------------------------------------------------------------------
//G        EXEC PGM=FABPTGEN,PARM='GEN'
//STEPLIB  DD   DISP=SHR,DSN=&HLQ..SHHPSLMD0
//SYSPUNCH DD   DISP=(NEW,PASS,DELETE),DSN=&&SOURCE,
//             DCB=(RECFM=FB,BLKSIZE=800),SPACE=(TRK,(1,1)),UNIT=SYSDA
//SYSPRINT DD   SYSOUT=*
//SYSUDUMP DD   DUMMY
```

Figure 81. Example JCL for creating the site default table module FABPCTL0 (Part 1 of 2)

```
//*--------------------------------------------------------------------
//*  ASSEMBLE & LINK ==> SITE DEFAULT TABLE MODULE (FABPCTL0)
//*--------------------------------------------------------------------
//ASM      EXEC PGM=ASMA90,COND=(4,LT,G),
//             PARM='OBJECT,NODECK,LIST,XREF(SHORT)'
//SYSLIN   DD   DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(5,5,0)),
//             DCB=(BLKSIZE=400),DSN=&&OBJECT
//SYSUT1   DD   DISP=(,DELETE),UNIT=SYSDA,SPACE=(CYL,(10,5))
//SYSPUNCH DD   DUMMY
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DISP=(OLD,DELETE,DELETE),DSN=&&SOURCE
//*
//L        EXEC PGM=IEWL,COND=(4,LT,ASM),REGION=4096K,
//             PARM='LIST,REFR,REUS,AMODE=31,RMODE=ANY'
//SYSPRINT DD   SYSOUT=*
//SYSLIN   DD   DISP=(OLD,DELETE,DELETE),DSN=&OBJECT
//*
//         PEND
//*
//*-------------------------------------------------------------------*
//*  FABPTGEN (PARM='GEN') - HDPC SITE DEFAULT GENARATION UTILITY    *
//*-------------------------------------------------------------------*
//GO       EXEC HDPCTGEN,HLQ=HPS
//*-------------------------------------*
//* SPECIFY SITE DEFAULT VALUES         *
//*-------------------------------------*
//G.PROCCTL DD  *
 PROC TYPE=ALL
 OPTION HISTORY=YES
 REPORT COMPFACT=YES,SEGIO=YES
 END
/*
//L.SYSLMOD DD  DISP=SHR,DSN=HPS.TABLELIB(FABPCTL0)
//*
//
```

Figure 81. Example JCL for creating the site default table module FABPCTL0 (Part 2 of 2)

Chapter 6. JCL examples for HD Pointer Checker

There are many ways to use the HD Pointer Checker utility. The examples given here represent some of the common things that they can do. By studying and understanding these examples, you can learn the techniques to monitor the condition of your databases.

Topics:

Example 1: Standard database analysis—Prevention

The example shown in Figure 82 on page 255 shows how to run HD Pointer Checker and HD Tuning Aid as a preventive measure. This is a typical way when you perform a regularly scheduled run (see Chapter 10, “Database repair guidelines,” on page 271).

The major consideration points are:

- Specify TYPE=ALL on the PROC statement.

  This creates sort records for pointers and segments in all the databases you specify. This also causes all the databases you specify to be checked and the database error messages to be printed.
- Specify EPSCHK=YES, or do not specify EPSCHK=keyword on the PROC statement.

  This causes Extended Pointer Set (EPS) to be evaluated. The sort records for pointers that reside in EPS are created. The EPS check function checks not only the pointers (direct and indirect pointers) but also ILKs included in the EPS against the target segment and the ILE (Indirect List Entry).
- Specify KEYSIN=YES on the OPTION statement.

  This causes the KEYSIN data set to be created. Since the IBM-supplied, cataloged procedure (FABPPTA) runs HD Tuning Aid (in addition to HD Pointer Checker), it is required that HD Tuning Aid input data set be created by HD Pointer Checker.
- Specify ERRLIMIT=YES on the OPTION statement.

  This causes only the first 100 messages to be printed and this is the default. If you want to print all error messages, specify ERRLIMIT=NO. However, if your databases are large and have many errors, a certain processing time is required to print all of the messages.
- Specify INCORE=YES, or do not specify INCORE= keyword on the OPTION statement.

  This means you are checking as many pointers as possible in the SCAN process. This minimizes the CPU in general, I/O, and run time used by HD Pointer Checker. If this run uncovers some (unexpected) pointer errors, the listing produced by the BLOCKMAP process may be incomplete.
- Specify DATASET=IMAGECOPY on the DATABASE statement.

  This means you are running with an image copy data set as input database. You do not want to impact database users with your HD Pointer Checker run. Using image copy data set allows you to schedule your HD Pointer Checker job whenever you wish, without having any effect on the real databases.
- You do not need to specify DD statements for the database data set or the image copy data set. The data sets to be processed are dynamically allocated, according to the specifications in the DFSMDA or the RECON data sets.
- You must specify DATASET=REAL for PHIDAM primary index, if you want to check the primary index database. Because image copy cannot be taken for the primary index database.
- You do not need to specify any other optional parameter on the control statements.

  Do not turn off the checking of certain pointers. All pointers should be verified by this HD Pointer Checker run.

A special consideration point for processing a large database is:

- Specify SCANGROUP= on the DATABASE statement.

The scan tasks with different scan group numbers are processed in parallel. This shortens scan time. Especially when your database is a HALDB with many partitions, consider the use of the SCANGROUP= option.

```
//EXAMPLE1 JOB --- use normal job statement parameters here ---
//*
//JOBLIB DD DSN=HPS.SHPSLMD0,DISP=SHR
//       DD DSN=IMSVS.RESLIB,DISP=SHR
//*
//*
//PCRUN   EXEC FABPPTA,
//             PSB=PSBSMUAL
//* ----------------------------------------------------------------------------
//HDPCPRO.PROCCTL DD *
 PROC TYPE=ALL,EPSCHK=YES
  OPTION KEYSIN=YES,ERRLIMIT=YES,INCORE=YES
  DATABASE DB=DSSTUIVN,DD=DSSTUIV0,OVERFLOW=DSSTUIV1,DATASET=IMAGECOPY
  DATABASE DB=DSSCHXIN,DD=DSSCHXI0,PRIMEDB=DSSCHHVN,DATASET=IMAGECOPY
  DATABASE DB=DSFACXVN,DD=DSFACXV0,PRIMEDB=DSFACHON,DATASET=IMAGECOPY
  DATABASE DB=DSFDAXVN,DD=DSFDAXV0,OVERFLOW=DSFDAXV1,PRIMEDB=DSFACHON,
           DATASET=IMAGECOPY
  DATABASE DB=DSSCHHVN,DD=DSSCHHV0,DATASET=IMAGECOPY
  DATABASE DB=DSFACHON,DD=DSFACHO0,DATASET=IMAGECOPY
  DATABASE DB=DSCRSDVN,DD=DSCRSDV0,DATASET=IMAGECOPY
  DATABASE DB=DSCRSDVN,DD=DSCRSDV1,DATASET=IMAGECOPY
  DATABASE DB=DSCLSDVN,DD=DSCLSDV0,DATASET=IMAGECOPY
  DATABASE DB=PHDAM0A1,PART=*ALL,DATASET=IMAGECOPY
  DATABASE DB=PHDAM0B1,PART=*ALL,DATASET=IMAGECOPY
  DATABASE DB=PHIDAM01,PART=*ALL,DD=(A,B,C),DATASET=IMAGECOPY
  DATABASE DB=PHIDAM01,PART=*ALL,DD=X,DATASET=REAL
  DATABASE DB=PSINDEX1,PART=*ALL,DATASET=IMAGECOPY
 END
/*
//* ----------------------------------------------------------------------------
```

Figure 82. Example 1: JCL for HD Pointer Checker

Example 2: Standard database analysis—Corrupted database

You should use the example shown in Figure 83 on page 257 to run HD Pointer Checker when a database is damaged.

The major consideration points are:

- Specify TYPE=ALL on the PROC statement.

  This creates sort records for pointers and segments in all the databases specified. This also checks all the databases specified and prints the database error messages.
- Specify EPSCHK=YES, or do not specify EPSCHK= keyword on the PROC statement.

  This causes Extended Pointer Set (EPS) to be evaluated. The sort records for pointers that reside in EPS are created. The EPS check function checks not only the pointers (direct and indirect pointers) but also ILKs included in the EPS against the target segment and the ILE (Indirect List Entry).
- Specify ERRLIMIT=NO on the OPTION statement.

  This causes all error messages to be printed. Only the first 100 messages are printed, whether ERRLIMIT=YES is specified or ERRLIMIT=keyword is not specified at all.
- Specify INCORE=NO on the OPTION statement.

  This means you are not checking the pointers in the SCAN process. The actual pointer checking is done in the CHECK process. Because you may wind up fixing the database with zapping, you may need the listing produced by the BLOCKMAP processor. This listing is complete only if the in-core checking is not used.
- Specify DIAGDUMP=ERROR on the OPTION statement.

  This means a block map and dump for a database data set block, which contains an error, is printed.
- Specify DATASET=REAL, or not specify DATASET=keyword on the DATABASE statement.

  This means you are running with real databases (not image copies) as input. Since you may wind up fixing the database with zapping, you may need the absolute disk addresses of the invalid pointers. These are only available when real databases as input are used.
- You do not need to specify DD statements for the database data set or the image copy data set. The data sets to be processed are dynamically allocated, according to the specifications in the DFSMDA or the RECON data sets.
- You must specify DATASET=REAL for PHIDAM primary index, if you want to check the primary index database. Because image copy cannot be taken for the primary index database.
- Specify SCANGROUP= on the DATABASE statement. The scan tasks with different scan group numbers are processed in parallel. This shortens scan time.
- Do not specify any other optional parameter on the control statements.

  Do not turn off the checking of certain pointers. When your database is broken, all kind of pointer checkings should be done.

```
//EXAMPLE2 JOB --- use normal job statement parameters here ---
//*
//JOBLIB DD DSN=HPS.SHPSLMD0,DISP=SHR
//       DD DSN=IMSVS.RESLIB,DISP=SHR
//*
//*
//PCRUN   EXEC FABPP,
//             PSB=PSBSMUAL
//* ----------------------------------------------------------------------------
//HDPCPRO.PROCCTL DD *
 PROC TYPE=ALL,EPSCHK=YES
  OPTION ERRLIMIT=NO,INCORE=NO,DIAGDUMP=ERROR
  DATABASE DB=DSSTUIVN,DD=DSSTUIV0,OVERFLOW=DSSTUIV1
  DATABASE DB=DSSCHXIN,DD=DSSCHXI0,PRIMEDB=DSSCHHVN
  DATABASE DB=DSFACXVN,DD=DSFACXV0,PRIMEDB=DSFACHON
  DATABASE DB=DSFDAXVN,DD=DSFDAXV0,OVERFLOW=DSFDAXV1,PRIMEDB=DSFACHON
  DATABASE DB=DSSCHHVN,DD=DSSCHHV0
  DATABASE DB=DSFACHON,DD=DSFACHO0
  DATABASE DB=DSCRSDVN,DD=DSCRSDV0
  DATABASE DB=DSCRSDVN,DD=DSCRSDV1
  DATABASE DB=DSCLSDVN,DD=DSCLSDV0
  DATABASE DB=PHDAM0A1,PART=*ALL,DATASET=IMAGECOPY
  DATABASE DB=PHDAM0B1,PART=*ALL,DATASET=IMAGECOPY
  DATABASE DB=PHIDAM01,PART=*ALL,DD=(A,B,C),DATASET=IMAGECOPY
  DATABASE DB=PHIDAM01,PART=*ALL,DD=X,DATASET=REAL
  DATABASE DB=PSINDEX1,PART=*ALL,DATASET=IMAGECOPY
/*
//* ----------------------------------------------------------------------------
```

Figure 83. Example 2: JCL for HD Pointer Checker

Example 3: Standard database analysis—Prevention with HASH

You should use the example shown in Figure 84 on page 259 when you want to use HASH Check Function to know that the database you are going to use is in an operable state.

Do the following:

- Specify TYPE=ALL on the PROC statement. This parameter creates the sort records for pointers and segments in all the specified databases. This parameter also checks the specified databases and prints a message when an error is found.
- Specify HASH=FORCE on the PROC statement. This parameter makes the HASH Check function apply to the databases that are not subject to the restrictions, but the databases that are not applicable to the HASH Check function are processed using the standard pointer checking technique.

  INCORE=NO is forced when the HASH Check function is active. EPSCHK=NO is forced when the HASH Check function is active.
- Specify ERRLIMIT=NO on the OPTION statement. This parameter causes all error messages to be printed. Only the first 100 messages are printed if ERRLIMIT=YES is specified or if ERRLIMIT=keyword is not specified at all.
- Specify DATASET=IMAGECOPY on the DATABASE statement. This parameters indicates that you are running the application with an image copy data set as the input database and you don't want to affect other database users with your HD Pointer Checker run. Using the image copy data set allows you to schedule your HD Pointer Checker job whenever you wish, without having any effect on the real databases.
- Specify SCANGROUP= on the DATABASE statement. The scan tasks with different scan group numbers are processed in parallel. This shortens scan time.
- You do not need to specify DD statements for the database data set or the image copy data set. The data sets to be processed are dynamically allocated, according to the specifications in the DFSMDA or the RECON data sets.

You do not need to specify any other optional parameter on the control statements.

Do not turn off the checking of certain pointers. All pointers should be verified by this HD Pointer Checker run.

```
//EXAMPLE4 JOB --- use normal job statement parameters here ---
//*
//JOBLIB DD DSN=HPS.SHPSLMD0,DISP=SHR
//       DD DSN=IMSVS.RESLIB,DISP=SHR
//*
//*
//PCRUN   EXEC FABPP,
//             PSB=PSBGRAL
//* ----------------------------------------------------------------------------
//PROCCTL  DD  *
PROC TYPE=ALL,HASH=FORCE
OPTION ERRLIMIT=NO
DATABASE DB=DBPID10,DD=DSID10,DATASET=IMAGECOPY,SCANGROUP=01
DATABASE DB=DBPIX10,DD=DSIX10,PRIMEDB=DBPID10,DATASET=IMAGECOPY,
SCANGROUP=01
DATABASE DB=DBPSX10,DD=DSSX10,PRIMEDB=DBPID10,DATASET=IMAGECOPY,
SCANGROUP=01
DATABASE DB=DBPSX11,DD=DSSX11,PRIMEDB=DBPID10,DATASET=IMAGECOPY,
SCANGROUP=01
DATABASE DB=DBPID20,DD=DSID20,DATASET=IMAGECOPY,
SCANGROUP=01
DATABASE DB=DBPIX20,DD=DSIX20,PRIMEDB=DBPID20,DATASET=IMAGECOPY,
SCANGROUP=01
DATABASE DB=DBPHD30,DD=DSHD30,DATASET=IMAGECOPY,
SCANGROUP=01
DATABASE DB=DBPHD30,DD=DSHD31,DATASET=IMAGECOPY,
SCANGROUP=01
DATABASE DB=DBPHS40,DD=DSHS40,OVERFLOW=DSHS41,DATASET=IMAGECOPY,
SCANGROUP=02
DATABASE DB=DBPID50,DD=DSID50,DATASET=IMAGECOPY,
SCANGROUP=02
DATABASE DB=DBPIX50,DD=DSIX50,PRIMEDB=DBPID50,DATASET=IMAGECOPY,
SCANGROUP=02
DATABASE DB=DBPHD60,DD=DSHD60,DATASET=IMAGECOPY,
SCANGROUP=02
DATABASE DB=DBPID70,DD=DSID70,DATASET=IMAGECOPY,
SCANGROUP=02
DATABASE DB=DBPIX70,DD=DSIX70,PRIMEDB=DBPID70,DATASET=IMAGECOPY,
SCANGROUP=02
DATABASE DB=DBPSX70,DD=DSSX70,PRIMEDB=DBPID70,DATASET=IMAGECOPY,
SCANGROUP=02
DATABASE DB=DBPSX71,DD=DSSX70,PRIMEDB=DBPID70,DATASET=IMAGECOPY,
SCANGROUP=02
DATABASE DB=DBPHD80,DD=DSHD80,DATASET=IMAGECOPY,
SCANGROUP=02
DATABASE DB=DBPHD90,DD=DSHD90,DATASET=IMAGECOPY,
SCANGROUP=02
DATABASE DB=DBPHSA0,DD=DSHSA0,OVERFLOW=DSHSA1,DATASET=IMAGECOPY,
SCANGROUP=02
DATABASE DB=PHDAM0A1,PART=*ALL,DATASET=IMAGECOPY,
SCANGROUP=02
DATABASE DB=PHDAM0B1,PART=*ALL,DATASET=IMAGECOPY,
SCANGROUP=02
END
/*
```

Figure 84. Example 3: JCL for HD Pointer Checker

Example 4: Standard database analysis—Multiple jobs

> **Note**
>
> The examples shown in Figure 85 on page 261, Figure 86 on page 262, and Figure 87 on page 262 are not recommended. If, however, you have problems of region size as described as follows, do as the example shows.

You can divide HD Pointer Checker job into two process jobs, SCAN and CHECK, and redivide the SCAN job into multiple jobs. But, usually, it is not recommended to run in multiple jobs. If you encounter a shortage of region size—caused by too many database data sets, you may have to run the jobs in multiple SCAN jobs and CHECK jobs. You can evade the region shortage problem by running the SCAN in multiple jobs. To validate the pointers, you need to process in one CHECK job all databases that have logical or index relationship. After all of the SCAN jobs are completed, specify all work data sets written by the SCAN jobs as input for the CHECK job. The CHECK job must be run in a single job.

If you do not have such problems with the region size, it is strongly recommended that you run HD Pointer Checker in a single job by using TYPE=ALL rather than run it in multiple jobs. When you do so, the performance is better, the work data sets become smaller, and the JCL statements are simpler.

In both SCAN and CHECK jobs, you do not need to specify DD statements for the database data set or the image copy data set. The data sets to be processed are dynamically allocated, according to the specifications in the DFSMDA or the RECON data sets.

SCAN job

In this job, HD Pointer Checker scans the databases and creates work data sets.

The major considerations are:

- Specify TYPE=SCAN on the PROC statement.

  This creates sort records for pointers and segments in work data sets.
- Specify the DD statements of the work data set as a permanent data set. When you use a FABPP, FABPPD, or FABPPA procedure, the work data sets are allocated as temporary ones. You need to specify the data set names explicitly.

  For the necessary work data sets, see “FABPMAIN JCL” on page 37.
- Optional parameters to be specified.

  You may need to specify optional parameters on the control statements, depending on whether your purpose is the prevention run or the broken database checking run. Refer to “Example 1: Standard database analysis—Prevention” on page 254 and “Example 2: Standard database analysis—Corrupted database” on page 256.

CHECK job

HD Pointer Checker validates and evaluates by using the work data sets created in the SCAN jobs. Then HD Pointer Checker continues the internal sort and BLOCKMAP process.

- Specify TYPE=CHECK on the PROC statement.

  Specify all work data sets—that is SORTEX*nn*, MERGIN*nn*, SORTIL*nn*, SORTE2*nn*, and MERGI2*nn*—created by the SCAN jobs as the input for the check. For the CHECK JCL, assign a serial number starting from 01 for *nn*, which might be different from *nn* of the SCAN jobs. For the details, see the following examples and “FABPMAIN JCL” on page 37.

```
 //EXAMPL31 JOB --- use appropriate job statement parameters here --- //*
 //PCRUN   EXEC FABPP,
 //             PSB=PSBSMUAL
 //PROCCTL  DD  *
   PROC TYPE=SCAN,EPSCHK=YES,IXKEYCHK=YES
   DATABASE DB=DSSTUIVN,DD=DSSTUIV0,OVERFLOW=DSSTUIV1,
   SCANGROUP=01
   DATABASE DB=DSSCHXIN,DD=DSSCHXI0,PRIMEDB=DSSCHHVN,
   SCANGROUP=01
   DATABASE DB=DSFACXVN,DD=DSFACXV0,PRIMEDB=DSFACHON,
   SCANGROUP=01
   DATABASE DB=DSFDAXVN,DD=DSFDAXV0,OFLOW=DSFDAXV1,PRIMEDB=DSFACHON,
   SCANGROUP=01
   DATABASE DB=PHDAM0A1,PART=*ALL,SCANGROUP=02
   DATABASE DB=PHDAM0B1,PART=*ALL,SCANGROUP=02
   DATABASE DB=PHIDAM01,PART=*ALL,DD=(A,B,C),SCANGROUP=02
   DATABASE DB=PHIDAM01,PART=*ALL,DD=X,SCANGROUP=02
   DATABASE DB=PSINDEX1,PART=*ALL,SCANGROUP=02
/*
//* DD STATEMENTS FOR SORTEX01 WORK DATA SETS HERE:
//SORTEX01 DD DSN=HPS.JOB1.SORTEX01,DISP=(NEW,CATLG,DELETE),
//  UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//*
//* DD STATEMENTS FOR WORK DATA SETS OF SCANGROUP 01 HERE:
//MERGIN01 DD DSN=HPS.JOB1.MERGIN01,DISP=(NEW,CATLG,DELETE),
//  UNIT=SYSALLDA,SPACE=(CYL,(100,100))
//SORTIL01 DD DSN=HPS.JOB1.SORTIL01,DISP=(NEW,CATLG,DELETE),
//  UNIT=SYSALLDA,SPACE=(CYL,(50,50))
//MERGI201 DD DSN=HPS.JOB1.MERGI201,DISP=(NEW,CATLG,DELETE),
//  UNIT=SYSALLDA,SPACE=(CYL,(50,50))
//SORTE201 DD DSN=HPS.JOB1.SORTE201,DISP=(NEW,CATLG,DELETE),
//  UNIT=SYSALLDA,SPACE=(CYL,(50,50))
//*
//* DD STATEMENTS FOR WORK DATA SETS OF SCANGROUP 02 HERE:
//MERGIN02 DD DSN=HPS.JOB1.MERGIN02,DISP=(NEW,CATLG,DELETE),
//  UNIT=SYSALLDA,SPACE=(CYL,(100,100))
//SORTIL02 DD DSN=HPS.JOB1.SORTIL02,DISP=(NEW,CATLG,DELETE),
//  UNIT=SYSALLDA,SPACE=(CYL,(50,50))
//MERGI202 DD DSN=HPS.JOB1.MERGI202,DISP=(NEW,CATLG,DELETE),
//  UNIT=SYSALLDA,SPACE=(CYL,(50,50))
//SORTE202 DD DSN=HPS.JOB1.SORTE202,DISP=(NEW,CATLG,DELETE),
//  UNIT=SYSALLDA,SPACE=(CYL,(50,50))
```

Figure 85. Example 4: JCL-1 for multiple SCAN jobs of HD Pointer Checker

```
 //EXAMPL32 JOB --- use appropriate job statement parameters here --- //*
 //PCRUN   EXEC FABPP,
 //             PSB=PSBSMUAL
 //PROCCTL  DD  *
   PROC TYPE=SCAN,EPSCHK=YES,IXKEYCHK=YES
   DATABASE DB=DSSCHHVN,DD=DSSCHHV0,SCANGROUP=01
   DATABASE DB=DSFACHON,DD=DSFACHO0,SCANGROUP=01
   DATABASE DB=DSCRSDVN,DD=DSCRSDV0,SCANGROUP=01
   DATABASE DB=DSCRSDVN,DD=DSCRSDV1,SCANGROUP=01
   DATABASE DB=DSCLSDVN,DD=DSCLSDV0,SCANGROUP=01
/*
//* DD STATEMENTS FOR SORTEX01 WORK DATA SETS HERE:
//SORTEX01 DD DSN=HPS.JOB2.SORTEX01,DISP=(NEW,CATLG,DELETE),
//  UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//*
//* DD STATEMENTS FOR WORK DATA SETS OF SCANGROUP 01 HERE:
//MERGIN01 DD DSN=HPS.JOB2.MERGIN01,DISP=(NEW,CATLG,DELETE),
//  UNIT=SYSALLDA,SPACE=(CYL,(100,100))
//SORTIL01 DD DSN=HPS.JOB2.SORTIL01,DISP=(NEW,CATLG,DELETE),
//  UNIT=SYSALLDA,SPACE=(CYL,(50,50))
//MERGI201 DD DSN=HPS.JOB2.MERGI201,DISP=(NEW,CATLG,DELETE),
//  UNIT=SYSALLDA,SPACE=(CYL,(50,50))
//SORTE201 DD DSN=HPS.JOB2.SORTE201,DISP=(NEW,CATLG,DELETE),
//  UNIT=SYSALLDA,SPACE=(CYL,(50,50))
```

Figure 86. Example 4: JCL-2 for HD Pointer Checker multiple SCAN jobs

```
//EXAMPL33 JOB --- use appropriate job statement parameters here --- //*
//PCRUN   EXEC FABPP,
//             PSB=PSBSMUAL
//HDPCSCAN.PROCCTL  DD  *
  PROC TYPE=CHECK
/*
//* DD STATEMENTS FOR SORTEXnn FROM SCAN JOBS HERE:
//SORTEX01 DD DSN=HPS.JOB1.SORTEX01,DISP=(OLD,DELETE)
//SORTEX02 DD DSN=HPS.JOB2.SORTEX01,DISP=(OLD,DELETE)
//*
//* DD STATEMENTS OF WORK DATA SETS FROM SCAN JOBS HERE:
//MERGIN01 DD DSN=HPS.JOB1.MERGIN01,DISP=(OLD,DELETE)
//MERGIN02 DD DSN=HPS.JOB1.MERGIN02,DISP=(OLD,DELETE)
//MERGIN03 DD DSN=HPS.JOB2.MERGIN01,DISP=(OLD,DELETE)
//SORTIL01 DD DSN=HPS.JOB1.SORTIL01,DISP=(OLD,DELETE)
//SORTIL02 DD DSN=HPS.JOB1.SORTIL02,DISP=(OLD,DELETE)
//SORTIL03 DD DSN=HPS.JOB2.SORTIL01,DISP=(OLD,DELETE)
//MERGI201 DD DSN=HPS.JOB1.MERGI201,DISP=(OLD,DELETE)
//MERGI202 DD DSN=HPS.JOB1.MERGI202,DISP=(OLD,DELETE)
//MERGI203 DD DSN=HPS.JOB2.MERGI201,DISP=(OLD,DELETE)
//SORTE201 DD DSN=HPS.JOB1.SORTE201,DISP=(OLD,DELETE)
//SORTE202 DD DSN=HPS.JOB1.SORTE202,DISP=(OLD,DELETE)
//SORTE203 DD DSN=HPS.JOB2.SORTE201,DISP=(OLD,DELETE)
```

Figure 87. Example 4: JCL-3 for HD Pointer Checker multiple CHECK job

Chapter 7. Operational strategy recommended for HD Pointer Checker

This chapter outlines some methods that can help an installation become more effective and more efficient in the use of IMS. These methods help increase database availability and integrity, while saving time for the programmer and the analyst.

Topics:

Good operating procedures

In order to use IMS successfully, establish and implement sound operating procedures. Especially important are sound backup and recovery method. DBRC can be very helpful in this area.

At reorganization time

All IMS databases need to be reorganized occasionally. In order to use HD Pointer Checker most effectively, adopt the following procedure for reorganizing the databases:

1. Reorganize the database.
2. Create the image copy.
3. Begin production use of the database.
4. Run HD Pointer Checker (and HD Tuning Aid) with image copy input.
5. Print and *retain* HD Pointer Checker (and HD Tuning Aid) listings.

By running HD Pointer Checker with image copy input, the following is verified:

- The database has no errors. Its IMS structure is valid.
- The image copy tape is usable. Tapes are prone to operator errors. Once you successfully read the image copy data set, you will be greatly confident that you could use it for recovery if necessary.

During general use of the database

Between reorganizations, the following procedures help minimize difficulty and maximize performance:

1. Make image copies regularly.
2. Run HD Pointer Checker (and HD Tuning Aid) regularly with image copy input.
3. Compare the listings with the reorganization-time listings.

It is important to run HD Pointer Checker (and HD Tuning Aid) regularly. Run them every **n** days or every **n** image copies. How often you run them depends on how valuable the data is and how often it is updated.

A key reason for regular HD Pointer Checker runs is to discover any database damage quickly. If database damage is found shortly after it occurs, the repair process is usually much easier. For example, pointer errors (if left untreated) can spread rapidly. If such errors are found early, the correction requires much less of the programmer and/or analyst's time.

HD Pointer Checker and HD Tuning Aid print a lot of performance-related information. In some ways, the reorganization-time listing represents the best possible condition of your database. By comparing your current run listing with the listing made immediately after reorganization, you can evaluate the changes that have occurred. These changes may help explain changes experienced in recent database performance (response-time increase, EXCP increase, and so on).

Use this information to help determine when to reorganize a database. This can help in either of these two ways:
1. It can indicate the need for an earlier-than-planned reorganization.
2. It can indicate that a planned reorganization is unnecessary.

In either case, computer resources are saved.

During development

When developing an application with a new database, include the creation of suitable HD Pointer Checker (and HD Tuning Aid) JCL as part of the development process. By setting up HD Pointer Checker jobs before the database goes into production, create these types of jobs in advance:

- HD Pointer Checker (and HD Tuning Aid) with real database input (no-in-core checking)
- HD Pointer Checker (and HD Tuning Aid) with image copy input (in-core checking)
- HD Pointer Checker with real database input (dumping blocks)
- FABPCHRO with real database input (finding disk addresses).

During regression testing

When installing a new release level of system such as IMS and z/OS, most installations run a series of tests before placing the new release into production. These regression tests should *always* include an HD Pointer Checker run against *all* of the databases (using image copy input). The data is too important to take any risks. HD Pointer Checker is a critical part of database protection and recovery process. Be sure that you are prepared to debug a database error, if the need arises.

Chapter 8. HD Pointer Checker online considerations

To get valid results from HD Pointer Checker, input databases must all be in a stable condition. The collection of input databases must represent a "snapshot" at a given instant. No update operations to any of the databases can be in progress when this "snapshot" is taken. Incomplete insert, replace, or delete operations usually cause HD Pointer Checker to detect errors. “Database consistency” on page 267 describes ways to ensure that the databases are consistent when running HD Pointer Checker.

Although it is generally undesirable, it is still possible to run HD Pointer Checker with an inconsistent set of input databases. This can be the case if the output from the IMS Online Database Image Copy utility is used as input to HD Pointer Checker This chapter describes an operational technique for doing this successfully.

Warning: The technique described below is *not* the recommended way to check the databases.

Use the output data sets from the IMS Online Database Image Copy utility as input to HD Pointer Checker to find error messages in HD Pointer Checker reports. Some or all of these error messages may be a result of database updates that were in progress when the Online Database Image Copy job was running. Analyze these error messages to determine whether they represent real database errors, or whether they are merely a result of using the so-called "fuzzy" image copy as input. If HD Pointer Checker is run regularly, this situation can be managed.

Consecutive HD Pointer Checker run listings must be compared. Whenever the same error message occurs on two consecutive run listings, you can be sure it a real error. Then analyze the errors and repair the databases. How to do this is described in Chapter 10, “Database repair guidelines,” on page 271.

All error messages that disappear on the following HD Pointer Checker run are quite likely to be a result of using the Online Database Image Copy data sets as input. It is probably safe to ignore such error messages, since they do not represent database corruption.

Although this technique is cumbersome and error-prone, it can be used successfully with due attention. However, this method of checking the databases is *not recommended*.

Chapter 9. DBRC and HD Pointer Checker

This chapter presents the method for running both HD Pointer Checker and HD Tuning Aid under DBRC.

Topics:

Background

The HD Pointer Checker processor (FABPMAIN) of the HD Pointer Checker utility and the HD Tuning Aid utility run under the IMS batch region controller (DFSRRC00). The reason for this is to obtain access to certain DL/I control blocks. Neither utility accesses databases with DL/I calls. For the rest of this section, "running IMS HP Pointer Checker" means running one of the above two modules (under IMS).

IMS treats IMS HP Pointer Checker like any other batch application program and calls DBRC to obtain authorization. If authorization is denied, then the IMS HP Pointer Checker program cannot run. When a database experiences a pointer problem, HD Pointer Checker may have to be run to determine the extent of database damage. Therefore, it is important to plan how to run HD Pointer Checker in the environment—before such an emergency arises.

Database consistency

When running HD Pointer Checker with image copy input, what is happening to the real databases has no bearing on the IMS HP Pointer Checker run. Your concern, in this case, is that DBRC does not deny authorization.

When running HD Pointer Checker with real database input, it is important that there are no updates to any of the databases during the IMS HP Pointer Checker run. Of course, DBRC is not expected to deny authorization.

One way to be sure that all of the databases remain consistent during the IMS HP Pointer Checker run is to use DISP=OLD on database DD statements. A disadvantage of this is that other users are denied read access to the databases during the IMS HP Pointer Checker run.

IMS installation choices

The primary factor that governs how to run HD Pointer Checker is how the *DBRC* operand is specified on the *IMSCTRL* macro during IMS installation. There are two cases to be considered:

- DBRC=(FORCE) has been specified.

  This means that DBRC is active in both the online and batch environments. It also means that DBRC cannot be turned off by using a parameter on the EXEC statement in JCL.
- DBRC=(FORCE) has not been specified.

This specifies whether DBRC is active in the online and batch environments. To override this, use a parameter on the EXEC statement.

Running IMS HP Pointer Checker while DBRC is inactive

Warning: The following method is applicable only when DBRC=(FORCE) has *not* been specified during IMS installation.

The simplest way to run IMS HP Pointer Checker is to have DBRC inactive during the run. Do this by coding *N* in the 14th parameter on the EXEC statement. The IRLM should also be disabled by coding *N* in the 15th parameter on the EXEC statement. The following are suitable EXEC statements for HD Pointer Checker steps that run under the region controller:

```
//HDPCPRO  EXEC PGM=DFSRRC00,
//            PARM='region,FABPMAIN,psb,,,,,,,,,,,N,N'
```

When DL/I is specified, PSB should have PROCOPT=G. The above suggestions apply to all HD Pointer Checker runs, whether the inputs are image copies or real databases.

Summary

In this case, there is probably no change from the way of the current IMS HP Pointer Checker run in a system without DBRC (except for specifying the 14th and 15th parameters). The important points are:

- Use PROCOPT=G
- Disable the IRLM
- Deactivate DBRC.

Running IMS HP Pointer Checker while DBRC is active

If specified DBRC=(FORCE) has been specified, DBRC is active during the HD Pointer Checker run. All of the suggestions given in the following sections apply whenever running IMS HP Pointer Checker while DBRC is active, even if DBRC=(FORCE) has not been specified.

The main difficulty in running IMS HP Pointer Checker under DBRC is that DBRC may deny authorization and not let the program run. It may be necessary to circumvent DBRC in order to run HD Pointer Checker.

There are two considerations that apply when IMS HP Pointer Checker is to be run under DBRC:

- Whether *FORCER* or *NOFORCER* has been used in the *INIT.RECON* command.
- Whether image copies or real databases are used as input to HD Pointer Checker.

Using FORCER on INIT.RECON command

If *FORCER* is used, it causes some difficulties in running IMS HP Pointer Checker:

- If you would like to process image copy data sets, run HD Pointer Checker in ULU region without DBD name.

  Otherwise, HD Pointer Checker cannot run with image copy input. The image copy data set names are different from those of the real database. This causes the database authorization call (from IMS to DBRC) to fail. Message DFS047I (with CODE=20) is issued for each database referenced by the PSB or DBD.

- A DD statement must be provided for all database data sets that are in the DMB, even when all of them are not to be processed. Otherwise, authorization fails. For example, DD statements must be provided for all databases and index databases that are also in the DMB—even though HD Pointer Checker may not process all of the above databases. If one of the DD statements is omitted, message DFS047I (with CODE=20) is issued for that data set.

Since the above difficulties are so severe, running IMS HP Pointer Checker under DBRC with FORCER is not recommended.

Note: To circumvent the difficulties caused by FORCER, use a different set of RECON data sets when running HD Pointer Checker. These "null" RECON data sets can be defined with the following JCL:

```
//LIST    EXEC PGM=DSPURX00
//SYSPRINT  DD SYSOUT=A
//IMS       DD DISP=SHR,DSN=IMSVS.DBDLIB
//SYSIN     DD *
  INIT.RECON
/*
//RECON1    DD DISP=SHR,DSN=IMSVS.TEST.RECON1
//RECON2    DD DISP=SHR,DSN=IMSVS.TEST.RECON2
```

Use the "real" RECON data sets (in which FORCER is specified) for all of the other IMS processing.

Using NOFORCER on INIT.RECON command

It is easier to run HD Pointer Checker when using *NOFORCER* on the *INIT.RECON* command. Then you do not have to worry about being denied authorization. Message DFS194W or DFS3341I may be issued for some or all of the databases that are referenced by the PSB, but HD Pointer Checker still runs normally.

The IRLM should also be disabled and a PSB with PROCOPT=G should be used in the HD Pointer Checker run. The following are suitable EXEC statements for HD Pointer Checker steps that run under the region controller:

```
//HDPCPRO  EXEC PGM=DFSRRC00,
//              PARM='region,FABPMAIN,psb,,,,,,,,,,,Y,N'
```

Summary

All of the above apply to running IMS HP Pointer Checker with either image copy or real database input. The important steps are:

- Use PROCOPT=G
- Disable the IRLM
- Use "null" RECON data sets if your "real" RECON data sets have FORCER.

Notes:

1. The above method is *required* when DBRC=(FORCE) has been specified during IMS installation.
2. The above method is *optional* when DBRC=(FORCE) has *not* been specified during IMS installation.

Conclusion

If DBRC is active, a HD Pointer Checker utility job is affected. Advance planning can avoid any negative effects that DBRC could have on the IMS HP Pointer Checker job. It is important to do this planning, because IMS HP Pointer Checker needs DBRC to process HALDBs, and to do dynamic allocation of image copy data sets. It is often run in an "emergency" situation.

Chapter 10. Database repair guidelines

This chapter provides guidelines for repairing a database. It discusses why you need database repair and how you can repair your database.

Topics:

- "Why pointers are important"
- "How a database can become corrupted" on page 272
- "Various methods of repairing a corrupted database" on page 273
- "Repairing a database by using HD Pointer Checker and zapping" on page 274

Why pointers are important

The basic structure of an IMS database segment is shown in Figure 88. It is a product-sensitive programming interface. See "Programming interface information" on page 732 to understand the restrictions associated with this type of materials.

Each segment contains two distinct parts: the prefix part and the data part. The *data* part contains the information that is used and processed by your application programs. The *prefix* part contains control information (including direct pointers) that is used by IMS. IMS uses these pointers to access the database segments. It follows a chain of pointers to navigate through the database.

Product-Sensitive Programming Interface

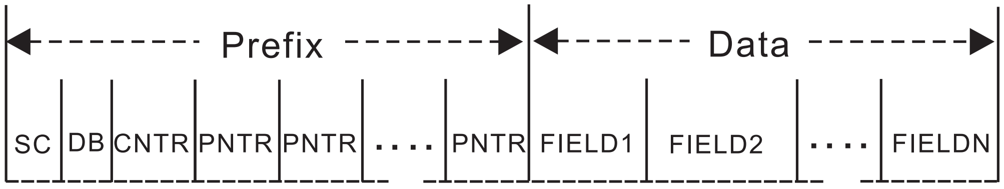


Figure 88. IMS segment structure

End of Product-Sensitive Programming Interface

The pointers that are contained in the segment prefixes are called *direct pointers*. Each direct pointer contains a 4-byte address of a target segment. IMS calculates the expected segment code of a target segment before attempting to access it. If the actual value stored at the pointer address is not the expected segment code, then the IMS buffer handler issues an 85x abend. In the online environment, an 85x abend error causes a *pseudo* abend to be issued by the buffer handler. A *fatal abend* is issued in the batch IMS environment.

How a database can become corrupted

Human error is the most common cause of a corrupted database. Table 30 shows the most common ways that pointer errors are introduced into IMS databases. Although the list is not all-inclusive, the errors most commonly reported by users are explained in Table 30.

Table 30. Ways pointers get damaged

Cause of damage	When it happens
Update with wrong DBD (ACB)	JCL error in batch job
Improper recovery procedures	Missing log
Failure to use emergency restart	After cancel, abend, or power failure
Failure to run batch backout	After cancel or abend
Software errors	
Hardware errors	Unnoticed I/O error

Updating with the wrong DBD (or ACB)

The DBD is the road map that IMS uses to interpret the database block. If a database is updated with the wrong DBD, results are unpredictable—and pointer errors may occur.

This is usually the result of a JCL error. A test DBD library could be accidentally used in a production update job. Production databases might be accidentally used in a test job. This usually occurs in batch jobs.

Improper recovery procedures

Using improper recovery procedures is a common way pointers get damaged. When running the IMS Database Change Accumulation utility and the Database Recovery utility, it is important to include all of the log tapes. If a log data set is omitted from the recovery process, a corrupted database is the likely result.

When recovering a database, IMS restores segments, pointers, and free space elements. If a log is left out of a recovery attempt, then a segment could be stored in the range of an incorrectly restored free space element.

Usually, IMS reclaims the free space and creates or updates a free space element when a segment is deleted. All pointers pointing to targets that are in the free space area are set to zero. If a log is left out of a recovery attempt, then these pointers may not get set to zero.

Failure to use emergency restart

Emergency restart is an extension of IMS's normal restart process. It is initiated by a master terminal operator command whenever it is necessary to restart IMS after an IMS, z/OS, hardware, or power failure. It should also be used whenever a prior execution of the IMS system was not terminated with a successful checkpoint.

If there is a power failure, the memory contents are lost (IMS buffers, too). If the system is brought back up without emergency restart, the database is probably damaged. Database changes made by in-flight transactions do not get backed out.

If the operator cancels the IMS DB/DC or CICS/IMS DB control region, the situation is similar to that of a power failure. If the system is brought back up without emergency restart, the database is probably damaged.

Failure to run Batch Backout

If the IMS Batch Backout utility is not run when it is necessary, a corrupted database is the likely result. A batch backout may be needed after a batch job is canceled or abnormally terminates. The circumstances under which this utility should be run are described in the following:

- *IMS Version 8 Operations Guide*
- *IMS Version 9 Operations Guide*
- *IMS Version 10 Operations Guide*

Software errors

Software errors in z/OS or IMS *program*s could result in pointer errors.

Hardware errors

Hardware failures could also result in pointer errors. If the operator does not notice an I/O error message and recover it appropriately, the database could be damaged.

Various methods of repairing a corrupted database

When a database is discovered to be corrupted, you must repair it immediately. There are three basic ways:

- **Use standard IMS recovery methods**: This is the best way to repair a database. It requires much less expertise, and usually can be accomplished in less time.
- **Modify the database manually with a zapping utility**: This method requires a lot of expertise to accomplish successfully. It is highly error-prone, even when done by an experienced programmer. The person who decides where and how to zap the database must have a good knowledge of the internal formats of IMS databases.
- **Reorganize the database**: This method can be used only for certain kinds of errors involving logical pointers or counter fields.

There are times when it is impossible to repair a database with standard IMS recovery methods. In this case, the HD Pointer Checker and the IBM IMS Database Repair Facility for z/OS (also referred to as IMS Database Repair Facility) are very valuable tools for the systems programmer or database administrator.

It may also be desirable to use zapping methods when your recovery job is known to be long. You may determine that you can analyze the database errors and repair them with zaps in a shorter time than is required to recover the databases. If you have expertise in the use of IMS, this may be a cost-effective way to repair a large database.

Repairing a database by using HD Pointer Checker and zapping

This section describes how to repair a database using HD Pointer Checker and zapping. Perform the steps outlined in Figure 89.

1. Image-copy the corrupted databases.
2. Run HD Pointer Checker on the corrupted databases.
3. Obtain dumps of the invalid block.
4. Analyze the results.
5. Repair the databases.
6. Image-copy the repaired databases.
7. Run HD Pointer Checker on the image copy of the repaired databases.

Figure 89. Database repair process

Image-copying the corrupted databases

Warning: Do not, under any circumstances, omit this step.

Before beginning the repair process, it is mandatory to make back-up image copies of your databases. Back up all databases that are in any way connected with those databases that have errors. If you make a mistake during the repair process, the databases can be recovered from these image copies. This guards against inadvertent loss of more data.

Running HD Pointer Checker on the corrupted databases

Run HD Pointer Checker to find out where the databases are damaged. All databases that are in any way connected with the damaged databases must be checked with HD Pointer Checker. The recommended control statements that should be used are shown in Figure 90 on page 275.

Notice that HD Pointer Checker is run using the real databases as input (that is, DATASET=IMAGECOPY is not specified on the DATABASE statement). This ensures that the absolute disk addresses of the invalid pointers are printed on HD Pointer Checker reports. Since the databases are probably down anyway (because of the damage), this may not have an adverse effect on production. If it is appropriate, image copy input can still be used.

When INCORE=NO is specified on the OPTION statement, in-core pointer checking is not done. This is necessary even though it requires more CPU and elapsed time. Complete the results from the BLOCKMAP process, which can be achieved when TYPE=ALL is specified on the PROC statement. This is the only way to make it happen.

ERRLIMIT=NO on the OPTION statement causes a message to be printed for every error that is detected. Normally, you want to see every error message, even if there are several hundred of them. If there is an excessively large number of messages generated, limit the number that are printed by specifying ERRLIMIT=YES or not specifying the ERRLIMIT= keyword at all.

```
 PROC TYPE=ALL
 OPTION INCORE=NO,ERRLIMIT=NO
***HISAM***
 DATABASE DB=dbdname,DD=ddname,OVERFLOW=ddname
***INDEX DB (HISAM INDEX/SECONDARY INDEX***
 DATABASE DB=dbdname,DD=ddname,OVERFLOW=ddname,PRIMEDB=dbdname
***HIDAM/HDAM***
 DATABASE DB=dbdname,DD=ddname
```

Figure 90. Control statements—Database repair with HD Pointer Checker

Obtaining dumps of the invalid block

In order to debug a corrupted database, you should obtain a dump of the block containing the segment pointing to the damaged area. You should use IDCAMS to dump the block for HISAM or the index databases. For details of IDCAMS, refer to *z/OS DFSMS Access Method Services for Catalogs*.

The best way to get the dump for the HDAM or the HIDAM databases is to specify BLOCKDUMP=(rba,nnn) on the DATABASE statement. This statement provides a block dump and a block map. The dump provided by this BLOCKDUMP option is easier to use than comparable IDCAMS dumps when repairing a database error. By using this dump, you will be able to know how the block is deblocked by the BLOCKDUMP option and you can understand the error messages better.

The control statement format is shown in Figure 91.

```
PROC TYPE=SCAN
DATABASE DB=dbdname,DD=ddname,BLOCKDUMP=(rba,nnn)
```

Figure 91. Control statements—Database repair with BLOCKDUMP

Analyzing the results

Once the error messages and the dumps of all appropriate database blocks are obtained from HD Pointer Checker, do some analysis. The purposes of this analysis are the following:

- Find the real errors
- Determine the best repair method
- Rebuilding the database to correct the problem
- Investigating the cause of the problem.

Each of these is discussed below.

Finding the real errors

The HD Pointer Checker often produces more error messages than there are errors. This results because the error messages describe what HD Pointer Checker "thinks" the errors are. For example, the occurrence of a particular error may cause several other error messages. Therefore, repairing one place in the database might eliminate several error messages. Your own analysis must always be the final judge of what is wrong with the database.

Sometimes the damage is extensive enough to cause HD Pointer Checker to incorrectly deblock part of a block. When this happens, a lot of totally invalid error messages may be produced. There is no substitute for human intelligence in a situation like this.

It is possible to get error messages describing situations that are not really errors. An "orphan" segment is a good example of this. If a "segment" that is not the target of any pointer is found, an error message is printed. Since IMS can never access such a segment, this is (in some respects) not really an error. It is merely unreclaimed free space. Such an "error" disappears when the database is reorganized.

The important thing is to determine exactly where the real errors are.

Some of the HD Pointer Checker reports help you perform this task. The HISAM Segment Level Statistics report and the Segment and Pointer Statistics report show what the prefix of each segment looks like. The HD Tuning Statistics report gives the lengths of the prefix and data part of the segments. These reports are useful aids when looking at dumps of database blocks.

Error conditions reported by HD Pointer Checker should be analyzed carefully before taking any action. HD Pointer Checker can detect and report error conditions that both would and would not cause IMS abends. This occurs because HD Pointer Checker uses DFP access methods rather than IMS to find segments (targets) or validate pointers in a database. Some of the "errors" reported by HD Pointer Checker could be removed by a normal database reorganization.

Warning: A database can be corrupted, even if it cannot be made to abends.

Sometimes you can use the IMS Test Program (DFSDDLT0) to duplicate problems and verify that an existing abend condition must be corrected. This technique must be used carefully. It is very easy to construct catastrophically corrupted databases for which IMS cannot be made to abend with DFSDDLT0.

Determining the best repair method

Warning: Do not try to repair a database by "zapping" without a good knowledge of internal IMS database formats.

When the place and type of errors are detected, repair the database using the following ways:
- Applying IMS recovery utilities
- Zapping
- Reorganizing.

Each method is described in the following sections.

IMS recovery utilities: When possible, database recovery is the accepted way to correct database errors.

Zapping: "Zapping" means modifying or changing parts of a database using some means other than the standard IMS database calls. Any of the following programs is an acceptable of "zapping" the databases:
- IMS Database Repair Facility (see *IMS Database Repair Facility User's Guide*)
- AMASPZAP (see *z/OS Service Aid*)
- IMS Utility Control Facility (see *IMS Utilities Reference*)

Select "zapping" as a database repair method for one of these two reasons:

- It may be the only technique available. For example, if the image copy or log is damaged, database recovery is impossible.
- It may be the fastest way to get the databases back online. If only a few pointer error conditions must be corrected, and if qualified personnel are available to repair the database, "zapping" may provide significant savings in time.

The formats of the various IMS database records are described in *IMS Diagnosis Guide and Reference*.

Reorganization: Under certain circumstances, reorganizing the database can correct an invalid pointer condition. The following points must be considered:

- The HD Reorganization Unload utility issues unqualified GN calls to read a database. If the invalid pointer is a physical child or physical twin pointer, the GN call fails. Therefore, reorganization cannot be used.
- Logical parent or logical twin pointers are rebuilt using reorganization. However, the logical parent pointer may be followed by DL/I during unload. If the logical child segment contains the logical parent pointer (PTR=LP) and the concatenated key of the logical parent is not physically stored, DL/I follows the bad logical parent pointer to construct the concatenated key of the logical parent. This causes an abend.
- If DBR= is specified for Database Prereorganization and logical parent pointers exist in a database, the HD Reorganization Unload utility saves the old logical parent pointer. During the reload process, this pointer is placed in the type 10 work data set record. The reload or scan of the logical parent database creates type 00 records containing the old RBA of the logical parent segment. Database Prefix Resolution then sorts and matches the two records using those two fields. Bad logical parent pointers produce the error message DFS879I.
- When the user executes HD Reorganization Unload and HD Reorganization Reload and specifies DBIL= for Database Prereorganization for a logical child or logical parent database, HD Reorganization Unload saves the logical parent concatenated key instead of the logical parent pointer in the type 10 work data set record. HD Reorganization Reload or Database Scan of the logical parent database saves the logical parent concatenated key in the type 00 work data set record. Database Prefix Resolution sorts and matches the records using these fields. In this manner, the records match, and the new logical parent pointers are created.
- When DBIL= is specified for a database, it must also be specified for a database containing a logical child whose logical parent resides in the DBIL= database. Consider the example in Figure 92 on page 278. If DBIL= is specified for database DB2, also specify DBIL= for DB1. This can have a cascading effect through many databases. DBIL= means "initial load" to the utilities. By definition, if DB2 is being "initially loaded," segment B could not have previously existed. Therefore, no attempt to resolve the B-to-C relationship is made unless DBIL= is specified for DB1.

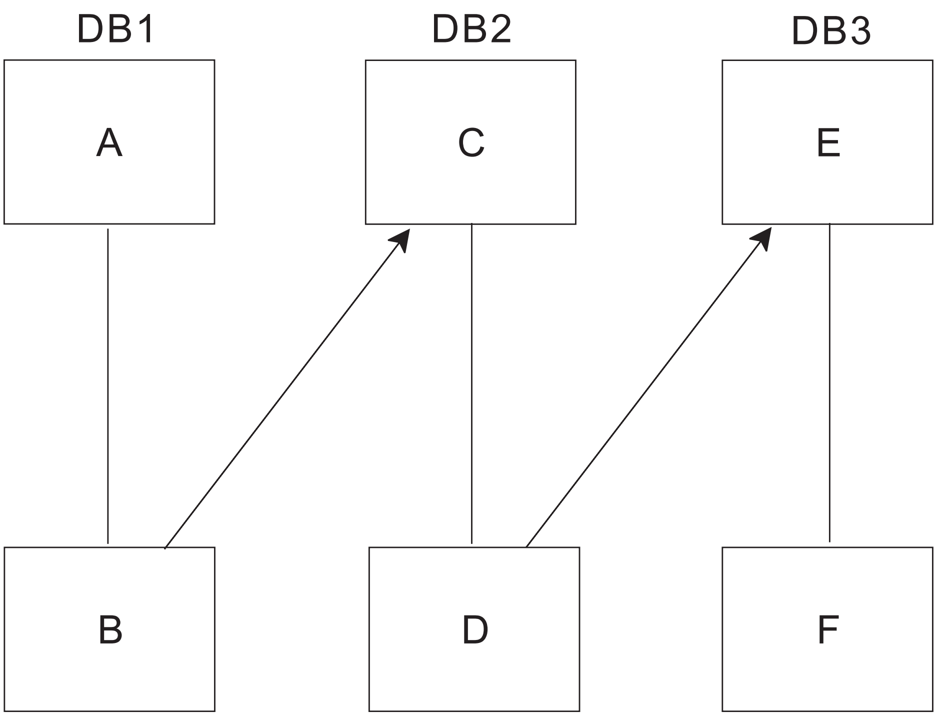


Figure 92. DBIL= Example 1

- Reorganization can also be used to reset faulty counters. It is necessary to use DBIL= for Database Prereorganization. It is mandatory that the logical parent database and *all* databases containing logical children of the logical parent be unloaded and reloaded with DBIL=. This allows Database Prefix Resolution to count all the logical children and ensures that the correct value is placed into the counter by Database Prefix Update.

  If the logical child database contains another logical child segment type whose logical parent resides in yet another database, that logical parent database must be reorganized with DBIL=.

  Reorganize all three databases in Figure 93 on page 279, specifying DBIL= for each. If DB1 or DB3 is omitted, is specified as DBR=, or is scanned, the counter is invalid.

  Reorganize all three databases in Figure 94 on page 279, specifying DBIL= for each, if a bad counter exists in either DB1 or DB3.

  Note: As a standard practice, DBR= should be specified for Database Prereorganization when reorganizing. It is the most efficient in terms of sorting during the Database Prefix Resolution step.

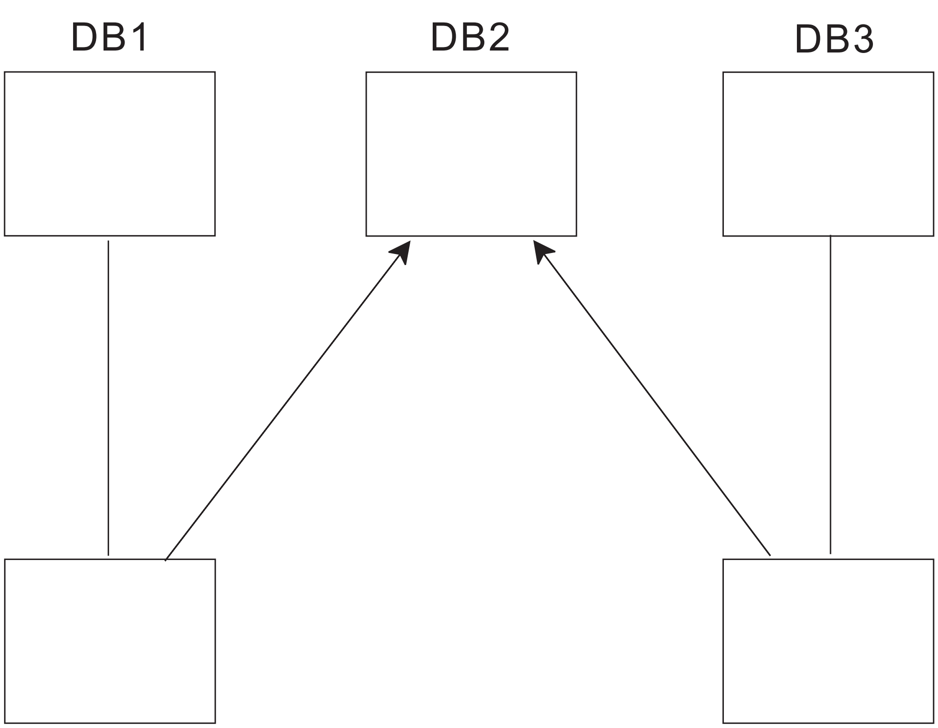


Figure 93. DBIL= Example 2

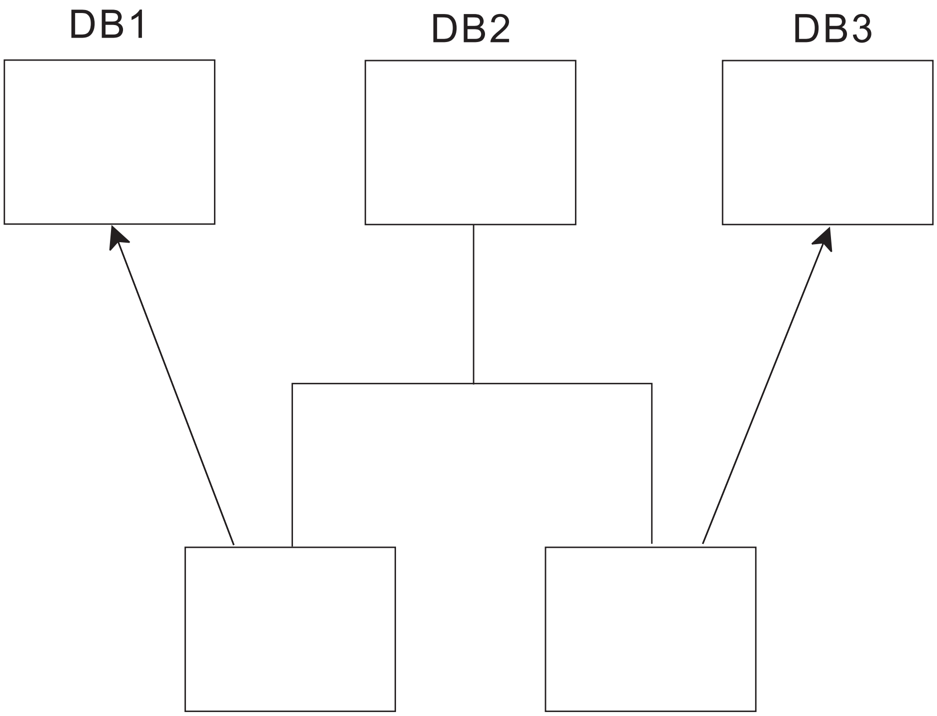


Figure 94. DBIL= Example 3

Rebuilding the databases to correct the problem

This section explains how to repair a database by zapping the HD primary databases only. If an index database is damaged, rebuild it by reorganizing the primary database or by using some other means.

A database can be repaired by using the IMS recovery utilities, or by modifying the contents of the database with a ZAP utility (such as IMS Database Repair Facility). Normally, the IMS recovery utilities should be used (rather than zapping pointers) to recover the databases. Sometimes, zapping may be required or desirable. Table 31 lists the abbreviations and full names of pointers.

Table 31. Abbreviations and full names of pointers

Pointer	Description
HF	Hierarchic Forward
HB	Hierarchic Backward
PTF	Physical Twin Forward
PTB	Physical Twin Backward
PP	Physical Parent
LTF	Logical Twin Forward
LTB	Logical Twin Backward
LP	Logical Parent
LCF	Logical Child First
LCL	Logical Child Last
PCF	Physical Child First
PCL	Physical Child Last
IN	Index
OF	Index, Overflow
SX	Secondary Index
SXO	Secondary Index, Overflow
RAP	Root Anchor Point
VLS	Variable Length Segment (Split)

ZAPPING: Pointer zapping may be necessary when the user attempts to unload a database via the IMS HD Reorganization Unload utility. The database will then end abnormally with a return code of 85x. This could happen if a physical pointer is damaged (PTF or PCF). Zapping may also be necessary when the previous image copy has been lost or destroyed, or when the log tape is missing.

Pointer zapping is desirable when the user has a very large database that may requires a long time to recover or reorganize.

The goal of zapping is to make the database structurally correct to IMS. If the pointer and the target are arbitrarily hooked to achieve this goal, delete the database record (and reinsert any lost data) when the system is backed up.

Inherent problems: Usually, database damages are concentrated to one or two database blocks. Some problems are inherent in repairing IMS databases because of the flexibility IMS offers the user in selecting the pointers.

To save DASD space, the database may have only forward pointers for dependent segments. This makes repair much more difficult than if the user selected both forward and backward twin pointers. The probability of both the forward and backward chains being broken is much less than the probability of only the forward chain being broken. To identify the parent of a dependent segment is very difficult unless the two segments are connected by a physical parent (PP) pointer.

The end of a twin chain is indicated by a PTF or LTF pointer with a value of binary zero. If you have a dangling chain, indicated by the FABP0960E error message, you could consider zapping the PTF or LTF pointer to zero and end the chain there. Zapping a pointer to zero may cause some unexpected results - especially if the segments in the chain are targets of a secondary index or a logical relationship.

How to use HD Pointer Checker: Before zapping a segment, it is necessary to understand the total effect that the zap can have. Knowing which segments point to the segment to be zapped helps design zaps that do not introduce new errors into the database. When TYPE=ALL is specified on the PROC statement as input for the PROCCTL data set, HD Pointer Checker prints a list of all segments that point to the 'invalid' segments; that is, segments identified in your HD Pointer Checker error messages. (This list can be incomplete if you run HD Pointer Checker with in-core pointer checking.) Analyze carefully each pointer that points to the segment to be zapped. Consider the effect that the zap has on all segments pointing to the about-to-be-changed segment.

Pointers that help in repair:

Physical parent/child relationship: The physical parent (PP) pointer can be used to find the physical parent of a dependent segment. The physical child first (PCF) pointer can be used to find the start of a physical twin chain. The physical child last (PCL) pointer can be used to find the end of a physical twin chain.

Logical parent/child relationship: The logical parent (LP) pointer can be used to find the logical parent of a dependent segment. The logical child first (LCF) pointer can be used to find the start of a logical twin chain. The logical child last (LCL) pointer can be used to find the end of a logical twin chain.

Since these pointers are in a parent/child relationship, if one of them is broken, you may be able to reconstruct the broken chain by starting from either the parent, or the child, depending on which pointer is valid.

Investigating the cause of the problem

You must conduct a thorough investigation to determine the exact cause of the database damage experienced. If the original source of the problem is not corrected, it will almost certainly happen again.

Usually, to find out how a database became damaged is not easy. A database damage is not noticeable until online transactions abend. To find out what happened, study the console logs for operation in previous hours.

Repairing the databases

Create and run the batch job that repairs the databases using the particular method selected. If "recovery" or "reorganization" is selected as repair technique, use standard IMS techniques. Before using a "zapping" program, you must decide what changes result in a "clean" database.

Always zap the minimum number of bytes that result in a clean database. Sometimes it is necessary to reorganize the database after the zapping to complete the repair. It is a good practice to do a reorganization after a zap, even if technically it would not be required to obtain a valid database.

The objective of the repair step is to obtain databases that are valid from an IMS standpoint. If data was lost because of either the damage or the repair method, re-create that data. Use IMS application programs or DFSDDLT0, after the repair process has been completed.

Image-copying the repaired databases

Warning: Do not, under any circumstances, omit this step.

At the end of the repair process, it is mandatory that you make back-up image copies of your databases. All databases that have been changed in any way must be backed up.

Running HD Pointer Checker on the image copy of the repaired databases

Warning: Do not, under any circumstances, omit this step.

Before bringing the databases back online, verify that the databases are valid, from an IMS standpoint. The best way to do this is to run HD Pointer Checker, using the new image copies as input. Two very important facts are verified in this way:

1. The databases are completely repaired and contain no errors.
2. The image copies are usable.

A HD Pointer Checker run should show no errors. All return codes must be zero.

Run HD Pointer Checker to ensure all the errors have been cleaned up and new ones have not been introduced. You can easily add some new damage of your own. If an error is made in zapping pointers, messages like the following may appear:

```
PTx & PTy POINT TO SAME TARGET

LTx & LTy POINT TO SAME TARGET

MORE THAN 1 PTx (TO SAME TARGET)
```

The message terms "PTx & PTy" and "LTx & LTy" represent various combinations of invalid pointers to a target.

Chapter 11. Reported by HD Pointer Checker slack bytes, unknown data, and T2 errors

This chapter provides information about slack bytes, unknown data, and T2 errors.

Topics:

- "Database format for slack bytes"
- "How IMS reclaims space" on page 284
- "Validation of free space element" on page 285
- "T2 errors" on page 285

Database format for slack bytes

Figure 95, Figure 96, Figure 97, Figure 98, Figure 99, and Figure 100 show the formats of HDAM and HIDAM control intervals. OSAM database blocks have exactly the same format, except that the RDF/CIDF fields are not present. If a HIDAM root segment has twin backward pointers, then the RAP area is not present.

FSAP	RAP area	Bitmap	RDF/CIDF

Figure 95. HDAM bitmap block format (Root addressable area)

FSAP	FSEs Segments Slack bytes	Bitmap	RDF/CIDF

Figure 96. HDAM bitmap block format (Overflow area)

FSAP	RAP area	FSEs Segments Slack bytes	RDF/CIDF

Figure 97. HDAM non-bitmap block format (Root addressable area)

FSAP	FSEs Segments Slack bytes	RDF/CIDF

Figure 98. HDAM non-bitmap block format (Overflow area)

FSAP	RAP area	Bitmap	RDF/CIDF

Figure 99. HIDAM bitmap block format

FSAP	RAP area	FSEs Segments Slack bytes	RDF/CIDF

Figure 100. HIDAM non-bitmap block format

How IMS reclaims space

This section describes Diagnosis, Modification, and Tuning Information. See "Programming interface information" on page 732 to understand the restrictions associated with this type of material.

Diagnosis, Modification, and Tuning Information

Disk space is often reclaimed by IMS so that it can be reused at a later time. This can happen in the following situations:

- IMS deletes a segment from the database.
- IMS replaces a variable-length segment, and the replacement is shorter than the original.
- IMS replaces a variable-length segment, the replacement is longer than the original, and a split segment is created.

There are two ways that IMS releases the no-longer-used disk space:

- If at least 8 bytes are released, IMS creates a free space element.
- If fewer than 8 bytes are released, IMS treats the space as slack bytes.

A free space element is subject to use by IMS at any time. Slack bytes are never reused by IMS. The only way to eliminate them is to reorganize the database.

A single DL/I delete or replace call can generate a maximum of seven slack bytes. It is possible to generate more than seven consecutive slack bytes by using several delete or replace calls. The following example illustrates how this might happen. Figure 101 on page 285 shows the status of the database after the three operations.

1. Database contains a 100-byte variable-length segment.
2. Replace the 100-byte segment with a 94-byte segment. Six slack bytes occupy the 6 bytes that follow the updated segment.
3. Replace the 94-byte segment with a 90-byte segment. Four slack bytes occupy the 4 bytes that follow the updated segment. Now a total of 10 slack bytes follow the 90-byte segment.

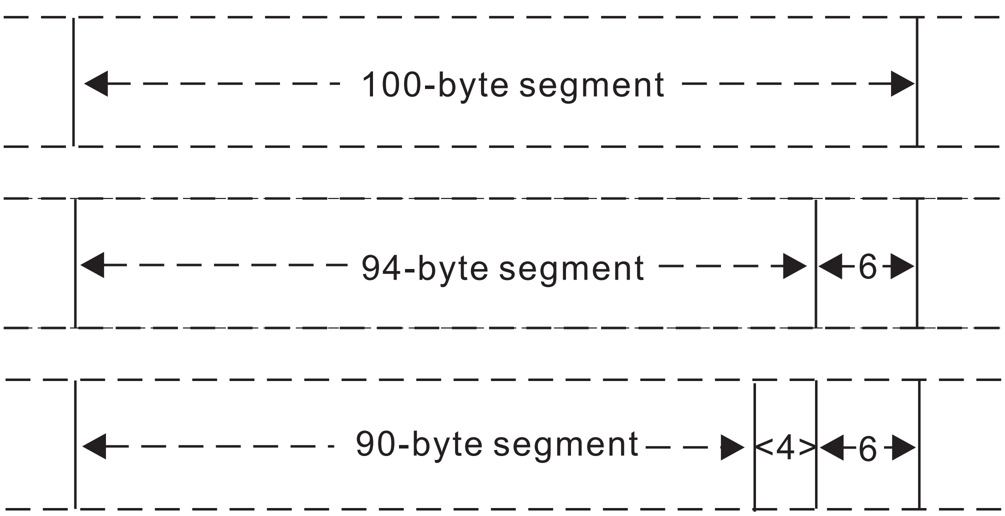


Figure 101. How more than seven slack bytes can occur

End of Diagnosis, Modification, and Tuning Information

Validation of free space element

HD Pointer Checker validates the free space element (FSE). The FSE is a prefix of a free space, and contains the length of the free space and the pointer to the next FSE address as shown in Figure 102.

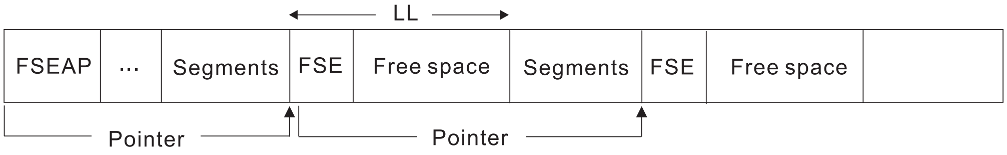


Figure 102. Structure of the free space element (FSE)

HD Pointer Checker checks whether the length and pointer information are correct or not. However, HD Pointer Checker does not check the contents of the free space following the FSE. IMS might clear the free space with X'00' or it might leave the values that are contained in the existing segment data. HD Pointer Checker, therefore, regards any data contained in the free space following the FSE as correct.

T2 errors

HD Pointer Checker prints an error message every time it detects an occurrence of more than seven slack bytes (this is the default). The above scenario shows that error message FABP0410E (also known as a T2 error) can occur as a result of normal, valid delete or replace operations. T2 errors that occur this way are *not* really errors.

T2 errors can also be the result of the following database damage:

- The wrong DBD may have been used to scan the database.

- An improper recovery attempt may have been made to recover the database.

Therefore, you must analyze an occurrence of more than seven slack bytes. However, the specification of the T2CHK option in the PROCCTL data set will allow the user to easily ignore the short and/or known T2s that are not really errors. The way to specify the T2CHK option is described in "FABPMAIN PROCCTL data set" on page 71.

The T2 information is reported by the following reports:

- Scan of HISAM Database report produced by the SCAN processor
- Validation of a Pointer to a Target at SCAN report produced by the SCAN processor
- Validation of a Pointer to a Target report produced by the CHECK processor.

If only T2 errors are detected for the database throughout HD Pointer Checker run, the return code for HD Pointer Checker run is set to 2. The T2 errors are reported in the HD Pointer Checker Summary report.

T2 processing

After HD Pointer Checker reads a database block (using either the VSAM or QSAM access method), it classifies each byte in the block. It does this during a sequential pass over the block.

Each segment or free space element is expected to be followed by another segment or free space element. If an invalid segment code is found to be following a segment or free space element, HD Pointer Checker assumes that a slack byte has been detected. HD Pointer Checker now tries to locate the next valid segment or free space element. It does this by testing every second byte for a valid segment code. (Because segments always begin on bytes with even relative byte addresses, it is sufficient to test every other byte.) Once a valid segment code (or free space element) is found, and a rudimentary validity check is satisfied, HD Pointer Checker assumes it is back on track in its deblocking scan of the database block. If the number of bytes of unknown data it detects is more than seven (the default value), an error message is printed.

It is possible that HD Pointer Checker may not get back on track correctly. A coincidence that fools the program could occur. If this happens, other error messages that are not valid may appear. In practice, such a coincidence is very rare.

T2CHK option

By using the T2CHK option, the user can easily ignore the short and/or known T2s that are not really errors in HISAM, HDAM, HIDAM, PHDAM, and PHIDAM databases. For this option, the following specifications are needed:

- The minimum value of T2 record length to be reported (for HDAM, HIDAM, PHDAM, and PHIDAM).

  This causes the T2 record whose length is shorter than the specified minimum value not to be reported. It allows the user to ignore short T2s that may not be really errors.
- The maximum number of T2 records not to be reported (for HISAM, HDAM, HIDAM, PHDAM, and PHIDAM).

By specifying T2 record threshold value for this specification, the known T2 is not reported. It allows the user to ignore known T2s that may not be really errors and will continue to be present until the database reorganization.

What should be done for T2 errors

If T2 error messages appear in your HD Pointer Checker reports, investigate them and determine their cause. Use HD Pointer Checker block maps and block dumps to verify whether the slack bytes were caused by normal update operations.

If only normal operations have caused the T2 errors, do one of the following:
- Ignore the T2 messages (because they are not actual errors).
- Reorganize the database (which eliminates all slack bytes).

If it is ascertained that the unknown data could not have occurred naturally, repair the database. Refer to Chapter 10, “Database repair guidelines,” on page 271 for instructions on how to repair databases.

Chapter 12. Estimating runtime resources

This chapter explains how to estimate runtime resources of HD Pointer Checker.

Topics:

- "Estimating the work data set size for HD Pointer Checker"
- "Estimating the storage needed for HD Pointer Checker" on page 295

Estimating the work data set size for HD Pointer Checker

This section explains how to estimate the size of the work data set for HD Pointer Checker.

When HD Pointer Checker runs with the Standard Check—that is, with no HASH Checking—several work data sets are required. In the following cases, you will need to estimate the sizes of the work data sets:

- If TYPE=ALL is specified, HD Pointer Checker allocates the work data sets dynamically. In most cases, you do not need to specify the DDs of work data sets, because HD Pointer Checker estimates the size of work data sets on the basis of the sizes of the database data sets or the DSSIZE= specifications. If there is not enough space on your disks, you will get a dynamic allocation error or a B37 abend when allocating the work data sets. In this case, you must specify the DD statements for the work data sets.
- If TYPE=SCAN is specified, HD Pointer Checker does not allocate the work data sets dynamically; you must specify the DD statements for the work data sets.
- If CHECKREC=YES is specified, CHECKREC DD is also required, but it is not allocated dynamically by HD Pointer Checker; you must specify the CHECKREC DD statement.

When HD Pointer Checker runs with HASH checking, the work data sets used are very small. Do one or two of the following:

- If TYPE=ALL is specified, HD Pointer Checker allocates the work data sets dynamically. You do not need to specify any DD statements of the work data set.
- If TYPE=SCAN is specified, you must specify the SORTEX01 DD statement. SPACE=(TRK,(1,1)) is enough.
- If CHECKREC=YES is specified, you must specify the CHECKREC DD statement. SPACE=(TRK,(1,1)) is enough.

All work data sets are sequential data sets. They can be allocated on either a tape or a disk. To run HD Pointer Checker successfully with work data sets on a disk, make sure each work data set has enough space for all of its records. Otherwise, the job will terminate abnormally.

Because an HD Pointer Checker job often runs for a relatively long time, it is important that you allocate the spaces correctly at the outset. To estimate the sizes of the spaces, follow the steps shown below. The following sections give guidelines to help you estimate the sizes of the HD Pointer Checker work data sets on the basis of these steps.

1. Run HD Pointer Checker with PTRCHK=NO.
2. Estimate the size of MERGIN*nn*.
3. If necessary, estimate the sizes of additional data sets:
   - Estimate the size of data set for the TYPE=SCAN process.

- Estimate the sizes of the data sets for the IXKEYCHK=YES process.
- Estimate the sizes of the data sets for the EPSCHK=YES process.
- Estimate the size of the data set for the CHECKREC=YES process.

4. Divide the *nn* data sets into scan groups.
5. Convert the results to proper space units.

Running HD Pointer Checker with PTRCHK=NO

Run HD Pointer Checker with PROC TYPE=SCAN and OPTION PTRCHK=NO to get some statistics reports that contain information for the estimation. For options other than TYPE=SCAN and PTRCHK=NO, specify the same values as in an actual run.

To calculate the sizes of the work data sets, fill in the worksheets 1 and 2, shown in Table 32 and Table 33, following the instructions below. To fill in the worksheets, refer to the statistics reports.

Table 32. Worksheet 1 for estimating the sizes of the work data sets

A. DB name	B. DB organization	C. Total number of segments	D. Total number of FSEs	E. Total number of internal pointers	F. Total number of external pointers	G. Total number of index target segments in indexed DB (ITS=ISS)	H. Total number of index source segments in indexed DB	I. Total number of index pointer segments in index DB	J. Total number of pointers in EPS

Table 33. Worksheet 2 for estimating the sizes of the work data sets

A. DB name	i. MERGIN*nn*	ii. SORTEX01	iii. MERGI2*nn*	iv. SORTE2*nn*	v. IXKEY	vi. SORTIL*nn*	vii. SORTOL	viii. CHECKREC

Referring to the DMB Directory report, fill in columns A and B for all databases.

Column A: DB name
Put the names of the databases in column A, referring to ″DBD NAME″ in the report.

Column B: DB organization
Put the organization types of the databases in column B, referring to ″DBD-ORG″ in the report.

For other columns, the guidelines for estimation are described in the following sections.

Estimating the size of MERGINnn

HD Pointer Checker always uses MERGIN*nn*. To calculate the value for MERGIN*nn* in worksheet 2, do as follows:

1. Fill in columns C and D for all databases other than an index database.

   Column C: Total number of segments
   Refer to ″DATABASE RECORD STATISTICS″ in the Database Statistics report. The report shows the total occurrence of segments. Put that number in column C in worksheet 1.

Column D: Total number of FSEs
Refer to ″NUMBER OF FREE SPACE ELEMENTS″ in the HD Data Set Statistics report. Because this report is printed per data set, add up the values of the FSEs of all data sets in the database. Put the sum in column D in worksheet 1.

2. Calculate MERGIN*nn* in worksheet 2 by the following formula:

```
MERGINnn = 36 x C + 22 x D Bytes
```

Note: Figures must be taken from columns C and D of the same database row in worksheet 1. The same applies for other columns in worksheet 2, which are explained in the following sections.

If you do not specify IXKEYCHK=YES, EPSCHK=YES, or CHECKREC=YES, only MERGIN*nn* is used. Go to “Dividing the nn data sets into scan groups” on page 294.

Estimating the sizes of additional data sets

This section explains how to estimate the sizes of additional data sets.

Estimating the size of the work data set for TYPE=SCAN (SORTEX01)

The SORTEX01 data set is needed in the TYPE=SCAN process. To calculate the value for SORTEX01 in worksheet 2, do as follows:

1. Fill in columns E and F in worksheet 1.

Column E: Total number of internal pointers
For an HD database (including HALDBs) or a HISAM database, refer to ″TOTAL POINTER STATISTICS″ in the Database Statistics report, and put the following in column E:

- If you specify INCORE=YES, the value of ″BEYOND″ in the row ″TOTALS (SAME DBDS)″
- If you specify INCORE=NO, the value of ″TOTAL″ under ″USED″ in the row ″TOTALS (SAME DBDS)″

For an index database, refer to ″TOTALS″ in the Scan of Index Database report, and put the following in column E:

- If column B is INDEX, the sum of ″TOTAL VALID SEGMENTS″ of INDEX and OVERFLOW
- If column B is PSINDEX, the sum of ″TOTAL VALID SEGMENTS″ of INDEX for all partitions

Column F: Total number of external pointers
Take the value of the column ″TOTAL″ in the row TOTALS(NOT SAME DBDS). The row TOTALS(NOT SAME DBDS) is next to the row ″TOTALS(SAME DBDS).″ Put the value in column F.

2. Calculate SORTEX01 in worksheet 2 by the following formula:

```
SORTEX01 = 40 x (E + F) Bytes
```

Estimating the sizes of the work data sets for IXKEYCHK=YES (MERGI2nn, SORTE2nn, and IXKEY)

This section tells how to estimate the sizes of the data sets that are needed when you specify PROC IXKEYCHK=YES and HASH=NO.

Estimating the size of MERGI2nn: Records are written in MERGI2*nn* during the scan process of TYPE=ALL or TYPE=SCAN. To calculate the value for MERGI2*nn* in worksheet 2, do as follows:

1. Fill in column G in worksheet 1.

   Column G: Total number of index target segments in indexed DB (ITS=ISS)

   Referring to the Scan of Index Database report, get the following values for each index database:

   - For primary index database, the sum of ″TOTAL VALID SEGMENTS″ of INDEX and OVERFLOW.
   - For secondary index database, do as follows if ″SOURCE SEGMENT NAME″ and ″TARGET SEGMENT NAME″ under the title ″SECONDARY INDEX DEFINITION″ are the same.
     - If column B is INDEX, the sum of ″TOTAL VALID SEGMENTS″ of INDEX and OVERFLOW.
     - If column B is PSINDEX, the sum of ″TOTAL VALID SEGMENTS″ of INDEX for all partitions.

   In this report, you will find the corresponding indexed database in ″INDEXED DATABASE=.″ Put the above values in column G of the row for the indexed database in worksheet 1. If the indexed database has two or more target segment types, add up the values for all index databases, and put the sum in column G.

   Note: Fill in column G in the row for the *indexed* database, not for the *index* database.

2. Calculate MERGI2*nn* in worksheet 2 by the following formula:

   ```
   MERGI2nn = 34 x G Bytes
   ```

Estimating the size of SORTE2nn: Records are written in SORTE2*nn* during the scan process TYPE=ALL or TYPE=SCAN. To calculate the value for SORTE2*nn* in worksheet 2, do as follows:

1. Fill in columns H and I in worksheet 1.

   Columns H and I: Total number of index source segments in indexed DB and total number of index pointer segments in index DB

   From the Scan of Index Database report, get the following values:

   - If column B is INDEX, the sum of ″TOTAL VALID SEGMENTS″ of INDEX and OVERFLOW.
   - If column B is PSINDEX, the sum of ″TOTAL VALID SEGMENTS″ of INDEX for all partitions.

   In this report, you will find the corresponding indexed database in ″INDEXED DATABASE=.″ Put the above value in column H of the row for the indexed database in worksheet 1. With the same value, also fill in column I of the row for the index database in worksheet 1.

   Note: Fill in column H in the row for the *indexed* database, not for the *index* database. If the indexed database has two or more source segment type, sum up the values of all index databases.

2. Calculate SORTE2*nn* in worksheet 2 by the following formula:

```
SORTE2nn = (56 + key length ) x H  Bytes ... in case of indexed database
SORTE2nn = (56 + key length ) x I  Bytes ... in case of index database
```

Where *key length* is the value shown in ″DB INDEX KEYLENGTH-1″ plus 1.

Estimating the size of IXKEY: IXKEY is used during the check process TYPE=ALL or TYPE=CHECK. Only one data set is allocated for it. Calculate IXKEY in worksheet 2 from the values of MERGIN2*nn* and SORTE2*nn* in worksheet 2, as follows:

```
IXKEY = (SUM of MERGI2nn) + (SUM of SORTE2nn)
```

Estimating the sizes of the work data sets for EPSCHK=YES (SORTILnn and SORTOL)

The following processes are required only for HALDBs that have logical or index relationships.

Estimating the size of SORTILnn: Records are written in SORTIL*nn* during the scan process of TYPE=ALL or TYPE=SCAN. To calculate the value for SORTIL*nn* in worksheet 2, do as follows:

1. Fill in column J in worksheet 1.

   Column J: Total number of pointers in EPS

   Put the following value in column J:
   - If column B is PSINDEX, the same number as in column I.
   - If column B is PHDAM or PHIDAM, refer to ″SEGMENT AND POINTER STATISTICS″ in the Database Statistics report. Take the values in ″TOTAL″ under ″USED″ from the ELP and the ELC rows, and add up the values for all segments in the database. Put the sum in column J.

2. Calculate SORTIL*nn* in worksheet 2 by the following formula:
   - If IXKEYCHK=YES is specified for PSINDEX:
     ```
     SORTILnn = (56 + key length) x J Bytes
     ```
     Where *key length* is the value shown in ″DB INDEX KEYLENGTH-1″ of the Scan of Index Database report plus 1.
   - If IXKEYCHK= NO is specified:
     ```
     SORTILnn = 44 x J Bytes
     ```

Estimating the size of SORTOL: SORTOL is used during the check process TYPE=ALL or TYPE=CHECK. Only one data set is allocated for it in an HD Pointer Checker job. To calculate the value for SORTOL, add all values of SORTIL*nn* in worksheet 2 as follows:

```
SORTOL = SUM of SORTILnn
```

Estimating the size of the work data set for CHECKREC=YES (CHECKREC)

Records are written in CHECKREC during the check process TYPE=ALL or TYPE=CHECK. Only one data set is allocated for it in an HD Pointer Checker job. To calculate the value for CHECKREC in worksheet 2, add all values of MERGIN*nn* and SORTEX01 in worksheet 2 as follows:

```
CHECKREC = SUM of ( MERGINnn + SORTEX01 )
```

Dividing the nn data sets into scan groups

Next, you will need to estimate the sizes of the work data sets with suffix *nn* that are used by each scan group.

The following examples show how to make these estimates.

Example 1

Assume that you specify DBA and DBB to be processed by scan group 01, and DBC to be processed by scan group 02. Also assume that each database requires MERGIN*nn* as shown in Table 34.

Table 34. Worksheet 2 for Example 1

A. DB name	i. MERGIN*nn*	ii. SORTEX01	iii. MERGI2*nn*	iv. SORTE2*nn*	v. IXKEY	vi. SORTIL*nn*	vii. SORTOL	viii. CHECKREC
DBA	1000							
DBB	2000							
DBC	5000							

You can estimate the size of MERGIN01 and MERGIN02 as follows:

- Size of MERGIN01= 1000 + 2000 = 3000 bytes
- Size of MERGIN02= 5000 bytes

Example 2

In addition to the assumptions made in Example 1, assume that DBA has two data sets, and the size of data set 1 is 1GB and that of data set 2 is 4GB. Also assume that you specify the processing of each scan group as follows:

- Scan group 01: Data set 1 of DBA, and DBB
- Scan group 02: Data set 2 of DBA, and DBC

The MERGIN*nn* for each data set of DBA requires the following size:

- For data set 1: (1000 × 1G/(1G+4G)) = 200 bytes
- For data set 2: (1000 × 4G/(1G+4G)) = 800 bytes

Then your estimates of the sizes of MERGIN01 and MERGIN02 will be as follows:

- Size of MERGIN01 = 200 + 2000 = 2200 bytes
- Size of MERGIN02 = 800 + 5000 = 5800 bytes

You can estimate the sizes of other work data sets with suffix *nn* in the same way as you did for MERGIN*nn*. For SORTEX01, which is used per scan job step, however, you need to estimate the size as follows:

- If you run all of scan processes in one job, add all SORTEX01 columns.
- If you run the scan processes in multiple job steps, divide worksheet 2 by scan jobs and calculate SORTEX01, MERGIN*nn*, and the other work data sets by each worksheet.

Converting the results to proper space units

The calculations in the preceding sections give the sizes of all the large HD Pointer Checker work data sets in bytes. In order to use them for your JCLs, you need to convert these figures to appropriate space units. If you are using a disk for your work data sets, convert them into cylinder units. If you are using a tape, convert them into tape reel units. Make sure you provide enough volume for the spaces.

Dynamic allocation

If the DD statements are not specified in the HD Pointer Checker JCL, HD Pointer Checker allocates the work data sets dynamically. The UNIT parameter of the DYNALLOC macro is SYSALLDA. The work data sets will be allocated on the DASD volume that is defined as the SYSALLDA unit.

The volume count parameter and space parameter are calculated by HD Pointer Checker on the basis of the data set size or the OPTION DSSIZE= parameter in the PROCCTL data set. If there is not enough space or enough number of volumes in the SYSALLDA unit for dynamic allocation, specify the DD statements in the HD Pointer Checker JCL.

The space information for the dynamic allocation is shown in message FABP1101I. The messages are printed in the PROCCTL Statement report.

Estimating the storage needed for HD Pointer Checker

The following formulas are for use in estimating the amount of storage required for HD Pointer Checker. The storage size depends on certain options in the PROCCTL data set.

Use these formulas for rough estimation. The storage size actually required may differ from the estimated value.

Below 16MB 2.5MB + number of scan groups × 250KB

Above 16MB

REPORT CHAINDIST=NO

For 10 or fewer scan groups.

(1) PROC HASH=YES

10MB + number of scan groups × 1.2MB

(2) PROC HASH=NO (*No* is the default value of the HASH option.)

(1) + 2.8MB

(3) PROC IXKEYCHK=YES

(2) + 2 × 2.8MB

(4) EPSCHK=YES (*Yes* is the default value of the EPSCHK option for HALDB.)

(2) + 2.8MB

For more than 10 scan groups:

Use 280K instead of 2.8MB in formulas (1) to (4).

REPORT CHAINDIST=YES (YES is the default value of the CHAINDIST option.)

When an HDAM database or PHDAM database is processed, a number of bytes equal to the number of RAPs in the database need to be added to the formulas 1 through 4. The additional area is used only while a data set that has root segments is being scanned; it disappears as soon as the scan is completed. Therefore, if you process more than two HDAMs or PHDAMs serially within a scan group, the maximum size of the additional storage is equal to the maximum number of RAPs in the databases. If you process more than two HDAMs or PHDAMs in parallel in different scan groups, the additional storage size is the sum of the number of RAPs of all databases.

The number of RAPs can be calculated by:
(Number of RAPs per block) × (Number of blocks in RAA)

Chapter 13. Improving the performance of HD Pointer Checker

This chapter presents methods for improving the performance of HD Pointer Checker. Certain options in a PROCCTL data set may affect performance. For details about each option, see "FABPMAIN PROCCTL data set" on page 71.

Topics:

Parallel scan of data set

In general, most of the elapsed time for an HD Pointer Checker job is spent on reading database data sets or image copy data sets. You can reduce the elapsed time by reading the data sets in parallel. To call for parallel reading, use the SCANGROUP option in a DATABASE statement, and specify a different scan group number for each data set.

HASH Check

A HASH Check is a quick and rough pointer-checking function that considerably improves the elapsed time. To run the HASH Check, specify HASH=YES in a PROC statement. It can determine only whether a database contains errors; it cannot identify any broken pointer or segment. We recommend that you run the HASH Check in regular operations, and run the Standard Check only when errors are detected in a HASH Check.

The default is NO.

INCORE Check

An INCORE Check is specified on the INCORE option in an OPTION statement. When it is specified, pointers that point near segments are checked in a SCAN process. Generally, this option may considerable reduce the elapsed time by reducing CPU time of CHECK. However, the increase in the SCAN CPU time is sometimes though rarely, greater than the reduction of CPU time of CHECK, and it makes the elapsed time longer.

The default is YES.

Other options that affect performance

The three options just discussed can yield a relatively large improvement in performance. Those following may have less effect.

Options that may improve performance when the default is accepted:

PROC CHECKREC, IXKEYCHK

OPTION KEYSIN

REPORT COMPFACT

Options that may improve performance when NO is specified (the default for each of these is YES):

```
PROC EPSCHK, VLSSUMM
OPTION SPIXCHK, HOMECHK, CHAINDIST
```

With the IBUFF option in an option statement, in general, the default value gives the best performance. The best value, however depends on the system.

If you have enough real storage, you can improve the performance of the Standard Check with PROC TYPE=ALL by using VIO for files MERGIN*nn*. This greatly speeds the CHECK process. You request VIO by supplying DD statements for files MERGIN*nn*. By default, HD Pointer Checker allocates MERGIN*nn* on disk.

Chapter 14. HD Pointer Checker options for debugging

The options described in this chapter are usually used for debugging purpose only. If these options are specified, you may get an extremely large report, or some pointer errors undetected.

Topics:

- “PROC statement”
- “OPTION statement” on page 300

PROC statement

This section explains the PROC statement options.

CHECK=

Specifies the certain kinds of pointers to be turned off. It is also used to cause all input records to be printed. If this option is not supplied, it results in complete pointer checking without generating any unnecessary reports. It is the *recommended* way of running the CHECK process.

This option can be specified when TYPE=ALL or CHECK is specified.

ALL

This option specifies all input records (as well as all error messages) to be printed. If this option is specified, you will get an extremely large report.

(CHK,*nnnnnn*)

This option specifies *certain* kinds of pointer checking to be turned off by the CHECK processor. You can control the pointer checking with the 6-digit *nnnnnn* part:

Position **Description**

1 This control byte indicates whether or not you want to check HIDAM index pointers. Use one of the following codes:

1 HIDAM/PHIDAM index pointers are checked. Use this selection to run the HD Pointer Checker. This is the default value.

0 HIDAM/PHIDAM index pointers are not checked.

2 This control byte indicates whether or not you want to check physical pointers. Use one of the following codes:

1 Physical pointers are checked. Use this selection to run the HD Pointer Checker. This is the default value.

0 Physical pointers are not checked.

3 This control byte indicates whether or not you want to check logical pointers. Use one of the following codes:

1 Logical pointers are checked. This is the *recommended* way to run the HD Pointer Checker. This is the default value.

0 Logical pointers are not checked.

4 This control byte indicates whether you want to check the counter field versus the number of logical child segments. Use one of the following codes:

1 You want to check the counter field versus the number of logical child segments. Use this selection to run the HD Pointer Checker. This is the default value.

0 You do not want to check the counter field versus the number of logical child segments.

5 This control byte indicates whether you want to check physically-paired segments. Use one of the following codes:

1 Physically paired segments are checked. Use this selection to run the HD Pointer Checker. This is the default value.

0 Physically paired segments are not checked. **This control byte should not be turned off.**

6 This control byte indicates whether or not you want to print all 24 hash formulas, regardless of whether they apply to the segment type or not.

1 No printing is generated. This is the default value.

0 Printing is generated.

OPTION statement

This section explains the OPTION statement.

DIAG=

Specifies whether you want to print dumps of some internal control blocks. This option can be specified when TYPE=ALL or SCAN is specified.

This option must be used for debugging purpose only.

YES

The dumps of some internal control blocks are printed.

NO

Any dump of the internal control blocks is not printed. This is the default.

PRINTDATA=

Specifies whether you want to print the pointer data that is extracted by the program. If you specify YES, you may get an extremely large report that will be of little use.

Abbreviations **PDATA** and **PD** can be used for **PRINTDATA**. This option can be specified when TYPE=ALL or SCAN is specified.

This option must be used for debugging purpose only.

YES

The extracted pointer data is printed.

NO

Any extracted pointer data is not printed. This is the default.

NOCHKP=

Specifies whether you want to bypass the checking of certain kinds of pointers. This option can be specified for HDAM and HIDAM databases and can be specified when TYPE=ALL or SCAN is specified.

This option must be used for debugging purpose only. When this option is specified, HISTORY=YES cannot be specified.

HF (or PHF)
This option specifies that hierarchical forward pointers are not checked.

HB (or PHB)
This option specifies that hierarchical backward pointers are not checked.

PTF
This option specifies that physical twin forward pointers are not checked.

PTB
This option specifies that physical twin backward pointers are not checked.

PP
This option specifies that physical parent pointers are not checked.

LTF
This option specifies that logical twin forward pointers are not checked.

LTB
This option specifies that logical twin backward pointers are not checked.

LP
This option specifies that logical parent pointers are not checked.

LC
This option specifies that logical child pointers are not checked.

PC
This option specifies that physical child pointers are not checked.

RAP
This option specifies that root anchor point (RAP) pointers are not checked.

VLS
This option specifies that variable-length split pointers are not checked.

When this option is not specified, that is the default, HD Pointer Checker checks all applicable pointers. The **NOCHKP=** field in the separator page for DB/DSG reports contains blank.

Chapter 15. Tailoring HD Pointer Checker for IMS Database Recovery Facility

IMS Database Recovery Facility (IMS DRF) Version 3 Release 1 can call the HD Pointer Checker HASH Check function.

This chapter describes the steps necessary to run the HASH Check function under the IMS Database Recovery Facility environment.

Related reading: For additional information to run the HASH Check function in IMS Database Recovery Facility, see *IMS Database Recovery Facility for z/OS User's Guide and Reference* (SC18-9407).

Topics:

- "SHPSLMD0 APF authorization"
- "FABPATH0 procedure"

SHPSLMD0 APF authorization

In the IMS Database Recovery Facility environment, the HD Pointer Checker program runs as an authorized program facility (APF) program.

To invoke the HASH Check function in IMS Database Recovery Facility, the IMS HP Pointer Checker load module library (SHPSLMD0) must be APF-authorized.

FABPATH0 procedure

When IMS Database Recovery Facility (IMS DRF) calls the HASH Check function, HD Pointer Checker starts a FABPATH0 address space. The FABPATH0 builds environmental setting for the HASH pointer checking. A FABPATH0 procedure must exist in one of the libraries in the SYS1.PROCLIB concatenation. The member name must be FABPATH0.

Figure 103 shows a sample for the FABPATH0 procedure.

```
//FABPATH0 PROC  HPPCLIB1=,HPPCLIB2=
//ATH0PROC EXEC  PGM=FABPAUTH
//STEPLIB  DD DISP=SHR,DSN=&HPPCLIB1
//         DD DISP=SHR,DSN=HALDB partition selection exit
//         DD DISP=SHR,DSN=&HPPCLIB2
//SYSUDUMP DD SYSOUT=*
```

Figure 103. FABPATH0 JCL procedure

EXEC statement

This is a required statement. Specify the EXEC statement as follows:

```
EXEC PGM=FABPAUTH
```

STEPLIB DD

This is a required statement. Specify two DD statements as follows:

```
//STEPLIB  DD DISP=SHR,DSN=&HPPCLIB1
//         DD DISP=SHR,DSN=&HPPCLIB2
```

&HPPCLIB1 is an IMS HP Pointer Checker load module library.
&HPPCLIB2 is an IMS RESLIB.

Do not remove or override &HPPCLIB1 and &HPPCLIB2. HD Pointer Checker obtains the above two data set names from the IMS Database Recovery Facility master address space JCL and passes the data set names to the &HPPCLIB1 and &HPPCLIB2 parameters. The data sets must be catalogued.

If you process a HALDB, and the HALDB uses a partition selection exit routine, FABPATH0 refers to the partition selection exit module. It is required that you do either of the following tasks:

- Store the partition selection exit module in the IMS RESLIB.
- Add a DD statement to the STEPLIB concatenation, and specify a data set name that contains the partition selection exit module.

SYSUDUMP

This optional output data set defines the output from a system ABEND dump routine. It is used only when a dump is required. Though optional, it is recommended that you include this data set.

In addition, FABPATH0 refers to the IMS and RECON*x* DD statements. These DD statements are basically not required in the FABAPTH0 procedure JCL. HD Pointer Checker obtains the data set names from the IMS Database Recovery Facility master address space JCL. The data sets must be catalogued.

The contents of the IMS and RECON DDs are follows:

- IMS DD: A data set that contains the DBD load modules.
- RECON1, RECON2, and RECON3 DDs: RECON data sets

If the IMS and RECON*x* DD statements are specified in the FABAPTH0 JCL, the data set names in the FABPATH0 JCL are used.

Part 3. HD Tuning Aid

Chapter 16. Overview of HD Tuning Aid

This chapter explains the overview of the HD Tuning Aid utility.

Topics:

Program functions

The HD Tuning Aid utility produces the following reports that describe the distribution of root segments in HDAM, HIDAM, PHDAM, or PHIDAM databases:

- The Actual Roots per Block report prints the actual number of roots that are stored in each database block.
- The Assigned Roots per Block report prints the number of roots that are randomized to each block.
- The Assigned Roots per RAP report prints the number of roots that are randomized to each root anchor point (RAP).

The last two of these reports are produced only for HDAM or PHDAM.

HD Tuning Aid also produces a report giving summary information about a HALDB.

- HALDB Process Summary report prints summary information about the processed partitions.

Typical uses

Typical uses of HD Tuning Aid are evaluations of the following for performance and tuning analysis:

- Current randomizer performance in HDAM or PHDAM databases
- Current root segment locations in HIDAM or PHIDAM databases
- Potential randomizer performance in HDAM or PHDAM databases
- Potential randomizer performance in HIDAM or PHIDAM databases that could be converted to HDAM or PHDAM

HD Tuning Aid can estimate the actual performance of partition selection for PHDAM or PHIDAM and randomizing parameter for PHDAM.

It can also estimate the performance of the following types of conversion:

- Conversion from an HDAM, HIDAM, or PHIDAM database to a PHDAM database
- Changing the number of partitions of a PHDAM database
- Changing the partition selection and/or the DBD randomizing parameters
- Conversion from a PHDAM or a PHIDAM database to an HDAM database

When the database to be processed is a HALDB, or when a simulation of conversion to a HALDB is processed, HD Tuning Aid generates reports for each partition in addition to the ordinary reports by database.

Program structure

HD Tuning Aid consists of two programs plus a DFSORT execution:

- FABTROOT produces the Actual Roots per Block report. It also creates the RAPSIN data set that is used as input to DFSORT.
- DFSORT is used to sort the RAPSIN data set.
- FABTRAPS produces the Assigned Roots per RAP report and the Assigned Roots per Block report.

If a HALDB is processed, FABTROOT program need run under the IMS batch region controller. FABTROOT program must be run under the IMS batch region controller, if a user randomizing routine requires to access IMS control blocks for non-HALDB. Otherwise, both programs (FABTROOT and FABTRAPS) run as a standard batch job.

HD Tuning Aid can run with multiple IMS versions/releases without reinstalling the product, as far as the version/release is supported.

Data flow

Figure 104 on page 309 shows the HD Tuning Aid data flow for a non-HALDB. The HD Tuning Aid reads control statements from the CTL data set, root keys from a data set that are created by scan process of HD Pointer Checker, and information about processing databases from the DBD/PSB library of IMS. The process consists of three JOB steps. Some of the reports are generated by the first step. To create the rest of the reports, the first step writes data into a work data set, which is passed to DFSORT utilities to be sorted and fed to the last step.

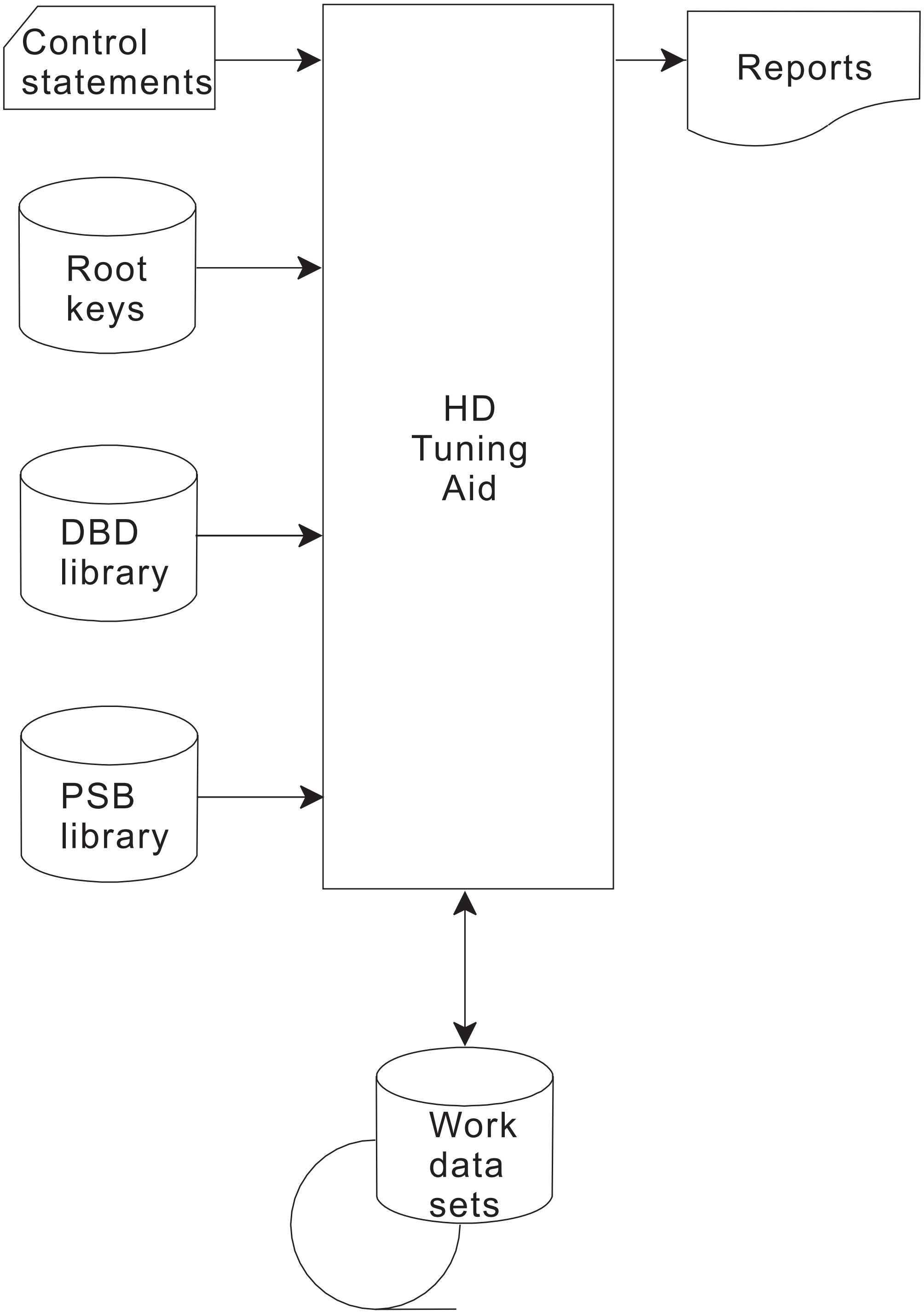


Figure 104. HD Tuning Aid data flow for non-HALDB

Figure 105 on page 310 shows the HD Tuning Aid data flow for a HALDB. In case of a HALDB, the HD Tuning Aid gets access to the RECON data sets and IMS RESLIB to retrieve information about the HALDB.

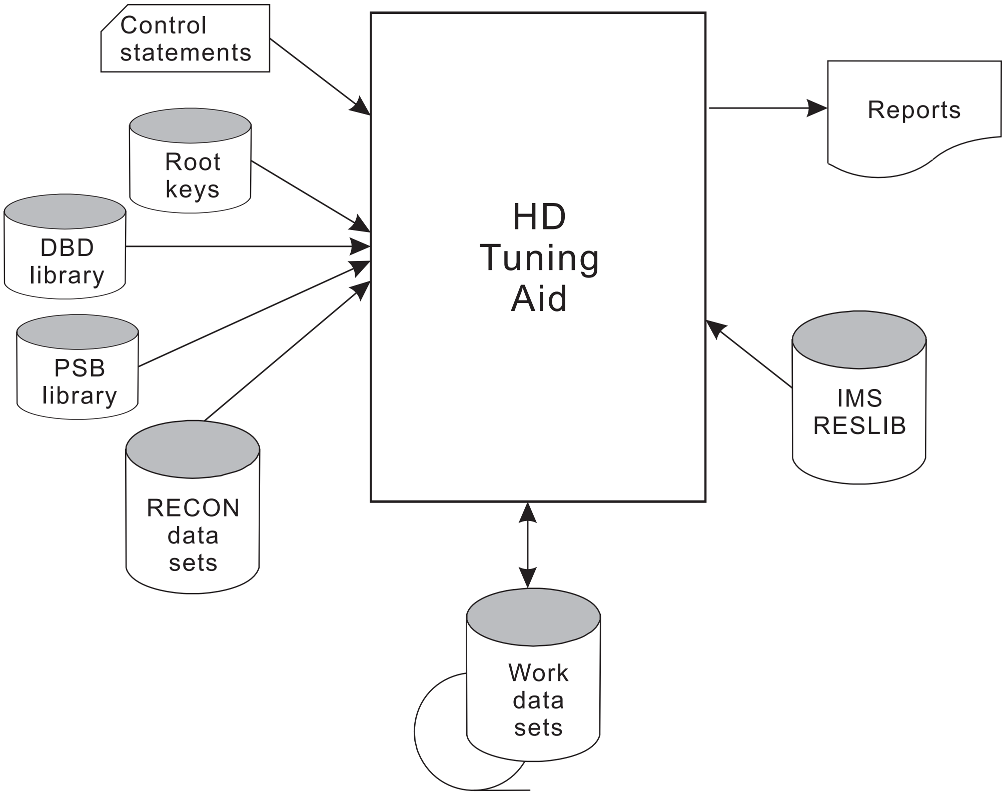


Figure 105. HD Tuning Aid data flow for HALDB

Restrictions and considerations

HD Tuning Aid is applicable only to HDAM, HIDAM, PHDAM, or PHIDAM databases.

Your randomizing routine may have its own restrictions that can affect the way you must run HD Tuning Aid. It must be capable of processing the key values that you provide as input. If your randomizer refers to IMS control blocks, you are required to run HD Tuning Aid under the IMS batch region controller.

To use HALDB, users will need to keep to following points in mind:

- If you use multiple KEYSIN data sets created by multiple HD Pointer Checker runs, you must make sure that all the KEYSIN data sets you specify on the KEYSIN DD statement are concatenated in ascending order of the database number and partition ID.
- HD Tuning Aid can accept all or some of the partitions of a PHDAM or PHIDAM database as input. If you want to run an HD Tuning Aid job for particular partitions, first run an HD Pointer Checker against those partitions to create a desired KEYSIN data set.
- When you process the KEYSIN data set of PHIDAM or PHDAM, or simulate conversion to PHDAM, you must run the HD Tuning Aid job under the IMS batch environment and specify DBRC=Y for PARM=(DLI,FABTROOT,&PSB,,,,,,,,,,Y,N)

of DFSRRC00. Also, RECON data sets are required. If the RECON data sets are not defined as dynamic allocation data sets, you must specify them in RECONx DD of step DFSRRC00.

- If the input PHIDAM or PHDAM database uses a partition selection exit, you must put the load module in the STEPLIB data set. If you intend to simulate the process that uses highkey without using the partition selection exit, you must put the load module in the STEPLIB data set. If you want to simulate the process using another partition selection exit, you must put the partition selection exit in the IMS2 data set.

Chapter 17. Operating instructions for HD Tuning Aid

In order to use HD Tuning Aid, you must complete the following steps:

- Code the JCL for the three HD Tuning Aid and DFSORT steps.
- Code the input data for DFSORT and one of the HD Tuning Aid programs.
- Make a test run.
- Interpret the output reports to verify that the process has completed successfully.
- Place the run JCL (and input data) into production use.

Once the production JCL is created and stored, HD Tuning Aid can easily be run by using the stored JCL.

DFSORT Application Programming Guide describes how to use the DFSORT utility. This section describes how to use the HD Tuning Aid program.

To use HD Tuning Aid, you must run three programs. They may be run in either a single job or in several jobs. However, it is usually much easier to run all the steps in one job.

A typical job stream executes both the HD Pointer Checker and HD Tuning Aid. The HD Tuning Aid part of such a job contains the following steps:

1. **FABTROOT:** This program prints the Actual Roots per Block report and creates sort records used to create other reports.
2. **DFSORT:** This program product sorts all of the sort records from the previous job step.
3. **FABTRAPS:** This HD Tuning Aid program prints the Assigned Roots per Rap report and the Assigned Roots per Block report.

Topics:

- "Job control language"
- "Input" on page 322
- "Output" on page 329

Job control language

There are two ways to run HD Tuning Aid:

- As a standard (non-IMS) batch job
- Under the IMS batch region controller

The two procedures need different JCL. HALDB can be processed in the IMS batch region controller.

FABTROOT JCL

To run FABTROOT, supply an EXEC statement PARM and the appropriate DD statements. Table 35 summarizes the DD statements that are appropriate for all FABTROOT runs. Table 36 on page 314 summarizes the extra DD statements that you need if you run FABTROOT under IMS. Actual JCL requirements are as follows:

Table 35. FABTROOT DD statements for all runs

DDNAME	Use	Format	Need
IMS	Input	PDS	Required

Table 35. FABTROOT DD statements for all runs (continued)

DDNAME	Use	Format	Need
IMS2	Input	PDS	Required
KEYSIN	Input		Required
CTL	Input	LRECL=80	Optional
PR8	Output	LRECL=133	Required
RAPSIN	Output	LRECL=42	Required
STEPLIB	Input	PDS	Required
SYSUDUMP	Output	SYSOUT	Optional

Table 36. Additional FABTROOT DD statements for IMS runs

DDNAME	Use	Format	Need
PROCLIB	Input	PDS	Optional
IEFRDER	Not used	DUMMY	Required
DFSRESLB	Input	PDS	Required
DFSVSAMP	Input		Required
PR10	Output	LRECL=133	Optional
SYSPRINT	Output	SYSOUT	Required
RECONx	Input	Recon data set	Optional

EXEC

If you are running this program as a standard MVS™ batch program, code as follow:

```
//      EXEC PGM=FABTROOT,PARM=NOIMS
```

If you are running this program as an IMS batch program, this statement must be in the following form:

```
//     EXEC PGM=DFSRRC00,
//     PARM='DLI,FABTROOT,psbname,,,,,,,,,,,dbrc,N'
//     imsplex'
```

If HALDB is processing, you must run the program as a batch program.

The PARM parameter has the same format as that used in the DLIBATCH procedure. The parameters shown need to be coded. The *psbname* must define a LANG=ASSEM PSB. It must refer (either directly or indirectly) to all input databases. If HALDB is processing, *dbrc* must be Y.

IMS DD

This required input data set is a library (partitioned data set) that contains your PSB and DBD load modules. It must contain all DBDs that are referenced (either directly or indirectly) by your PSB. If your PSB and DBDs are not in the same library, all appropriate libraries must be concatenated.

IMS2 DD

This required input data set is a library (partitioned data set) containing the randomizing modules. If the partition selection exit name that are specified in CTL DD statements, IMS2 data set must contain the partition selection exit module defined in CTL.

PR8 DD

This required output data set contains the reports produced by module FABTROOT. If BLKSIZE is coded on the DD statement, it must be a multiple of 133.

PR10 DD

This data set is optional and applicable only for HALDB. This optional output data set contains reports produced by FABTROOT. If BLKSIZE is coded on the DD statement, it must be a multiple of 133.

KEYSIN DD

This required input data set contains root segment keys that are written by the SCAN process of HD Pointer Checker.

RAPSIN DD

This required output data set contains HDAM block/RAP assignments for all keys that are in the KEYSIN data set. BLKSIZE, if coded, must be a multiple of 42.

CTL DD

This optional input data set contains the values of DBD parameters and or partition selections that you want to override. This optional input data set contains your overrides of certain DBD parameters. Use this data set whenever you want to perform iterative analysis of randomizer performance. It describes the databases that are processed. It also contains optional user's requests for printing the reports.

RECONx DD

These optional data sets are the RECON data set. They are required when the input DBDs are HALDB and when RECON data sets are not allocated dynamically.

STEPLIB DD

This optional input data set is a library (partitioned data set) containing HD Tuning Aid modules. If a partition selection exit is defined to your input PHDAM or PHIDAM database in the RECON data sets, STEPLIB data set must contain the partition selection exit module.

SYSUDUMP DD

This defines output from a system abend dump routine. It is used only when a dump is required. Although optional, it is highly recommended that you include this data set.

PROCLIB DD

This statement defines the library that contains all IMS-generated, cataloged procedures and jobs. This statement is applicable only if you are running this program as an IMS batch program, in which case it is optional. If it is omitted, a warning message may be issued by IMS. You may ignore this warning message.

IEFRDER DD

This statement defines the primary system log data set. This statement is applicable only if you are running this program as an IMS batch program, in which case it is required and should be coded as DUMMY.

DFSRESLB DD

This statement defines the data set that contains the IMS load modules. This statement is applicable only if you are running this program as an IMS batch program, in which case it is required.

DFSVSAMP DD
This input data set contains the buffer information required by the DL/I buffer handler. This statement is applicable only if you are running this program as an IMS batch program, in which case it is required.

SYSPRINT DD
This output data set contains messages produced by IMS. Since the HD Tuning Aid's IMS activity consists only of a GSCD call, no data is usually written to the SYSPRINT data set. This statement is applicable only if you are running this program as an IMS batch program, in which case it is required.

DFSORT JCL

To run DFSORT, supply the appropriate DD statements. Table 37 summarizes the DD statements. Actual JCL requirements are as follows:

Table 37. DFSORT DD statements

DDNAME	Use	Format	Need
SORTIN	Input	LRECL=42	Required
SYSIN	Input		Required
SORTOUT	Output	LRECL=42	Required
SYSOUT	Output	SYSOUT	Required
SYSUDUMP	Output	SYSOUT	Optional
SORTWK01	Work data set		Required
SORTWK02	Work data set		Required
SORTWK03	Work data set		Required
SORTWK04	Work data set		Required
SORTWK05	Work data set		Required
SORTWK06	Work data set		Required

EXEC
This statement must be in the following form:

```
//      EXEC PGM=SORT
```

SORTIN DD
This input data set is the RAPSIN data set created by FABTROOT.

SYSIN DD
This input data set contains DFSORT control statements. The sort parameter is "SORT FIELDS=(1,8,BI,A)."

SORTOUT DD
This output data set contains the sorted records. LRECL must be 42, and BLKSIZE must be a multiple of 42.

SYSOUT DD
This output data set contains the messages produced by DFSORT.

SYSUDUMP DD (or SYSABEND)
This defines output from a system abend dump routine. It is used only for debugging, when a dump is required.

SORTWK*nn* DD
These are intermediate storage data sets used by DFSORT. See *DFSORT Application Programming Guide* for more information on how to code SORTWKnn DD statements.

FABTRAPS JCL

To run FABTRAPS, supply the appropriate DD statements. Table 38 summarizes the DD statements that are appropriate for all FABTRAPS runs. Actual JCL requirements are as follows:

Table 38. FABTRAPS DD statements

DDNAME	Use	Format	Need
KEYSOUT	Input	LRECL=42	Required
PR9	Output	LRECL=133	Required
PR9X	Output	LRECL=133	Required
SYSUDUMP	Output	SYSOUT	Optional

EXEC

The EXEC statement must be in the following form:

```
//     EXEC PGM=FABTRAPS
```

KEYSOUT DD

This required input data set contains the root keys that are sorted.

PR9 DD

This required output data set contains reports produced by module FABTRAPS. If BLKSIZE is coded on the DD statement, it must be a multiple of 133.

PR9X DD

This required output data set contains reports produced by module FABTRAPS. If BLKSIZE is coded on the DD statement, it must be a multiple of 133.

SYSUDUMP DD

This defines output from a system abend dump routine. It is used only when a dump is required. Although optional, it is highly recommended that you include this data set.

JCL procedures

To run HD Tuning Aid, use the IBM-supplied, cataloged procedures shown in Figure 5 on page 56, Figure 8 on page 65, Figure 106, and Figure 107 on page 320, or prepare a similar procedure of your own. A procedure like the one shown in Figure 5 on page 56 is recommended, because it runs both HD Pointer Checker and HD Tuning Aid at the same time. To run only HD Tuning Aid, use a procedure similar to that shown in Figure 106 or Figure 107 on page 320.

For the examples provided in Chapter 18, “JCL examples for HD Tuning Aid,” on page 345, assume that the IBM-supplied, cataloged procedures are used.

FABTMVS procedure cannot be used for HALDB. Only FABTIMS or FABPPTA can be used for HALDB.

```
//*******************************************************************  00010000
//*   Licensed Materials - Property of IBM                           *  00020000
//*                                                                  *  00030000
//*   5655-K53                                                       *  00040000
//*                                                                  *  00050000
//*   (c) Copyright IBM Corp. 2000, 2006 All Rights Reserved.        *  00060000
//*                                                                  *  00070000
//*   US Government Users Restricted Rights - Use,                   *  00080000
//*   duplication or disclosure restricted by GSA ADP                *  00090000
//*   Schedule Contract with IBM Corp.                               *  00100000
//*                                                                  *  00110000
//*******************************************************************  00120000
//       PROC U=SYSDA,                 UNIT FOR WORK DATA SETS          00130000
//            CYL='1,1',              SPACE FOR WORK DATA SETS          00140000
//            PARM2=,                          PARM FOR DFSORT          00150000
//            PRTBLK=6118,  (133*46) BLKSIZE OF PRINT DATA SETS         00160000
//            TEMPBLK=8400,  (42*200) BLKSIZE OF WORK DATA SETS         00170000
//            KEYSIN=,           DSN OF INPUT FILE OF ROOT KEYS         00180000
//            DBDLIB='IMSVS.DBDLIB',                <<--------<         00190000
//            USERLIB='IMSVS.RESLIB',     <<--< USER RANDOMIZER         00200000
//            RESLIB='IMSVS.RESLIB',                <<--------<         00210000
//            DBTSRC='HPS.HPSSAMP',                 <<--------<         00220000
//            DBTLIB='HPS.SHPSLMD0'                 <<--------<         00230000
//*----------------------------------------------------------------     00240000
//* HD TUNING AID - BATCH MVS                                           00250000
//*    USE THIS PROCEDURE IF THE RANDOMIZER DOES NOT NEED               00260000
//*    ACCESS TO IMS CONTROL BLOCKS.  IBM MODULES DFSHDC10,             00270000
//*    DFSHDC20, DFSHDC30, AND DFSHDC40 USE THIS PROCEDURE.             00280000
//*----------------------------------------------------------------     00290000
//STEP1   EXEC PGM=FABTROOT,PARM=NOIMS                                  00300000
//STEPLIB   DD DSN=&DBTLIB,DISP=SHR                                     00310000
//RAPSIN    DD DSN=&&RAPSIN,DISP=(,PASS,DELETE),                        00320000
//            UNIT=&U,SPACE=(CYL,(&CYL)),                               00330000
//            DCB=BLKSIZE=&TEMPBLK                                      00340000
//IMS       DD DSN=&DBDLIB,DISP=SHR                                     00350000
//IMS2      DD DSN=&RESLIB,DISP=SHR                                     00360000
//          DD DSN=&USERLIB,DISP=SHR                                    00370000
//PR8       DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK                             00380000
//KEYSIN    DD DSN=&KEYSIN,DISP=OLD                                     00390000
//SYSUDUMP  DD SYSOUT=A                                                 00400000
```

Figure 106. HD Tuning Aid JCL procedure for MVS batch (FABTMVS) (Part 1 of 2)

```
//*-------------------------------------------------------------        00410000
//STEP2   EXEC PGM=SORT,PARM='&PARM2',COND=(0,LT,STEP1)                 00420000
//SORTIN    DD DSN=*.STEP1.RAPSIN,DISP=(OLD,DELETE,DELETE),             00430000
//             DCB=(LRECL=42,BLKSIZE=&TEMPBLK,RECFM=FB)                 00440000
//SORTOUT   DD DSN=&&KEYSOUT,DISP=(,PASS,DELETE),                       00450000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                              00460000
//             DCB=(LRECL=42,BLKSIZE=&TEMPBLK,RECFM=FB)                 00470000
//SORTWK01  DD UNIT=&U,SPACE=(CYL,(&CYL))                               00480000
//SORTWK02  DD UNIT=&U,SPACE=(CYL,(&CYL))                               00490000
//SORTWK03  DD UNIT=&U,SPACE=(CYL,(&CYL))                               00500000
//SORTWK04  DD UNIT=&U,SPACE=(CYL,(&CYL))                               00510000
//SORTWK05  DD UNIT=&U,SPACE=(CYL,(&CYL))                               00520000
//SORTWK06  DD UNIT=&U,SPACE=(CYL,(&CYL))                               00530000
//SYSOUT    DD SYSOUT=A                                                 00540000
//SYSIN     DD DSN=&DBTSRC(FABPSORT),DISP=SHR                           00550000
//*-------------------------------------------------------------        00560000
//STEP3   EXEC PGM=FABTRAPS,COND=(0,LT,STEP2)                           00570000
//STEPLIB   DD DSN=&DBTLIB,DISP=SHR                                     00580000
//PR9       DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK                             00590000
//PR9X      DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK                             00600000
//KEYSOUT   DD DSN=*.STEP2.SORTOUT,DISP=(OLD,DELETE,DELETE),            00610000
//             DCB=(LRECL=42,BLKSIZE=&TEMPBLK,RECFM=FB)                 00620000
//SYSUDUMP  DD SYSOUT=A                                                 00630000
//*-------------------------------------------------------------        00640000
```

Figure 106. HD Tuning Aid JCL procedure for MVS batch (FABTMVS) (Part 2 of 2)

```
//********************************************************************* 00010000
//*   Licensed Materials - Property of IBM                            * 00020000
//*                                                                   * 00030000
//*   5655-K53                                                        * 00040000
//*                                                                   * 00050000
//*   (c) Copyright IBM Corp. 2000, 2006 All Rights Reserved.         * 00060000
//*                                                                   * 00070000
//*   US Government Users Restricted Rights - Use,                    * 00080000
//*   duplication or disclosure restricted by GSA ADP                 * 00090000
//*   Schedule Contract with IBM Corp.                                * 00100000
//*                                                                   * 00110000
//********************************************************************* 00120000
//        PROC PSB=,                                    PSB NAME        00130000
//             DBRC=N,                          DBRC=Y OR DBRC=N        00140000
//             U=SYSDA,                  UNIT FOR WORK DATA SETS        00150000
//             CYL='1,1',               SPACE FOR WORK DATA SETS        00160000
//             PARM2=,                           PARM FOR DFSORT        00170000
//             PRTBLK=6118,  (133*46) BLKSIZE OF PRINT DATA SETS        00180000
//             TEMPBLK=8400,  (42*200) BLKSIZE OF WORK DATA SETS        00190000
//             KEYSIN=,           DSN OF INPUT FILE OF ROOT KEYS        00200000
//             DBDLIB='IMSVS.DBDLIB',                <<--------<        00210000
//             PSBLIB='IMSVS.PSBLIB',                <<--------<        00220000
//             USERLIB='IMSVS.RESLIB',     <<--< USER RANDOMIZER        00230000
//             RESLIB='IMSVS.RESLIB',                <<--------<        00240000
//             DBTSRC='HPS.HPSSAMP',                 <<--------<        00250000
//             DBTLIB='HPS.SHPSLMD0'                 <<--------<        00260000
//*-------------------------------------------------------------        00270000
//* HD TUNING AID - UNDER IMS                                           00280000
//*    USE THIS PROCEDURE ONLY IF THE RANDOMIZER NEEDS                  00290000
//*    ACCESS TO IMS CONTROL BLOCKS.                                    00300000
//*-------------------------------------------------------------        00310000
//STEP1   EXEC PGM=DFSRRC00,                                            00320000
//             PARM='DLI,FABTROOT,&PSB,,,,,,,,,,&DBRC,N',               00330000
//             REGION=1000K,TIME=(,30)                                  00340000
//STEPLIB   DD DSN=&DBTLIB,DISP=SHR                                     00350000
//          DD DSN=&RESLIB,DISP=SHR                                     00360000
//SYSUDUMP  DD SYSOUT=A                                                 00370000
//RAPSIN    DD DSN=&&RAPSIN,DISP=(,PASS,DELETE),                        00380000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                              00390000
//             DCB=BLKSIZE=&TEMPBLK                                     00400000
//IMS       DD DSN=&DBDLIB,DISP=SHR                                     00410000
//          DD DSN=&PSBLIB,DISP=SHR                                     00420000
//IMS2      DD DSN=&RESLIB,DISP=SHR                                     00430000
//          DD DSN=&USERLIB,DISP=SHR                                    00440000
//*         DD DSN=USER.IMS.RANLIB,DISP=SHR                             00450000
//IEFRDER   DD DUMMY,DCB=BLKSIZE=1408                                   00460000
//DFSRESLB  DD DSN=&RESLIB,DISP=SHR                                     00470000
//DFSVSAMP  DD DSN=&DBTSRC(FABPVSAM),DISP=SHR                           00480000
//PR8       DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK                             00490000
//PR10      DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK                             00500000
//KEYSIN    DD DSN=&KEYSIN,DISP=OLD                                     00510000
```

Figure 107. HD Tuning Aid JCL procedure under IMS (FABTIMS) (Part 1 of 2)

```
//*-------------------------------------------------------------        00520000
//STEP2   EXEC PGM=SORT,PARM='&PARM2',COND=(0,LT,STEP1)                 00530000
//SORTIN    DD DSN=*.STEP1.RAPSIN,DISP=(OLD,DELETE,DELETE),             00540000
//             DCB=(LRECL=42,BLKSIZE=&TEMPBLK,RECFM=FB)                 00550000
//SORTOUT   DD DSN=&&KEYSOUT,DISP=(,PASS,DELETE),                       00560000
//             UNIT=&U,SPACE=(CYL,(&CYL)),                              00570000
//             DCB=(LRECL=42,BLKSIZE=&TEMPBLK,RECFM=FB)                 00580000
//SORTWK01  DD UNIT=&U,SPACE=(CYL,(&CYL))                               00590000
//SORTWK02  DD UNIT=&U,SPACE=(CYL,(&CYL))                               00600000
//SORTWK03  DD UNIT=&U,SPACE=(CYL,(&CYL))                               00610000
//SORTWK04  DD UNIT=&U,SPACE=(CYL,(&CYL))                               00620000
//SORTWK05  DD UNIT=&U,SPACE=(CYL,(&CYL))                               00630000
//SORTWK06  DD UNIT=&U,SPACE=(CYL,(&CYL))                               00640000
//SYSOUT    DD SYSOUT=A                                                 00650000
//SYSIN     DD DSN=&DBTSRC(FABPSORT),DISP=SHR                           00660000
//*-------------------------------------------------------------        00670000
//STEP3   EXEC PGM=FABTRAPS,COND=(0,LT,STEP2)                           00680000
//STEPLIB   DD DSN=&DBTLIB,DISP=SHR                                     00690000
//SYSUDUMP  DD SYSOUT=A                                                 00700000
//PR9       DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK                             00710000
//PR9X      DD SYSOUT=A,DCB=BLKSIZE=&PRTBLK                             00720000
//KEYSOUT   DD DSN=*.STEP2.SORTOUT,DISP=(OLD,DELETE,DELETE),            00730000
//             DCB=(LRECL=42,BLKSIZE=&TEMPBLK,RECFM=FB)                 00740000
//*-------------------------------------------------------------        00750000
```

Figure 107. HD Tuning Aid JCL procedure under IMS (FABTIMS) (Part 2 of 2)

Input

This section describes all the input that you must specify to run HD Tuning Aid. This includes the various control statement data sets (CTL DD or SYSIN DD) and the EXEC statement parameters.

FABTROOT EXEC statement PARM

For a description of the EXEC statement PARM, see “FABTROOT JCL” on page 313.

FABTROOT CTL data set

This section explains the FABTROOT CTL data set.

Function

The CTL data set contains your description of the processing to be done by the HD Tuning Aid. It describes the databases that will be processed, and it contains optional user’s requests.

Use the CTL data set when you are analyzing potential DBD changes. The CTL data set is used to override certain DBD parameters. This enables you to analyze the effects of changing some of the randomization parameters without actually reorganizing your database. If control statements are supplied, only the Actual Roots per Block report is not produced.

You also use the CTL data set when you are analyzing potential partition selection changes for HALDB. This enables you to analyze the effects of changing partition selection without actually reorganizing your database.

Format

This control data set usually resides in the input stream. However, it can be defined as a sequential data set or as a member of a partitioned data set. It must contain one 80-byte fixed-length record for each database that you want to process. BLKSIZE, if coded, must be a multiple of 80. The order of the control statements is not significant; they can be specified in any order. The CTL data set can be coded as shown in Figure 108:

```
//STEP1.CTL DD *
SD144P    DFSHDC40 0001000  004  08192  001  007  03000  HDAM
/*
```

Figure 108. Format of the FABTROOT CTL data set

The CTL statement can include three statements: %OPTION, DB, and PART.

The %OPTION statement specifies the run-time option for this utility. The DB statement and the PART statement override certain DBD parameters. The DB statement is specified for each database. The maximum number you can specify is 100. The PART statement must be coded after each DB statement.

For the HALDB processing, you must specify the DB statement, and optionally PART statements.

The DB statement gives information that affect all partitions of HALDB. The PART statement gives individual information on each partition. The specification by the PART statement overrides the specification of the DD statement.

%OPTION Statement: This optional statement specifies the run-time option of this utility. If it is specified, it must be the first statement in the CTL data set.

```
         1         2         3         4         5
12345678901234567890123456789012345678901234567890123456789012345678
-----------------------------------------------------------
%OPTION x
```

Position | **Description**

1 - 7
The keyword %OPTION is required.

8
blank

9
Optional. Specify the type of codes to be returned when the KEYSIN data set is empty, by one of the following:

I Code 0 is returned with message FABT3522I.

W Code 4 is returned with message FABT3522W.

E Code 8 is returned with message FABT3522E. This is the default value.

blank The same as E.

10 - 72
blanks

DB statement: Line 1: This statement can be specified for both HALDB and non-HALDB. It gives the values of changing randomizing parameters for a database, and controls reports to be printed.

```
         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
--------------------------------------------------------------------------------
dbdname kpmod      rbn      anch blksz  kst  kln  bytes  HDAM PARTINI--S
                                                         PHDAMPARTDELPSH
                                                              AUTOINI
```

Position | **Description**

1
Required. Left-justified. The name of the DBD for the database to be processed.

9
Optional. This field can contain either of the following two codes:

1 The keys for this database are skipped.

blank The keys for this database are processed. This is the default value.

10
Optional. This field can contain either of the following two codes:

1 Discontinue reading the control statements. Any control statements that follow this control statement are not effective.

blank Continue reading the control statements. This is the default value.

11 Optional. 1 to 8 characters, left-justified. The name of a user-supplied randomizing module. This field is used to override the (default) DBD value.

20 Optional. Used to override the (default) DBD value. It must contain the maximum relative block number that you want to allow a randomizing module to produce for this database. This value determines the number of control intervals or blocks in the root addressable area. It must be a 7-digit, unsigned decimal integer (with leading zeros if needed).[1]

29 Optional. Used to override the (default) DBD value. It contains the number of RAPs desired in each control interval or block in the root addressable area. It must be a 3-digit, unsigned decimal integer (with leading zeros if needed).[1]

34 Optional. Used to override the (default) DBD value. It contains the database block size (or control interval size). It must be a 5-digit, unsigned decimal integer (with leading zeros if needed).[1]

41 Optional. Used to override the (default) DBD value. This value must be the starting position of the key field in bytes relative to the beginning of the segment. It must be a 3-digit, unsigned decimal integer (with leading zeros if needed). The starting position for the first byte of a fixed-length segment is 001.

46 Optional. Used to override the (default) DBD value. This value must be the length of the root segment's key field in bytes. It must be a 3-digit, unsigned decimal integer (with leading zeros if needed).

51 Optional. Used to override the (default) DBD value. This must be the maximum number of bytes of a database record that can be stored in the root addressable area. It also must be continuous, and not interrupted by calls to another database record. It must be a 5-digit, unsigned decimal integer (with leading zeros if needed).[1]

58 Optional. This field can contain one of the following three kinds of strings:

HDAM This entry indicates that the input DBD is not HDAM. This optional field is used to override the database organization that is specified in DBD. If the input DBD is not HDAM, but must be simulated as HDAM, this field is required.

PHDAM This entry indicates that the input DBD is not PHDAM. This optional field is used to override the database organization that is specified in the DBD. If the input DBD is not PHDAM, but must be simulated as PHDAM, this field is required. The PARTINI or AUTOINI is also Required at column 63.

blank No override of the database organization. This is a default value.

63 Optional. This is effective when the database simulates as PHDAM. This field can contain one of the following four kinds of strings:

[1] The override value applies to all partitions except those otherwise specified in the PART statement.

PARTINI This entry indicates that each partition definition must be followed and simulated as a PHDAM database. The partition definitions using PART statements are needed for all partitions.

PARTDEL This entry indicates that the input DBD is PHDAM or PHIDAM and simulated as an HDAM database. No following PART statement is needed.

AUTOINI This entry indicates HD Tuning Aid to assign the partition names automatically, and to simulate a PHDAM database. Only partition high keys or partition selection strings are required in the PART statement.[2]

blank Blank means there is to be no partition deletion or partition initialization. This is the default.

70 Optional. This is effective when the database simulates as PHDAM. This field can contain one of the following three codes:

- This option requests that the "DBD Parameters and Overrides Reports" not be printed.

P This option requests that the "DBD Parameters and Overrides Reports" be printed with partition override information.

blank This option requests that the "DBD Parameters and Overrides Reports" be printed without the partition information. This is the default value.

71 Optional. This statement is effect when the database simulates as PHDAM. This field can contain one of the following three CODES:

- This option requests that the "HALDB Process Summary Report" not be printed.

S This option requests that the "HALDB Process Summary Report" be printed with all partition information.

blank This option requests that "HALDB Process Summary Report" be printed with any root key assigned partition information. This is the default value.

72 Optional. This statement is effective when the database simulates as PHDAM. If the input DBD is not PHDAM, S or H is required. This field can contain one of three codes:

S This option requests that the partition selection exit module name be changed or that the partition selection method be replaced with the partition selection exit. The name of the new partition selection exit module has to be defined on the second line of the DB statement.

H This option requests that the method for partition selection be replaced with the high key method.

[2] If 'AUTOINI' is specified, HD Tuning Aid assigns the partition names automatically. The naming rule is that the first three characters are the same as the first three characters of the database name and the last four characters are assigned numbers from 0001 to 1001, in ascending order.

blank This option requests that the method for partition selection not be replaced.

Note: If HDAM or PHDAM is in column 58, all DBD parameters must be specified in the DB statement or the PART statement.

DB statement: Line 2: This statement is effective when the database simulates a PHDAM.

```
         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
--------------------------------------------------------------------------------
          pselexnm
```

Position **Description**

11 Optional. 1 to 8 characters, left-justified. The name of a user-supplied partition selection exit module—It is used to override the value in the RECON data set. If "S" is specified in column 72 of the preceding DB statement, this field is required.

PART statement for HALDB: Line 1: This statement is effective when the database simulates as PHDAM.

```
         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
--------------------------------------------------------------------------------
 partnm   mod      rbn      anch blksz            bytes  ADD         - +
                                                         DEL
```

This statement specifies the DBD parameters for a certain partition.

If "PARTINI" is specified on the DB statement, this statement is required. If "AUTOINI" is specified on the DB statement, it is not required.

Position **Description**

2 Required if this line is coded. 1 to 7 characters, left-justified. The name of the partition to be processed. If you want to simulate change of parameters or deletion of a partition, specify the name of the partition that actually exists. If you want to simulate conversion of a database to a PHDAM or add a new partition to a PHDAM, do not specify a name of a partition that actually exists.

11 Optional. 1 to 8 characters, left-justified. The name of a user-supplied randomizing module for this partition. This field is used to override the (default) DBD value.

20 Optional. This field is used to override the (default) DBD value. It gives the maximum relative block number that you want to allow a randomizing module to produce. This value determines the number of control intervals or blocks in the root addressable area of a partition. It must be a 7-digit, unsigned decimal integer (with leading zeros if needed).

29 Optional. This field is used to override the (default) DBD value. It contains the number of RAPs desired in each control interval or block in the root addressable area of a partition. It must be a 3-digit, unsigned decimal integer (with leading zeros if needed).

34 Optional. This field is used to override the (default) DBD value. This field contains the database block size (or the control interval size). It must be a 5-digit, unsigned decimal integer (with leading zeros if needed).

51
: Optional. This field is used to override the (default) DBD value. This must be the maximum number of bytes of a database record that can be stored in the root addressable area by continuous ISRT calls not interrupted by calls to another database record. It must be a 5-digit, unsigned decimal integer (with leading zeros if needed).

58
: Optional. This field can contain one of the following three kinds of strings:

  ADD This optional field requests that a new partition be added to the PHDAM database. [3]

  If ADD is specified, the following is needed:
  - All DBD parameters must be specified by the DB statement or the PART statement.
  - The partition selection string or high key must be specified by the PART statement line 2 and after.

  DEL This optional field requests that this partition be deleted from the PHDAM database.

  blank This means to change the randomizing or partitioning parameter of the existence partition. This is the default value.

70
: Optional. This option is ignored when "-" is specified in column 70 in the DB statement. This field can contain one of the following two codes:

  - Don’t print the "DBD Parameters and Overrides Reports" partition information for this partition.

  blank Print the partition "DBD Parameters and Overrides Reports" information for this partition. This is the default value.

72
: Optional. This field can contain one of following two codes:

  + This option indicates that the partition selection string or the partition high key is coded on the next line.

  blank This option indicates that the partition selection string or the partition high key is not coded on the next line.

PART statement for HALDB: Line 2 and after: This statement is effective when the database simulates as PHDAM.

```
PART Statement for HALDB: line 2 and after
         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
--------------------------------------------------------------------------------
 X'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'       +
 C'ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc'+
```

These statements give the partition selection string or the high key.

[3] HD Tuning Aid assigns a partition ID to an additional partition continuously. But, the partition IDs are not continuous if the partition deletion operations have been issued. The assigned partition ids by HD Tuning Aid may not be as same as the actual partition IDs assigned by IMS.

If "AUTOINI" or "PARTINI" is specified on the DB statement, or "ADD" is specified on the PART statement line 1, these statements are required.

The column 72 of PART statement line 1 is required if these statements exists.

Position **Description**

2 Required if this line is coded. Can contain one of the following:

X Specify strings as hexadecimal code.

C Specify string as character.

3

’ Required if this line is coded. Start a string.

4 Required if this line is coded. Specify a partition high key or a partition selection string. The string can be as long as up to column 70 in each line. If a string exceeds that length, it can be split and continued on the next line. Then every line of the string must start with C’ or X’, and end with an apostrophe ’.

72 Optional. This field is coded as follows:

+ This option indicates that the partition selection string or the partition high key is continued to the next line.

blank This option indicates that the partition selection string or the partition high key ends on this line.

Considerations about HALDB: When the input database is HALDB and a partition selection exit is used, do the following:

1. Store the partition selection exit routine defined to the RECON data set in a STEPLIB data set. Store the partition selection exit routine specified by the CTL statement in an IMS2 data set.
2. The original partition selection exit in the STEPLIB is invoked in the following sequence:
   a. invoked with the INIT parameter.
   b. invoked with the FIRST parameter.
   c. invoked with the NEXT parameter.
3. If the partition selection exit or the partition selection string are specified in the CTL statement, the original partition selection exit is invoked with the TERM parameter. Then the partition selection exit in the IMS2 is invoked.

DFSORT SYSIN data set

This section explains the DFSORT SYSIN data set.

Function

The SYSIN data set contains DFSORT program control statements. They define the type of sort operation to be performed and the sort control fields to be used.

Format

This control data set usually is defined as a sequential data set or as a member of a partitioned data set; however, it can also reside in the input stream. It usually contains 80-byte, fixed-length records. This data set must not be defined as RECFM=U.

The SYSIN data set can be coded as shown in Figure 109. Two control statements (SORT and END) are required.

```
//SYSIN     DD *
 SORT   FIELDS=(1,8,BI,A),FILSZ=En
 END
/*
```

Figure 109. Format of the DFSORT SYSIN data set

SORT control statement

The SORT statement is required. It describes the control fields (in the input records) on which the program will sort.

- The FIELDS=(1,8,BI,A) operand is required. It must be coded as shown in Figure 109.
- The FILSZ=E*n* operand is optional. It helps optimize DFSORT main storage and intermediate storage allocation/use. *n* is the estimated number of records to be sorted.

END control statement

The END statement causes DFSORT to discontinue reading the SYSIN data set. It is optional.

For more information about the DFSORT program, refer to *DFSORT Application Programming Guide*.

Output

The HD Tuning Aid output consists of some printed reports and a work data set. These reports are contained on three data sets:

- PR8—produced by FABTROOT
- PR9—produced by FABTRAPS
- PR9X—produced by FABTRAPS
- PR10—produced by FABTROOT

The following is a work data set:

- RAPSIN—produced by FABTROOT

FABTROOT PR8 data set

This section explains the FABTROOT PR8 data set.

Function

This data set contains all reports produced by module FABTROOT:

- Control Card Format report
- Control Card report
- DBD Parameters and Overrides report
- Actual Roots per Block report

Format

This data set contains 133-byte, fixed-length records, and block size (if coded in your JCL) must be a multiple of 133. Code your DD statement as follows:

```
//PR8       DD SYSOUT=A
```

Control Card Format report

The report shown in Figure 110 describes the fields on the records in the CTL data set. It is printed for the convenience of the user.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - HDTA              "CONTROL CARD FORMAT REPORT"                           PAGE:     1
5655-K53                                                   DATE: 07/10/2006  TIME: 15.21.05                         FABTROOT - V2.R2

>>> DB STATEMENT  **** FIRST LINE ****                             >>> PART STATEMENT **** FOR HALDB : FIRST LINE ****

COLUMNS  1- 8: DBD NAME OF KEY FILE TO BE RANDOMIZED               *** FOLLOWING FIELDS OPT - USED TO OVERRIDE DEFAULT DB DEFINITION

*** FOLLOWING FIELDS OPT - USED TO OVERRIDE DEFAULT DB DEFINITION  COLUMNS  2- 9: PARTITION NAME
                                                                   COLUMNS 58-60: ADD     - ADD THIS PARTITION
COLUMN      9: 1     - IF THIS KEY FILE IS TO BE SKIPPED                          DEL     - DELETE THIS PARTITION
               BLANK - PROCESS THIS KEY FILE (DEFAULT)             COLUMN     72: +       - USE OR CHANGE PARTITION HIGH KEY OR
COLUMN     10: 1     - IF STOP AFTER THIS KEY FILE PROCESSED                                PARTITION SELECTION STRING
               BLANK - CONTINUE PROCESSING KEY FILE (DEFAULT)
COLUMNS 11-18: HDAM ALGORITHM NAME                                 *** FOLLOWING FIELD OPT - USED TO REQUEST REPORT DISPLAY
COLUMNS 20-26: NUMBER OF BLOCKS IN ROOT ADDRESSABLE AREA
COLUMNS 29-31: NUMBER OF RAPS PER BLOCK                            COLUMN     70: FOR DBD PARAMETERS AND OVERRIDES REPORT
COLUMNS 34-38: BLOCK SIZE                                                         -       - DO NOT PRINT FOR THIS PARTITION
COLUMNS 41-43: STARTING POSITION IN SEGMENT (POS. 1 = 001)                        BLANK   - PRINT FOR THIS PARTITION
COLUMNS 46-48: KEY LENGTH
COLUMNS 51-55: BYTE LIMIT                                          *** LEADING ZEROES REQUIRED IN ALL NUMERIC FIELDS
COLUMNS 58-62: HDAM    - HIDAM, PHDAM, PHIDAM CONVERT TO HDAM
               PHDAM   - HDAM, HIDAM, PHIDAM CONVERT TO PHDAM      *** FIELDS STARTING IN COLUMNS 11, 20, 29, 34 AND 51 IS SAME AS
COLUMNS 63-69: PARTINI - HDAM, HIDAM CONVERT TO PARTITIONED HDAM       DB STATEMENT
                           OR PHDAM
               PARTDEL - PARTITIONED DB OR HALDB CONVERT TO HDAM
               AUTOINI - HDAM, HIDAM CONVERT TO PHDAM USING        >>> PART STATEMENT **** FOR HALDB : SECOND LINE ****
                           AUTOMATIC PARTITION DEFINITION
COLUMN     72: S       - USE OR CHANGE PARTITION SELECTION EXIT    *** FOLLOWING FIELDS OPT - USED TO OVERRIDE DEFAULT DB DEFINITION
               H       - USE OR CHANGE PARTITION HIGH KEY
                                                                   COLUMNS  2-71: PARTITION HIGH KEY OR PARTITION SELECTION STRING
*** FOLLOWING FIELDS OPT - USED TO REQUEST REPORT DISPLAY          COLUMN     72: +        - CONTINUE SPECIFIED VALUES TO NEXT LINE

COLUMN     70: FOR DBD PARAMETERS AND OVERRIDES REPORT
               -     - DO NOT PRINT                                >>> PART STATEMENT **** FOR PARTITIONED DB ****
               BLANK - PRINT FOR WHOLE DATABASE (DEFAULT)
               P     - PRINT FOR WHOLE DATABASE AND EACH           *** FOLLOWING FIELDS OPT - USED TO OVERRIDE DEFAULT DB DEFINITION
                         PARTITIONS
COLUMN     71: FOR HALDB PROCESS SUMMARY REPORT                    COLUMNS 11-18: PARTITION DDNAME
               -     - DO NOT PRINT                                COLUMNS 20-26: NUMBER OF BLOCKS IN ROOT ADDRESSABLE AREA FOR
               BLANK - PRINT FOR EACH PARTITIONS RANDOMIZED                       THIS PARTITION
                       SOME DATA (DEFAULT)                         COLUMNS 34-38: BLOCK SIZE FOR THIS PARTITION
               S     - PRINT FOR EACH PARTITIONS

*** LEADING ZEROES REQUIRED IN ALL NUMERIC FIELDS

*** FIELDS STARTING IN COLUMNS 20, 29, 34, 41, 46
    AND 51 MUST NOT BE ZERO IF GIVEN

*** IF INPUT KEY FILE IS HIDAM OR PHIDAM, 58-62 = 'HDAM' OR
    'PHDAM' AND GIVE ** ALL ** OTHER PARAMETERS, NO DEFAULTS
    FROM HIDAM OR PHIDAM DBDS ARE USED

>>> DB STATEMENT **** SECOND LINE ****

*** FOLLOWING FIELDS OPT - USED TO OVERRIDE DEFAULT DB DEFINITION

COLUMNS 11-18: PARTITION SELECTION EXIT NAME
```

Figure 110. FABTROOT—Control Card Format report

Control Card report

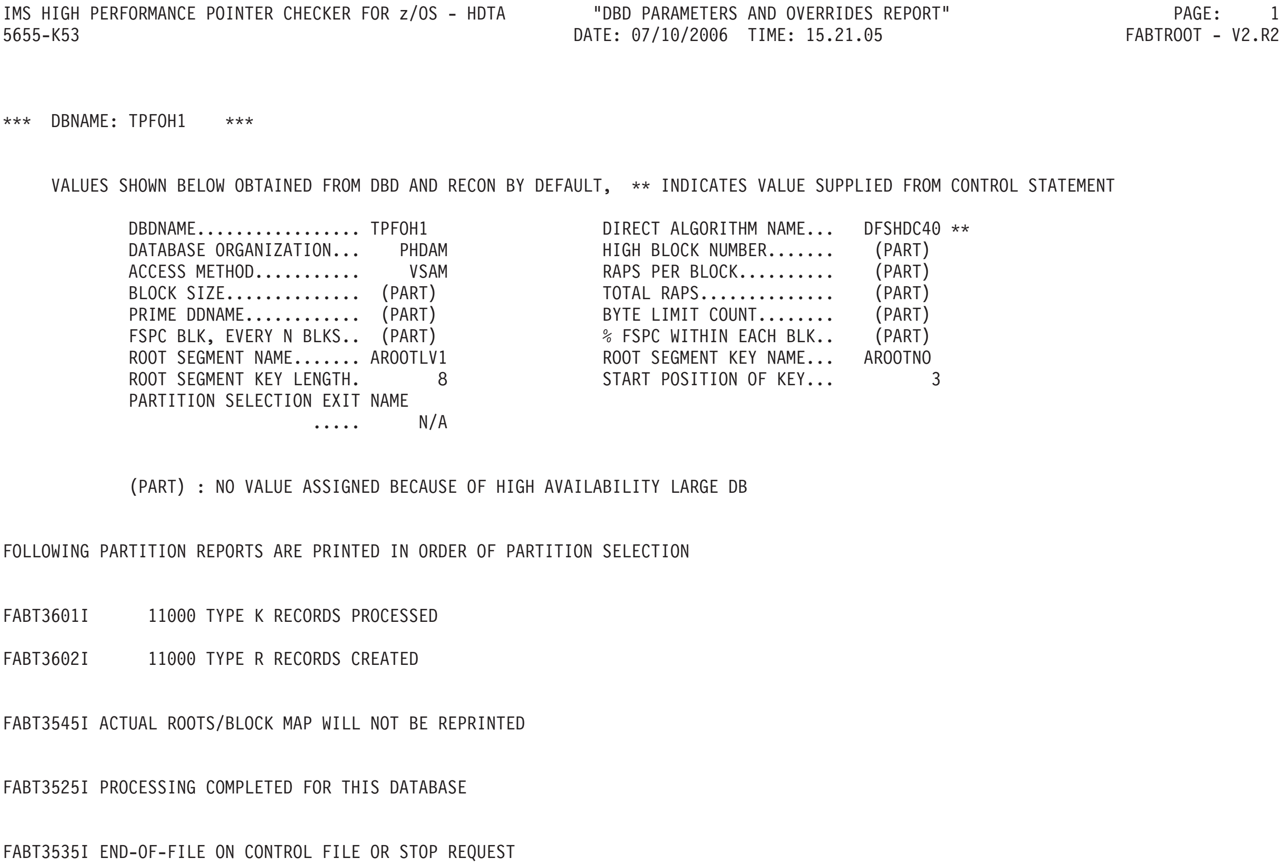
The report shown in Figure 111 contains a printed copy of the control statement that the user provided in the CTL data set.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - HDTA                 "CONTROL CARD REPORT"                               PAGE:     1
5655-K53                                                   DATE: 07/10/2006  TIME: 15.21.05                         FABTROOT - V2.R2


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

T0902001 DFSHDC40 0000002 002 16384 001 008 00150 PHDAMAUTOINIPSH
 C'00000001'
```

Figure 111. FABTROOT—Control Card report

DBD Parameters and Overrides report

The report shown in Figure 112 contains a printed copy of the DBD parameters that were used in the HD Tuning Aid job. Any parameters that were overridden by the user's control statement are flagged.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - HDTA         "DBD PARAMETERS AND OVERRIDES REPORT"                  PAGE:     1
5655-K53                                                    DATE: 07/10/2006  TIME: 15.21.05                 FABTROOT - V2.R2


***  DBNAME: TPFOH1    ***


     VALUES SHOWN BELOW OBTAINED FROM DBD AND RECON BY DEFAULT,  ** INDICATES VALUE SUPPLIED FROM CONTROL STATEMENT

             DBDNAME................. TPFOH1                DIRECT ALGORITHM NAME...   DFSHDC40 **
             DATABASE ORGANIZATION...    PHDAM              HIGH BLOCK NUMBER.......    (PART)
             ACCESS METHOD...........     VSAM              RAPS PER BLOCK..........    (PART)
             BLOCK SIZE..............  (PART)               TOTAL RAPS..............    (PART)
             PRIME DDNAME............  (PART)               BYTE LIMIT COUNT........    (PART)
             FSPC BLK, EVERY N BLKS..  (PART)               % FSPC WITHIN EACH BLK..    (PART)
             ROOT SEGMENT NAME....... AROOTLV1              ROOT SEGMENT KEY NAME...   AROOTNO
             ROOT SEGMENT KEY LENGTH.        8              START POSITION OF KEY...          3
             PARTITION SELECTION EXIT NAME
                                 .....     N/A


             (PART) : NO VALUE ASSIGNED BECAUSE OF HIGH AVAILABILITY LARGE DB


FOLLOWING PARTITION REPORTS ARE PRINTED IN ORDER OF PARTITION SELECTION


FABT3601I      11000 TYPE K RECORDS PROCESSED

FABT3602I      11000 TYPE R RECORDS CREATED


FABT3545I ACTUAL ROOTS/BLOCK MAP WILL NOT BE REPRINTED


FABT3525I PROCESSING COMPLETED FOR THIS DATABASE


FABT3535I END-OF-FILE ON CONTROL FILE OR STOP REQUEST
```

Figure 112. FABTROOT—DBD Parameters and Overrides report

Actual Roots per Block report

The report shown in Figure 113 describes where root segments are physically stored in your HDAM, HIDAM, PHDAM, or PHIDAM database.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - HDTA            "ACTUAL ROOTS PER BLOCK REPORT"                          PAGE:     1
5655-K53                                                   DATE: 07/10/2006  TIME: 16.22.33                         FABTROOT - V2.R2


***  DBNAME: HDAMDB2   ***

             DBDNAME................. HDAMDB2                 DIRECT ALGORITHM NAME...   DFSHDC40
             DATABASE ORGANIZATION...     HDAM                HIGH BLOCK NUMBER.......        100
             ACCESS METHOD...........     VSAM                RAPS PER BLOCK..........          1
             BLOCK SIZE..............     1024                TOTAL RAPS..............        100

COUNT OF 0-35 REPRESENTED AS FOLLOWS:
  SYMBOL:     1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z
  COUNT : 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
COUNT GREATER THAN 35 =  *
END OF DATA BLOCKS    =  $

         EACH ENTRY IN THE TABLE BELOW REPRESENTS THE ACTUAL NUMBER OF ROOTS THAT ARE STORED IN THE CORRESPONDING BLOCK
                                                                      -----
 BLOCK #  0........1.........2.........3.........4.........5.........6.........7.........8.........9.........0
          1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

       0 |    22  2 1 1 2 222 1  11 21112 2 111  2 111121 2 21 1     21111 1 1 1  11   1 1    1111  21 1  21
     100 |$

          COUNT OF BLOCKS WITHOUT ROOTS        =       47          COUNT OF BLOCKS WITH MORE THAN 35 ROOTS =        0

          AVG. COUNT OF ROOTS PER ACTIVE BLOCK =      1.3          MAX. COUNT OF ROOTS IN ONE BLOCK        =        2

FABT3601I         70 TYPE K RECORDS PROCESSED

FABT3602I         70 TYPE R RECORDS CREATED

FABT3525I PROCESSING COMPLETED FOR THIS DATABASE
```

Figure 113. FABTROOT—Actual Roots per Block report

FABTRAPS PR9 data set

This section explains the FABTRAPS PR9 data set.

Function

This data set contains the Assigned Roots per RAP report produced by module FABTRAPS.

Format

This data set contains 133-byte, fixed-length records. Block size (if coded in your JCL) must be a multiple of 133. Code your DD statement as follows:

```
//PR9       DD SYSOUT=A
```

Assigned Roots per RAP report

The report shown in Figure 114 contains a map of all root anchor points (RAPs) in the database. It gives the actual number of root segments that are randomized to each RAP. It also includes some totals.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - HDTA           "ASSIGNED ROOTS PER RAP REPORT"                        PAGE:     1
5655-K53                                                  DATE: 07/10/2006  TIME: 15.21.06                       FABTRAPS - V2.R2


***  DBNAME: TPFOH1    ***


             DBDNAME................. TPFOH1                DIRECT ALGORITHM NAME...     (PART)
             DATABASE ORGANIZATION...    PHDAM              HIGH BLOCK NUMBER.......     (PART)
             ACCESS METHOD...........     VSAM              RAPS PER BLOCK..........     (PART)
             BLOCK SIZE.............   (PART)               TOTAL RAPS..............     (PART)
             PARTITION SELECTION EXIT NAME
                                .....    N / A


             (PART) : NO VALUE ASSIGNED BECAUSE OF HIGH AVAILABILITY LARGE DB


FOLLOWING PARTITION REPORTS ARE PRINTED IN ORDER OF PARTITION SELECTION
***  PARTITION NAME: TPFOH1A  ***


             PARTITION ID...........        1               DIRECT ALGORITHM NAME...   DFSHDC40
             BLOCK SIZE.............        0               HIGH BLOCK NUMBER.......       4500
                                                            RAPS PER BLOCK..........          1
             PARTITION HIGH KEY............                 TOTAL RAPS..............       4500
             X'FFFFFFFFFFFFFFFF'


COUNT OF 0-35 REPRESENTED AS FOLLOWS:
  SYMBOL:     1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z
  COUNT : 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
COUNT GREATER THAN 35 =  *
END OF DATA BLOCKS    =  $


         EACH ENTRY IN THE TABLE BELOW REPRESENTS THE NUMBER OF ROOTS THAT HAVE RANDOMIZED TO THE CORRESPONDING RAP
                                                                -----
  RAP #  0........1.........2.........3.........4.........5.........6.........7.........8.........9.........0
         1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

      0 | 1 13113232531442232111 12116322222123212 421 332232515143232312511323131411421341214511131214333134
    100 |5 5432114 34323242 113117333122223232124143516 21343252354311241245233231132123542537 222223 2123352
    200 |11433 24223 33413 23251322132124222112163212334 12 321322241 1325212 121115  1263  15532511241124114

------ For formatting purposes, several lines have been deleted.------


   4300 |54231311133362143 22524 13623225232116313431236 34255332323333  3144151 23334 2122123 23 51732232242
   4400 |11 344  44134114423221434323241121 52131 431611211357234246222512332335211 24 26232 162436245341333
   4500 |$


          COUNT OF RAPS NOT USED           =        410         COUNT OF RAPS WITH MORE THAN 35 ROOTS   =          0

          AVG. COUNT OF ROOTS ON ACTIVE RAP =       2.6         MAX. COUNT OF ROOTS ON ONE RAP          =          9


FABT3602I      11000 TYPE R RECORDS PROCESSED


NOTE :  SYNONYM - ANY ROOT RANDOMIZED TO THE SAME RAP  (RAP COUNT GREATER THAN 1)
----

FABT3700I NO MORE RECORDS FOR THIS FILE
```

Figure 114. FABTRAPS—Assigned Roots per RAP report

FABTRAPS PR9X data set

This section explains the FABTRAPS PR9X data set.

Function

This data set contains the Assigned Roots per Block report produced by module FABTRAPS.

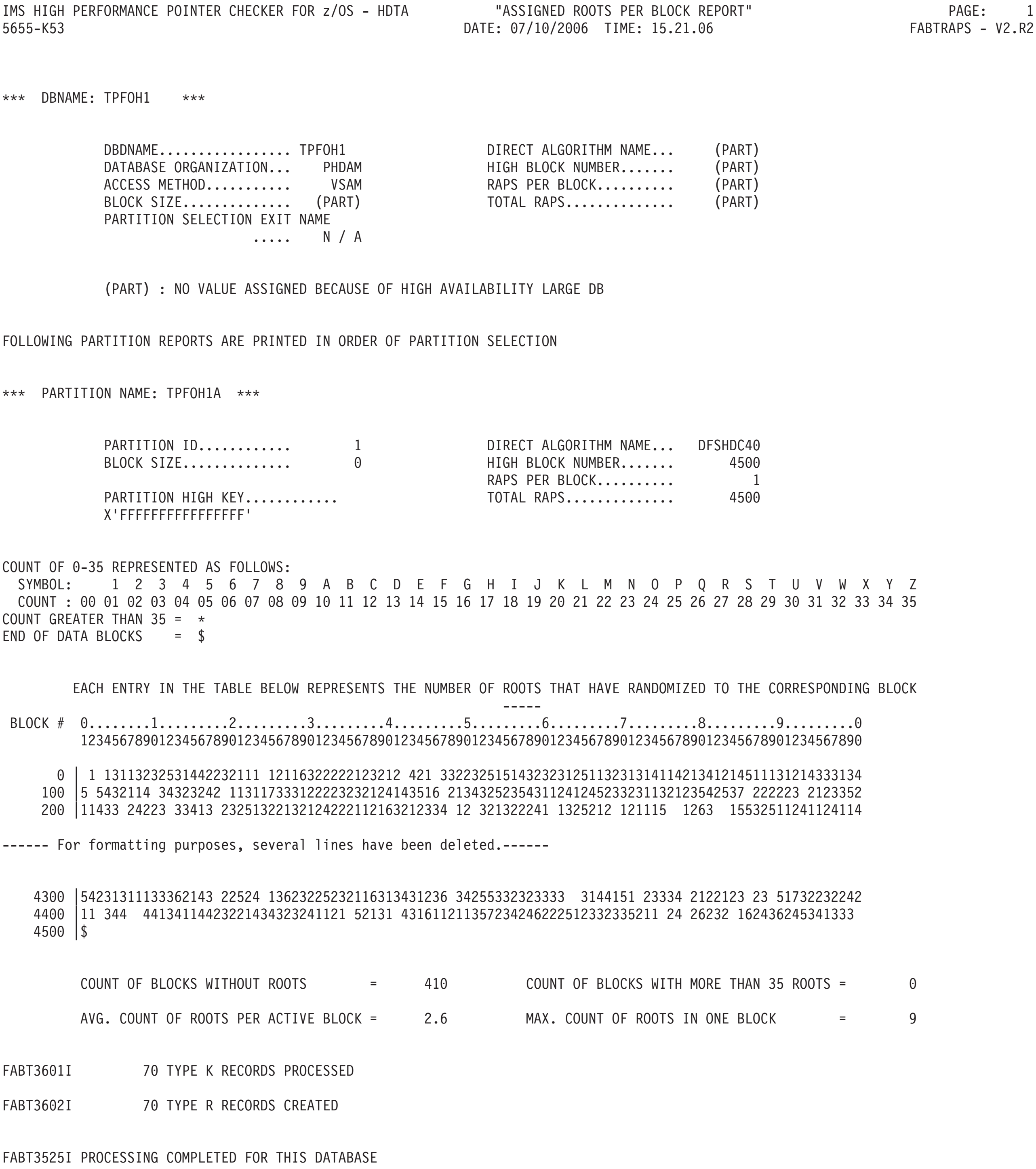
Format

This data set contains 133-byte, fixed-length records, and block size (if coded in your JCL) must be a multiple of 133. Code your DD statement as follows:

```
//PR9X      DD SYSOUT=A
```

Assigned Roots per Block report

The report shown in Figure 115 contains a map of all blocks (or control intervals) in the database. It gives the actual number of root segments that are randomized to each block. It also includes some totals.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - HDTA           "ASSIGNED ROOTS PER BLOCK REPORT"                        PAGE:     1
5655-K53                                                   DATE: 07/10/2006  TIME: 15.21.06                        FABTRAPS - V2.R2


***  DBNAME: TPFOH1    ***


             DBDNAME................. TPFOH1                  DIRECT ALGORITHM NAME...     (PART)
             DATABASE ORGANIZATION...    PHDAM                HIGH BLOCK NUMBER.......     (PART)
             ACCESS METHOD...........     VSAM                RAPS PER BLOCK..........     (PART)
             BLOCK SIZE.............   (PART)                 TOTAL RAPS..............     (PART)
             PARTITION SELECTION EXIT NAME
                                .....    N / A


             (PART) : NO VALUE ASSIGNED BECAUSE OF HIGH AVAILABILITY LARGE DB


FOLLOWING PARTITION REPORTS ARE PRINTED IN ORDER OF PARTITION SELECTION


***  PARTITION NAME: TPFOH1A  ***


             PARTITION ID...........        1                 DIRECT ALGORITHM NAME...   DFSHDC40
             BLOCK SIZE.............        0                 HIGH BLOCK NUMBER.......       4500
                                                              RAPS PER BLOCK..........          1
             PARTITION HIGH KEY............                   TOTAL RAPS..............       4500
             X'FFFFFFFFFFFFFFFF'


COUNT OF 0-35 REPRESENTED AS FOLLOWS:
  SYMBOL:     1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z
  COUNT : 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
COUNT GREATER THAN 35 =  *
END OF DATA BLOCKS    =  $


         EACH ENTRY IN THE TABLE BELOW REPRESENTS THE NUMBER OF ROOTS THAT HAVE RANDOMIZED TO THE CORRESPONDING BLOCK
                                                                 -----
 BLOCK #  0........1.........2.........3.........4.........5.........6.........7.........8.........9.........0
          1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

        0 | 1 13113232531442232111 12116322222123212 421 332232515143232312511323131411421341214511131214333134
      100 |5 5432114 34323242 113117333122223232124143516 21343252354311241245233231132123542537 222223 2123352
      200 |11433 24223 33413 23251322132124222112163212334 12 321322241 1325212 121115  1263  15532511241124114

------ For formatting purposes, several lines have been deleted.------


     4300 |54231311133362143 22524 13623225232116313431236 34255332323333  3144151 23334 2122123 23 51732232242
     4400 |11 344  44134114423221434323241121 52131 431611211357234246222512332335211 24 26232 162436245341333
     4500 |$


          COUNT OF BLOCKS WITHOUT ROOTS        =      410         COUNT OF BLOCKS WITH MORE THAN 35 ROOTS =        0

          AVG. COUNT OF ROOTS PER ACTIVE BLOCK =      2.6         MAX. COUNT OF ROOTS IN ONE BLOCK        =        9


FABT3601I        70 TYPE K RECORDS PROCESSED

FABT3602I        70 TYPE R RECORDS CREATED


FABT3525I PROCESSING COMPLETED FOR THIS DATABASE
```

Figure 115. FABTRAPS—Assigned Roots per Block report

FABTROOT PR10 data set

This section explains the FABTROOT PR10 data set.

HALDB Process Summary report

The report shown in Figure 116 describes which partitions are processed by HD Tuning Aid. The following figure shows an example of the HALDB Process Summary report for PHDAM. The processed partition is indicated by "*" in the "P" column. The "P" means it is processed. The format of this report can be controlled by a column 71 of a DB statement line-1 in a CTL data set. If no control statement is specified in the CTL data set, only processed partitions are displayed in this report.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - HDTA        "HALDB PROCESS SUMMARY REPORT"                               PAGE:     1
5655-K53                                                   DATE: 07/10/2006  TIME: 15.21.05                         FABTROOT - V2.R2

*** DBDNAME: TPFOH1   ***
  DBDNAME........................... TPFOH1
  DATABASE ORGANIZATION............    PHDAM
  ACCESS METHOD....................     VSAM
  NUMBER OF PARTITIONS DEFINED......       1
  NUMBER OF PARTITIONS PROCESSED....       1
  PARTITION SELECTION EXIT..........     N/A

*** PARTITIONS LISTED IN ORDER OF PARTITION SELECTION ***

  P SEQ   NAME        ID  PARTITION HIGH KEY

  * 0001  TPFOH1A      1  X'FFFFFFFFFFFFFFFF'
```

Figure 116. FABTROOT—HALDB Process Summary report

FABTROOT RAPSIN data set

This section describes the record type of the FABTROOT RAPSIN data set.

Function

The RAPSIN data set contains the block/RAP assignments used by module FABTRAPS to produce the randomization reports.

Format

This is a sequential data set with RECFM=FB and LRECL=42. BLKSIZE must be coded on the DCB parameter of your DD statement.

Record types

There are four record types in the RAPSIN data set:

- Header record: one per database.
- Part record: one per partition.
- Highkey record: more than one per partition.
- RAP record: one per root segment.

HD Tuning Aid uses Header record and RAP record for non-HALDB, and uses all types of records for HALDB.

The data set can contain records of more than one database. All records for a single database are contiguous. Figure 117 on page 341 and Figure 118 on page 342 show how the data set is organized. It is a product-sensitive programming interface. See “Programming interface information” on page 732 to understand the restrictions associated with this type of material.

Product-Sensitive Programming Interface

```
Header record for first database
Part record of first partition for first database
Highkey record of first partition for first database
RAP record of first partition for first database
RAP record of first partition for first database
........
........
Part record of second partition for first database
Highkey record of second partition for first database
RAP record of second partition for first database
RAP record of second partition for first database
........
Header record for 2nd database
Part record of first partition for second database
Highkey record of first partition for second database
RAP record of first partition for second database
.......
```

Figure 117. RAPSIN data set format

```
Header record for first database
RAP record for first database
RAP record for first database
RAP record for first database
... ... ...
Header record for second database
RAP record for second database
RAP record for second database
RAP record for second database
... ... ...
Header record for third database
RAP record for third database
RAP record for third database
RAP record for third database
... ... ...
```

Figure 118. RAPSIN data set format

End of Product-Sensitive Programming Interface

Header record

Position	Length	Description
1	2	Database number in hexadecimal
3	2	Zero in hexadecimal
5	8	DBD name
13	4	Database block size in hexadecimal
17	1	C'V' if the database is VSAM
18	1	C'1' if the database is HDAM C'2' if the database is HIDAM C'3' if the database is PHDAM C'4' if the database is PHIDAM
19	8	Module name of partition selection exit
27	8	Randomizing module name
35	4	RAA block number in hexadecimal
39	2	RAP number in each control interval or block (in hexadecimal)
41	2	RESERVED

Part record

Position	Length	Description
1	2	Database number in hexadecimal
3	2	Partition number in hexadecimal Partition definition order for PDB Partition selection order for HALDB
5	8	ZERO in hexadecimal
13	4	Database block size in hexadecimal
17	8	DD name in case of PDB Partition name in case of HALDB
25	2	Partition ID for HALDB
27	8	Randomizing module name
35	4	RAA block number in hexadecimal
39	2	RAP number in each control interval or block (in hexadecimal)
41	2	Reserved

Highkey record

Position	Length	Description
1	2	Database number in hexadecimal
3	2	Partition selection order in hexadecimal
5	3	ZERO in hexadecimal
8	1	Sequence number in Highkey record
9	34	Partition high key string or partition selection string

RAP record

Position	Length	Description
1	2	Database number in hexadecimal
3	2	Data set group number for non-HALDB Partition selection order for HALDB (in hexadecimal)
5	3	Block number (in hexadecimal) to which this root segment is randomized
8	1	RAP number (in hexadecimal) to which this root segment is randomized
9	1	C’R’
10	4	Relative byte address (in hexadecimal) at which this root segment is randomized
14	29	The first 29 bytes of the sequence field of this root

Chapter 18. JCL examples for HD Tuning Aid

There are many ways to use the HD Tuning Aid utility. The examples in this chapter represent some of the typical tasks that HD Tuning Aid can perform. By studying and understanding these examples, you can learn the techniques to use to monitor the condition of your databases.

Topics:

Example 1: Running HD Tuning Aid with HD Pointer Checker

This example shows how to run HD Pointer Checker and HD Tuning Aid together. Run HD Tuning Aid as in the example when you are performing your regularly scheduled HD Pointer Checker run.

When you run HD Tuning Aid this way, do not use any HD Tuning Aid control statements. All of the control statements and JCL in Figure 119 apply to HD Pointer Checker.

KEYSIN is an optional parameter of the OPTION statement. When KEYSIN=YES is specified, HD Pointer Checker creates the KEYSIN data set that is used as input to HD Tuning Aid.

See Part 2, “HD Pointer Checker,” on page 21 for a description of the FABPPTA cataloged JCL procedure and the HD Pointer Checker aspects of this example.

```
//EXAMPLE1 JOB --- use normal job statement parameters here ---
//*
//JOBLIB DD DSN=HPS.SHPSLMD
//       DD DSN=IMSVS.RESLIB,DISP=SHR
//*
//*
//PCRUN   EXEC FABPPTA,
//            PSB=PSBSMUAL
//* ----------------------------------------------------------------------------
//HDPCPRO.PROCCTL DD *
 PROC TYPE=ALL
  OPTION KEYSIN=YES,ERRLIMIT=NO,INCORE=YES
  DATABASE DB=DSSTUIVN,DD=DSSTUIV0,OVERFLOW=DSSTUIV1,DATASET=IMAGECOPY
  DATABASE DB=DSSCHXIN,DD=DSSCHXI0,PRIMEDB=DSSCHHVN,DATASET=IMAGECOPY
  DATABASE DB=DSFACXVN,DD=DSFACXV0,PRIMEDB=DSFACHON,DATASET=IMAGECOPY
  DATABASE DB=DSFDAXVN,DD=DSFDAXV0,OVERFLOW=DSFDAXV1,PRIMEDB=DSFACHON,
           DATASET=IMAGECOPY
  DATABASE DB=DSSCHHVN,DD=DSSCHHV0,DATASET=IMAGECOPY
  DATABASE DB=DSFACHON,DD=DSFACHO0,DATASET=IMAGECOPY
  DATABASE DB=DSCRSDVN,DD=DSCRSDV0,DATASET=IMAGECOPY
  DATABASE DB=DSCRSDVN,DD=DSCRSDV1,DATASET=IMAGECOPY
  DATABASE DB=DSCLSDVN,DD=DSCLSDV0,DATASET=IMAGECOPY
```

Figure 119. Example 1: JCL

Example 2: Iterative tuning analysis

In this example, an HDAM database (SD148P) and an HIDAM database (SD144P) are being analyzed. The purpose of the analysis is to determine good values for the DBD parameters that pertain to randomization. Once the analysis is complete, the DBDs will be changed accordingly, and the databases will be reorganized. SD144P will be converted to HDAM organization.

It is assumed that the KEYSIN data set contains root keys for both databases (SD148P and SD144P). Figure 120 shows the JCL and control statements that must be coded by the user.

HD Tuning Aid is being run twice for each database. Since module FABTROOT only makes one pass through the KEYSIN data set, HD Tuning Aid must be executed twice. Each execution of HD Tuning Aid processes both databases once.

Since SD144P is an HIDAM DBD, it is required that **all** fields on the control statement be used, even if the actual DBD value is not being changed.

```
//EXAMPLE2  JOB --- use normal job statement parameters here ---
//*
//TUNE1   EXEC FABTIMS,
//             CYL='10,10',            SPACE FOR WORK FILES
//             PARM2='SIZE=(MAX)',          PARM FOR DFSORT
//             KEYSIN='HPS.TEST.KEYSIN',         INPUT KEYS
//             DBDLIB='HPS.TEST.DBDLIB'
//STEP1.CTL DD *
SD144P    DFSHDC40  001000  004  08192  001  007  03000  HDAM
SD148P    DFSHDC40  001500  002  04096
//TUNE2   EXEC FABTIMS,
//             CYL='10,10',            SPACE FOR WORK FILES
//             PARM2='SIZE=(MAX)',          PARM FOR DFSORT
//             KEYSIN='HPS.TEST.KEYSIN',         INPUT KEYS
//             DBDLIB='HPS.TEST.DBDLIB'
//STEP1.CTL DD *
SD144P    DFSHDC20  001000  004  08192  001  007  03000  HDAM
SD148P    DFSHDC20  001500  002  04096
/*
//
```

Figure 120. Example 2: JCL

Example 3: Running HD Tuning Aid under IMS

In this example, an HDAM database (DBDP33J) is being analyzed to examine the effects of changing the size of the root addressable area. The randomizing routine accesses some IMS control blocks, so HD Tuning Aid must be run under IMS.

Figure 121 on page 349 shows the JCL and control statements that must be coded by the user. It is assumed that the KEYSIN data set (HPS.DBDP33J.KEYSIN) has been created in an earlier HD Pointer Checker job.

HD Tuning Aid is being run four times as follows:

- Step TUNE1 runs HD Tuning Aid with no control statements. This produces the following reports:
  1. Actual Roots per Block report
  2. Assigned Roots per Block report
  3. Assigned Roots per RAP report
- Steps TUNE2, TUNE3, and TUNE4 run HD Tuning Aid with a control statement. They produce the following reports:
  1. DBD Parameters and Overrides report
  2. Assigned Roots per Block report
  3. Assigned Roots per RAP report

Since module FABTROOT only makes one pass through the KEYSIN data set, HD Tuning Aid must be executed several times to do this kind of iterative analysis.

Since DBDP33J is an HDAM DBD, only the DBD parameters that are being overridden need to be specified on the FABTROOT control statement. This example produces reports based on changing two DBD parameters:

- The number of blocks in the root addressable area
- The number of root anchor points per block.

```
//EXAMPLE3  JOB --- use normal job statement parameters here ---
//*
//TUNE1   EXEC FABTIMS,
//             PSB=P33J001G,                        PSB NAME
//             CYL='10,10',            SPACE FOR WORK FILES
//             PARM2='SIZE=(MAX)',          PARM FOR DFSORT
//             KEYSIN='HPS.DBDP33J.KEYSIN',       INPUT KEYS
//             DBDLIB='HPS.TEST.DBDLIB',
//             PSBLIB='HPS.TEST.PSBLIB'
//TUNE2   EXEC FABTIMS,
//             PSB=P33J001G,                        PSB NAME
//             CYL='10,10',            SPACE FOR WORK FILES
//             PARM2='SIZE=(MAX)',          PARM FOR DFSORT
//             KEYSIN='HPS.DBDP33J.KEYSIN',       INPUT KEYS
//             DBDLIB='HPS.TEST.DBDLIB',
//             PSBLIB='HPS.TEST.PSBLIB'
//STEP1.CTL DD *
DBDP33J             001000  004
/*
//TUNE3   EXEC FABTIMS,
//             PSB=P33J001G,                        PSB NAME
//             CYL='10,10',            SPACE FOR WORK FILES
//             PARM2='SIZE=(MAX)',          PARM FOR DFSORT
//             KEYSIN='HPS.DBDP33J.KEYSIN',       INPUT KEYS
//             DBDLIB='HPS.TEST.DBDLIB',
//             PSBLIB='HPS.TEST.PSBLIB'
//STEP1.CTL DD *
DBDP33J             002000  002
/*
//TUNE4   EXEC FABTIMS,
//             PSB=P33J001G,                        PSB NAME
//             CYL='10,10',            SPACE FOR WORK FILES
//             PARM2='SIZE=(MAX)',          PARM FOR DFSORT
//             KEYSIN='HPS.DBDP33J.KEYSIN',       INPUT KEYS
//             DBDLIB='HPS.TEST.DBDLIB',
//             PSBLIB='HPS.TEST.PSBLIB'
//STEP1.CTL DD *
DBDP33J             001500  003
/*
//
```

Figure 121. Example 3: JCL

Examples of control statements for PHDAM

This section provides example control statements for PHDAM.

Example 4: Changing randomizing parameters for all partitions consistently

SAMPDBI is a PHDAM database. This set of CTL statements is the simulation of changing the randomizer name of each partition. HD Tuning Aid assigns the same randomizer name to each partition.

```
//CTL DD *
SAMPDB1    DFSHDC40
/*
```

Example 5: Changing randomizing parameters for individual partitions

This sample is for an input PHDAM database with two partitions. Change only the randomizer name of the first partition.

```
//CTL DD *
SAMPDB1
 SAMPP1    DFSHDCAA
/*
```

Example 6: Changing the high key value of partition

SAMPDBI is a PHDAM database. This sample shows how to change the high key value of a specific partition.

```
//CTL DD *
SAMPDB1
 SAMPP1    DFSHDCAA                                                     +
 C'1234000000000000100000000'                                           +
 C'00000000000000000000000000000'
/*
```

The partition high key is assigned as `12340000000000001000000000000000000000000000000000000.`

Example 7: Changing the partition selection

SAMPDB1 is a PHDAM database. This sample shows how to a change a partition selection from a high key selection to a partition selection exit.

```
//CTL DD *
SAMPDB1                                                                  S
          DFSPSE00
 SAMPP1                                                                  +
 C'1234000000000000000000000'
 SAMPP2                                                                  +
 C'5678900000000000000000000'
/*
```

Example 8: Adding and deleting some partitions

SAMPDBI is a PHDAM database. This is a sample of two additions to and one deletion for a PHDAM.

```
//CTL  DD *
SAMPDB1
 SAMPP4   RMOD3    1000     10   4096             0      ADD            +
 C'300'
 SAMPP5   RMOD4    2000     20   2048             0      ADD            +
 C'700'
 SAMPP1                                                  DEL
/*
```

Table 39 shows the original configuration of the PHDAM database.

Table 39. Original configuration of the PHDAM database

Part name	Part ID	High key	Randomizer	rbn	anch	blksz	bytes
SAMPP1	1	300	RMOD1	1,000	40	4,096	0
SAMPP2	2	500	RMOD2	1,000	20	2,048	0
SAMPP3	3	600	RMOD2	2,000	30	2,048	100

Table 40 shows the modified configuration of PHDAM database that is to be used by HD Tuning Aid for its simulation.

Table 40. Modified configuration of PHDAM database (Adding and deleting some partitions)

Part name	Part ID	High key	Randomizer	rbn	anch	blksz	bytes
SAMPP2	2	500	RMOD2	1,000	20	2,048	0
SAMPP3	3	600	RMOD2	2,000	30	2,048	100
SAMPP4	4	300	RMOD3	1,000	10	4,096	0
SAMPP5	5	700	RMOD4	2,000	20	2,048	0

Analyzing conversion to PHDAM

These samples show two methods of converting from HDAM, HIDAM, or PHIDAM to PHDAM.

Example 9: Defining individual partition

In this sample, HD Tuning Aid simulates a PHDAM database. The PHDAM has three partitions, and each of them has different DBD parameters.

```
//CTL  DD *
SAMPDB1            1000                                  PHDAMPARTINI  H
 SAMPP1   RMOD1             40   4096             0                    +
 C'300'
 SAMPP2   RMOD2             20   2048             0                    +
 C'500'
 SAMPP3   RMOD2    2000     30   2048           100                    +
 C'600'
/*
```

Table 41 shows the modified configuration of PHDAM database that is to be used by HD Tuning Aid for its simulation.

Table 41. Modified configuration of PHDAM database (Defining individual partition)

Part name	Part ID	High key	Randomizer	rbn	anch	blksz	bytes
SAMPP1	1	300	RMOD1	1,000	40	4,096	0
SAMPP2	2	500	RMOD2	1,000	20	2,048	0
SAMPP3	3	600	RMOD2	2,000	30	2,048	100

Example 10: Defining all partitions consistently: Simple method

In this sample, HD Tuning Aid simulates a PHDAM database. The PHDAM has three partitions, and all three have the same DBD parameters. Partition names are automatically assigned by HD Tuning Aid. Partition high key strings are required for every partition individually.

```
//CTL  DD *
SAMPDB1   RMOA0    1000     10   2048             0     PHDAMAUTOINI  H
 C'300'
 C'500'
 C'600'
/*
```

Table 42 shows the modified configuration of PHDAM database that is to be used by HD Tuning Aid for its simulation.

Table 42. Modified configuration of PHDAM database (Defining all partitions consistently)

Part name	Part ID	High key	Randomizer	rbn	anch	blksz	bytes
SAM0001	1	300	RMOA0	1,000	10	2,048	0
SAM0002	2	500	RMOA0	1,000	10	2,048	0
SAM0003	3	600	RMOA0	1,000	10	2,048	0

If 'AUTOINI' is specified, HD Tuning Aid assigns the partition name automatically. HD Tuning Aid follows the rule that the first three characters are the same as in the database name and the last four characters are numbered from 0001 to 1001, in ascending order.

Analyzing conversion to HDAM

This section provides an example that shows how to evaluate the conversion from PHDAM or PHIDAM to HDAM.

Example 11: Defining HDAM

This sample shows the procedure for evaluation of conversion from PHDAM or PHIDAM to HDAM.

```
//CTL  DD *
SAMPDB1   RMOA0    1000     10   2048             0     HDAM PARTDEL
/*
```

Part 4. DB Historical Data Analyzer

Chapter 19. Overview of DB Historical Data Analyzer

This chapter gives an overview of DB Historical Data Analyzer.

Topics:

Program functions

DB Historical Data Analyzer uses the data produced by HD Pointer Checker to create the following reports when the program is run as a batch job:

- HISTORY Data Set by DB-DS report, which shows all the entries in the HISTORY data set by database data set order.
- HISTORY Data Set by Key Date report, which shows all the entries in the HISTORY data set by the HD Pointer Checker run date order.
- HD Pointer Checker Summary report, which shows the summary of HD Pointer Checker run results for all the specified database data sets.
- HD Analysis report, which shows concise data of the HD Pointer Checker run results for each database data set.

In addition to creating the above reports, DB Historical Data Analyzer, when run under TSO/ISPF, creates

- HD Analysis graph, which shows the trend of various aspects of a database data set (such as the amount of free space, total number of segment occurrences, and the percentage of blocks with free space) on a graphic terminal, in the form of a graph, chart, or table.

The HD Analysis graph contains the data collected by the two IMS HP Pointer Checker components: HD Pointer Checker and Space Monitor.

To produce graph charts, the Interactive Chart Utility (ICU) of Graphical Data Display Manager Presentation Graphic Feature is a prerequisite, and this program must be run under TSO/ISPF. A hard copy can also be obtained on printers supported by GDDM.

Export Utility is a subcomponent of DB Historical Data Analyzer. It runs as a batch job. It exports the data produced by HD Pointer Checker.

Typical uses

DB Historical Data Analyzer is used as follows:

- To obtain a concise summary report of HD Pointer Checker output
- To obtain a graph, chart, or table image that shows the trend of space allocation/use and other database analysis data of IMS full-function database data sets

Export Utility is used to export the database statistics data from the HISTORY data set to a flat file that can be processed by user application programs.

Program structure

The program structure of DB Historical Data Analyzer is described separately for the MVS batch environment and TSO/ISPF environment.

MVS batch environment

- DB Historical Data Analyzer consists of one load module (FABGHIST).
- HISTORY Data Set Migration utility consists of one load module (FABGFMIG).

Export Utility

Export Utility consists of FABGXEXP and several load modules. Export Utility runs as a batch job. FABGXEXP is the first program that runs in the job step.

TSO/ISPF environment

- DB Historical Data Analyzer consists of several ISPF panel modules, message modules, and two load modules (FABGPANL and FABGGRAF).

  FABGP000 is the first panel module invoked by TSO/ISPF ISPSTART command to start the dialog with DB Historical Data Analyzer. The two load modules, FABGPANL and FABGGRAF, are called during the dialog. FABGGRAF invokes ICU to generate and display the HD Analysis graph.

Data flow

DB Historical Data Analyzer operates in either MVS batch environment or TSO/ISPF environment.

Running in MVS batch environment

Figure 122 on page 361 shows the general data flow of DB Historical Data Analyzer when it is run in the MVS batch environment.

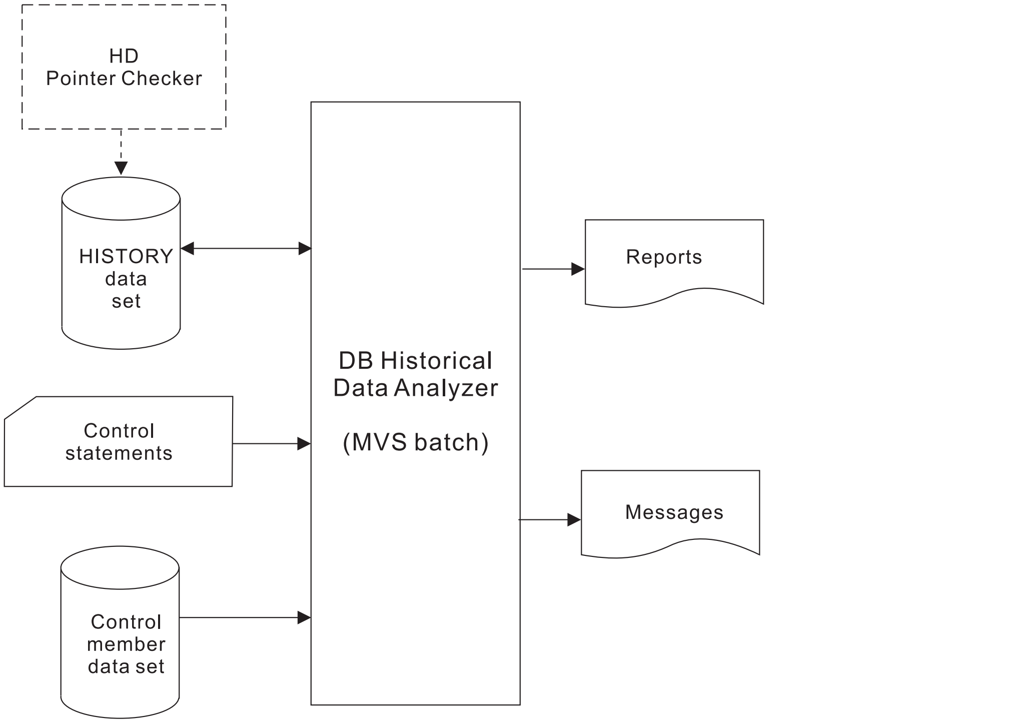


Figure 122. Data flow for DB Historical Data Analyzer (MVS Batch)

- DB Historical Data Analyzer uses IMS database data set statistics and analysis information maintained in the **HISTORY data set**. The data is collected in the HISTORY data set by HD Pointer Checker.

  DB Historical Data Analyzer provides functions to initialize, reorganize, and delete entries from the HISTORY data set.
- The user describes the function to be performed by this utility in the input data stream or in a data set in the form of **control statements**. Control statements describe the functions and/or the database data sets to be processed.
- If the above mentioned control statement contains a MEMBER= keyword parameter, DB Historical Data Analyzer refers to the **control member data set**, which contains the names of database data sets to be processed. This is the same data set as is used by IMS HP Pointer Checker Space Monitor.
- DB Historical Data Analyzer produces the following four types of reports:
  - HISTORY Data Set by DB-DS report
  - HISTORY Data Set by Key Date report
  - HD Pointer Checker Summary report
  - HD Analysis report

Running Export Utility

Figure 123 on page 362 shows the general data flow of Export Utility.

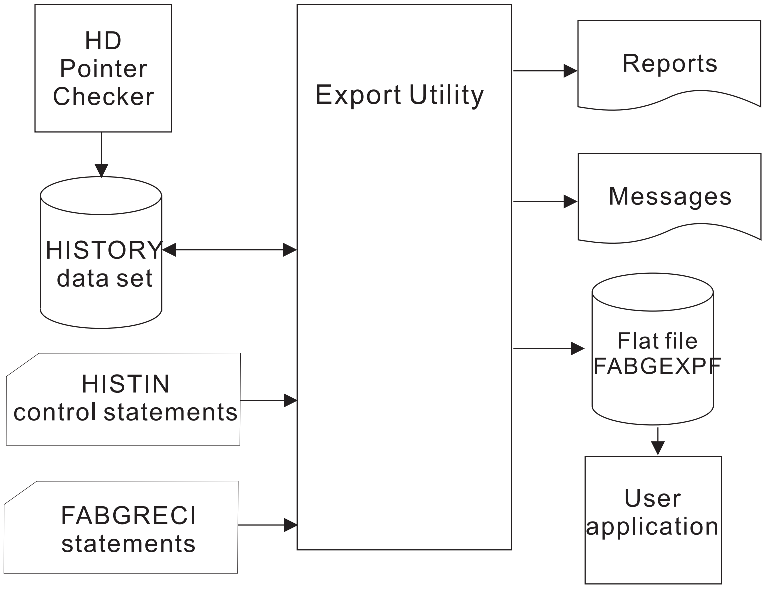


Figure 123. Data flow for Export Utility

- Export Utility uses IMS database data set statistics and analysis information maintained in the HISTORY data set. The data is collected in the HISTORY data set by HD Pointer Checker.

  Export Utility exports data from the HISTORY data set to a sequential file. The exported file is referred to as a *flat file*, and the record is referred to as a *flat record*. User's application can process the flat records as an input data.
- The flat record can be created in a format that is specified by the user or a predefined format provided by Export Utility. The definition of a user's format is referred to as a *flat record definition*. The flat record definitions are stored in the FABGRECI data set.
- The user describes the function to be run by this utility in the input data stream or in a data set in the form of the HISTIN control statements.
- Export Utility produces the following reports:
  - HISTORY Data Set Message report
  - FABGRECI Statement report
  - History Attribute report
  - History Export Summary report

Running in TSO/ISPF environment

Figure 124 on page 363 shows the general data flow of DB Historical Data Analyzer when it is run in the TSO/ISPF environment.

- When run in TSO/ISPF environment, DB Historical Data Analyzer uses two data sets from other IMS HP Pointer Checker utilities. One is the **HISTORY data set**, containing entries created by HD Pointer Checker. The other is the **Space Monitor graph record data set**, created by IMS HP Pointer Checker Space Monitor, which contains space management information. The Space Monitor graph record data set is optional.
- DB Historical Data Analyzer refers to the **control member data set**, which contains the names of database data sets. This is the same data set as is used by IMS HP Pointer Checker Space Monitor. This control member data set is optional.

- The user operates from a display unit and selects necessary items on the panels to specify the type of chart to create.
- DB Historical Data Analyzer retrieves the necessary information from the HISTORY data set and the Space Monitor graph record data set, and passes it to GDDM Interactive Chart Utility (ICU). Then the ICU generates a chart (HD Analysis graph) and displays it on a display unit. The user can then communicate with ICU interactively to customize the chart format, if necessary, or print a hard copy of the chart on a printer supported by GDDM.

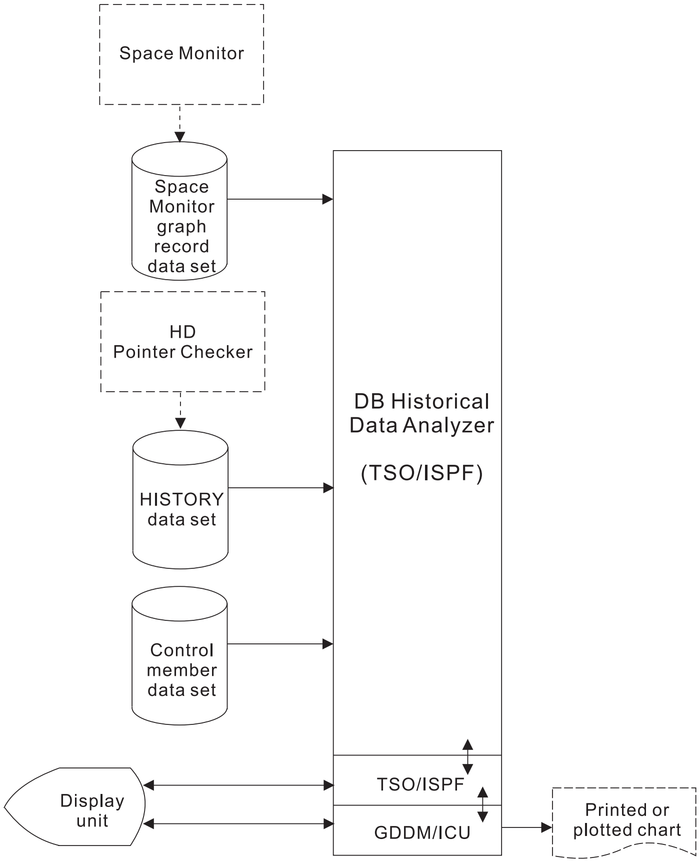


Figure 124. Data flow for DB Historical Data Analyzer (TSO/ISPF)

Restrictions and considerations

Restrictions

- A database whose attribute has been changed by DBD regeneration cannot be treated in the same way as before DBD regeneration. In this situation, all HISTORY data set entries for the database must be deleted before the database is processed by HD Pointer Checker. Otherwise, the results are unpredictable. This is also applied to the migration situation from non-HALDB to HALDB.
- DB Historical Data Analyzer applies only to the following database organizations:
  – HDAM
  – HIDAM (primary and index)
  – HISAM (including SHISAM)
  – Secondary index
  – PHDAM (partitioned HDAM)
  – PHIDAM (partitioned HIDAM)

  - PSINDEX (partitioned secondary index)

  Indirect List Data Sets (ILDS) used for PHDAM and PHIDAM are not analyzed.
- Export Utility exports the data only to the following database organizations:
  - HDAM
  - HIDAM (excluding index)
  - HISAM (including SHISAM)
  - PHDAM (partitioned HDAM)
  - PHIDAM (partitioned HIDAM)

  The data of the following database organization are not processed:
  - HIDAM and PHIDAM indexes
  - Secondary index
  - PSINDEX (partitioned secondary index)
  - Indirect List Data Sets (ILDS) used for PHDAM and PHIDAM
- In order to produce graph charts, Interactive Chart Utility (ICU) of Graphical Data Display Manager Presentation Graphic Feature is a prerequisite, and this program must be run under TSO/ISPF.
- The HISTORY data set must be periodically reorganized to avoid performance problems with DB Historical Data Analyzer and the shortage of the available space on the HISTORY data set.
- The database administrator should be familiar with the functions and operations of TSO/ISPF and GDDM Interactive Chart Utility (ICU) to produce the HD Analysis graph.
- While running DB Historical Data Analyzer in the TSO/ISPF environment, you cannot invoke DB Historical Data Analyzer from the DB Historical Data Analyzer panel, or have two dialog sessions in the split screen mode.
- HISTORY data sets used under DBT Version 2 Release 3 do not have upward compatibility with IMS HP Pointer Checker utilities. You must migrate them by the procedure described in “Migrating from DBT Version 2 Release 3 format” on page 370.

Considerations for HALDB Online Reorganization (OLR)

- For the HALDB partition which is OLR capable, DB Historical Data Analyzer and Export Utility show the DD names and database data set names of the active data set groups (either (A-J&X) or (M-V&Y)) at the time of HD Pointer Checker’s run.
- DB Historical Data Analyzer and Export Utility can work while a HALDB partition is in cursor-active status.

Chapter 20. Operating instructions for DB Historical Data Analyzer in the MVS batch environment

This chapter describes how to run a FABGHIST program of DB Historical Data Analyzer, which is a batch job. For the operation of this program in TSO/ISPF environment, see Chapter 22, “Operating instructions for DB Historical Data Analyzer in the TSO/ISPF environment,” on page 447. For the operation of Export Utility, see Chapter 21, “Operating instructions for Export Utility,” on page 409.

This chapter gives an introduction to the product.

Topics:

- “Preparation for running DB Historical Data Analyzer”
- “Job control language” on page 366
- “Input” on page 367
- “Output” on page 379

Preparation for running DB Historical Data Analyzer

To run DB Historical Data Analyzer as a batch job, follow the steps described below:

1. Allocate and initialize a VSAM KSDS for the HISTORY data set. DB Historical Data Analyzer provides a function to initialize the HISTORY data set. For the description on how to create a HISTORY data set, refer to “HISTORY data set (HISTORY)” on page 367.

   IMS HP Pointer Checker Historical Data Analyzer provides a function to migrate the HISTORY data set used under DBT Version 2 Release 3 into the required format. Refer to “HISTORY data set (HISTORY)” on page 367.
2. At the time of initialization, attributes of the HISTORY data set are follows:
   - MULTIENT option is ″NO″

     The multiple entries option is inactive. In this status, one database data set entry per day is recorded in the HISTORY data set.
   - EXPORTABLE option is ″NO″

     The HISTORY data set is not exportable. In this status, Export Utility does not export data from the HISTORY data set.
   - HISTLOCK option is ″GROUP″

     The HISTORY lock attribute is set to ″GROUP″. In this status, the HD Pointer Checker job issues ENQ MACRO with QNAME=FAB@HIST, RNAME=FAB@HIST, and SCOPE=SYSTEMS, before updating the HISTORY data set.
   - | MULTIIMSID option is ″NO″

     | The Multiple-IMSID option is disabled. In this status, the IMSID parameter of | the DBHDA PROC statement is ignored.

   For the description on how to change options, see “HISTORY data set (HISTORY)” on page 367.
3. Make sure that HD Pointer Checker is run in advance to create entries in the HISTORY data set.
4. Specify control statements to describe the functions to be performed. The control statements can reside in the input stream, or in a data set called HISTIN. Refer to “HISTIN data set” on page 371 for the description of control statement specifications.

5. Ensure that a control member data set exists, if a MEMBER= option parameter is coded on the control statement. Refer to “Control member data set (SPMNMBR)” on page 379 for the description of this data set.
6. Code the JCL as described in “Job control language.”

Then, run the DB Historical Data Analyzer job.

Job control language

JCL for the DB Historical Data Analyzer is explained in this section.

FABGHIST JCL

To run DB Historical Data Analyzer, supply an EXEC statement and the appropriate DD statements. Table 43 summarizes the DD statements.

Table 43. DB Historical Data Analyzer DD statements

DDNAME	Use	Format	Need
HISTORY	Input/Output	KSDS	Required
HISTIN	Input	LRECL=80	Required
SPMNMBR	Input	PDS	Optional
HISTPRT	Output	LRECL=133	Required
HISTMSG	Output	LRECL=133	Required
SYSUDUMP	Output	SYSOUT	Optional

EXEC This statement must be in the following form:

```
//      EXEC PGM=FABGHIST
```

HISTORY DD

This required VSAM KSDS data set contains the summary information of the HD Pointer Checker run results. This data set must be allocated before you run DB Historical Data Analyzer.

DISP=OLD must be used if you want to maintain the HISTORY data set by specifying TYPE=CREATE, DELETE, UPDATE, or REORG on the PROC control statement. Otherwise, DISP=SHR should be used.

HISTIN DD

This required input data set contains control statements, which describe your specification of the processing to be done by DB Historical Data Analyzer.

SPMNMBR DD

This optional input partitioned data set (PDS) contains one or more members, each of which consists of one or more control statements from Space Monitor.

This data set is required only when the MEMBER= option parameter of the DATABASE statement in the HISTIN data set is specified; otherwise, this DD statement is ignored even if coded.

HISTPRT DD

This required output data set contains reports. BLKSIZE, if coded, must be a multiple of 133.

HISTMSG DD
This required output data set contains messages. BLKSIZE, if coded, must be a multiple of 133.

SYSUDUMP DD
This statement defines output from a system abend dump routine. It is used only when a dump is required. Although optional, it is recommended that you include this data set.

Input

This section describes all the input data sets that are required in order to run DB Historical Data Analyzer in the MVS batch environment. This includes the control statement data set (HISTIN DD).

HISTORY data set (HISTORY)

The HISTORY data set is a VSAM KSDS data set that contains the IMS database data set statistics and analysis information obtained from the HD Pointer Checker output data.

This data set is used for creating reports or exporting. It is also an input to Space Monitor.

DB Historical Data Analyzer has the following functions to maintain the HISTORY data set. These are:
- Initializing the HISTORY data set
- Updating options of the HISTORY data set
- Reorganizing the HISTORY data set
- Deleting entries from the HISTORY data set

The user can invoke the above functions with the control statements specified in HISTIN data set (see “HISTIN data set” on page 371).

The HISTORY data set has four optional attributes, EXPORTABLE, MULTIENT, HISTLOCK, and MULTIIMSID. For details of the options, see “OPTION control statement” on page 376.

Format

This section describes product-sensitive programming interface information. See “Programming interface information” on page 732 to understand the restrictions associated with this type of material.

Product-Sensitive Programming Interface

The key of the HISTORY data set record is concatenation of the following fields and the length of key field is 23 byte:

Date The date on which the specified real database data set was scanned by HD Pointer Checker. If an image copy is used, it is the date when the image copy data set was created.

Database name
The name of the DBD as coded in the NAME= keyword of the DBD macro in the DBD.

Partition name

The name of the partition as defined through the IMS HALDB Partition Definition Utility (DFSHALDB). If the database is a non-HALDB, this field is filled with binary zeros.

Data set group number

The ordinal number of the data set group.

In case of a database data set record, this field consists of two fields.

- Sequential number within a day

  The first byte of the data set group number. It is the sequential number of database data set entry within a day. The sequential number starts with X'00', therefore, the value of this field will be the same as the other records, if only one entry is stored per day. When the Multiple-IMSID option is enabled, the top bit is turned on.
- Data Set Group Number

  The second one byte is the ordinal number of the data set group. When the Multiple-IMSID option is enabled, the first 4 bits indicate the IMSID number and the next 4 bits indicate the DSG number.
- IMSID number

  The IMSID number is the number assigned for each IMSID entry in the IMS record. IMSID record is created when the Multiple-IMSID option is enabled.

Record sequence number

The ordinal number of the record within the entry. (One entry consists of multiple records.)

The first byte after the key and the 24th-byte field (offset 23 X'17') contains the following data:

Public format flag

Contains the flag showing whether this record is a public format. If this field is X'00', it is a record made public to the user. If it is other than X'00', it is a non-public record. If EXPORTABLE=YES, some of the history record entries are generated in non-public format. In user applications, ignore and do not process the non-public records. Refer to "Record types (EXPORTABLE=YES/NO)" on page 11 for the details of the public and non-public format.

End of Product-Sensitive Programming Interface

Refer to Appendix E, "Layout of HISTORY data set record," on page 729 for the detailed layout of the public record.

Creating HISTORY data set

Figure 125 on page 369 presents a sample JCL for allocating and initializing the HISTORY data set. The following parameters are required for the DEFINE CLUSTER command:

- INDEXED
- UNIQUE
- KEYS(23,0)
- SHAREOPTIONS(3,3)
- RECORDSIZE(240,240)

```
//***************************************************
//**  ALLOCATE HISTORY DATA SET                   **
//***************************************************
//ALLOC   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=A
//SYSIN     DD *
  DEFINE CLUSTER ( NAME(HIST.HISTORY) -
                   VOLUMES(IMSDBT) CYLINDERS(3,2) -
                   INDEXED UNIQUE KEYS(23,0) -
                   SHAREOPTIONS(3,3) -
                   RECORDSIZE(240,240) -
                   CONTROLINTERVALSIZE(4096) ) -
         DATA    ( NAME(HIST.HISTORY.DATA) ) -
         INDEX   ( NAME(HIST.HISTORY.INDEX) )
/*
//***************************************************
//**  INITIALIZE HISTORY DATA SET                 **
//***************************************************
//INIT    EXEC PGM=FABGHIST
//STEPLIB   DD DISP=SHR,DSN=HPS.SHPSLMD0
//HISTORY   DD DISP=OLD,DSN=HIST.HISTORY
//HISTPRT   DD SYSOUT=A
//HISTMSG   DD SYSOUT=A
//SYSUDUMP  DD SYSOUT=A
//HISTIN    DD *
  PROC TYPE=CREATE
  ENDPROC
/*
```

Figure 125. Sample JCL for creating a HISTORY data set

For the sample JCLs for reorganizing and deleting entries from the HISTORY data set, see Chapter 23, “JCL examples for DB Historical Data Analyzer,” on page 475.

Changing options of HISTORY data set

Figure 126 shows a sample JCL stream for changing options of the HISTORY data set. This step is optional.

```
//UPDATE  EXEC PGM=FABGHIST
//STEPLIB   DD DISP=SHR,DSN=HPS.SHPSLMD0
//HISTORY   DD DISP=OLD,DSN=HIST.HISTORY
//HISTPRT   DD SYSOUT=A
//HISTMSG   DD SYSOUT=A
//SYSUDUMP  DD SYSOUT=A
//HISTIN    DD *
  PROC TYPE=UPDATE
  OPTION MULTIENT=YES,EXPORTABLE=YES,HISTLOCK=DATASET,
         MULTIIMSID=YES
  ENDPROC
```

Figure 126. Sample JCL for changing options for a HISTORY data set

Estimating the size of HISTORY data set

Figure 127 on page 370 shows the formulae to calculate the total number of records in a HISTORY data set and the data set size. The quotient is rounded up.

```
Total number of records = N1 + N2 + N3
Size = total number of records x 240

N1 =  ( A / 32 ) + A * B * ( C / 9 )

n2 =  ((A * B) / 25 ) + (A * B) * ( 2 + D / 2 )

n3 = A * B ( 2 + ( D + E ) / 3 )
```

Figure 127. Formula to estimate the size of the HISTORY data set

N1 through N3 and A though E used in these formulae mean as follows:

N1 Calculate N1, which can be calculated as in the formula.

N2 Calculate n2, which can be calculated as in the formula, for each database data set that is to be processed by HD Pointer Checker. N2 is the total of n2 of all database data set.

N3 Calculate N3 only when you export the contents of the HISTORY data set to a flat file (EXPORTABLE=YES) by Export Utility. Calculate n3, which can be calculated as in the formula, for each database data set that is to be processed by HD Pointer Checker. N3 is the total of n3 of all database data set.

A: Number of HD Pointer Checker processing days that are recorded in the HISTORY data set.

B: Number of HD Pointer Checker processing per day. If MULTIENT=NO is specified, it is always 1.

C: Number of database data sets processed by one HD Pointer Checker run.

D: Number of segment types contained in the database data set.

E: Sum of number of pointer types that segment types in the database data set have.

Migrating from DBT Version 2 Release 3 format

Figure 128 on page 371 presents a sample JCL for converting the HISTORY data set from DBT Version 2 Release 3 format to the required format.

If you want to change the optional attributes of the HISTORY data set, change them after migration.

```
//************************************************
//** ALLOCATE HISTORY DATA SET                  **
//************************************************
//ALLOC   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=A
//SYSPRINT  DD *
  DEFINE CLUSTER ( NAME(HIST.HISTORY) -
                   VOLUME(IMSDBT) CYLINDERS(3,2) -
                   INDEXED UNIQUE KEYS(23,0) -
                   SHAREOPTIONS(3,3) -
                   RECORDSIZE(240,240) -
                   CONTROLINTERVALSIZE(4096) ) -
         DATA    ( NAME(HIST.HISTORY.DATA) ) -
         INDEX   ( NAME(HIST.HISTORY.INDEX) )
/*
//************************************************
//** MIGRATE HISTORY DATA SET                   **
//************************************************
//MIGRATE EXEC PGM=FABGFMIG
//STEPLIB   DD DISP=SHR,DSN=HPS.SHPSLMD0
//OLDHIST   DD DISP=SHR,DSN=DBT.V230.HIST.HISTORY
//NEWHIST   DD DISP=OLD,DSN=HIST.HISTORY
```

Figure 128. Sample JCL for converting HISTORY data set

HISTIN data set

This section describes the HISTIN data set.

Function

This data set contains the user's description of the processing to be done by DB Historical Data Analyzer through the batch job.

Format

This data set usually resides in the input stream. However, it can also be defined 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.

This data set can contain one or more combinations of three types of control statements: PROC, DATABASE, and ENDPROC. Control statements can be coded as shown in Figure 129.

```
//HISTIN DD *
 PROC TYPE=SUMMARY
  DATABASE DB=DB01,DD=DD01
  DATABASE DB=DB02,DD=DD02
 ENDPROC
 PROC TYPE=ANALYSIS
  DATABASE MEMBER=APPL01
 ENDPROC
 PROC TYPE=DELETE
  DATABASE DB=DB03,DD=DD03,FROM=08102000,TO=08312000
 ENDPROC
 PROC TYPE=REORG
 ENDPROC
/*
```

Figure 129. Control statements (Example)

Control statement syntax

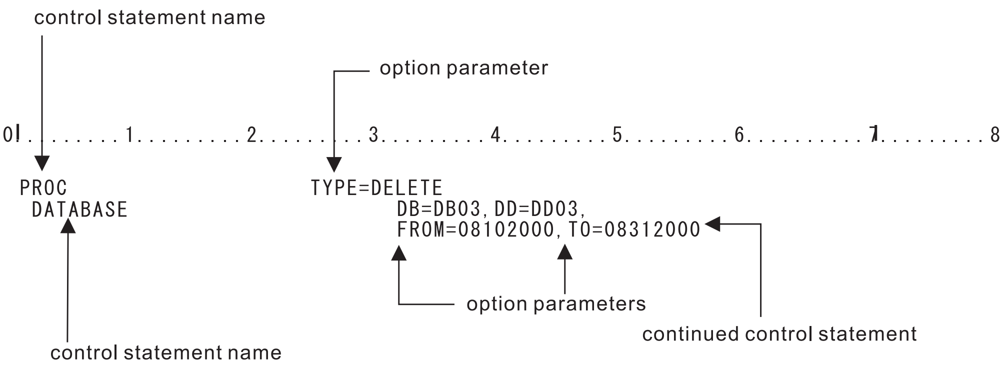
Figure 130 and the following description present the coding conventions that you must follow in writing control statements in HISTIN data set:

- A control statement can be coded onto one or more control statement records. Control statement names (PROC, DATABASE, and ENDPROC) and option parameters must be coded within column 2 and column 72. A control statement name must be the first entry in the control statement.
- Option parameters follow the control statement name, separated by one or more blanks. (ENDPROC has no option parameters.) A control statement name and the first option parameter must be written in the same control statement record. When more than one option parameter is coded, they must be separated by commas. No blanks are allowed between the option parameters and the commas, or within the option parameters.

  Option parameters can be continued onto more than one control statement record. In this case, the control statement that starts with a control statement name must be completed with an option parameter including a comma that follows it. The succeeding option parameters can be continued onto the following control statement records that begin in any column from column 2.

  Option parameters are not positional parameters; they can be specified in any order.

  A null value is not allowed for any option parameter.
- Comments may follow the last option parameter on each control statement record, separated by at least one blank.
- A comment statement must begin with an asterisk in column 1.



```
0|........1.........2.........3.........4.........5.........6.........7.........8
 PROC                    TYPE=DELETE
  DATABASE                      DB=DB03,DD=DD03,
                                FROM=08102000,TO=08312000
```

Figure 130. Sample control statement format in HISTIN data set

Notational conventions

The following symbols should be coded as they appear in the command format:

comma	,
equal sign	=

The following symbols are used to illustrate the command format:

Brackets []
Indicates optional choices, one of which may be selected.

Braces {}
Indicates choices, one of which ***must*** be selected.

Vertical bar |
Separates the operand alternatives within the brackets and braces.

PROC control statement

The PROC control statement specifies the *function* to be run. One or more PROC control statements can be specified at a time.

In terms of TYPE=SUMMARY, ANALYSIS, and DELETE, each PROC control statement may be followed by one or more DATABASE control statements. TYPE=UPDATE needs to be followed by an OPTION statement. Each PROC control statement must have a matching ENDPROC control statement.

The syntax of the PROC control statement is shown in Figure 131.

```
PROC  TYPE= { LIST | SUMMARY | ANALYSIS | DELETE | REORG | CREATE | UPDATE | ATTRLIST }
      [ ,KEYDATAFORM= { FULL | DATEONLY } ]
      [ ,IMSID= { (ims1,ims2,...) | *ALL } ]
```

Figure 131. PROC control statement syntax

TYPE=

Specifies the type of process to be performed.

LIST Specifies to generate two types of reports: the HISTORY Data Set by DB-DS report and HISTORY Data Set by Key Date report.

A DATABASE control statement is not required (if coded, it is ignored).

SUMMARY

Specifies to generate the HD Pointer Checker Summary report. This report summarizes the HD Pointer Checker run results for all the database data sets specified by the succeeding DATABASE control statements. The summary data for each database data set group is obtained from its last entry in the HISTORY data set, which is usually the result of the most recent HD Pointer Checker run for the database data set group.

When TYPE=SUMMARY is specified, FROM= and/or TO= parameters are not required for the associated DATABASE control statements (if coded, they are ignored). If DATABASE control statements are not specified, the report will contain information of all the database data sets that have entries in the HISTORY data set.

ANALYSIS

Specifies to generate the HD Analysis report for each database data set group specified by the succeeding DATABASE control statements. If DATABASE control statements are not specified, a report is printed for each database data set group that has entries in the HISTORY data set, using each of their latest entries.

DELETE

Specifies to delete entries from the HISTORY data set. At least one DATABASE control statement must follow.

REORG

Specifies to reorganize the HISTORY data set. A DATABASE control statement is not required (if coded, it is ignored).

CREATE

Specifies to initialize a new HISTORY data set. A DATABASE control statement is not required (if coded, it is ignored).

Notes:

1. The multiple entries option is inactive at initialization time. If you want to store multiple database data set entries per day, you need to run the FABGHIST program with the TYPE=UPDATE and OPTION MULTIENT=YES control statement.
2. The exportable option is inactive at initialization time. If you want to run Export Utility, you must run the FABGHIST program with the TYPE=UPDATE and OPTION EXPORTABLE=YES control statement.
3. The HISTORY lock option is set as GROUP at initialization. If you use more than one HISTORY data set among the HD Pointer Checker jobs, it is recommended that you change the HISTORY lock option by running the FABGHIST program with the TYPE=UPDATE and the OPTION HISTLOCK=DATASET control statement.
4. The Multiple-IMSID option is disabled at initialization time. If you want to use the IMSID parameter of the DBHDA PROC statement, you need to enable the Multiple-IMSID option by running the FABGHIST program with the TYPE=UPDATE and the OPTION MULTIIMSID=YES control statements.

UPDATE

Change the HISTORY data set option. The optional value is specified by the OPTION control statement.

ATTRLIST

Specifies to generate the HISTORY Attribute report. A DATABASE control statement is not required.

KEYDATAFORM=

Specifies the format of the data that will be printed in the KEYDATA fields of the HISTORY Data Set by Key Date report and the HISTORY Data Set by DB-DS report. This option is supported only for the TYPE=LIST process.

FULL

The data in the KEYDATA fields are shown with date and time. Each HISTORY data set record is shown in the report. This is the default.

DATEONLY

The data in the KEYDATA fields are shown with date only. When HISTORY data set records of the same DBNAME/DDNAME exist within the same day but with different time, they are shown as one entry in the report.

IMSID=

The report is sorted by IMSIDs. This parameter can be specified for TYPE=LIST, SUMMARY, ANALYSIS, or DELETE, and takes affect only when the Multiple-IMSID option is enabled. In case of TYPE=DELETE, you can specify the IMSIDs of the records in the HISTORY data set that are to be deleted.

(*ims1***,***ims2***,...)**

Specifies IMSIDs to be reported or to be deleted. You can specify up to 10 IMSIDs.

***ALL**
Records in the HISTORY data set of any IMSIDs are the target of the processing. This is the default when the Multiple-IMSID option is enabled.

DATABASE control statement

The DATABASE control statement specifies the database data set to be processed. A DATABASE control statement, if coded, must be preceded by a PROC control statement. The syntax of the DATABASE control statement is shown in Figure 132.

```
DATABASE { DB={dbname|ALL},DD={ddname|ALL}  |  MEMBER=member-name }
         [,FROM={mmddyyyy|mmddyy}]
         [,TO={mmddyyyy|mmddyy}]
         [,KEEP=ddd]
```

Figure 132. DATABASE control statement syntax

DB= Specifies the name of the DBD to be processed. **ALL** can be specified to process all the database data set groups that have entries in the HISTORY data set.

For HALDB, specify the master database name.

This parameter must be specified if MEMBER= option parameter is *not* specified. This parameter has no default value.

DD= Specifies the ddname of the database data set group to be processed. **ALL** can be specified to process all the database data set groups of the DBD specified by the DB= option parameter. If the database data set group consists of a primary data set and an overflow data set, either of them can be specified.

For HALDB, specify each ddname created by the concatenation of the partition name and the data set suffix character, A through J or X.

For the HALDB which is Online Reorganization (OLR) capable, you can specify the data set suffix character M through V or Y, as well as A through J or X.

Whichever suffix is specified, DB Historical Data Analyzer searches and processes both A and M sides of the history record entries. Therefore, the statistics in the report shows the side that was active when HD Pointer Checker created the history record entry.

This parameter must be specified if the DB= option parameter is specified. If DB=ALL is specified, DD=ALL must be specified. This parameter has no default value.

MEMBER=member-name
Specifies the member name of the SPMNMBR data set that contains one or more Space Monitor control statements. All the database data sets specified in the control statements of the member are processed. If non-IMS data sets are specified in the SPMNMBR data set, they are ignored.

This parameter must be specified if the DB= option parameter is *not* specified.

FROM=mmddyyyy or mmddyy

Specifies the date of the entry within this database data set group; the processing starts with this entry. If this parameter is not specified, processing starts with the first entry within this database data set group. If neither the "FROM=" nor "TO=" option parameter is specified, only the most recent entry is processed.

The FROM date must be either in the form "mmddyyyy" or "mmddyy" (mm=month, dd=date, yyyy or yy=year). IBM recommends to use mmddyyyy format for correct specification of a year later than 1999.

If TYPE=SUMMARY is specified for the preceding PROC control statement, this option parameter is not required (if coded, it is ignored).

TO=mmddyyyy or mmddyy

Specifies the date of the entry within this database data set group; the processing ends on this entry. If this parameter is not specified, processing ends on the last entry within this database data set group. If neither the "FROM=" nor "TO=" option parameter is specified, only the most recent entry is processed.

The TO date must be either in the form "mmddyyyy" or "mmddyy" (mm=month, dd=date, yyyy or yy=year). IBM recommends to use mmddyyyy format for correct specification of a year later than 1999.

If TYPE=SUMMARY is specified for the preceding PROC control statement, this option parameter is not required (if coded, it is ignored).

KEEP=ddd

Specifies the retention days of the history records of this database data set group. The minimum number is 1 and the maximum number is 999. Specify this parameter with TYPE=DELETE in the preceding PROC control statement. The FABGHIST program will delete the history records of which life time exceeds this retention days. For example, if the FABGHIST program runs with KEEP=10 and today is January 23th, the records created before January 13th will be deleted.

To delete the history records, you must specify either KEEP= or a set of FROM= and TO=.

If this parameter is specified when TYPE=DELETE is not specified, it is ignored.

OPTION control statement

An OPTION control statement specifies an option for the HISTORY data set. The OPTION control statement must be preceded by a TYPE=UPDATE control statement. The syntax of the OPTION control statement is shown in Figure 133.

```
OPTION [ MULTIENT= { YES | NO } ] [ ,EXPORTABLE= { YES | NO } ]
       [ ,HISTLOCK= { GROUP | DATASET } ]
       [ ,MULTIIMSID= { YES | NO } ]
```

Figure 133. OPTION control statement syntax

MULTIENT=

Specifies to store multiple database data set entries per day in the HISTORY data set. You can change the multiple entries option at any time of the day.

YES Activates the multiple database data set entries per day. After YES is specified, HD Pointer Checker keeps more than one database data set entries per day. The entries can be stored up to a maximum of 25 entries per day. The updates over 25th entries will not override or be recorded.

NO Deactivates the multiple database data set entries per day. HD Pointer Checker stores only one entry per day in the HISTORY data set. If there are more than one times of HD Pointer Checker run for the database data set with HISTORY=YES option in a day, the entry is overridden.

At the time the HISTORY data set created, MULTIENT is set to NO.

When multiple database data set entries are stored per day, DB Historical Database Analyzer does as follows:

- Batch job processes the multiple entries. Multiple data for one day will be shown in the DB Historical Data Analyzer reports.
- The ISPF GDDM interface does not process the multiple entries. The last data of the day is shown on the GDDM panel.
- Export Utility exports the multiple entries.

Notes:

1. Even if MULTIENT=NO is specified, the multiple entries already written are not removed from the HISTORY data set. If you need to delete the existing entries, run the FABGHIST program with the TYPE=DELETE control statement.
2. If some entries already exist for the day, and you specify MULTIENT=NO, the existing entries remain but the last entry is overridden by the update information.
3. There is a compatibility issue for selecting MULTIENT=YES. For details, see “Format (MULTIENT=YES/NO)” on page 11.

EXPORTABLE=
Specifies the HISTORY data set to be processed by Export Utility.

YES Change the HISTORY data set to exportable. When EXPORTABLE is set to YES, HD Pointer Checker creates the exportable history records. Export Utility processes the HISTORY data set, if EXPORTABLE is set to YES.

NO Change the HISTORY data set to unexportable. When EXPORTABLE is set to NO, HD Pointer Checker creates the unexportable history records, and Export Utility does not export data. If EXPORTABLE is set to NO, Export Utility does not process the HISTORY data set.

At the time the HISTORY data set created, EXPORTABLE is set to NO.

Notes:

1. Even if the HISTORY data set is exportable, Export Utility cannot process the history record entries, which were created during unexportable status.
2. Any of the existing history records are not deleted by changing this attribute. If you want to delete them, run the FABGHIST program with the TYPE=DELETE control statement.

3. Even if exportable records remain in the HISTORY data set, of which EXPORTABLE attribute is NO, Export Utility does not process the records.
4. There is a compatibility issue for selecting EXPORTABLE=YES. For details, see "Record types (EXPORTABLE=YES/NO)" on page 11.

HISTLOCK=

Specifies the HISTORY lock option. The HD Pointer Checker jobs are serialized to update a HISTORY data set by the ENQ and DEQ macros. The HISTORY lock option is an optional function for selecting a minor name of an ENQ macro for serialization of updating HISTORY data sets.

GROUP

HD Pointer Checker issues the ENQ macro with the following parameters:

```
QNAME (major name) = CL8'FAB@HIST'
RNAME (minor name) = CL8'FAB@HIST'
CONTROL TYPE       = EXCLUSIVE
SCOPE              = SYSTEMS
```

DATASET

HD Pointer Checker issues the ENQ macro with the following parameters:

```
QNAME (major name) = CL8'FAB@HIST'
RNAME (minor name) = CL44'(HISTORY data set name)'
CONTROL TYPE       = EXCLUSIVE
SCOPE              = SYSTEMS
```

When the HISTORY data set is created, HISTLOCK is set to GROUP.

If more than one HISTORY data set has the GROUP attribute, the HD Pointer Checker jobs issue the ENQ macro with the same RNAME. It makes all the HD Pointer Checker jobs get into the same queue, that is for a group HISTORY data sets, and only one HD Pointer Checker job can update the HISTORY data set at a time.

By specifying HISTLOCK=DATASET to the HISTORY data sets, the HD Pointer Checker jobs get into individual queue of each HISTORY data set, and update each HISTORY data set in parallel.

GROUP is for keeping backward compatibility with lower level of HD Pointer Checker. It is recommended that you select HISTLOCK=DATASET.

| **MULTIIMSID=**

| Specifies the Multiple-IMSID option.

| **YES**

| Changes the HISTORY data set to enable the Multiple-IMSID option. When the Multiple-IMSID option is enabled, the DBHDA report shows each HISTORY record entry sorted by IMSIDs. Only the HISTORY records that are stored by HDPC while Multiple-IMSID is enabled are reported. HISTORY data set can have up to 15 different IMSIDs.

| **NO**

| Changes the HISTORY data set to disable the Multiple-IMSID option. Only the HISTORY records that are stored by HDPC while Multiple-IMSID is disabled are reported. DBHDA reports the HISTORY

data set entries without sorting by IMSIDs. When HISTORY data set is created by the TYPE=CREATE option, the Multiple-IMSID option is disabled.

Notes:

1. When Multiple-IMSID options is enabled, DBHDA processes only the HISTORY data set records that are stored by HDPC while the option is enabled. When the option is disabled, DBHDA processes only the records that are stored while the option is disabled. This also applies to the DELETE operation.
2. There is a compatibility issue for selecting MULTIIMSID=YES. For details, see “Format (MULTIIMSID=YES/NO)” on page 12.

ENDPROC control statement

The ENDPROC control statement specifies the end of the user’s specification by a PROC control statement. Each PROC control statement must have an associated ENDPROC control statement. The syntax of the ENDPROC control statement is shown in Figure 134.

```
ENDPROC
```

Figure 134. ENDPROC control statement syntax

This control statement has no option parameters.

Control member data set (SPMNMBR)

This input data set is an OS-partitioned data set (PDS), whose members contain one or more control statements. Each control statement describes a database data set to be analyzed by DB Historical Data Analyzer.

This data set is the same as the one used by Space Monitor (SPMNMBR data set).

This data set is required when the MEMBER= option parameter of the DATABASE control statement in the HISTIN data set is specified.

Note: Control statements that define non-IMS data sets are ignored.

For a detailed description of the format of the control statements, refer to Part 5, “Space Monitor,” on page 489.

Output

This section describes the output that DB Historical Data Analyzer produces when it is run as a batch job.

HISTMSG data set

This section describes the HISTMSG data set.

Overview

This data set contains the HISTORY Data Set Message report.

The report is described in detail in the following section.

HISTORY Data Set Message report

This report shows the control statements specified in the HISTIN data set. The following control statements are reported:

- PROC statement
- DATABASE statement
- OPTION statement
- ENDPROC statement

This report also shows messages from the DB Historical Data Analyzer FABGHIST program.

Figure 135 shows a sample of the HISTORY Data Set Message report.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBHDA         "HISTORY DATA SET MESSAGE REPORT"                        PAGE:     1
5655-K53                                                  DATE: 07/10/2006  TIME: 17.37.24                         FABGHIST - V2.R2

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

 PROC TYPE=ATTRLIST
 ENDPROC

FABG1010I REQUESTED PROCESS ENDED NORMALLY (TYPE = ATTRLIST)
 PROC TYPE=ANALYSIS
 ENDPROC

FABG1010I REQUESTED PROCESS ENDED NORMALLY (TYPE = ANALYSIS)
 PROC TYPE=LIST
 ENDPROC

FABG1010I REQUESTED PROCESS ENDED NORMALLY (TYPE = LIST    )
 PROC TYPE=SUMMARY
 ENDPROC

FABG1010I REQUESTED PROCESS ENDED NORMALLY (TYPE = SUMMARY )

FABG1000I FABGHIST ENDED NORMALLY
```

Figure 135. HISTMSG—HISTORY Data Set Message report

HISTPRT data set

This section describes the HISTPRT data set.

Overview

This data set contains the following reports:

- HISTORY Data Set by Key Date report
- HISTORY Data Set by DB-DS report
- HD Pointer Checker Summary report
- HD Analysis report
- History Attribute report

Each of the above reports is described in detail in the following sections.

HISTORY Data Set by Key Date report

This report contains a list of all the entries in the HISTORY data set sorted by the key date and time.

DB Historical Data Analyzer generates this report when TYPE=LIST is specified for the PROC control statement in the HISTIN data set.

Figure 136 shows a sample of the HISTORY Data Set by Key Date report.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBHDA        "HISTORY DATA SET BY KEY DATE REPORT"                       PAGE:     1
5655-K53                                                   DATE: 07/10/2006  TIME: 17.37.25                         FABGHIST - V2.R2


KEY DATA      DBNAME   DDNAME         DBNAME   DDNAME         DBNAME   DDNAME         DBNAME   DDNAME         DBNAME   DDNAME
----------    -------- --------       -------- --------       -------- --------       -------- --------       -------- --------
07/10/2006    HDAMDB2  HDAMDS4        HISAMDB1 HISAMDS1       HISAMDB1 HISAMDS2       TPFOH1   TPFOH1AA       TPFOH1   TPFOH1AB
  17:33:53    TPFOH2   TPFOH2AA       TPFOH2   TPFOH2AX       TPFOH3   TPFOH3AA       TPFOX1   TPFOX1AA
```

Figure 136. HISTPRT—HISTORY Data Set by Key Date report

This report contains the following information:

KEY DATA
The date and time when the real database data set of the database was scanned by the HD Pointer Checker run. If an image copy data set is processed instead of the real database data set, the time stamp of the image copy data set is used.

DBNAME
The name of the DBD as coded in the NAME= keyword of the DBD macro in the DBD that was processed by the HD Pointer Checker run.

DDNAME
The ddname of the data set group.

(*) This indicator shows that the associated database data set was scanned, but that the HD Pointer Checker validation/evaluation has not been performed yet.

Attention

(E) This indicator shows that the entry is invalid and needs to be deleted by the PROC TYPE=DELETE statement. An invalid HISTORY data set entry may be created when HD Pointer Checker run fails while processing the database data set group, and a rerun is not done for the same database data set group on the same day.

HISTORY Data Set by DB-DS report

This report contains a list of all the entries of the HISTORY data set sorted by the database data set name.

DB Historical Data Analyzer generates this report when TYPE=LIST is specified for the PROC control statement in the HISTIN data set.

Figure 137 on page 382 shows a sample of the HISTORY Data Set by DB-DS report.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBHDA         "HISTORY DATA SET BY DB-DS REPORT"                         PAGE:     1
5655-K53                                                  DATE: 07/10/2006  TIME: 17.37.25                        FABGHIST - V2.R2

DBNAME   DDNAME      ENTRIES (SHOWN BY KEY DATA)
-------- --------    ---------------------------------------------------------------------------------------------------------------
HDAMDB2  HDAMDS4     07/10/2006-17:33:53
HISAMDB1 HISAMDS1    07/10/2006-17:33:53
HISAMDB1 HISAMDS2    07/10/2006-17:33:53
TPFOH1   TPFOH1AA    07/10/2006-17:33:53
TPFOH1   TPFOH1AB    07/10/2006-17:33:53
TPFOH2   TPFOH2AA    07/10/2006-17:33:53
TPFOH2   TPFOH2AX    07/10/2006-17:33:53
TPFOH3   TPFOH3AA    07/10/2006-17:33:53
TPFOX1   TPFOX1AA    07/10/2006-17:33:53
```

Figure 137. HISTPRT—HISTORY Data Set by DB-DS report

This report contains the following information:

DBNAME
The name of the DBD as coded in the NAME= keyword of the DBD macro in the DBD that was processed by the HD Pointer Checker run.

DDNAME
The ddname of the data set group.

ENTRIES (SHOWN BY KEY DATE)
The date and time when the real database data set of the database was scanned by the HD Pointer Checker run. If an image copy data set is processed instead of the real database data set, the time stamp of the image copy data set is used.

(*) This indicator shows that the associated database data set was scanned on this date, but that the HD Pointer Checker validation/evaluation has not been performed yet.

Attention

(E) This indicator shows that the entry is invalid and needs to be deleted by the PROC TYPE=DELETE statement. An invalid HISTORY data set entry may be created when HD Pointer Checker run fails while processing the database data set group, and a rerun is not done for the same database data set group on the same day.

HD Pointer Checker Summary report

This report summarizes the entire HD Pointer Checker run results. Two lines of information are produced for each database data set processed by the HD Pointer Checker run.

DB Historical Data Analyzer generates this report when TYPE=SUMMARY is specified for the PROC control statement in the HISTIN data set. The summary data for each database data set group is retrieved from its last entry in the HISTORY data set, which is usually the result of the *most recent* HD Pointer Checker run (with HISTORY=YES option) for the database data set group.

This report is also generated at the end of the HD Pointer Checker run with TYPE=ALL, SCAN, or CHECK option. For a detailed description of the runs, see the parameter description of “PROC statement” on page 73.

See Figure 138 on page 383 for the sample output of the HD Pointer Checker Summary report.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBHDA        "HD POINTER CHECKER SUMMARY REPORT"                    PAGE:     1
5655-K53                                                   DATE: 07/10/2006  TIME: 17.37.25                     FABGHIST - V2.R2

DBNAME/        DDNAME/          C-DATE/    D-DATE/    D-TIME/    CHK-DATE/  CHK-TIME/ DATA-SET SIZE     F-SPACE %/  DETECTED ERRORS
DB# DSG# DBLG# DB-ORGANIZATION  ACCM BLKSZ LRECL DBTYPE DEVICE   %SEGMS IN OFLW       CYL'S     BYTES     BYTES     TOTAL  UNKNOWN
--- ---- ----- ---------------  ---- ----- ----- ------------    -------------------  ----   ---------  ----------  -----  -------

HDAMDB2        HDAMDS4          07/06/2006 07/10/2006 17.33.53   07/10/2006 17.33.53                          8 %
N/A 01   N/A   HDAM             ESDS  1024  1017   REAL  3390     98                      5     2533376      215670     0         0

HISAMDB1       HISAMDS1         07/06/2006 07/10/2006 17.33.53   07/10/2006 17.33.53
N/A 01   N/A   HISAM            KSDS  8192   510   REAL  3390     99                                                    0         0

HISAMDB1       HISAMDS2         07/06/2006 07/10/2006 17.33.53   07/10/2006 17.33.53
N/A 01   N/A   HISAM       OFLW ESDS  8192   510   REAL  3390    N/A                                                    0         0

TPFOH1         TPFOH1AA         07/06/2006 07/10/2006 17.33.53   07/10/2006 17.33.53                          1 %
N/A  A   N/A   PHDAM            ESDS   512   505   REAL  3390     73                     27    10160128      147298     0         0

TPFOH1         TPFOH1AB         07/06/2006 07/10/2006 17.33.53   07/10/2006 17.33.53                         87 %
N/A  B   N/A   PHDAM            ESDS   512   505   REAL  3390      0                      1      375808      329850     0         0

TPFOH2         TPFOH2AA         07/06/2006 07/10/2006 17.33.53   07/10/2006 17.33.53                         10 %
N/A  A   N/A   PHIDAM           ESDS   512   505   REAL  3390      0                      6     2257408      241226     0         0

TPFOH2         TPFOH2AX         07/06/2006 07/10/2006 17.33.53   07/10/2006 17.33.53
N/A  X   N/A   PHIDAM IDX       KSDS   512    14   REAL  3390    N/A

TPFOH3         TPFOH3AA         07/06/2006 07/10/2006 17.33.53   07/10/2006 17.33.53                         37 %
N/A  A   N/A   PHDAM            ESDS   512   505   REAL  3390     65                      2      752128      280350     0         0

TPFOX1         TPFOX1AA         07/06/2006 07/10/2006 17.33.53   07/10/2006 17.33.53
N/A  A   N/A   PSINDEX          KSDS   512    54   REAL  3390    N/A
```

Figure 138. HISTPRT—HD Pointer Checker Summary report

This report contains the following information:

DBNAME
This is the name of the DBD as coded in the NAME= keyword of the DBD macro in the DBD that was processed by the HD Pointer Checker run.

DDNAME
The ddname of the data set group.

C-DATE
This is the date when the database data set was actually created (if available). If the specified database data set is an image copy data set, this is the date when the image copy database data set was created.

D-DATE, D-TIME
If the specified data set is a real database data set, then these are the date and time when HD Pointer Checker SCAN process was performed.

If the specified data set is an image copy database data set, then these are the date and time when the image copy data set was created.

CHK-DATE, CHK-TIME
These are the date and time when HD Pointer Checker CHECK process was performed.

DATA-SET SIZE
This is the data set size shown in cylinders (CYL'S) and in bytes (BYTES).

This field is applies to HDAM and HIDAM databases.

F-SPACE

This is the amount of reusable database free space according to the bit map block. The value is shown in bytes (BYTES) and in percentage (%).

This field is applies to HDAM and HIDAM databases.

DETECTED ERRORS

The total number of errors is shown under TOTAL. This includes the total number of unknown areas (T2 records), which is shown under UNKNOWN.

KEY is the error type indicator. It indicates an error of the index key check.

HASH is the error type indicator. It indicates an error found during the HASH check.

For a detailed description of the T2 record, index key check, and HASH check, see “Detecting errors in a database” on page 27.

DB#

This is the database number used to identify the database throughout the HD Pointer Checker run. This field always shows "N/A."

DSG#

This is the data set group number. It is the ordinal number of the data set group. 1 to 10 is shown for non-HALDB, A to J and X for HALDB.

DBLG#

This is the database logical group number which indicates that physical databases with the same database logical group number are logically related to each other. This number is used throughout the HD Pointer Checker run. This field always shows "N/A."

DB-ORGANIZATION

This is the IMS organization used for this database. One of the following is shown:

- SHISAM
- HISAM
- HISAM OFLW
- HDAM
- HIDAM
- HIDAM INDEX
- HIDAM INDEX OFLW
- 2NDARY INDX
- 2NDARY INDX OFLW
- SHR 2ND IDX
- SHR 2ND IDX OFLW
- PHDAM
- PHIDAM
- PHIDAM IDX
- PSINDEX

ACCM

This is the access method used for this database data set (OSAM, ESDS, or KSDS).

BLKSZ

This is the block size or control interval (CI) size of the database data set.

LRECL

This is the logical record length of the database data set.

DBTYPE

This is the type (REAL or IMGCPY) of database data set. **REAL** indicates the real database data set, and **IMGCPY** indicates the image copy data set. If an image copy is used as an input data set, and if it was more than 5 days old when HD Pointer Checker SCAN process was performed, an asterisk * is shown after IMGCPY.

DEVICE

This is the device type of the database data set. If an image copy data set is used, the type of the image copy data set (TAPE or DASD) is shown.

%SEGMS IN OFLW

This is the percentage of segments in the overflow area.

This field applies to HISAM, HDAM, HIDAM, PHDAM and PHIDAM databases.

HD Analysis report

This report contains concise information of HD Pointer Checker output that is maintained in the HISTORY data set. The major advantages of this report are:

- Volume of HD Pointer Checker output can be reduced.

  Many of the HD Pointer Checker reports are produced by options. The user, however, may request the HD Analysis report instead of requesting the other HD Pointer Checker reports to reduce the volume of output.
- Analysis of the HD Pointer Checker output becomes much easier.
- All essential data for each database data set is summarized in one to several pages, depending on the number of dependent segment types.

DB Historical Data Analyzer generates this report when TYPE=ANALYSIS is specified for the PROC control statement in the HISTIN data set. DB Historical Data Analyzer generates this report ***for each HALDB partition***, ***for each database data set group***, and ***for each date and time***. In other words, if the HISTORY data set has three entries for a database data set group within the period specified by the FROM= or TO= parameters of the associated DATABASE control statement, three reports are generated for the database data set group.

If the PROC control statement with TYPE=ANALYSIS option has no associated DATABASE control statements, DB Historical Data Analyzer generates this report for each database data set group in the HISTORY data set, using their latest entries in the HISTORY data set.

This report contains the following information on the specified database data set group of the specified date and of the previous date, if it exists:

- Report heading
- Database description
- HD Pointer Checker run time options
- Database data set space utilization
- HISAM statistics
- HIDAM index statistics or partitioned HIDAM (PHIDAM) index statistics
- Secondary index statistics or partitioned secondary index (PSINDEX) statistics
- Pointer validation
- Free space statistics
- Segment distribution statistics
- HD tuning statistics
- Segment and pointer occurrences

See Figure 139 on page 397 to Figure 144 on page 407 for the sample output of the HD Analysis report.

Note

The information to be provided on the report varies depending on the type of database data set group. For example, for a primary index database, only the information under the headings "REPORT HEADING", "DATABASE DESCRIPTION," "RUN TIME OPTION," "SPACE UTILIZATION," and "HIDAM INDEX STATISTICS" are generated.

In the following description, the term ***DB*** indicates database.

<REPORT HEADING>

Report heading applies to all database types. The following fields are included in the report heading:

DBNAME

This is the name of the DBD as coded in the NAME= keyword of the DBD macro.

DDNAME

This is the ddname of the this data set group.

DSG#

This is the data set group number. It is the ordinal number of the data set group. 1 to 10 is shown for non-HALDB, A to J or X for HALDB.

*** *database type* ***

This is the type of the database. One of the following is shown:

*** **HISAM DB** ***

HISAM database

*** **PRIME DB** ***

HDAM, HIDAM, PHDAM database, or the data set group A-J of PHIDAM database

*** **PRIMARY INDEX DB** ***

HIDAM index database, or the data set group X of PHIDAM database

*** **SECONDARY INDEX DB** ***

Secondary Index database, or PSINDEX database

SPECIFIED DATA

This is the data of which the date and time when the actual database data set was scanned by HD Pointer Checker or, if an image copy data set was used, when the image copy data set was created. Under this heading, actual data of each item is shown.

PREVIOUS DATA

This is the data of which the date and time of the ***previous*** HD Pointer Checker run when the actual database data set was scanned or, if an image copy data set was used, when the image copy data set was created. Under this heading, actual data of each item of the previous run is shown, if it exists in the HISTORY data set.

<DATABASE DESCRIPTION>

This section applies to all database types. The following fields are included in this section:

DATABASE ORGANIZATION

This is the IMS organization used for this database. One of the following is shown:

- SHISAM
- HISAM
- HDAM
- HIDAM
- HIDAM INDEX
- 2NDARY INDX
- SHR 2ND IDX
- PHDAM
- PHIDAM
- PHIDAM IDX
- PSINDEX

ACCESS METHOD
This is the access method used for this database data set. One of the following is shown:

- OSAM
- KSDS
- KSDS/ESDS
- ESDS

PRIME DD NAME
This is the ddname of the this data set group. This field applies to HISAM, HDAM, HIDAM, PHDAM, and PHIDAM databases.

INDEXED PRIME DB NAME
This is the name of the DBD (as coded in the NAME= keyword of the DBD macro in the DBD) of the prime database that is being indexed.

This field applies only to index databases.

INDEX DD NAME
This is the ddname (as coded in the DD1= keyword of the DATASET macro in the DBD) of the index database.

This field applies only to index databases.

OVERFLOW DD NAME
This is the ddname (as coded in the OVFLW= keyword of the DATASET macro in the DBD) of this data set group.

This field applies only to HISAM and index databases. The field is left blank if the database has no overflow data set.

INPUT DATABASE
This is the data set type of this database data set group (REAL, IMGCPY(TAPE), or IMGCPY(DASD)) used for the HD Pointer Checker run.

CREATION DATE
This is the date when the database data set group was actually created (if available). If the specified database data set is an image copy data set, this is the date when the image copy was created.

The following fields (REAL DATABASE DEVICE, BLOCK SIZE, LOGICAL RECORD LENGTH, and KEY LENGTH) are the data for the primary data set of the database data set group.

REAL DATABASE DEVICE
This is the type of the device on which the data set resides.

This field is left blank if an image copy data set is used.

BLOCK SIZE
This is the block size or control interval (CI) size of the database data set.

LOGICAL RECORD LENGTH
This is the logical record length of the database data set.

KEY LENGTH
This is the root key length of the database data set.

<ENVIRONMENT>
This section applies to all database types. The following field is included in this section:

IMSID
This is the IMSID of the IMS subsystem in which the database data set was scanned by HD Pointer Checker.

<RUN TIME OPTION>
This section applies to all database types. The following HD Pointer Checker run time options are shown:

- HASH
- EPSCHK
- IXKEYCHK
- HOMECHK
- INCORE
- T2CHK
- ZEROCTR

For a detailed description of the options, see the parameter description of "PROC statement" on page 73 and "OPTION statement" on page 87.

<SPACE UTILIZATION>
This section applies to all database types.

For HISAM and index databases, space utilization information is shown under the two headings (PRIME and OVFLOW). Under the heading **PRIME**, space utilization information of the primary data set is shown. Under the heading **OVFLOW**, space utilization information of the overflow data set is shown.

The following fields are included in this section:

TYPE
This is the space allocation type (TRKS: allocated by unit of track; CYLS: allocated by unit of cylinder).

PRIMARY ALLOCATION
This is the primary allocation in cylinders or in tracks.

SECONDARY ALLOCATION
This is the secondary allocation in cylinders or in tracks.

EXTENTS
This is the number of extents.

ALLOCATED SPACE (TRKS)
This is the total allocated space shown in the unit of tracks.

USED SPACE (%)
This is the percentage of the used space within the allocated space.

CI-SPLITS
This is the total number of CI splits detected.

This field applies only to the primary data sets of the secondary index databases.

CA-SPLITS
This is the total number of CA splits detected.

This field applies only to the primary data sets of the secondary index databases.

<HISAM (or HIDAM INDEX or SECONDARY INDEX) STATISTICS>

This section applies to HISAM databases, HIDAM index databases, secondary index databases, the data set group X of PHIDAM databases, and PSINDEX databases.

The following fields are the descriptions of the database data set:

TOTAL SEGMENTS IN PRIME DB

This is the number of database segments that were detected in the primary data set of the database.

DELETED SEGMENTS IN PRIME DB

This is the number of deleted segments that were detected in the primary data set of the database.

POINTERS IN PRIME DB (T6 RECORDS)

This is the number of pointers that were detected in the KSDS part of this index database.

This field applies only to index databases.

POINTERS IN PRIME DB (T8 RECORDS)

This is the number of pointers that were detected in the KSDS part of this HISAM data set group.

This field applies only to HISAM databases.

TOTAL SEGMENTS IN OVERFLOW DB

This is the number of database segments that were detected in the overflow data set of the database.

DELETED SEGMENTS IN OVERFLOW DB

This is the number of deleted segments that were detected in the overflow data set of the database.

POINTERS IN OVERFLOW DB (T7 RECORDS)

This is the number of pointers that were detected in the overflow (ESDS or OSAM) part of this index database.

This field applies only to index databases.

POINTERS IN OVERFLOW DB (T9 RECORDS)

This is the number of pointers that were detected in the overflow (ESDS or OSAM) part of this HISAM data set group.

This field applies only to HISAM databases.

TOTAL OSAM/ESDS BLOCKS/CIS IN DB

This is the number of blocks or control intervals in OSAM/ESDS index database overflow data set.

This field applies only to index databases.

<POINTER VALIDATION>

This section applies to HDAM, HIDAM, PHDAM, and PHIDAM databases. The following fields contain pointer validation and evaluation of the HD Pointer Checker run:

VALIDATION ERRORS IN SCAN PROCESSING

This is the total number of validation errors detected in SCAN process.

VALIDATION ERRORS IN CHECK PROCESSING
This is the total number of validation errors detected in CHECK process.

EVALUATION ERRORS IN CHECK PROCESSING
This is the total number of evaluation errors detected in CHECK process. If KEY is shown, it indicates that errors were detected during the index key check. If HASH is shown, it indicates that errors were detected during the HASH check.

TOTAL BLOCKS IN DATA SET
This is the number of database data set blocks or control intervals that were processed by HD Pointer Checker.

HIGH RBA IN HEX
This is the relative byte address of the last segment in the data set group shown in hexadecimal.

HIGH RBA IN DEC
This is the relative byte address of the last segment in the data set group shown in decimal.

% POINTERS VALIDATED IN MEMORY
This is the percentage of pointers validated by in-core check of the SCAN process. When the HASH check function has been selected, this field shows N/A.

<FREE SPACE STATISTICS>
This section applies to HDAM, HIDAM, PHDAM, and PHIDAM databases. The following fields contain statistical information on the free space:

FREE SPACE ELEMENTS
This is the number of free space elements in this data set group.

BLOCKS WITH SPACE (BITMAP)
This is the number of blocks with space from the viewpoint of bit maps, and its percentage within the data set in this data set group.

REUSABLE FREE SPACE (BITMAP)
This is the amount of reusable free space from the viewpoint of bit maps, and its percentage within the data set in this data set group.

Note: Reusable free space is a free space element equal to or larger than the longest segment in the data set (as defined in the DBD).

NOT REUSABLE FREE SPACE (BITMAP)
This is the amount of nonreusable free space from the viewpoint of bit maps, and its percentage within the data set in this data set group.

Note: Nonreusable free space is a free space element smaller than the longest segment in the data set (as defined in the DBD).

EMPTY BLOCKS
This is the number of empty blocks and its percentage within the data set in this data set group.

<SEGMENT DISTRIBUTION STATISTICS>
This section applies to HISAM, HDAM, HIDAM, PHDAM, and PHIDAM databases. The following fields contain statistical information on segment distribution:

SEGMENTS IN DATA SET
This is the number of segments in this data set group.

ROOT SEGMENTS
This is the number of root segments in this data set group.

DEPENDENT SEGMENTS
This is the number of dependent segments in this data set group.

ROOTS WITH NO DEPENDENTS IN SAME BLOCK
This is the number of root segments that have no dependent segment in the same block as their root segment, and its percentage within the total number of root segments that have dependent segments (in the case of an HISAM, it is the percentage within the total number of root segments).

This field applies to the primary data set groups of HISAM, HDAM, HIDAM, PHDAM, and PHIDAM databases.

When the data set group is processed by the HASH check function, this field shows N/A, because that function forces INCORE=NO.

AVG. COUNT OF DEPS IN SAME BLK AS ROOT
This is the average number of dependent segments that are in the same block as their root segment.

This field applies to the primary data set groups of HDAM, HIDAM, PHDAM, and PHIDAM databases.

When the data set group is processed by the HASH check function, this field shows N/A, because that function forces INCORE=NO.

DEPENDENTS NOT IN SAME BLOCK WITH ROOT
This is the number of dependent segments that are not in the same block as their root segment, and its percentage within the total number of dependent segments.

This field applies to the primary data set groups of HDAM, HIDAM, PHDAM, and PHIDAM databases.

When the data set group is processed by the HASH check function, this field shows N/A, because that function forces INCORE=NO.

AVG. DATABASE RECORD LENGTH
This is the average length of a database record (including prefix and data portions of all segments). This does not include segments that are in the secondary data set group of the database.

This field applies only to the primary data set groups of HISAM databases.

<HD TUNING STATISTICS>
The following fields contain HD tuning statistics information. This section applies to HDAM, HIDAM, PHDAM, and PHIDAM databases.

DIRECT ALGORITHM NAME
This is the name of the randomizing module, as coded in the RMNAME= keyword of the DBD macro.

This field applies to the primary data set groups of HDAM and PHDAM databases.

LONGEST SEGMENT IN DATA SET
This is the length of the longest segment in this data set group.

HIGH BLOCK NUMBER (IN RAA)
If HDAM, this is the maximum relative block number in the root addressable area (RAA) that the randomizing module is permitted to produce (as coded in the RMNAME= keyword of the DBD macro) in this database.

If HIDAM, this is the total number of blocks in the data set group.

This field applies to the primary data set groups of HDAM, HIDAM, PHDAM, and PHIDAM databases.

RAPS PER BLOCK

If HDAM, this is the number of root anchor points (RAPs) in each CI (or block) in the root addressable area (as coded in the RMNAME= keyword of the DBD macro) of this database.

If HIDAM, this field always shows 1.

This field applies to the primary data set groups of HDAM, HIDAM, PHDAM, and PHIDAM databases.

TOTAL RAPS

This is the product of HIGH BLOCK NUMBER and RAPS PER BLOCK.

TOTAL RAPS = (HIGH BLOCK NUMBER) × (RAPS PER BLOCK)

This field applies to the primary data set groups of HDAM, HIDAM, PHDAM, and PHIDAM databases.

BYTE LIMIT COUNT

This is the maximum number of bytes of a database record that can be stored into the root addressable area in a series of inserts unbroken by a call to another database record (as coded in the RMNAME= keyword of the DBD macro) of this HDAM and PHDAM databases. N/A indicates that no limit is placed for the maximum number of bytes of a database record.

This field applies to the primary data set groups of HDAM and PHDAM databases.

AVG. DATABASE RECORD LENGTH

This is the average length of a database record (including prefix and data portions of all segments). This does not include segments that are in another data set group of the database.

This field does not apply to the secondary data set group in either of the following conditions:

- HD Pointer Checker did not process the data set group with the primary data set group of the same database simultaneously.

  Or,
- IMS Image Copy Extensions processed the data set group.

FREE SPACE SCAN CYLINDERS

This is the number of direct access device cylinders to be scanned when searching for available storage space during segment insertion operations (as coded in the SCAN= keyword of the DATASET macro in the DBD) of this data.

This field applies to the primary data set groups of HDAM, HIDAM, PHDAM, and PHIDAM databases.

FSPC BLK. EVERY N BLKS

This is the "free block frequency factor" (as coded in the FRSPC= keyword of the DATASET macro in the DBD) of this data set group. It specifies that every Nth control interval or block in this data set group is left as free space during database load or reorganization.

This field applies to the primary data set groups of HDAM, HIDAM, PHDAM, and PHIDAM databases.

% FSPC WITHIN EACH BLK

This is the "free block frequency factor" (as coded in the FRSPC= keyword of the DATASET macro in the DBD) of this data set group. It specifies the minimum percentage of each control interval or block in this data set group that is left as free space during database load or reorganization.

This field applies to the primary data set groups of HDAM, HIDAM, PHDAM, and PHIDAM databases.

NO. KEY RECORDS WRITTEN

This is the number of records written by the SCAN process to the KEYSIN data set (for use by the HD Tuning Aid utility).

This field applies to the primary data set groups of HDAM, HIDAM, PHDAM, and PHIDAM databases.

ROOTS IN HOME BLOCK

This is the number of HDAM root segments that are stored in the same blocks as they are assigned by the randomizing routine, and its percentage within the data set of this data set group.

This field applies to the primary data set groups of HDAM and PHDAM databases.

When the data set group is processed by IMS Image Copy Extensions (IMS ICE), this field shows N/A, because the HASH check function invoked by IMS Image Copy Extensions forces HOMECHK=NO.

ROOTS 1 BLOCK AWAY

This is the number of HDAM root segments that are stored in the blocks that immediately precede or follow the blocks to which they are randomized, and its percentage within the data set of this data set group.

This field applies to the primary data set groups of HDAM and PHDAM databases.

When the data set group is processed by IMS Image Copy Extensions (IMS ICE), this field shows N/A, because the HASH check function invoked by IMS Image Copy Extensions forces HOMECHK=NO.

ROOTS BEYOND

This is the number of HDAM root segments that are stored in the blocks that are neither adjacent to nor the same as the blocks to which they are randomized, and its percentage within the data set of this data set group.

This field applies to the primary data set groups of HDAM and PHDAM databases.

When the data set group is processed by IMS Image Copy Extensions (IMS ICE), this field shows N/A, because the HASH check function invoked by IMS Image Copy Extensions forces HOMECHK=NO.

ROOTS IN OVERFLOW

This is the number of HDAM root segments that are stored in the blocks that are not in the root addressable area of the database, and its percentage within the data set of this data set group.

This field applies to the primary data set groups of HDAM and PHDAM databases.

When the data set group is processed by IMS Image Copy Extensions (IMS ICE), this field shows N/A, because the HASH check function invoked by IMS Image Copy Extensions forces HOMECHK=NO.

BLOCKS WITHOUT ROOT (IN RAA)

This is the total number of blocks (in RAA, in the case of HDAM) which has root anchor points but has no root segments, and its percentage within the data set of this data set group.

This field applies to the primary data set groups of HDAM, HIDAM, PHDAM, and PHIDAM databases.

AVG. COUNT OF ROOTS PER ACT. BLK IN RAA

This is the average count of root segments per active block in root addressable area of the HDAM data set.

This field applies to the primary data set groups of HDAM and PHDAM databases.

AVG. COUNT OF ROOTS PER ACTIVE RAP

This is the average count of root segments per active root anchor points of the HDAM data set.

This field applies to the primary data set groups of HDAM and PHDAM databases.

COUNT OF RAPS NOT USED

This is the total number of root anchor points that do not chain any root segments yet, and its percentage within the data set of this data set group.

This field applies to the primary data set groups of HDAM, HIDAM, PHDAM, and PHIDAM databases.

<SEGMENT AND POINTER OCCURRENCES>

This section applies to HISAM, HDAM, HIDAM, PHDAM, and PHIDAM databases. The following fields contain segment and pointer occurrence information:

SC

This is the segment code (in hexadecimal) of the segment.

SEGNAME

The segment name (as coded in the SEGM macro in the DBD) of the segment.

POINTER TYPES

This is the type of pointer (CTR, PTF, PTB, PP, LTF, LTB, LP, PCF, PCL, LCF, LCL, EPS *LP, *LC). The number shown under each pointer type indicates the count of each pointer type in the prefix of the segment. If value **X** is shown, it means only one pointer type for the segment type.

CTR Counter.

PTF Physical twin forward. If value **H** is shown, it means hierarchical twin forward.

PTB Physical twin backward. If value **H** is shown, it means hierarchical twin backward.

PP Physical parent.

LTF Logical twin forward.

LTB Logical twin backward.

LP Logical parent.

PCF Physical child first.

PCL Physical child last.

LCF Logical child first.

LCL Logical child last.

EPS Extended Pointer Set (HALDB)

The following two values show the existence of a logical relationship when no direct pointer exists:

***LP** The source segment is a logical child, and the target segment is its logical parent. There is no direct logical parent pointer. Instead, the source segment has a symbolic pointer (the logical parent concatenated key) to its logical parent.

***LC** The source segment is a logical parent, and the target segment is its logical child. There is no logical child pointer.

PAIRING
This field shows the logical relationship of the segment.

TP
Indicates the type of the logical relationship.

U indicates that the segment has an unidirectional logical relationship.

P indicates that the segment has a bidirectional, physically-paired logical relationship.

V indicates that the segment has a bidirectional, virtually-paired logical relationship.

TARGT
Identifies the target segment of the database with which the specified segment has a logical relationship.

DB is the database number (in hexadecimal) that identifies the target database. This number used to identify the database throughout the HD Pointer Checker run.

SC is the segment code (in hexadecimal) that identifies the target segment.

PRFX LGTH
This is the prefix length of the segment.

SEGM LGTH
This is the length of the segment (including prefix). If the segment has a variable length, it shows the average length of the segment and the character 'V' follows the length value.

ACTUAL MAX. SEGMENT LENGTH
This is the actual maximum length of the segment in the database data set (including prefix).

TOTAL OCCURRENCES
This is the total number of occurrences of the segment in the database data set.

When the HISAM primary and overflow data sets are processed by IMS Image Copy Extensions (IMS ICE), this field shows N/A, because IMS Image Copy Extensions processes each data set separately.

OCCURRENCE IN OVERFLW
This is the total number of occurrences of the segment that are in the overflow area.

When the HISAM primary and overflow data sets are processed by IMS Image Copy Extensions (IMS ICE), this field shows N/A, because IMS Image Copy Extensions processes each data set separately.

OCC/RT

This is the average number of occurrences of the segment per root.

For any of the following database data sets processed by IMS Image Copy Extensions (IMS ICE), this field shows N/A, because IMS Image Copy Extensions processes each data set separately:
- The HISAM primary and secondary data sets
- The secondary data set groups

COUNT OF POINTERS POINTING TO

This section applies to HDAM, HIDAM, PHDAM, and PHIDAM databases. This is the total number of pointers that point to the target segments that are in the following blocks:

SAME-BLK

This is the number of pointers that point to the blocks containing the pointer.

ADJ-BLKS

This is the number of pointers that point to the blocks that immediately precede or follow the blocks containing the pointer.

BEYOND

This is the number of pointers that point to the blocks (in the same data set) that are neither adjacent to nor the same as the blocks containing the pointer.

EXTERNAL

This is the number of pointers that point to the blocks in another data set.

TOTAL PTRS

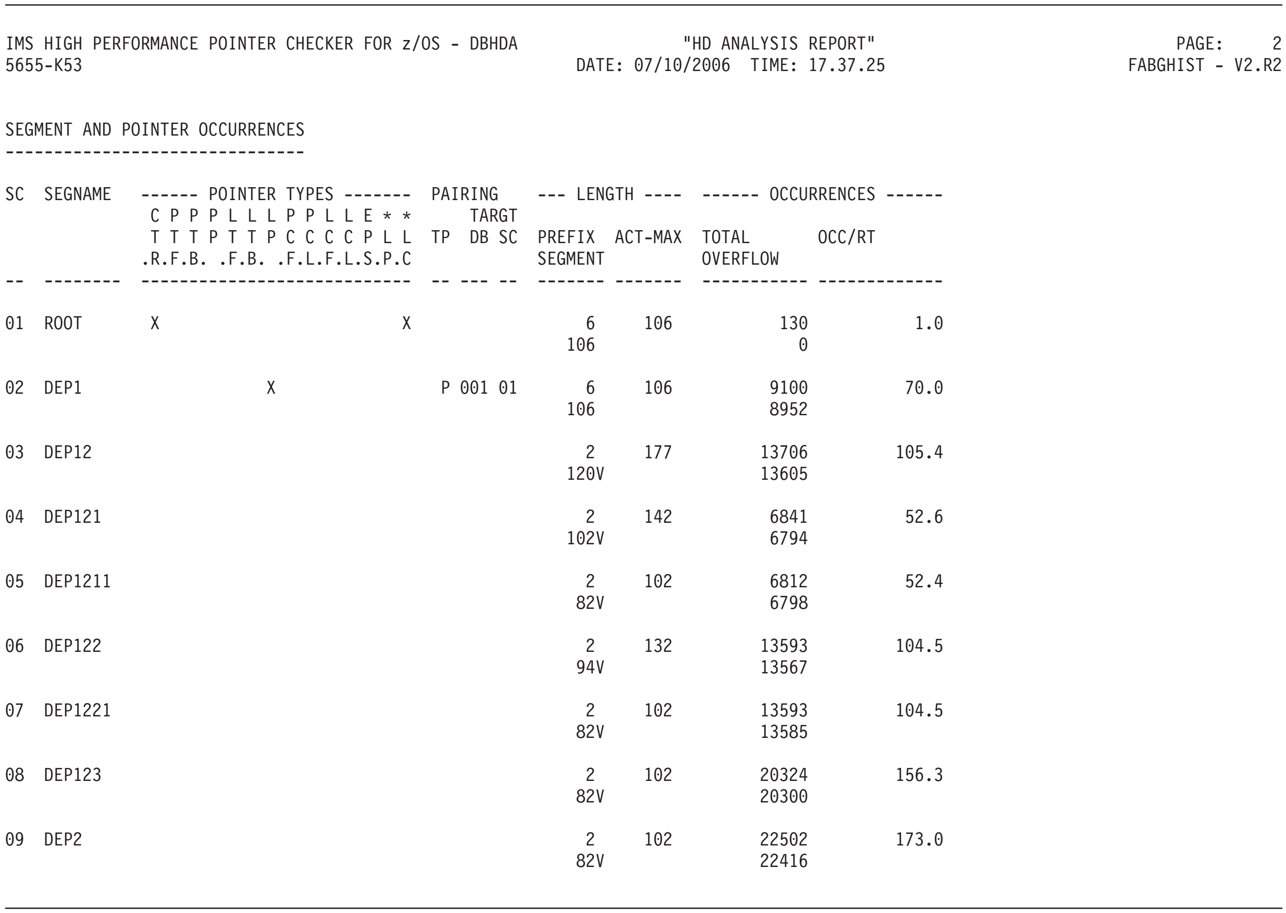
This is the total number of pointers that the specified segment contains.

This section applies to HISAM, HDAM, HIDAM, PHDAM, and PHIDAM databases.

Figure 139 on page 397 to Figure 144 on page 407 shows the sample output of the HD Analysis Report.

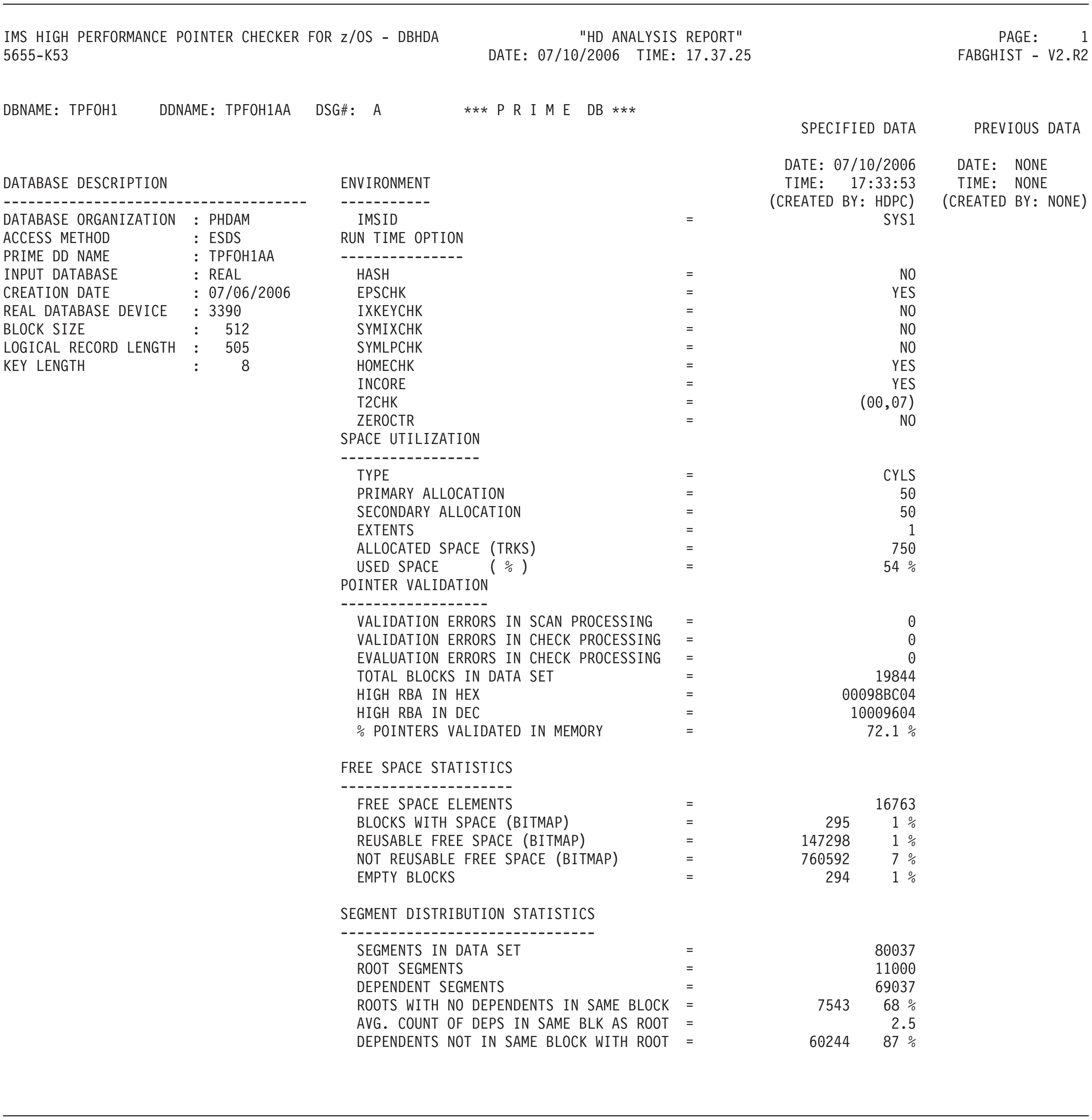
```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBHDA                 "HD ANALYSIS REPORT"                            PAGE:     1
5655-K53                                                   DATE: 07/10/2006  TIME: 17.37.25                         FABGHIST - V2.R2

DBNAME: HISAMDB1   DDNAME: HISAMDS1   DSG#: 01          *** H I S A M  DB ***
                                                                                                 SPECIFIED DATA       PREVIOUS DATA

                                                                                               DATE: 07/10/2006     DATE:  NONE
DATABASE DESCRIPTION                     ENVIRONMENT                                           TIME:   17:33:53     TIME:  NONE
-------------------------------------    -----------                                         (CREATED BY: HDPC)   (CREATED BY: NONE)
DATABASE ORGANIZATION  : HISAM             IMSID                                   =                       SYS1
ACCESS METHOD          : KSDS/ESDS       RUN TIME OPTION
PRIME DD NAME          : HISAMDS1        ---------------
OVERFLOW DD NAME       : HISAMDS2          HASH                                    =                         NO
INPUT DATABASE         : REAL              EPSCHK                                  =                        N/A
CREATION DATE          : 07/06/2006        IXKEYCHK                                =                        N/A
REAL DATABASE DEVICE   : 3390              SYMIXCHK                                =                         NO
BLOCK SIZE             :  8192             SYMLPCHK                                =                        YES
LOGICAL RECORD LENGTH  :   510             HOMECHK                                 =                        N/A
KEY LENGTH             :    10             INCORE                                  =                        N/A
                                           T2CHK                                   =                    (00,07)
                                           ZEROCTR                                 =                         NO
                                         SPACE UTILIZATION
                                         -----------------                                        PRIME  OVFLOW
                                           TYPE                                    =               CYLS    CYLS
                                           PRIMARY ALLOCATION                      =                 50      50
                                           SECONDARY ALLOCATION                    =                 50      20
                                           EXTENTS                                 =                  1       1
                                           ALLOCATED SPACE (TRKS)                  =                750     750
                                           USED SPACE      ( % )                   =                0 %    29 %
                                           CI-SPLITS                               =                  0
                                           CA-SPLITS                               =                  0
                                         POINTER VALIDATION
                                         ------------------
                                           EVALUATION ERRORS IN CHECK PROCESSING   =                          0

                                         HISAM STATISTICS
                                         ----------------
                                           TOTAL SEGMENTS IN PRIME DB              =                        584
                                           DELETED SEGMENTS IN PRIME DB            =                          0
                                           POINTERS IN PRIME DB (T8 RECORDS)       =                        278
                                           TOTAL SEGMENTS IN OVERFLOW DB           =                     106017
                                           DELETED SEGMENTS IN OVERFLOW DB         =                          0
                                           POINTERS IN OVERFLOW DB (T9 RECORDS)    =                      30082

                                         SEGMENT DISTRIBUTION STATISTICS
                                         -------------------------------
                                           SEGMENTS IN DATA SET                    =                     106601
                                           ROOT SEGMENTS                           =                        130
                                           DEPENDENT SEGMENTS                      =                     106471
                                           ROOTS WITH NO DEPENDENTS IN SAME BLOCK  =                  0     0 %
                                           AVG. DATABASE RECORD LENGTH             =                      74927
```

Figure 139. HISTPRT—HD Analysis report (HISAM database) (Part 1 of 2)



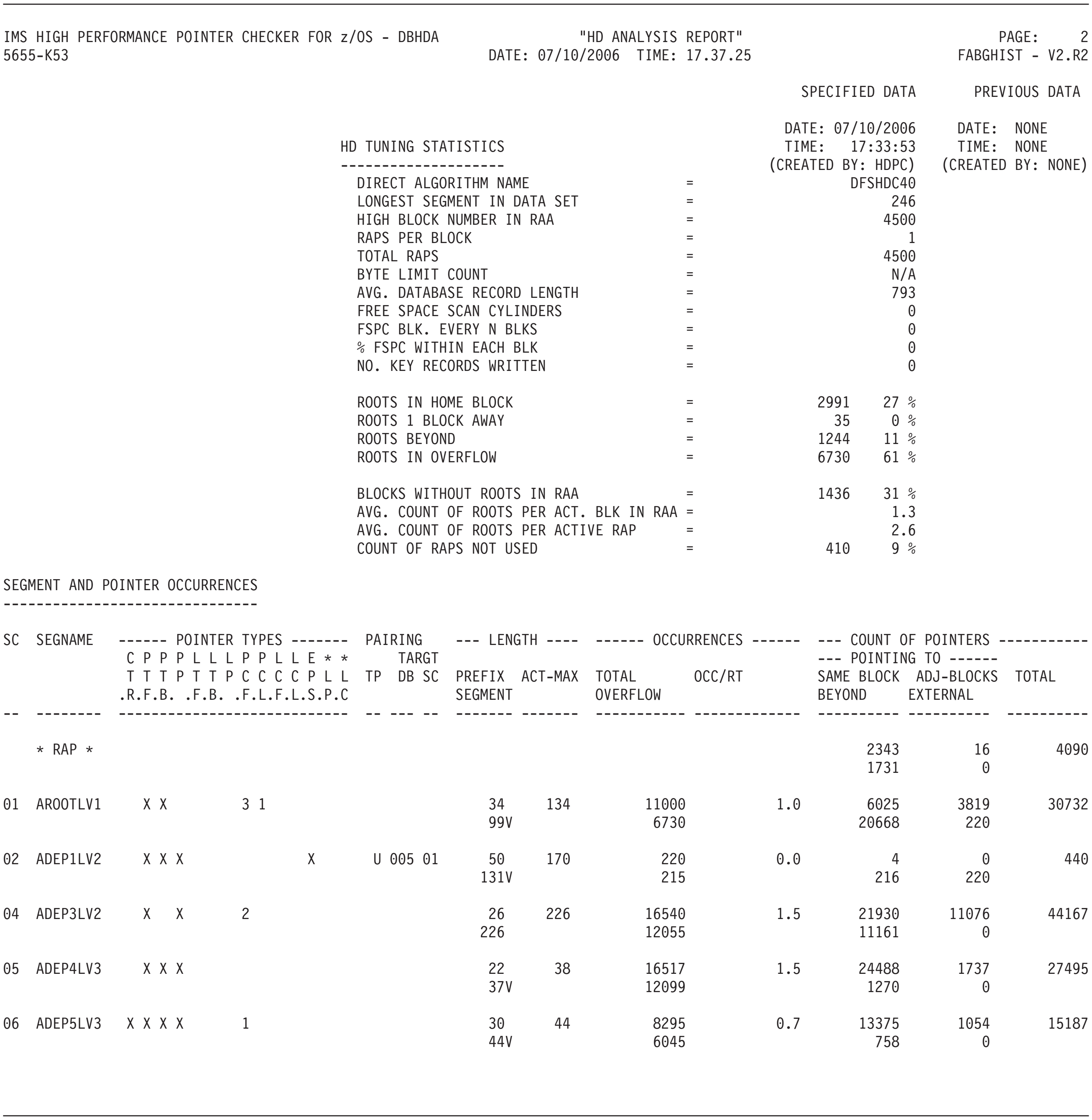
```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBHDA                 "HD ANALYSIS REPORT"                               PAGE:      2
5655-K53                                                   DATE: 07/10/2006  TIME: 17.37.25                         FABGHIST - V2.R2

SEGMENT AND POINTER OCCURRENCES
-------------------------------

SC  SEGNAME   ------ POINTER TYPES -------  PAIRING    --- LENGTH ----  ------ OCCURRENCES ------
               C P P P L L L P P L L E * *      TARGT
               T T T P T T P C C C C P L L  TP  DB SC  PREFIX  ACT-MAX  TOTAL       OCC/RT
              .R.F.B. .F.B. .F.L.F.L.S.P.C             SEGMENT          OVERFLOW
--  --------  ----------------------------  -- --- --  ------- -------  ----------- -------------

01  ROOT       X                         X                  6     106           130           1.0
                                                          106                     0

02  DEP1                   X                 P 001 01       6     106          9100          70.0
                                                          106                  8952

03  DEP12                                                   2     177         13706         105.4
                                                          120V                13605

04  DEP121                                                  2     142          6841          52.6
                                                          102V                 6794

05  DEP1211                                                 2     102          6812          52.4
                                                           82V                 6798

06  DEP122                                                  2     132         13593         104.5
                                                           94V                13567

07  DEP1221                                                 2     102         13593         104.5
                                                           82V                13585

08  DEP123                                                  2     102         20324         156.3
                                                           82V                20300

09  DEP2                                                    2     102         22502         173.0
                                                           82V                22416
```

Figure 139. HISTPRT—HD Analysis report (HISAM database) (Part 2 of 2)



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBHDA             "HD ANALYSIS REPORT"                             PAGE:     1
5655-K53                                                  DATE: 07/10/2006  TIME: 17.37.25                      FABGHIST - V2.R2

DBNAME: TPFOH1     DDNAME: TPFOH1AA   DSG#:  A         *** P R I M E  DB ***
                                                                                         SPECIFIED DATA       PREVIOUS DATA

                                                                                       DATE: 07/10/2006     DATE:  NONE
DATABASE DESCRIPTION                    ENVIRONMENT                                    TIME:   17:33:53     TIME:  NONE
-------------------------------------   -----------                                  (CREATED BY: HDPC)   (CREATED BY: NONE)
DATABASE ORGANIZATION  : PHDAM            IMSID                                 =                  SYS1
ACCESS METHOD          : ESDS           RUN TIME OPTION
PRIME DD NAME          : TPFOH1AA       ---------------
INPUT DATABASE         : REAL             HASH                                  =                    NO
CREATION DATE          : 07/06/2006       EPSCHK                                =                   YES
REAL DATABASE DEVICE   : 3390             IXKEYCHK                              =                    NO
BLOCK SIZE             :   512            SYMIXCHK                              =                    NO
LOGICAL RECORD LENGTH  :   505            SYMLPCHK                              =                    NO
KEY LENGTH             :     8            HOMECHK                               =                   YES
                                          INCORE                                =                   YES
                                          T2CHK                                 =               (00,07)
                                          ZEROCTR                               =                    NO
                                        SPACE UTILIZATION
                                        -----------------
                                          TYPE                                  =                  CYLS
                                          PRIMARY ALLOCATION                    =                    50
                                          SECONDARY ALLOCATION                  =                    50
                                          EXTENTS                               =                     1
                                          ALLOCATED SPACE (TRKS)                =                   750
                                          USED SPACE      ( % )                 =                  54 %
                                        POINTER VALIDATION
                                        ------------------
                                          VALIDATION ERRORS IN SCAN PROCESSING  =                     0
                                          VALIDATION ERRORS IN CHECK PROCESSING =                     0
                                          EVALUATION ERRORS IN CHECK PROCESSING =                     0
                                          TOTAL BLOCKS IN DATA SET              =                 19844
                                          HIGH RBA IN HEX                       =             00098BC04
                                          HIGH RBA IN DEC                       =              10009604
                                          % POINTERS VALIDATED IN MEMORY        =                72.1 %

                                        FREE SPACE STATISTICS
                                        ---------------------
                                          FREE SPACE ELEMENTS                   =                 16763
                                          BLOCKS WITH SPACE (BITMAP)            =           295     1 %
                                          REUSABLE FREE SPACE (BITMAP)          =        147298     1 %
                                          NOT REUSABLE FREE SPACE (BITMAP)      =        760592     7 %
                                          EMPTY BLOCKS                          =           294     1 %

                                        SEGMENT DISTRIBUTION STATISTICS
                                        -------------------------------
                                          SEGMENTS IN DATA SET                  =                 80037
                                          ROOT SEGMENTS                         =                 11000
                                          DEPENDENT SEGMENTS                    =                 69037
                                          ROOTS WITH NO DEPENDENTS IN SAME BLOCK =         7543    68 %
                                          AVG. COUNT OF DEPS IN SAME BLK AS ROOT =                  2.5
                                          DEPENDENTS NOT IN SAME BLOCK WITH ROOT =        60244    87 %
```

Figure 140. HISTPRT—HD Analysis report (PHDAM database primary data set group) (Part 1 of 3)



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBHDA                 "HD ANALYSIS REPORT"                               PAGE:     2
5655-K53                                                   DATE: 07/10/2006  TIME: 17.37.25                         FABGHIST - V2.R2
                                                                                                 SPECIFIED DATA       PREVIOUS DATA

                                                                                               DATE: 07/10/2006     DATE:  NONE
                                         HD TUNING STATISTICS                                  TIME:    17:33:53     TIME:  NONE
                                         --------------------                                (CREATED BY: HDPC)   (CREATED BY: NONE)
                                           DIRECT ALGORITHM NAME                   =                   DFSHDC40
                                           LONGEST SEGMENT IN DATA SET             =                        246
                                           HIGH BLOCK NUMBER IN RAA                =                       4500
                                           RAPS PER BLOCK                          =                          1
                                           TOTAL RAPS                              =                       4500
                                           BYTE LIMIT COUNT                        =                        N/A
                                           AVG. DATABASE RECORD LENGTH             =                        793
                                           FREE SPACE SCAN CYLINDERS               =                          0
                                           FSPC BLK. EVERY N BLKS                  =                          0
                                           % FSPC WITHIN EACH BLK                  =                          0
                                           NO. KEY RECORDS WRITTEN                 =                          0

                                           ROOTS IN HOME BLOCK                     =               2991   27 %
                                           ROOTS 1 BLOCK AWAY                      =                 35    0 %
                                           ROOTS BEYOND                            =               1244   11 %
                                           ROOTS IN OVERFLOW                       =               6730   61 %

                                           BLOCKS WITHOUT ROOTS IN RAA             =               1436   31 %
                                           AVG. COUNT OF ROOTS PER ACT. BLK IN RAA =                        1.3
                                           AVG. COUNT OF ROOTS PER ACTIVE RAP      =                        2.6
                                           COUNT OF RAPS NOT USED                  =                410    9 %

SEGMENT AND POINTER OCCURRENCES
-------------------------------

SC  SEGNAME   ------ POINTER TYPES -------  PAIRING    --- LENGTH ----  ------ OCCURRENCES ------  --- COUNT OF POINTERS -----------
               C P P P L L L P P L L E * *      TARGT                                              --- POINTING TO ------
               T T T P T T P C C C C P L L  TP  DB SC  PREFIX  ACT-MAX  TOTAL       OCC/RT         SAME BLOCK  ADJ-BLOCKS  TOTAL
              .R.F.B. .F.B. .F.L.F.L.S.P.C             SEGMENT          OVERFLOW                   BEYOND     EXTERNAL
--  --------  ----------------------------  -- --- --  ------- -------  ----------- ------------  ---------- ----------  ----------

    * RAP *                                                                                                  2343         16       4090
                                                                                                             1731          0

01  AROOTLV1     X X       3 1                            34     134       11000           1.0        6025       3819      30732
                                                         99V                6730                     20668        220

02  ADEP1LV2     X X X               X       U 005 01       50     170         220           0.0           4          0        440
                                                          131V                 215                       216        220

04  ADEP3LV2     X   X       2                              26     226       16540           1.5       21930      11076      44167
                                                           226               12055                     11161          0

05  ADEP4LV3     X X X                                      22      38       16517           1.5       24488       1737      27495
                                                           37V               12099                      1270          0

06  ADEP5LV3   X X X X       1                              30      44        8295           0.7       13375       1054      15187
                                                           44V                6045                       758          0
```

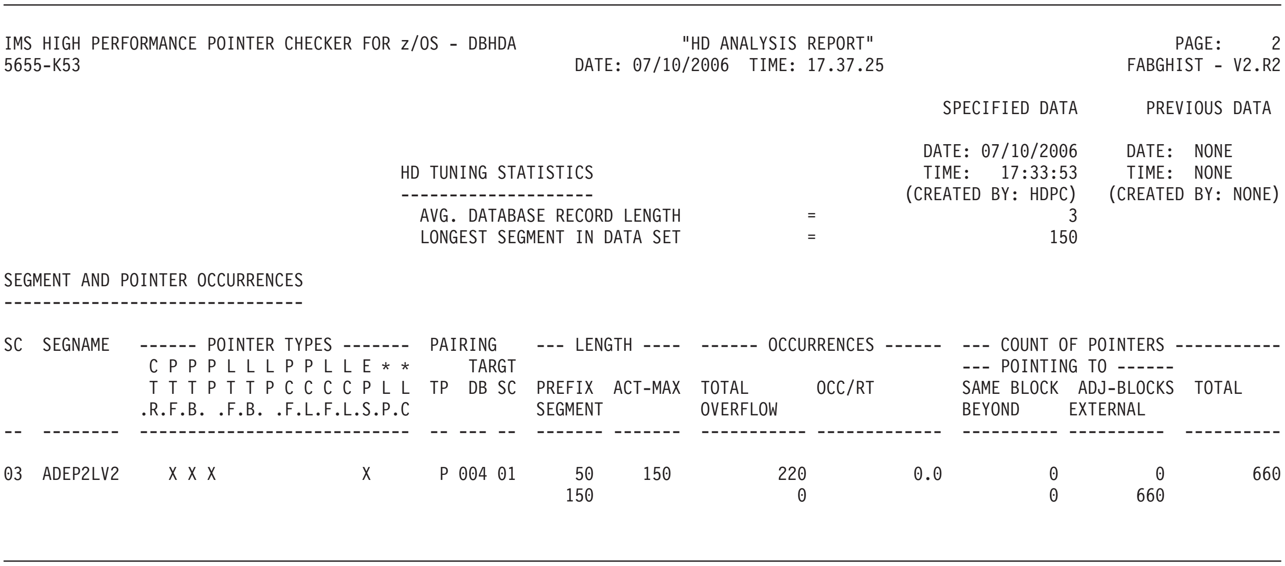
Figure 140. HISTPRT—HD Analysis report (PHDAM database primary data set group) (Part 2 of 3)

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBHDA                 "HD ANALYSIS REPORT"                                   PAGE:     3
5655-K53                                                   DATE: 07/10/2006  TIME: 17.37.25                                FABGHIST - V2.R2

SEGMENT AND POINTER OCCURRENCES
-------------------------------

SC  SEGNAME   ------ POINTER TYPES -------  PAIRING    --- LENGTH ----  ------ OCCURRENCES ------  --- COUNT OF POINTERS -----------
               C P P P L L L P P L L E * *      TARGT                                              --- POINTING TO ------
               T T T P T T P C C C C P L L  TP  DB SC  PREFIX  ACT-MAX  TOTAL       OCC/RT         SAME BLOCK  ADJ-BLOCKS  TOTAL
              .R.F.B. .F.B. .F.L.F.L.S.P.C             SEGMENT          OVERFLOW                   BEYOND     EXTERNAL
--  --------  ----------------------------  -- --- --  ------- -------  ----------- -------------  ---------- ----------  ----------

07  ADEP6LV4     X   X       1 1                           26     226        20573           1.8       22316       6711       48298
                                                          58V                15081                     19271          0

08  ADEP7LV5     X   X               X       U 003 06      46     246         6892           0.6           0          0       13784
                                                          246                 6892                      6892       6892
```

Figure 140. HISTPRT—HD Analysis report (PHDAM database primary data set group) (Part 3 of 3)

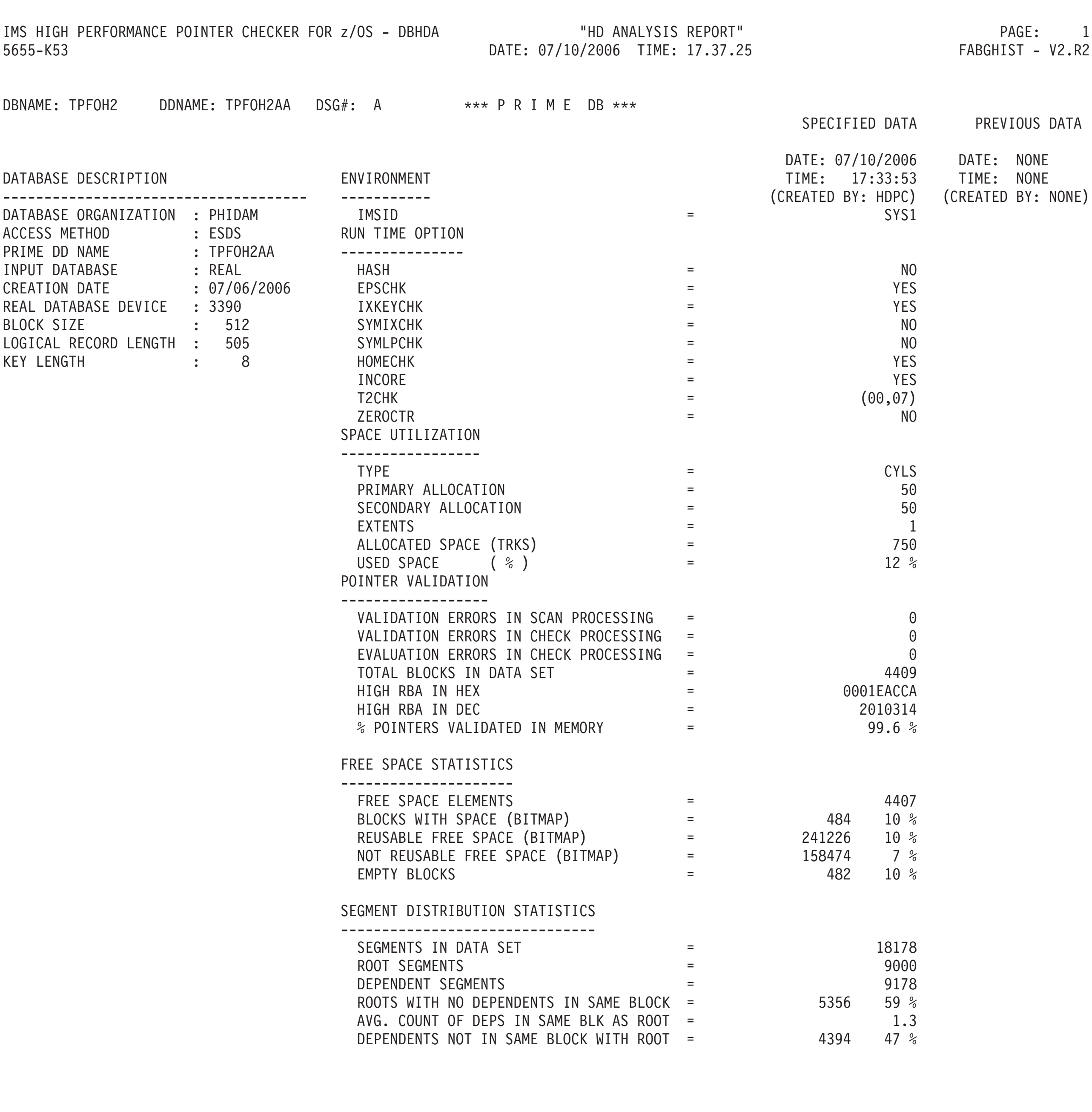
```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBHDA              "HD ANALYSIS REPORT"                               PAGE:     1
5655-K53                                                 DATE: 07/10/2006  TIME: 17.37.25                             FABGHIST - V2.R2


DBNAME: TPFOH1     DDNAME: TPFOH1AB   DSG#:  B         *** P R I M E  DB ***
                                                                                         SPECIFIED DATA      PREVIOUS DATA

                                                                                       DATE: 07/10/2006     DATE:  NONE
DATABASE DESCRIPTION                    ENVIRONMENT                                    TIME:   17:33:53     TIME:  NONE
-------------------------------------   -----------                                  (CREATED BY: HDPC)   (CREATED BY: NONE)
DATABASE ORGANIZATION  : PHDAM            IMSID                              =                    SYS1
ACCESS METHOD          : ESDS           RUN TIME OPTION
PRIME DD NAME          : TPFOH1AB       ---------------
INPUT DATABASE         : REAL             HASH                               =                      NO
CREATION DATE          : 07/06/2006       EPSCHK                             =                     YES
REAL DATABASE DEVICE   : 3390             IXKEYCHK                           =                      NO
BLOCK SIZE             :   512            SYMIXCHK                           =                      NO
LOGICAL RECORD LENGTH  :   505            SYMLPCHK                           =                      NO
KEY LENGTH             :     8            HOMECHK                            =                     YES
                                          INCORE                             =                     YES
                                          T2CHK                              =                 (00,07)
                                          ZEROCTR                            =                      NO
                                        SPACE UTILIZATION
                                        -----------------
                                          TYPE                               =                    CYLS
                                          PRIMARY ALLOCATION                 =                      50
                                          SECONDARY ALLOCATION               =                      50
                                          EXTENTS                            =                       1
                                          ALLOCATED SPACE (TRKS)             =                     750
                                          USED SPACE      ( % )              =                     2 %
                                        POINTER VALIDATION
                                        ------------------
                                          VALIDATION ERRORS IN SCAN PROCESSING    =                  0
                                          VALIDATION ERRORS IN CHECK PROCESSING   =                  0
                                          EVALUATION ERRORS IN CHECK PROCESSING   =                  0
                                          TOTAL BLOCKS IN DATA SET                =                734
                                          HIGH RBA IN HEX                         =          000009604
                                          HIGH RBA IN DEC                         =              38404
                                          % POINTERS VALIDATED IN MEMORY          =              0.0 %

                                        FREE SPACE STATISTICS
                                        ---------------------
                                          FREE SPACE ELEMENTS                =                     733
                                          BLOCKS WITH SPACE (BITMAP)         =             660    89 %
                                          REUSABLE FREE SPACE (BITMAP)       =          329850    87 %
                                          NOT REUSABLE FREE SPACE (BITMAP)   =            3650     0 %
                                          EMPTY BLOCKS                       =             659    89 %

                                        SEGMENT DISTRIBUTION STATISTICS
                                        -------------------------------
                                          SEGMENTS IN DATA SET                    =                220
                                          ROOT SEGMENTS                           =                N/A
                                          DEPENDENT SEGMENTS                      =                220
                                          ROOTS WITH NO DEPENDENTS IN SAME BLOCK  =                N/A
                                          AVG. COUNT OF DEPS IN SAME BLK AS ROOT  =                N/A
                                          DEPENDENTS NOT IN SAME BLOCK WITH ROOT  =                N/A
```

Figure 141. HISTPRT—HD Analysis report (PHDAM database secondary data set group) (Part 1 of 2)



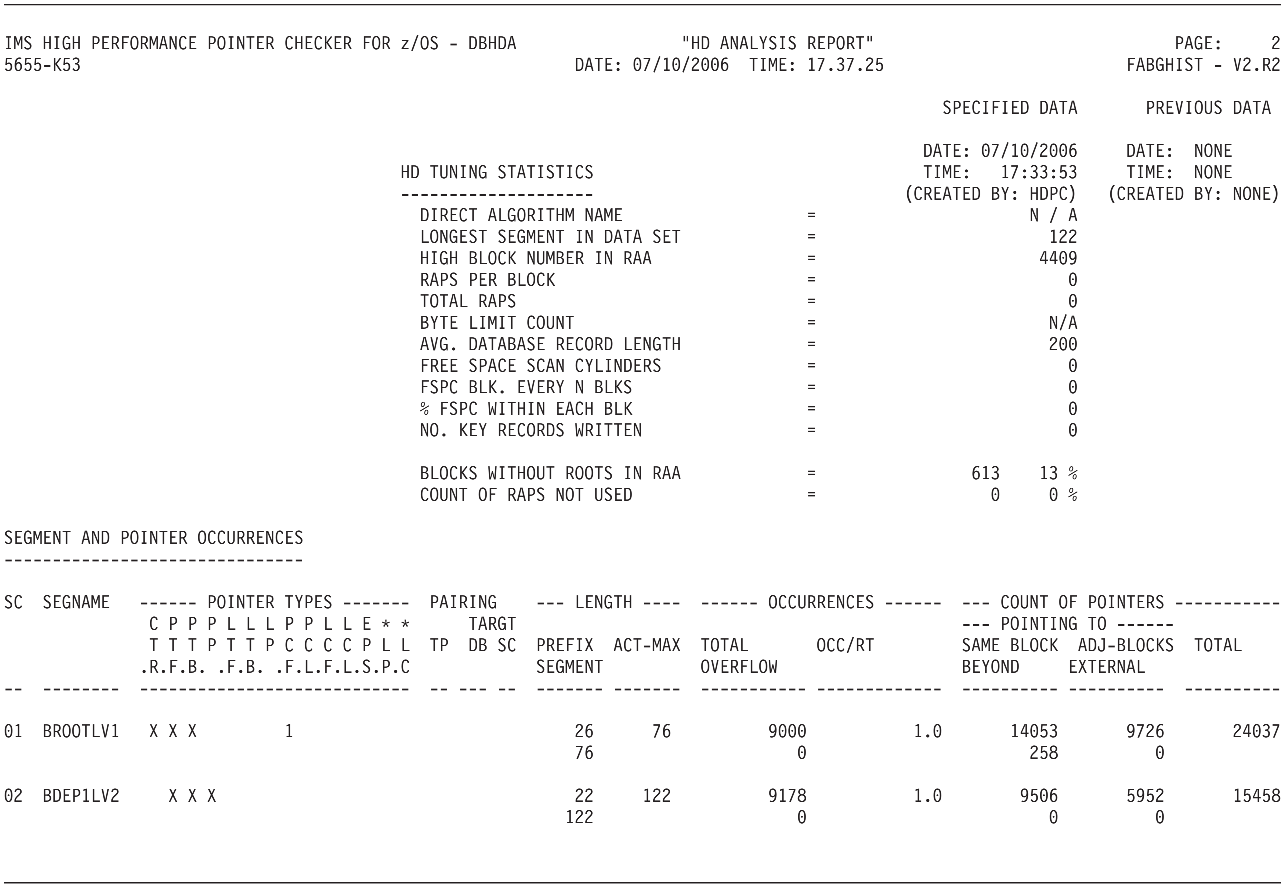
```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBHDA                 "HD ANALYSIS REPORT"                               PAGE:     2
5655-K53                                                   DATE: 07/10/2006  TIME: 17.37.25                         FABGHIST - V2.R2

                                                                                                 SPECIFIED DATA       PREVIOUS DATA

                                                                                               DATE: 07/10/2006     DATE:  NONE
                                         HD TUNING STATISTICS                                  TIME:   17:33:53     TIME:  NONE
                                         --------------------                                (CREATED BY: HDPC)   (CREATED BY: NONE)
                                           AVG. DATABASE RECORD LENGTH             =                          3
                                           LONGEST SEGMENT IN DATA SET             =                        150

SEGMENT AND POINTER OCCURRENCES
-------------------------------

SC  SEGNAME   ------ POINTER TYPES -------  PAIRING    --- LENGTH ----  ------ OCCURRENCES ------  --- COUNT OF POINTERS -----------
               C P P P L L L P P L L E * *      TARGT                                                --- POINTING TO ------
              T T T P T T T P C C C C P L L  TP  DB SC  PREFIX  ACT-MAX  TOTAL       OCC/RT         SAME BLOCK  ADJ-BLOCKS  TOTAL
              .R.F.B. .F.B. .F.L.F.L.S.P.C             SEGMENT          OVERFLOW                   BEYOND     EXTERNAL
--  --------  ----------------------------  -- --- --  ------- -------  ----------- -------------  ---------- ----------  ----------

03  ADEP2LV2     X X X               X       P 004 01      50     150           220           0.0           0          0         660
                                                          150                     0                         0        660
```

Figure 141. HISTPRT—HD Analysis report (PHDAM database secondary data set group) (Part 2 of 2)



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBHDA                 "HD ANALYSIS REPORT"                                PAGE:     1
5655-K53                                                   DATE: 07/10/2006  TIME: 17.37.25                          FABGHIST - V2.R2

DBNAME: TPFOH2     DDNAME: TPFOH2AA   DSG#:  A          *** P R I M E  DB ***
                                                                                                 SPECIFIED DATA       PREVIOUS DATA

                                                                                               DATE: 07/10/2006     DATE:  NONE
DATABASE DESCRIPTION                     ENVIRONMENT                                           TIME:   17:33:53     TIME:  NONE
-------------------------------------    -----------                                         (CREATED BY: HDPC)   (CREATED BY: NONE)
DATABASE ORGANIZATION  : PHIDAM            IMSID                                   =                       SYS1
ACCESS METHOD          : ESDS            RUN TIME OPTION
PRIME DD NAME          : TPFOH2AA        ---------------
INPUT DATABASE         : REAL              HASH                                    =                         NO
CREATION DATE          : 07/06/2006        EPSCHK                                  =                        YES
REAL DATABASE DEVICE   : 3390              IXKEYCHK                                =                        YES
BLOCK SIZE             :   512             SYMIXCHK                                =                         NO
LOGICAL RECORD LENGTH  :   505             SYMLPCHK                                =                         NO
KEY LENGTH             :     8             HOMECHK                                 =                        YES
                                           INCORE                                  =                        YES
                                           T2CHK                                   =                    (00,07)
                                           ZEROCTR                                 =                         NO
                                         SPACE UTILIZATION
                                         -----------------
                                           TYPE                                    =                       CYLS
                                           PRIMARY ALLOCATION                      =                         50
                                           SECONDARY ALLOCATION                    =                         50
                                           EXTENTS                                 =                          1
                                           ALLOCATED SPACE (TRKS)                  =                        750
                                           USED SPACE      ( % )                   =                       12 %
                                         POINTER VALIDATION
                                         ------------------
                                           VALIDATION ERRORS IN SCAN PROCESSING    =                          0
                                           VALIDATION ERRORS IN CHECK PROCESSING   =                          0
                                           EVALUATION ERRORS IN CHECK PROCESSING   =                          0
                                           TOTAL BLOCKS IN DATA SET                =                       4409
                                           HIGH RBA IN HEX                         =                  0001EACCA
                                           HIGH RBA IN DEC                         =                    2010314
                                           % POINTERS VALIDATED IN MEMORY          =                     99.6 %

                                         FREE SPACE STATISTICS
                                         ---------------------
                                           FREE SPACE ELEMENTS                     =                       4407
                                           BLOCKS WITH SPACE (BITMAP)              =                484    10 %
                                           REUSABLE FREE SPACE (BITMAP)            =             241226    10 %
                                           NOT REUSABLE FREE SPACE (BITMAP)        =             158474     7 %
                                           EMPTY BLOCKS                            =                482    10 %

                                         SEGMENT DISTRIBUTION STATISTICS
                                         -------------------------------
                                           SEGMENTS IN DATA SET                    =                      18178
                                           ROOT SEGMENTS                           =                       9000
                                           DEPENDENT SEGMENTS                      =                       9178
                                           ROOTS WITH NO DEPENDENTS IN SAME BLOCK  =               5356    59 %
                                           AVG. COUNT OF DEPS IN SAME BLK AS ROOT  =                        1.3
                                           DEPENDENTS NOT IN SAME BLOCK WITH ROOT  =               4394    47 %
```

Figure 142. HISTPRT—HD Analysis report (PHIDAM database) (Part 1 of 2)



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBHDA                 "HD ANALYSIS REPORT"                                PAGE:     2
5655-K53                                                   DATE: 07/10/2006  TIME: 17.37.25                          FABGHIST - V2.R2
                                                                                                 SPECIFIED DATA       PREVIOUS DATA

                                                                                               DATE: 07/10/2006     DATE:  NONE
                                         HD TUNING STATISTICS                                  TIME:   17:33:53     TIME:  NONE
                                         --------------------                                (CREATED BY: HDPC)   (CREATED BY: NONE)
                                           DIRECT ALGORITHM NAME                   =                      N / A
                                           LONGEST SEGMENT IN DATA SET             =                        122
                                           HIGH BLOCK NUMBER IN RAA                =                       4409
                                           RAPS PER BLOCK                          =                          0
                                           TOTAL RAPS                              =                          0
                                           BYTE LIMIT COUNT                        =                        N/A
                                           AVG. DATABASE RECORD LENGTH             =                        200
                                           FREE SPACE SCAN CYLINDERS               =                          0
                                           FSPC BLK. EVERY N BLKS                  =                          0
                                           % FSPC WITHIN EACH BLK                  =                          0
                                           NO. KEY RECORDS WRITTEN                 =                          0

                                           BLOCKS WITHOUT ROOTS IN RAA             =                613    13 %
                                           COUNT OF RAPS NOT USED                  =                  0     0 %

SEGMENT AND POINTER OCCURRENCES
-------------------------------

SC  SEGNAME   ------ POINTER TYPES -------  PAIRING    --- LENGTH ----  ------ OCCURRENCES ------  --- COUNT OF POINTERS -----------
               C P P P L L L P P L L E * *      TARGT                                              --- POINTING TO ------
               T T T P T T P C C C C P L L  TP  DB SC  PREFIX  ACT-MAX  TOTAL       OCC/RT         SAME BLOCK  ADJ-BLOCKS  TOTAL
              .R.F.B. .F.B. .F.L.F.L.S.P.C             SEGMENT          OVERFLOW                   BEYOND     EXTERNAL
--  --------  ----------------------------  -- --- --  ------- -------  ----------- -------------  ---------- ----------  ----------

01  BROOTLV1   X X X         1                              26      76         9000           1.0       14053       9726       24037
                                                            76                    0                       258          0

02  BDEP1LV2     X X X                                      22     122         9178           1.0        9506       5952       15458
                                                           122                    0                         0          0
```

Figure 142. HISTPRT—HD Analysis report (PHIDAM database) (Part 2 of 2)

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBHDA                 "HD ANALYSIS REPORT"                                PAGE:     1
5655-K53                                                   DATE: 07/10/2006  TIME: 17.37.25                          FABGHIST - V2.R2


DBNAME: TPFOH2     DDNAME: TPFOH2AX   DSG#:  X          *** PRIMARY  I N D E X  DB ***
                                                                                                 SPECIFIED DATA       PREVIOUS DATA

                                                                                               DATE: 07/10/2006     DATE:  NONE
DATABASE DESCRIPTION                     ENVIRONMENT                                           TIME:   17:33:53     TIME:  NONE
-------------------------------------    -----------                                         (CREATED BY: HDPC)   (CREATED BY: NONE)
DATABASE ORGANIZATION  : PHIDAM IDX        IMSID                                   =                       SYS1
ACCESS METHOD          : KSDS            RUN TIME OPTION
INDEXED PRIME DB NAME  : TPFOH2          ---------------
INDEX DD NAME          : TPFOH2AX          HASH                                    =                         NO
OVERFLOW DD NAME       :                   EPSCHK                                  =                         NO
INPUT DATABASE         : REAL              IXKEYCHK                                =                        YES
CREATION DATE          : 07/06/2006        SYMIXCHK                                =                         NO
REAL DATABASE DEVICE   : 3390              SYMLPCHK                                =                         NO
BLOCK SIZE             :   512             HOMECHK                                 =                        N/A
LOGICAL RECORD LENGTH  :    14             INCORE                                  =                        N/A
KEY LENGTH             :     8             T2CHK                                   =                        N/A
                                           ZEROCTR                                 =                        N/A
                                         SPACE UTILIZATION
                                         -----------------                                        PRIME
                                           TYPE                                    =               CYLS
                                           PRIMARY ALLOCATION                      =                 20
                                           SECONDARY ALLOCATION                    =                 10
                                           EXTENTS                                 =                  1
                                           ALLOCATED SPACE (TRKS)                  =                300
                                           USED SPACE      ( % )                   =                2 %
                                           CI-SPLITS                               =                  0
                                           CA-SPLITS                               =                  0

                                         HIDAM INDEX STATISTICS
                                         ----------------------
                                           TOTAL SEGMENTS IN PRIME DB              =                       9001
                                           DELETED SEGMENTS IN PRIME DB            =                          0
                                           POINTERS IN PRIME DB (T6 RECORDS)       =                       9001
                                           TOTAL SEGMENTS IN OVERFLOW DB           =                          0
                                           DELETED SEGMENTS IN OVERFLOW DB         =                          0
                                           POINTERS IN OVERFLOW DB (T7 RECORDS)    =                          0
                                           TOTAL OSAM/ESDS BLOCKS/CIS IN DB        =                          0
```

Figure 143. HISTPRT—HD Analysis report (PHIDAM index database)

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBHDA                 "HD ANALYSIS REPORT"                               PAGE:     1
5655-K53                                                   DATE: 07/10/2006  TIME: 17.37.25                         FABGHIST - V2.R2

DBNAME: TPFOX1     DDNAME: TPFOX1AA   DSG#:  A          *** SECONDARY  I N D E X  DB ***
                                                                                                 SPECIFIED DATA       PREVIOUS DATA

                                                                                               DATE: 07/10/2006     DATE:  NONE
DATABASE DESCRIPTION                     ENVIRONMENT                                           TIME:   17:33:53     TIME:  NONE
-------------------------------------    -----------                                         (CREATED BY: HDPC)   (CREATED BY: NONE)
DATABASE ORGANIZATION  : PSINDEX           IMSID                                   =                       SYS1
ACCESS METHOD          : KSDS            RUN TIME OPTION
INDEXED PRIME DB NAME  : TPFOH2          ---------------
INDEX DD NAME          : TPFOX1AA          HASH                                    =                         NO
OVERFLOW DD NAME       :                   EPSCHK                                  =                        YES
INPUT DATABASE         : REAL              IXKEYCHK                                =                        YES
CREATION DATE          : 07/06/2006        SYMIXCHK                                =                         NO
REAL DATABASE DEVICE   : 3390              SYMLPCHK                                =                         NO
BLOCK SIZE             :   512             HOMECHK                                 =                        N/A
LOGICAL RECORD LENGTH  :    54             INCORE                                  =                        N/A
KEY LENGTH             :    16             T2CHK                                   =                        N/A
                                           ZEROCTR                                 =                        N/A
                                         SPACE UTILIZATION
                                         -----------------                        PRIME
                                           TYPE                                    =               CYLS
                                           PRIMARY ALLOCATION                      =                 10
                                           SECONDARY ALLOCATION                    =                 10
                                           EXTENTS                                 =                  1
                                           ALLOCATED SPACE (TRKS)                  =                150
                                           USED SPACE      ( % )                   =               14 %
                                           CI-SPLITS                               =                  0
                                           CA-SPLITS                               =                  0

                                         SECONDARY INDEX STATISTICS
                                         -------------------------
                                           TOTAL SEGMENTS IN PRIME DB              =                       9178
                                           DELETED SEGMENTS IN PRIME DB            =                          0
                                           POINTERS IN PRIME DB (T6 RECORDS)       =                       9178
                                           TOTAL SEGMENTS IN OVERFLOW DB           =                          0
                                           DELETED SEGMENTS IN OVERFLOW DB         =                          0
                                           POINTERS IN OVERFLOW DB (T7 RECORDS)    =                          0
                                           TOTAL OSAM/ESDS BLOCKS/CIS IN DB        =                          0
```

Figure 144. HISTPRT—HD Analysis report (PSINDEX database)

History Attribute report

This report contains an attribute information of the HISTORY data set that is specified in the HISTORY DD statement.

DB Historical Data Analyzer generates this report when TYPE=ATTRLIST is specified for the PROC control statement in the HISTIN data set.

This report shows the current status of the following optional attributes:

- EXPORTABLE

  YES The HISTORY data set can be exportable by Export Utility.

  NO The HISTORY data set cannot be exportable by Export Utility.

- MULTIENT

  YES By running HD Pointer Checker with history option multiple times a day, more than one database data set entries can be stored in the HISTORY data set for each day.

  NO Only one database data set entries can be stored in the HISTORY data set for each day.

- HISTLOCK

  GROUP
  GROUP is selected for the HISTORY lock option.

  DATASET
  DATASET is selected for the HISTORY lock option.

Figure 145 shows a sample of the History Attribute report.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBHDA              "HISTORY ATTRIBUTE REPORT"                            PAGE:     1
5655-K53                                                   DATE: 07/10/2006  TIME: 17.37.25                         FABGHIST - V2.R2


OPTION               STATUS
-----------------    ---------------------------------------------------------------------------------------------------------------
EXPORTABLE           NO
MULTIENT             YES
HISTLOCK             GROUP
```

Figure 145. HISTPRT—History Attribute report

Chapter 21. Operating instructions for Export Utility

This chapter describes how to run Export Utility of DB Historical Data Analyzer. Export Utility can be run as a batch job, but it cannot be run in the TSO/ISPF environment.

For the operation of FABGHIST program of DB Historical Data Analyzer, see Chapter 20, “Operating instructions for DB Historical Data Analyzer in the MVS batch environment,” on page 365. For the operation of DB Historical Data Analyzer in TSO/ISPF environment, see Chapter 22, “Operating instructions for DB Historical Data Analyzer in the TSO/ISPF environment,” on page 447.

Topics:

Running Export Utility

Export Utility, a FABGXEXP program, exports data from the HISTORY data set to a flat file.

To run Export Utility, do as follows:

1. Allocate and initialize a VSAM KSDS for the HISTORY data set. DB Historical Data Analyzer, FABGHIST program, provides a function to initialize the HISTORY data set.

   Run FABGHIST program specifying TYPE=UPDATE OPTION EXPORTABLE=YES for activate EXPORTABLE option.

   For the description on how to create a HISTORY data set and change the attribute, see “PROC control statement” on page 373.
2. Make sure that HD Pointer Checker is run in advance to create entries in the HISTORY data set.
3. If you want to create the flat records in the user-defined format, prepare the flat record definition statements. The statements can reside in a member of a data set called FABGRECI. For the description of the flat record definitions, see “FABGRECI data set” on page 415. For example of using flat record definitions, see “Example 6: Creating the user-defined flat records” on page 483.

   If you want to create the flat records in the predefined format provided by IMS HP Pointer Checker, you need not prepare the FABGRECI data set. For example of using the predefined format, see “Example 5: Creating the predefined flat records” on page 482.
4. Specify control statements to describe the functions to be run. The control statements can reside in the input stream, or in a data set called HISTIN. See “HISTIN data set” on page 411 for the description of control statement specifications.

   If you want to do syntax checking for the flat record definitions, run Export Utility with specifying TYPE=CHECK in the HISTIN data set. The flat records will not be created, and the syntax checking will be done.
5. Code the JCL as described in “Job control language” on page 410. Then run the Export Utility job.

Job control language

This section describes JCL for Export Utility.

FABGXEXP JCL

To run Export Utility, supply an EXEC statement and the appropriate DD statements. Table 44 summarizes the DD statements.

Table 44. Export Utility DD statements

DDNAME	Use	Format	Need
HISTORY	Input	KSDS	Optional
HISTIN	Input	LRECL=80	Required
HISTMSG	Output	LRECL=133	Required
HISTPRT	Output	LRECL=133	Required

FABGEXPF	Output	DSORG=PS
		RECFM=VB
			Optional

FABGRECI	Input	PDS
		LRECL=80
		DSORG=PO
			Optional

EXEC

This statement must be in the following form:

```
//         EXEC PGM=FABGXEXP
```

HISTORY DD

This VSAM KSDS data set contains the summary information of the HD Pointer Checker run results. This data set must be allocated before you run Export Utility.

DISP=SHR should be used.

It is required if TYPE=EXPORT is specified in the HISTIN data set.

HISTIN DD

This required input data set contains control statements, which describe your specification of the processing to be done by Export Utility.

HISTMSG DD

This required output data set contains a report and messages. BLKSIZE, if coded, must be a multiple of 133.

HISTPRT DD

This required output data set contains reports and messages. BLKSIZE, if coded, must be a multiple of 133.

FABGEXPF DD

This output data set is referred to as a flat file. It contains the flat records that are exported from the HISTORY data set. If TYPE=EXPORT is specified in the HISTIN data set, it is required.

FABGRECI DD

This optional data set is a partitioned data set. Each member contains the flat file definitions. The flat file definitions describe the user-defined format of flat records.

A member name cannot be specified in the DD statement. Specify as in the following sample:

```
//FABGRECI   DD   DISP=SHR,DSN=fabgreci.data.set
```

If you want to generate the flat records in the predefined formats, do not specify this data set.

If you want to generate the flat records in the user-defined format, you must specify this data set.

Input

This section describes all the input data sets that are required to run Export Utility.

HISTORY data set (HISTORY)

Export Utility exports data from this HISTORY data set to a flat file (FABGEXPF data set).

For more details the HISTORY data set, see “HISTORY data set (HISTORY)” on page 367.

HISTIN data set

This section explains the HISTIN data set in Export Utility.

Function

This data set contains the user’s description of the processing to be done by Export Utility.

Format

This data set usually resides in the input stream. However, it can also be defined 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.

This data set can contain one PROC statement and one or more DATABASE and OPTION statements. Control statements can be coded as shown in Figure 146.

```
//HISTIN  DD  *
  PROC TYPE=EXPORT,MEMBER=(MEMB01,MEMB02,MEMB05),DBORG=(HDAM,HIDAM)
    OPTION IMSID=SYS1,
           FROM=12122003,TO=03032005
   DATABASE DB=DBA
   DATABASE DB=DBB
/*
```

Figure 146. HISTIN control statements (Example)

Unlike in the case of FABGHIST, the ENDPROC statement is not required for FABGXEXP. Do not specify ENDPROC.

Control statement syntax

The coding conventions is as same as that described in “Control statement syntax” on page 372.

Notational conventions

The following symbols should be coded as they appear in the command format:

comma	,
equal sign	=

The following symbols are used to illustrate the command format:

Brackets []
Indicates optional choices, one of which may be selected.

Braces {}
Indicates choices, one of which must be selected.

Vertical bar |
Separates the operand alternatives within the brackets and braces.

PROC control statement

The PROC control statement specifies the *function* to be run. One PROC control statement can be specified at a time.

The syntax of the PROC control statement is shown in Figure 147.

```
PROC  TYPE= { EXPORT | CHECK  }
      ,MEMBER= { member-name | (member-name, member-name,....) }
      ,DBORG= ( [HDAM] [,HIDAM] [,HISAM] [,PHDAM] [,PHIDAM] )
```

Figure 147. PROC control statement syntax

TYPE=, MEMBER=, and DBORG= are required for the PROC statement.

TYPE=
Specifies the type of process to be run.

EXPORT
Specifies to export data from the HISTORY data set to flat records in a flat file. The flat file is specified in the FABGEXPF DD statement.

When TYPE=EXPORT is specified, the PROC control statement can be followed by one or more DATABASE control statements and OPTION control statements.

HISTORY, FABGEXPF, HISTPRT, and HISTMSG DD statements are required.

CHECK
Specifies to check the syntax in flat record definitions in the FABGRECI statement. You can see the result of syntax checking in the FABGRECI Statement report in the HISTPRT data set. DATABASE statement or OPTION statement of the HISTIN control statement is not required.

FABGRECI, HISTMSG, and HISTPRT DD statements are required. HISTORY or FABGEXPF DD statement is not required because data is not exported.

It is recommended that you run Export Utility with TYPE=CHECK for the syntax checking before running it with TYPE=EXPORT.

MEMBER=

Specifies one or more member names of the predefined flat records or the user-defined flat record definition members. The flat record is generated in the format defined in these members. Specify at least one member. You can specify multiple member names up to 20.

When you generate the flat records in the predefined format, specify the member names listed in Table 56 on page 442.

When you generate the flat records in user-defined format, specify the member names in the FABGRECI partitioned data set, which contains the flat record definitions.

To specify one member, you can either enclose the member name within parentheses or not. For example, either MEMBER=HDPC0001 or MEMBER=(HDPC001) can be used. To specify multiple members, enclose the member names within parentheses. For example, specify as MEMBER=(HDPC0002,HDPC0003).

DBORG=

Specifies one or more database organization to be exported.

You have to specify at least one database organization.

To specify one database organization, you can either enclose the name within parentheses or not. For example, either DBORG=HDAM or DBORG=(HDAM) can be used. To specify more than one name, enclose the names within parentheses. For example, specify as DBORG=(HDAM,HIDAM).

DATABASE control statement

The DATABASE control statement specifies the database to be processed. A DATABASE control statement, if coded, must be preceded by a PROC control statement. The syntax of the DATABASE control statement is shown in Figure 148.

```
DATABASE DB={dbname|*ALL}
```

Figure 148. DATABASE control statement syntax

DB= Specifies the name of the DBD to be processed. For HALDB, specify the master database name.

If you specify *ALL, all database entries with the database organization that is specified by the DBORG= in the PROC statement are processed.

The default value is *ALL.

If the DATABASE statement is not specified, it is assumed that *ALL is specified.

OPTION control statement

The OPTION control statement specifies additional options for selecting the entries to be exported.

The OPTION control statement can be preceded by a PROC control statement or a DATABASE control statement. If it is preceded by the PROC control statement, the specifications are effective for all of the databases. If it is preceded by the DATABASE control statement, the specifications are effective only for the database specified by the DATABASE statement. If the OPTION statements are preceded by both the PROC and the DATABASE control statements, the specifications preceded by the DATABASE statement overrides the one preceded by the PROC statement and becomes effective for the database specified by the DATABASE statement.

The syntax of the OPTION control statement is shown in Figure 149.

```
OPTION   [FROM=mmddyyyy]
         [,TO=mmddyyyy]
         [,DAYS=nnnn | *ALL |1 ]
         [,IMSID=ims-id | *ALL]
```

Figure 149. OPTION control statement syntax

FROM=

Optional. Specifies the start date of the record entry that you want to export. Specifies the start date for the exporting process. The FROM date must be in the form ″mmddyyyy″ (mm=month, dd=date, yyyy=year).

TO= Optional. Specifies the end date of the record entry that you want to export. The TO date must be in the form ″mmddyyyy″ (mm=month, dd=date, yyyy=year).

DAYS=

Optional. Specify the number of days. Export Utility processes entries for the latest nnnn days within the database. If *ALL is specified, Export Utility processes entries for all dates.

DAYS determines the days for each database. The valid specified range for the DAYS parameter is 1 to 1000.

Example 1:

HDAMDBA contains entries for 01/01/2005, 01/02/2005, 01/03/2005 in the HISTORY data set, and HDAMDBB contains entries for 01/02/2005, 01/03/2005, 01/04/2005.

If DAYS=3 is specified as follows,
HDAMDBA exports entries for 01/01/2005, 01/02/2005, 01/03/2005,
and HDAMDBB exports entries for 01/02/2005, 01/03/2005, 01/04/2005.

```
PROC TYPE=EXPORT,MEMBER=MEM01,DBORG=HDAM
OPTION DAYS=3
DATABASE DB=HDAMDBA
DATABASE DB=HDAMDBB
```

Consideration for the date options:

You can specify the date option, which are ″FROM=″ and ″TO=″ combination or ″DAYS=″, for selecting the entries. If you specify only ″FROM=″, Export Utility processing starts with the specified date entry and ends on the last entry. If you specify only ″TO=″, Export Utility processing starts with the first entry and ends on the specified date entry.

If you do not specify any of them, Export Utility assumes DAYS=1 and the latest date entry within the database is exported.

Example 2:

HDAMDBA contains entries for 01/01/2005, 01/02/2005, and 01/03/2005 in the HISTORY data set, and HDAMDBB contains entries for 01/02/2005, 01/03/2005, 01/04/2005. (The same assumption as in Example 1.)

If none of the date options are specified, the last entry for each database is exported, therefore, 2005/01/03 for HDAMDBA and 01/04/2005 for HDAMDBB.

```
PROC TYPE=EXPORT,MEMBER=MEM01,DBORG=HDAM
DATABASE DB=HDAMDBA
DATABASE DB=HDAMDBB
```

If the MULTIENT option for the HISTORY data set is specified as YES, and there are multiple database entries for a day, all entries with the specified date will be exported.

IMSID=

Optional. Specify IMS ID of IMS system on which Export Utility processes. Only one IMS ID can be specified. If *ALL is specified or if IMSID= is not specified, all IMSIDs are processed.

If ″IMSID=″ is specified with the date option (″FROM=″ and ″TO=″ combination, or ″DAYS=″), Export Utility selects the entries by the date at first, and then selects the specified IMSID within the selected entries.

For example, if HDAMDBA has the entries of July 1st with IMSA, July 2nd with IMSA, and July 3rd with IMSB and the following control statements are specified, Export Utility selects the entries by the last 2 days, July 2nd with IMSA and July 3rd with IMSB, and then selects the entry that has IMSA. Therefore, the entry of July 2nd with IMSA is exported.

```
PROC TYPE=EXPORT,MEMBER=MEM01,DBORG=HDAM
OPTION DAYS=02,IMSID=IMSA
DATABASE DB=HDAMDBA
```

FABGRECI data set

This section explains the FABGRECI data set in Export Utility.

Function

This data set contains the flat record definitions for user-defined flat records.

This data set is an optional data set when TYPE=EXPORT is specified in the HISTIN data set. When this data set is not specified or DUMMY is specified, the flat records are created in the predefined formats provided by IMS HP Pointer Checker.

This data set is required when TYPE=CHECK is specified in the HISTIN data set.

Format

It is a partitioned data set that contains one more members. A member contains the flat record definition statements for a flat record.

It must contain 80-byte fixed-length records. BLKSIZE, if coded, must be a multiple of 80.

The member contains one RECORD statement and some FIELD statements. Statements can be coded in a member of the FABGRECI data set as shown in Figure 150 on page 416.

```
*........1.........2.........3.........4.........5.........6.........7..
*234567890123456789012345678901234567890123456789012345678901234567890123456789012
  RECORD TYPE=DSG,RECID=05
   FIELD NAME=DBDNAME,ATTR=C,LEN=008   DB NAME (CL8)
   FIELD NAME=DDNAME,ATTR=C,LEN=008    PRIMARY DDNAME (CL8)
   FIELD NAME=LRECL,ATTR=X,LEN=002     LRECL (XL2)
   FIELD ATTR=X,LEN=005,VALUE=000001
```

Figure 150. Flat record definition (Example)

From the above example definition, the flat record shown in Figure 151 is generated in the FABGEXPF data set.

```
C8C9C4D4 C4C2F140 C8C9C4D4 C4E2F140   07F90000 000001                       *HIDMDB1 HIDMDS1 .......         *
C8C9C4D4 C4C2F140 C8C9C4D4 C4E2F240   07F90000 000001                       *HIDMDB1 HIDMDS2 .......         *
C8C9C4D4 C4C2F140 C8C9C4D4 C4E2F340   07F90000 000001                       *HIDMDB1 HIDMDS3 .......         *
```

Figure 151. Flat record (Example)

Flat record definition syntax

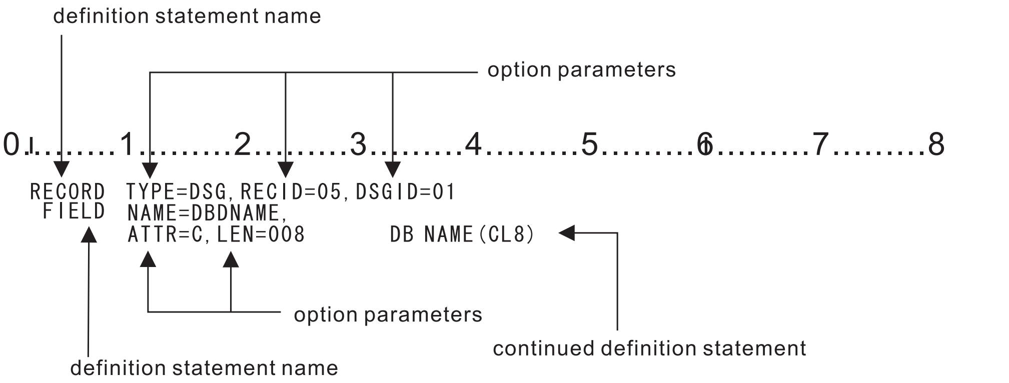
Figure 152 on page 417 and the following description present the coding conventions that you must follow in writing the flat record definition statements (definition statements) in FABGRECI data set:

- A definition statement can be coded onto one or more line. Definition statement names (RECORD and FIELD) and option parameters must be coded within column 2 and column 72. A definition statement name must be the first entry in the definition statement.
- Option parameters follow the definition statement name, separated by one or more blanks. A definition statement name and the first option parameter must be written in the same definition statement record. When more than one option parameter is coded, they must be separated by commas. No blanks are allowed between the option parameters and the commas, or within the option parameters.

  Option parameters can be continued onto more than one definition statement record. In this case, the definition statement that starts with a definition statement name must be completed with an option parameter including a comma that follows it. The succeeding option parameters can be continued onto the following definition statement records that begin in any column from column 2.

  Option parameters are not positional parameters; they can be specified in any order.

  A null value is not allowed for any option parameter.
- Comments may follow the last option parameter on each definition statement record, separated by at least one blank.
- A comment statement must begin with an asterisk in column 1.



```
0.........1.........2.........3.........4.........5.........6.........7.........8
  RECORD  TYPE=DSG,RECID=05,DSGID=01
   FIELD  NAME=DBDNAME,
          ATTR=C,LEN=008        DB NAME(CL8)
```

definition statement name
option parameters
option parameters
continued definition statement
definition statement name

Figure 152. Sample definition statement format in FABGRECI data set

Notational conventions

The following symbols should be coded as they appear in the syntax format:

comma ,
equal sign =

The following symbols are used to illustrate the syntax format:

Brackets []
Indicates optional parameter.

Braces {}
Indicates choices, one of which must be selected.

Vertical bar |
Separates the operand alternatives within the brackets and braces.

RECORD definition statement

The RECORD definition statement (RECORD statement) must be specified on the first line of a member of the FABGRECI data set. The RECORD statement can be followed by one or more FIELD statements.

The syntax of the RECORD statement is shown in Figure 153.

```
RECORD  TYPE= { DB | PART | DSG | SEGMENT | RAP | POINTER }
      ,RECID=xx
     [,DSGID=nn]
```

Figure 153. RECORD statement syntax

TYPE=
Required. Specifies the type of flat record.

DB
Specifies to generate the flat record for each database. The fields contained in this record show the information of whole database.

FIELD NAME=DBDNAME statement must be specified for this record type.

PART

Specifies to generate the flat record for each partition of HALDB. It can be specified to a HALDB. The fields contained in this record show the information of each partition.

FIELD NAME=PARTID statement must be specified for this record type.

DSG

Specifies to generate the flat record for each data set group. The fields contained in this record show the information of each data set group.

For HALDB, the flat record is generated per data set group and per partition. For example,

- If three data set groups are defined to non-HALDB, three flat records are created.
- If three data set groups are defined to HALDB and two partitions are defined to the HALDB, six flat records are created in total.

FIELD NAME=DSGID must be specified for this record type.

SEGMENT

Specifies to generate the flat record for each segment type. The fields contained in this record show the information of each segment type.

For HALDB, the flat record is generated per segment type and per partition.

FIELD NAME=SEGNAME statement must be specified with this record type.

RAP

Specifies to generate the flat record for each RAP (Root Anchor Point). The fields contained in this record show the information of RAP.

For HALDB, the flat record is generated per partition.

FIELD NAME=PTRTYPE must be specified with this record type.

POINTER

Specifies to generate the flat record for each the pointer types. The fields contained in this record show the information of each pointer type. For HALDB, the flat record is generated per pointer type and per partition.

FIELD NAME=PTRTYPE must be specified for this record type.

RECID=*xx*

Required. Specifies an identifier for the flat record. *xx* is two alphameric characters. This ID can be stored in the flat record.

DSGID=*nn*

Optional. Specifies a data set group number. nn is one of decimal numbers from 1 to 10. If this parameter is not specified, the flat records are generated from all of the data set groups. If this parameter is specified, the flat records are generated from the specified data set group.

In case of HALDB, the data set group is usually referred to by an alphabet (A, B, ... or, J). However, specify this parameter by a decimal number. For example, to process data set group A, specify DSGID=1. The flat records are generated for the specified data set group for every partition. You can only specify DSGID=*nn* when TYPE=DSG is specified.

FIELD definition statement

The FIELD definition statement (FIELD statement) specifies the field that are stored in a flat file. One or more FIELD statements are required in the FABGRECI member. The fields are stored in the flat record in the order of the FIELD statements.

The maximum number of fields in one flat record is 255. The maximum total length of the fields in one flat record is 32752 bytes.

The syntax of the FIELD statement is shown in Figure 154.

```
FIELD  {NAME= field name | VALUE= value }
      [,ATTR = { X | C | P | default value }]
      [,LEN=   { nnn | default value }]
```

Figure 154. FIELD statement syntax

NAME=

Specify the field name that is listed in Table 45 on page 421 through Table 55 on page 438.

Either ″NAME=″ or ″VALUE=″ is required.

ATTR=

Specify the attribute for the field.

X The field is stored into the flat record in hexadecimal.

C The field is stored into the flat record in character.

P The field is stored into the flat record in packed decimal.

If ″NAME=″ is specified and this parameter is not specified, the default attribute of the field is taken. The default attribute of each field name is described in Table 45 on page 421 through Table 55 on page 438.

If ″VALUE=″ is specified, it is required that you specify X, C, or P. No default value is taken.

LEN=

Specify the length of the field.

The length is how many bytes there are in the field. For packed decimal attribute, specify the length in bytes, not the number of digits. For example, specify LEN=16 for 31 digit packed decimal number.

If ″NAME=″ is specified and this parameter is not specified, the default length of the field is taken. The default length and maximum length of each field are described in Table 45 on page 421 through Table 55 on page 438.

If ″VALUE=″ is specified, it is required that you specify the length. The maximum length is as follows:

- For character or hexadecimal attribute, the maximum length is 256.
- For packed decimal attribute, the maximum length is 16.

Export Utility checks the overflow condition of each field and issues a warning message according to the following rules:

- In the TYPE=CHECK run of Export Utility, the length check is done. If there is a possibility of overflow, it is notified by RC=02 and a message from Export Utility.
- In the TYPE=EXPORT run of Export Utility, field overflow is checked for every attribute while exporting the data. If an overflow is caused, it is notified by RC=04 and message from Export Utility. If the specified length is shorter than the actual field, the data is truncated as follows:
  - For character attribute, the specified number of characters are stored from the left in the flat record and the rest is truncated.
  - For hexadecimal or packed decimal attribute, the number of digits are stored from the low order digits and the rest is truncated.

If the specified length is longer than the actual data, no message will be issued. The following data is padded to the extra column:

- For character attribute, the right columns are padded with blank (X’40’).
- For hexadecimal or packed decimal attribute, the left columns are padded with X’00’.

VALUE=

Specify the specific value to be stored in the flat record. The ″ATTR=″ and ″LEN=″ are required for this parameter. For example, if ″FIELD VALUE=’ABC’,LEN=3,ATTR=C ″ is specified, the word ’ABC’ is stored in the flat record.

The value has the following rules:

- For character attribute, the character must be enclosed in quotation marks ’’. The maximum number of characters specified in the value is 32. Blank can be specified in the value, but you cannot use an apostrophe (’).
- For hexadecimal attribute, the length of value must be a multiple of 2 and the maximum length is 32. (This means 16 byte long field.) The value can be specified from 01 to FF....FF (32 Fs).
- For packed decimal attribute, the value can be specified up to 31 digits signed decimal.

You can specify a larger or a shorter number to LEN= than the actual value length. The padding and truncated rules are described in the descriptions for the LEN= parameter.

Field names

The field names that can be specified in the FIELD statement are described in Table 45 on page 421 through Table 55 on page 438.

The field names can be specified in the order you like.

Table 45. Record ID field

Field name	**Default value**		**Record type that can be specified Y: Can be specified N: Cannot be specified**						**Effective database organization Y: Value is set N: Blank (X’40’) or null (X’00’) is set**					**Maximum length that can be specified 0: Cannot be specified**			**Description**
	Attribute	**Length**	**DB**	**PART**	**DSG**	**SEGM**	**RAP**	**PTR**	**HDAM**	**HIDAM**	**PHDAM**	**PHIDAM**	**HISAM**	**C**	**X**	**P**	
RECID	C	2	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	2	0	0	Record ID, which is assigned for the predefined flat record, or defined in the flat record definitions

Table 46 shows the field names for the IMS environment and runtime information when HD Pointer Checker SCAN process is run.

Table 46. Field names for the IMS environment and runtime information when HD Pointer Checker SCAN process is run

Field name	**Default value**		**Record type that can be specified Y: Can be specified N: Cannot be specified**						**Effective database organization Y: Value is set N: Blank (X’40’) or null (X’00’) is set**					**Maximum length that can be specified 0: Cannot be specified**			**Description**
	Attribute	**Length**	**DB**	**PART**	**DSG**	**SEGM**	**RAP**	**PTR**	**HDAM**	**HIDAM**	**PHDAM**	**PHIDAM**	**HISAM**	**C**	**X**	**P**	
SCANDATE	C	7	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	7	0	4	SCAN year and date by HD Pointer Checker: *yyyyddd*
SCANTIME	X	3	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	0	3	0	SCAN time: X’*hhmmss*’
SCANDATE5	C	5	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	5	0	3	SCAN julian date: *yyddd*
SCANYEA4	C	4	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	4	0	3	SCAN year (4-digit number): *yyyy*
SCANYEA2	C	2	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	2	0	2	SCAN year (Low order 2-digit number): *yy*
SCANDAYJ	C	3	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	3	0	2	SCAN Julian date: *ddd*

Table 46. Field names for the IMS environment and runtime information when HD Pointer Checker SCAN process is run (continued)

Field name	Default value		Record type that can be specified Y: Can be specified N: Cannot be specified						Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set					Maximum length that can be specified 0: Cannot be specified			Description
	Attribute	Length	DB	PART	DSG	SEGM	RAP	PTR	HDAM	HIDAM	PHDAM	PHIDAM	HISAM	C	X	P	
SCANMNTH	C	2	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	2	0	2	SCAN month: *mm*
SCANDAY	C	2	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	2	0	2	SCAN day: *dd*
SCANHOUR	C	2	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	2	0	2	SCAN hour: *hh*
SCANMINT	C	2	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	2	0	2	SCAN minute: *mm*
SCANSCND	C	2	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	2	0	2	SCAN second: *ss*
IMSID	C	4	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	4	0	0	IMS ID
IMSVER	C	5	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	5	0	0	IMS Version
IMSVER2	C	6	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	6	0	0	Extended IMS Version
DBRCOPT	C	1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	1	0	0	• DBRC=Y or N when HD Pointer Checker runs • ″Y″ or ″N″ is set

Table 47 shows the field names for DBD information.

Table 47. Field names for DBD information

Field name	Default value		Record type that can be specified Y: Can be specified N: Cannot be specified						Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set					Maximum length that can be specified 0: Cannot be specified			Description
	Attribute	Length	DB	PART	DSG	SEGM	RAP	PTR	HDAM	HIDAM	PHDAM	PHIDAM	HISAM	C	X	P	
DBDNAME	C	8	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	8	0	0	Database name

Table 47. Field names for DBD information (continued)

Field name	De-fault value		Record type that can be specified Y: Can be specified N: Cannot be specified						Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set					Maximum length that can be specified 0: Cannot be specified			Description
	Attribute	Length	DB	PART	DSG	SEGM	RAP	PTR	HDAM	HIDAM	PHDAM	PHIDAM	HISAM	C	X	P	
PARTNAME	C	7	N	Y	Y	Y	Y	Y	N	N	Y	Y	N	7	0	0	Partition name
PARTID	C	4	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	4	2	0	Partition ID
DBORG	C	6	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	6	0	0	Database organization: SHISAM/HISAM/HIDAM HDAM/PHIDAM/PHDAM
ACCESS	C	4	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	4	0	0	Access method
DBRECN	C	1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	1	0	0	• DBD registered in the RECON • ″Y″ or ″N″ is set
DBDSIZE	X	2	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	0	2	0	DBD size
DBDDATE	C	16	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	16	0	0	DBD assemble date
NO_OF_DSG	X	1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	0	1	0	The number of DSG defined in the DBD
NO_OF_SEG	X	1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	0	1	0	The number of segment type defined in the DBD
PINDXNM	C	8	Y	Y	Y	Y	Y	Y	N	Y	N	N	N	8	0	0	Name of primary index DBD of HIDAM

Table 48 on page 424 shows the field names for randomizing parameters defined in the DBD. The values in Table 48 on page 424 are set only when DSG=01. Otherwise, blank (X'40') or null (X'00') is set.

Table 48. Field names for randomizing parameters defined in the DBD

Field name	Default value		Record type that can be specified Y: Can be specified N: Cannot be specified						Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set					Maximum length that can be specified 0: Cannot be specified			Description
	Attribute	Length	DB	PART	DSG	SEGM	RAP	PTR	HDAM	HIDAM	PHDAM	PHIDAM	HISAM	C	X	P	
RANDNAME	C	8	N	N	Y	N	N	N	Y	N	Y	N	N	8	0	0	Randomizer name
NO_OF_RAA	X	4	N	N	Y	N	N	N	Y	N	Y	N	N	0	4	8	The number of RAA
NO_OF_RAP	X	2	N	N	Y	N	N	N	Y	Y	Y	N	N	0	2	4	The number of RAP
NO_OF_BYTLIM	X	2	N	N	Y	N	N	N	Y	N	Y	N	N	0	2	4	Byte limit

Table 49 shows the field names for statistics of RAPs, root segments, and database records. The values in Table 49 are set only when DSG=01. Otherwise, blank (X'40') or null (X'00') is set. For a non-HALDB, the values for each database is set. For a HALDB, the values for each partition is set.

Table 49. Field names for statistics of RAPs, root segments, and database records

Field name	Default value		Record type that can be specified Y: Can be specified N: Cannot be specified						Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set					Maximum length that can be specified 0: Cannot be specified			Description
	Attribute	Length	DB	PART	DSG	SEGM	RAP	PTR	HDAM	HIDAM	PHDAM	PHIDAM	HISAM	C	X	P	
NO_OF_ACTRAP	X	4	N	N	Y	N	N	N	Y	Y	Y	N	N	0	4	8	The number of active RAPs
NO_OF_NOUSRAP	X	4	N	N	Y	N	N	N	Y	Y	Y	N	N	0	4	8	The number of not used RAPs

Table 49. Field names for statistics of RAPs, root segments, and database records (continued)

Field name	Default value		Record type that can be specified Y: Can be specified N: Cannot be specified						Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set					Maximum length that can be specified 0: Cannot be specified			Description
	Attribute	Length	DB	PART	DSG	SEGM	RAP	PTR	HDAM	HIDAM	PHDAM	PHIDAM	HISAM	C	X	P	
NO_OF_TOTROTRA	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	4	8	• The number of root segment occurrences in RAA • If you want to set this field, run HD Pointer Checker with the HOMECHK=YES, DBDIST=YES, and CHAINDIST=YES options • For HIDAM or PHIDAM, the number of root segment occurrences is stored
NO_OF_ROTHOME	X	4	N	N	Y	N	N	N	Y	N	Y	N	N	0	4	8	• The number of root segment occurrences in HOME BLOCK • If you want to set this field, run HD Pointer Checker with the HOMECHK=YES, DBDIST=YES, and CHAINDIST=YES options
NO_OF_ROTHOM1	X	4	N	N	Y	N	N	N	Y	N	Y	N	N	0	4	8	• The number of root segment occurrences within (HOME BLOCK+1) and (HOME BLOCK-1) • If you want to set this field, run HD Pointer Checker with the HOMECHK=YES, DBDIST=YES, and CHAINDIST=YES options
NO_OF_ROTBYND	X	4	N	N	Y	N	N	N	Y	N	Y	N	N	0	4	8	• The number of root segment occurrences that exist above (HOME BLOCK+1) and under (HOME BLOCK-1) • If you want to set this field, run HD Pointer Checker with the HOMECHK=YES, DBDIST=YES, and CHAINDIST=YES options
NO_OF_RAAWOROT	X	4	N	N	Y	N	N	N	Y	Y	Y	N	N	0	4	8	The number of blocks without root segment in RAA

Table 49. Field names for statistics of RAPs, root segments, and database records (continued)

Field name	Default value		Record type that can be specified Y: Can be specified N: Cannot be specified						Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set					Maximum length that can be specified 0: Cannot be specified			Description
	Attribute	Length	DB	PART	DSG	SEGM	RAP	PTR	HDAM	HIDAM	PHDAM	PHIDAM	HISAM	C	X	P	
AVDBREC	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	4	8	Average database record length
NO_OF_ROTWODEP	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	4	8	• The number of root segment occurrences that are in the same block as the dependent segment occurrences • If you want to set this field, run HD Pointer Checker with the HASH=NO and INCORE=YES options
NO_OF_DEPWOROT	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	4	8	• The number of dependent segment occurrences that are in different blocks than the root segment block • If you want to set this field, run HD Pointer Checker with the HASH=NO and INCORE=YES options
NO_OF_AVDEP	X	2	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	2	4	• Average count of dependent segment occurrences in the same block • If you want to set this field, run HD Pointer Checker with the HASH=NO and INCORE=YES option
NO_OF_SYNRAP	X	4	N	N	Y	N	N	N	Y	N	Y	N	N	0	4	8	• The number of RAPs with synonym • If you want to set this field, run HD Pointer Checker with the HOMECHK=YES, DBDIST=YES, and CHAINDIST=YES options

Table 49. Field names for statistics of RAPs, root segments, and database records (continued)

Field name	Default value		Record type that can be specified Y: Can be specified N: Cannot be specified						Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set					Maximum length that can be specified 0: Cannot be specified			Description
	Attribute	Length	DB	PART	DSG	SEGM	RAP	PTR	HDAM	HIDAM	PHDAM	PHIDAM	HISAM	C	X	P	
NO_OF_ROTNOHOM	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	4	8	• The number of root segment occurrences in blocks other than HOME BLOCK • For HIDAM or PHIDAM, the number of root segment occurrences is stored
NO_OF_ROTSYN	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	4	8	• The number of root segment occurrences in the synonym chain • If you want to set this field, run HD Pointer Checker with the HOMECHK=YES, DBDIST=YES, and CHAINDIST=YES options
MXROTBLK	X	2	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	2	4	The maximum number of root segment occurrences per block
OVSEG%	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	4	8	The percentage of segment occurrences in overflow

Table 50 on page 428 shows the field names for statistics of segments. The values in Table 50 on page 428 will be set whatever DSG ID is specified. For a non-HALDB, the values for each database is set. For a HALDB, the values for each partition is set.

Table 50. Field names for statistics of segments

Field name	Default value		Record type that can be specified Y: Can be specified N: Cannot be specified						Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set					Maximum length that can be specified 0: Cannot be specified			Description
	Attribute	Length	DB	PART	DSG	SEGM	RAP	PTR	HDAM	HIDAM	PHDAM	PHIDAM	HISAM	C	X	P	
NO_OF_TOTROTDB	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	4	8	The number of root segment occurrences
NO_OF_TOTDEPDB	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	4	8	The number of dependent segment occurrences

Table 51 shows the field names for data set group and data set space information. Value for each data set is set. For a HALDB, each DSG information is generated per partition.

Table 51. Field names for data set group and data set space information

Field name	Default value		Record type that can be specified Y: Can be specified N: Cannot be specified						Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set					Maximum length that can be specified 0: Cannot be specified			Description
	Attribute	Length	DB	PART	DSG	SEGM	RAP	PTR	HDAM	HIDAM	PHDAM	PHIDAM	HISAM	C	X	P	
DSGID	C	2	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	2	1	0	Data Set Group identifier in 2-digit numbers from 01 to 10
DSGIDCHR	C	1	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	1	0	0	• Data Set Group identifier in character • For HALDB, character A to J, or M to V • For non-HALDB, 1 to 9, or A.
DDNAME	C	8	N	N	Y	N	N	N	Y	Y	Y	Y	Y	8	0	0	DD name

Table 51. Field names for data set group and data set space information (continued)

Field name	**De-fault value**		**Record type that can be specified Y: Can be specified N: Cannot be specified**						**Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set**					**Maximum length that can be specified 0: Cannot be specified**			**Description**
	Attribute	**Length**	**DB**	**PART**	**DSG**	**SEGM**	**RAP**	**PTR**	**HDAM**	**HIDAM**	**PHDAM**	**PHIDAM**	**HISAM**	**C**	**X**	**P**	
DSNAME	C	44	N	N	Y	N	N	N	Y	Y	Y	Y	Y	44	0	0	Database data set name or image copy data set name
SCANCYL	X	1	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	1	0	The number of direct-access device cylinders to be scanned when searching for available storage space during segment insertion operations
FBFF	X	1	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	1	0	The free block frequency factor (fbff) specified in DBD
FSPF	X	1	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	1	0	The free space percentage factor (fspf) specified in DBD
BLKSIZE	X	2	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	2	4	Data set block size
LRECL	X	2	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	2	4	Data set lrecl
ALOCTYPE	C	1	N	N	Y	N	N	N	Y	Y	Y	Y	Y	1	0	0	Data set allocation type C'C': CYL C'T': TRK C'B': BLK
NO_OF_PALC	X	2	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	2	0	Primary allocation quantity of direct access storage required for the data set
NO_OF_SALC	X	2	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	2	0	Secondary allocation quantity of direct access storage required for the data set
NO_OF_ALCSPC	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	4	0	The number of allocated tracks for the data set
NO_OF_USESPC	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	4	0	The number of tracks used by the data set
NO_OF_EXTENT	X	1	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	1	0	The number of data set extents
NO_OF_CISPLT	X	2	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	2	0	The number of VSAM CI split
NO_OF_CASPLT	X	2	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	2	0	The number of VSAM CA split
CREDATE	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	4	0	Data set creation date: X'*yyyymmdd*'

Table 51. Field names for data set group and data set space information (continued)

Field name	Default value		Record type that can be specified Y: Can be specified N: Cannot be specified						Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set					Maximum length that can be specified 0: Cannot be specified			Description
	Attribute	Length	DB	PART	DSG	SEGM	RAP	PTR	HDAM	HIDAM	PHDAM	PHIDAM	HISAM	C	X	P	
CRETIME	X	3	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	3	0	Data set creation time: X'*hhmmss*'
HIRBA	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	4	8	High used RBA in the data set
INCORE%	X	2	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	2	0	• The percentage of pointers validated in the incore checking • If you want to set this field, run HD Pointer Checker with the HASH=NO and INCORE=YES options
SUMSPC	X	5	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	5	8	Allocated bytes for the data set
FREEBYT	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	4	8	Free space bytes in the data set
ALCCYL	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	4	8	• Allocated cylinders for the data set • If you want to set this field, run HD Pointer Checker with the DATASET=REAL option
USEDDS%	C	5	N	N	Y	N	N	N	Y	Y	Y	Y	Y	5	0	0	Used rate to the data set capacity
OVUSEDDS%	C	5	N	N	Y	N	N	N	N	N	N	N	Y	5	0	0	Used rate to the data set capacity of overflow data set

Table 52 on page 431 shows the field names for statistics of a data set group (For HISAM, information of a primary data set). Value for each data set is set. For a HALDB, each DSG information is generated per partition.

Table 52. Field names for statistics of a data set group (For HISAM, information of a primary data set)

Field name	De-fault value		Record type that can be specified Y: Can be specified N: Cannot be specified						Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set					Maximum length that can be specified 0: Cannot be specified			Description
	Attribute	Length	DB	PART	DSG	SEGM	RAP	PTR	HDAM	HIDAM	PHDAM	PHIDAM	HISAM	C	X	P	
DSGMXSGL	X	2	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	2	4	Maximum length of segment in the data set
DSGMNSGL	X	2	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	2	4	Minimum length of segment in the data set
NO_OF_UNKNOWN	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	4	8	The number of unknown data. For HISAM database, this field is set only for a primary data set. **Note:** If the number of unknown data exceeds 32767 (X'00007FFF'), X'FFFFFFFF' is set in this field.
NO_OF_TOTPRSG	X	4	N	N	Y	N	N	N	N	N	N	N	Y	0	4	8	The number of segment occurrences in the data set (HISAM only)
NO_OF_DELPRSG	X	4	N	N	Y	N	N	N	N	N	N	N	Y	0	4	8	The number of deleted segment occurrences in the data set (HISAM only)
NO_OF_TOTBLK	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	4	8	The number of CIs or blocks in the data set
NO_OF_BMBLK	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	4	8	The number of bit map blocks in the data set
NO_OF_EMPBLK	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	4	8	The number of empty blocks in the data set
NO_OF_BLKWOFSE	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	4	8	The number of blocks without FSE in the data set
NO_OF_BLKWFSE	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	4	8	The number of blocks with FSE and segment occurrences in the data set
NO_OF_VSAMCI	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	4	8	The number of VSAM CI#0 in the data set
NO_OF_ERRBLK	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	4	8	The number of blocks with error in the data set
NO_OF_SLKBLK	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	4	8	The number of blocks with slack bytes in the data set

Table 52. Field names for statistics of a data set group (For HISAM, information of a primary data set) (continued)

Field name	Default value		Record type that can be specified Y: Can be specified N: Cannot be specified						Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set					Maximum length that can be specified 0: Cannot be specified			Description
	Attribute	Length	DB	PART	DSG	SEGM	RAP	PTR	HDAM	HIDAM	PHDAM	PHIDAM	HISAM	C	X	P	
NO_OF_ADDBLK	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	4	8	The number of blocks added after initial load or reorganization in the data set
NO_OF_AVALFSE	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	4	8	The number of FSEs that has enough length to store the longest segment in the data set
NO_OF_NAVLFSE	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	4	8	The number of FSEs that are too short to store the shortest segment in the data set
NO_OF_NSEGBLK	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	4	8	The number of blocks with no segment in the data set
NO_OF_UNFMBLK	X	4	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	4	8	The number of unformatted blocks in the data set
SUMSLK	X	5	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	5	8	Total byte of slack bytes in the data set
SUMPRE	X	5	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	5	8	Total byte of segment prefixes in the data set
SUMDAT	X	5	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	5	8	Total byte of segment data in the data set
SUMPAD	X	5	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	5	8	Total byte of segment padding areas in the data set
SUMAVFSE	X	5	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	5	8	Total byte of available FSEs in the data set, which has enough length to store the shortest segment
SUMNAFSE	X	5	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	5	8	Total byte of enough FSEs in the data set, which does not have enough length to store the shortest segment.
SUMUNKOW	X	5	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	5	8	Total byte of unknown data in the data set
SUMOVHED	X	5	N	N	Y	N	N	N	Y	Y	Y	Y	Y	0	5	8	Total byte of DL/I overhead in the data set
SUMVSMBT	X	5	N	N	Y	N	N	N	Y	Y	Y	Y	N	0	5	8	Total byte of VSAM CTL in the data set

Table 52. Field names for statistics of a data set group (For HISAM, information of a primary data set) (continued)

Field name	Default value		Record type that can be specified Y: Can be specified N: Cannot be specified						Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set					Maximum length that can be specified 0: Cannot be specified			Description
	Attribute	Length	DB	PART	DSG	SEGM	RAP	PTR	HDAM	HIDAM	PHDAM	PHIDAM	HISAM	C	X	P	
SUMOVFSE	X	5	N	N	Y					Y	Y	Y	N	0	5	8	Total byte of FSEs in overflow area in the data set

Table 53 shows the field names for data set information of HISAM overflow.

Table 53. Field names for data set information of HISAM overflow

Field name	Default value		Record type that can be specified Y: Can be specified N: Cannot be specified						Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set					Maximum length that can be specified 0: Cannot be specified			Description
	Attribute	Length	DB	PART	DSG	SEGM	RAP	PTR	HDAM	HIDAM	PHDAM	PHIDAM	HISAM	C	X	P	
OVDDNM	C	8	N	N	Y	N	N	N	N	N	N	N	Y	8	0	0	Overflow DD name
OVDSNM	C	44	N	N	Y	N	N	N	N	N	N	N	Y	44	0	0	Overflow data set name or image copy data set name
OVBLKSZ	X	2	N	N	Y	N	N	N	N	N	N	N	Y	0	2	4	Block size of overflow data set
OVLRECL	X	2	N	N	Y	N	N	N	N	N	N	N	Y	0	2	4	LRECL of overflow data set
OVALOCTP	C	1	N	N	Y	N	N	N	N	N	N	N	Y	1	0	0	Allocation type of overflow data set C'C': CYL C'T': TRK C'B': BLK
NO_OF_OVPALC	X	2	N	N	Y	N	N	N	N	N	N	N	Y	0	2	0	Primary allocation quantity of direct access storage required for overflow data set

Table 53. Field names for data set information of HISAM overflow (continued)

Field name	De-fault value		Record type that can be specified Y: Can be specified N: Cannot be specified						Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set					Maximum length that can be specified 0: Cannot be specified			Description
	Attribute	Length	DB	PART	DSG	SEGM	RAP	PTR	HDAM	HIDAM	PHDAM	PHIDAM	HISAM	C	X	P	
NO_OF_OVSALC	X	2	N	N	Y	N	N	N	N	N	N	N	Y	0	2	0	Secondary allocation quantity of direct access storage required for overflow data set
NO_OF_OVUNKNOWN	X	4	N	N	Y	N	N	N	N	N	N	N	Y	0	4	8	The number of unknown data in an overflow data set of HISAM database. **Note:** If the number of unknown data exceeds 32767 (X'00007FFF'), X'FFFFFFFF' is set in this field.
NO_OF_OVEXTNT	X	1	N	N	Y	N	N	N	N	N	N	N	Y	0	1	0	The number of overflow data set extents
NO_OF_OVCISPT	X	2	N	N	Y	N	N	N	N	N	N	N	Y	0	2	0	The number of overflow data set CI split
NO_OF_OVCASPT	X	2	N	N	Y	N	N	N	N	N	N	N	Y	0	2	0	The number of overflow data set CA split
NO_OF_OVALCSP	X	4	N	N	Y	N	N	N	N	N	N	N	Y	0	4	0	The number of allocated tracks for the overflow data set
NO_OF_OVUSESP	X	4	N	N	Y	N	N	N	N	N	N	N	Y	0	4	0	The number of tracks used in the overflow data set
NO_OF_TOTOVSG	X	4	N	N	Y	N	N	N	N	N	N	N	Y	0	4	8	The number of segment occurrences in the overflow data set
NO_OF_DELOVSG	X	4	N	N	Y	N	N	N	N	N	N	N	Y	0	4	8	The number of deleted segment occurrences in the overflow data set

Table 54 on page 435 shows the field names for segment information. For a non-HALDB, the values for each database is set. For a HALDB, the values for each partition is set.

Table 54. Field names for segment information

Field name	Default value		Record type that can be specified Y: Can be specified N: Cannot be specified						Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set					Maximum length that can be specified 0: Cannot be specified			Description
	Attribute	Length	DB	PART	DSG	SEGM	RAP	PTR	HDAM	HIDAM	PHDAM	PHIDAM	HISAM	C	X	P	
SEGNAME	C	8	N	N	N	Y	N	Y	Y	Y	Y	Y	Y	8	0	0	Segment name
SEGCODE	X	1	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	0	1	0	Segment code
SEGLVL	X	1	N	N	N	Y	N	N	Y	Y	Y	Y	Y	0	1	0	Segment hierarchical level
SEGPPCD	X	1	N	N	N	Y	N	N	Y	Y	Y	Y	Y	0	1	0	Segment code of physical parent
SEGPRE	X	2	N	N	N	Y	N	N	Y	Y	Y	Y	Y	0	2	4	Length of segment prefix part
SEGDAT	X	2	N	N	N	Y	N	N	Y	Y	Y	Y	Y	0	2	4	Length of segment data part
ACTMXVLS	X	2	N	N	N	Y	N	N	Y	Y	Y	Y	Y	0	2	4	Actual maximum length of the segment data
ACTMNVLS	X	2	N	N	N	Y	N	N	Y	Y	Y	Y	Y	0	2	4	Actual minimum length of the segment data
NO_OF_TOTOCC	X	4	N	N	N	Y	N	N	Y	Y	Y	Y	Y	0	4	8	The number of the segment occurrences
NO_OF_TOTOVOC	X	4	N	N	N	Y	N	N	Y	Y	Y	Y	Y	0	4	8	The number of segment occurrences in the overflow area or the data set
NO_OF_TOTSPOC	X	4	N	N	N	Y	N	N	Y	Y	Y	Y	N	0	4	8	The number of segment occurrences that are split
NO_OF_DEFSPOC	X	4	N	N	N	Y	N	N	Y	Y	Y	Y	N	0	4	8	The number of segment occurrences that are split into different blocks
NO_OF_SEGMINCT	X	4	N	N	N	Y	N	N	Y	Y	Y	Y	N	0	4	8	The number of segment occurrences that are shorter than the minimum length
NO_OF_TOTSEGLN	P	8	N	N	N	Y	N	N	Y	Y	Y	Y	Y	0	0	8	Total length of segment occurrences
TOTNOCMP	P	8	N	N	N	Y	N	N	Y	Y	Y	Y	Y	0	0	8	• Total length of segment occurrences • The length before it is edited by the segment edit/compression exit routine • If you want to set this field, run HD Pointer Checker with the COMPFACT=YES option

Table 54. Field names for segment information (continued)

Field name	**De-fault value**		**Record type that can be specified Y: Can be specified N: Cannot be specified**						**Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set**					**Maximum length that can be specified 0: Cannot be specified**			**Description**
	Attribute	**Length**	**DB**	**PART**	**DSG**	**SEGM**	**RAP**	**PTR**	**HDAM**	**HIDAM**	**PHDAM**	**PHIDAM**	**HISAM**	**C**	**X**	**P**	
NO_OF_SEGBSOCC	X	4	N	N	N	Y	N	N	Y	Y	Y	Y	Y	0	4	8	The number of segment occurrences in the base area • For HISAM, the base area means a primary data set • For the first data set group of HDAM or PHDAM, the base area means RAA • For the first data set group of HIDAM or PHIDAM, the base area means the blocks below the High key • Other than the previous conditions, the base area is whole area of the database data set
NO_OF_SEGBSSPL	X	4	N	N	N	Y	N	N	Y	Y	Y	Y	N	0	4	8	The number of segment split prefixes in the base area
NO_OF_SEGBSSDT	X	4	N	N	N	Y	N	N	Y	Y	Y	Y	N	0	4	8	The number of segment split data in the base area
NO_OF_SEGOVOCC	X	4	N	N	N	Y	N	N	Y	Y	Y	Y	Y	0	4	8	The number of segment occurrences in the overflow area • For HISAM, the overflow area means an overflow data set • For the first data set group of HDAM or PHDAM, the overflow area means an overflow area beyond the RAA • For the first data set group of HIDAM or PHIDAM, the overflow area means the blocks beyond the High key • Other than the previous conditions, no value is set for the overflow area. Null X'00' or zero is set in the field
NO_OF_SEGOVSPL	X	4	N	N	N	Y	N	N	Y	Y	Y	Y	N	0	4	8	The number of segment split prefixes in the overflow area

Table 54. Field names for segment information (continued)

Field name	Default value		Record type that can be specified Y: Can be specified N: Cannot be specified						Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set					Maximum length that can be specified 0: Cannot be specified			Description
	Attribute	Length	DB	PART	DSG	SEGM	RAP	PTR	HDAM	HIDAM	PHDAM	PHIDAM	HISAM	C	X	P	
NO_OF_SEGOVSDT	X	4	N	N	N	Y	N	N	Y	Y	Y	Y	N	0	4	8	The number of segment split data in the overflow area
COMPNAM	C	8	N	N	N	Y	N	N	Y	Y	Y	Y	Y	8	0	0	Segment edit/compression exit routine name
AVESGLN	X	2	N	N	N	Y	N	N	Y	Y	Y	Y	Y	0	2	4	Average length of segment occurrences
COMPFACT	C	6	N	N	N	Y	N	N	Y	Y	Y	Y	Y	6	0	0	• Compression factor. For the formula of compression factor, see "COMPRESSION FACTOR" on page 164 • If you want to set this field, run HD Pointer Checker with the COMPFACT=YES option

Table 55 on page 438 shows the field names for pointer information. For a non-HALDB, the values for each database is set. For a HALDB, the values for each partition is set.

Table 55. Field names for pointer information

Field name	Default value		Record type that can be specified Y: Can be specified N: Cannot be specified						Effective database organization Y: Value is set N: Blank (X'40') or null (X'00') is set					Maximum length that can be specified 0: Cannot be specified			Description
	Attribute	Length	DB	PART	DSG	SEGM	RAP	PTR	HDAM	HIDAM	PHDAM	PHIDAM	HISAM	C	X	P	
PTRTYPE	C	3	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	3	0	0	Pointer type: HF/HB/PTF/PTB/PP/LTF/LTB/LP/LCF/LCL/PCF/PCL/RAP/VLS/ELP/ELC
TSEGNAME	C	8	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	8	0	0	Target segment name
NO_OF_BLKM1CT	X	4	N	N	N	N	Y	Y	Y	Y	Y	Y	N	0	4	8	The number of pointers pointing to segments in the preceding block
NO_OF_SAMECT	X	4	N	N	N	N	Y	Y	Y	Y	Y	Y	N	0	4	8	The number of pointers pointing to segments in the same block
NO_OF_BLKP1CT	X	4	N	N	N	N	Y	Y	Y	Y	Y	Y	N	0	4	8	The number of pointers pointing to segments in the following block
NO_OF_BYNDCT	X	4	N	N	N	N	Y	Y	Y	Y	Y	Y	N	0	4	8	The number of pointers pointing to segments in blocks other than itself and the adjacent blocks within the data set
NO_OF_EXTRNCT	X	4	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	0	4	8	The number of pointers pointing to segments in blocks in other data sets
NO_OF_DIFBCT	X	4	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	0	4	8	The number of pointers pointing to segments in a different block

Output

This section describes the output that Export Utility produces.

HISTMSG data set

This section describes the HISTMSG data set, which is an output of Export Utility.

Overview

This data set contains the HISTORY data set by Message report.

The report is described in detail in the following section.

HISTORY Data Set Message report

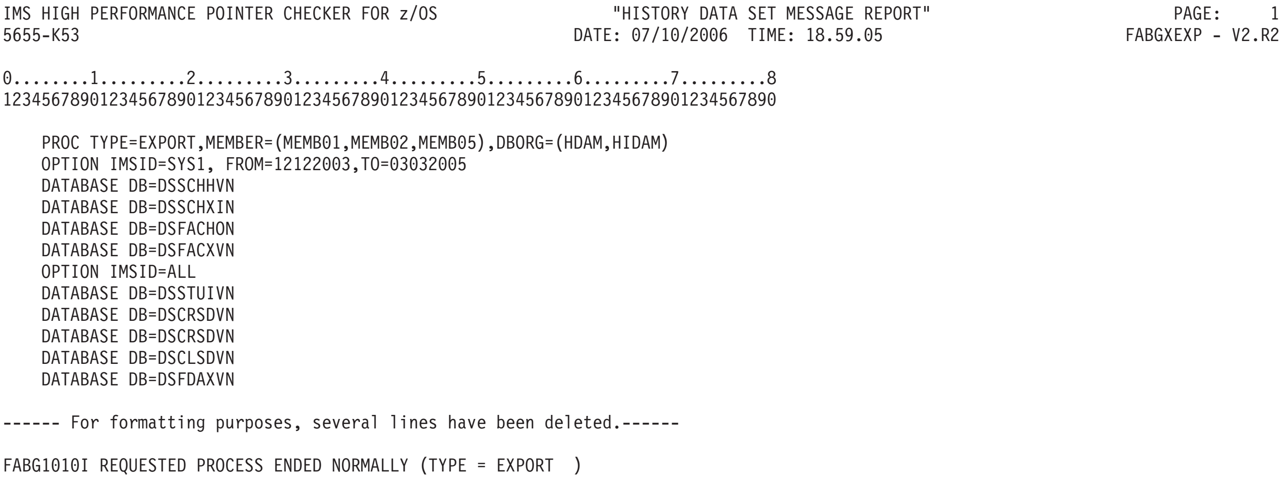
This report shows the control statements specified in the HISTIN data set. The following control statements are reported:

- PROC statement
- OPTION statement
- DATABASE statement

This report also shows the following messages from Export Utility:

- Warning and errors in analyzing the HISTIN control statements
- A result of the Export Utility run

Figure 155 shows a sample of the HISTORY Data Set Message report.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS              "HISTORY DATA SET MESSAGE REPORT"                    PAGE:     1
5655-K53                                               DATE: 07/10/2006  TIME: 18.59.05                    FABGXEXP - V2.R2

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

    PROC TYPE=EXPORT,MEMBER=(MEMB01,MEMB02,MEMB05),DBORG=(HDAM,HIDAM)
    OPTION IMSID=SYS1, FROM=12122003,TO=03032005
    DATABASE DB=DSSCHHVN
    DATABASE DB=DSSCHXIN
    DATABASE DB=DSFACHON
    DATABASE DB=DSFACXVN
    OPTION IMSID=ALL
    DATABASE DB=DSSTUIVN
    DATABASE DB=DSCRSDVN
    DATABASE DB=DSCRSDVN
    DATABASE DB=DSCLSDVN
    DATABASE DB=DSFDAXVN

------ For formatting purposes, several lines have been deleted.------

FABG1010I REQUESTED PROCESS ENDED NORMALLY (TYPE = EXPORT  )
```

Figure 155. HISTORY Data Set Message report

HISTPRT data set

This section describes the HISTPRT data set.

Overview

This data set contains the following reports:

- FABGRECI Statement report
- History Attribute report
- History Export Summary report

Each of the above reports is described in detail in the following sections.

FABGRECI Statement report

This report shows the name of FABGRECI member and the flat record definition statements that are specified in the member.

The member name is shown in the MEMBER NAME( ). The next lines are the statements in the member.

The following statements are reported:

- RECORD statement
- FIELD statement

This report also shows warning and error messages issued during analysis of the FABGRECI statement syntax and creating flat records.

Figure 156 shows a sample of the FABGRECI Statement report.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS                 "FABGRECI STATEMENT REPORT"                              PAGE:     1
5655-K53                                                  DATE: 07/10/2006  TIME: 18.59.05                          FABGXEXP - V2.R2

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

MEMBER NAME(FABPR001)
 RECORD TYPE=DB,RECID=A1
  FIELD NAME=DBDNAME,ATTR=C,LEN=8
  FIELD NAME=SCANDATE,ATTR=P,LEN=4
  FIELD NAME=SCANTIME,ATTR=X,LEN=3
  FIELD NAME=PARTID,ATTR=C,LEN=4
  FIELD VALUE=00,ATTR=X,LEN=1
  FIELD VALUE=' ',ATTR=C,LEN=1
  FIELD NAME=RECID,ATTR=C,LEN=2
  FIELD VALUE=00,ATTR=X,LEN=1
  FIELD NAME=IMSVER,ATTR=C,LEN=5
  FIELD NAME=IMSID,ATTR=C,LEN=4
  FIELD NAME=DBORG,ATTR=C,LEN=6
  FIELD NAME=ACCESS,ATTR=C,LEN=4
  FIELD VALUE=00,ATTR=X,LEN=1
  FIELD NAME=DBDSIZE,ATTR=X,LEN=2
  FIELD NAME=DBDDATE,ATTR=C,LEN=16
  FIELD NAME=DBRCOPT,ATTR=C,LEN=1
  FIELD NAME=DBRECN,ATTR=C,LEN=1
  FIELD NAME=PINDXNM,ATTR=C,LEN=8
  FIELD VALUE=00,ATTR=X,LEN=30
```

Figure 156. HISTPRT—FABGRECI Statement report

History Attribute report

This report contains an attribute information of the HISTORY data set that is specified in the HISTORY DD statement. Export Utility generates this report when TYPE=EXPORT is specified on the PROC control statement in the HISTIN data set.

For description of this report, see “History Attribute report” on page 407.

History Export Summary report

This report summarizes the result of the exporting processes. This report shows the number of flat records that were exported for each database. Export Utility generates this report when TYPE=EXPORT is specified for the PROC control statement in the HISTIN data set.

Figure 157 on page 441 shows a sample of the History Export Summary report.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS              "HISTORY EXPORT SUMMARY REPORT"                   PAGE:     1
5655-K53                                              DATE: 07/10/2006  TIME: 18.59.05                  FABGXEXP - V2.R2


HISTORY  DSNAME: TESTDS.PUBLIC.SAMPLE.HISTORY
FABGEXPF DSNAME: TESTDS.PUBLIC.SAMPLE.FLATADATA

DBDNAME                          = HISAMDB1
IMS ID                           = *ALL
DATE                             = 07/10/2006    - 07/10/2006


MEMBER   RECORD  RECORD    RECORD                COUNT
NAME     ID      TYPE      LENGTH           OF RECORDS
-------- ------- ------- -------- --------------------
FABPR001 A1      DB           102                    1
FABPR002 A2      DSG          138                    1
-------- ------- ------- -------- --------------------
                                                     2

-------------------------------- --------------------
TOTAL COUNT OF RECORDS                               2
```

Figure 157. HISTPRT—History Export Summary report

This report contains the following information:

HISTORY DSNAME
This is the name of the HISTORY data set. It is specified on the HISTORY DD statement in the Export Utility FABGXEXP JCL. The data in this data set is exported by Export Utility.

FABGEXPF DSNAME
This is the name of the flat file. It is specified on the FABGEXPF DD statement in the Export Utility FABGXEXP JCL. The flat records exported by Export Utility are stored in this data set.

DBD NAME
The DBD name (database name) that is processed by Export Utility.

IMS ID
The IMS ID that is processed by Export Utility.

EXPORTED DATE
The entries in the HISTORY data set that were taken by HD Pointer Checker in this period, are exported by Export Utility.

MEMBER NAME
This is a member name. If the FABGRECI DD statement is specified, this is a member name of the FABGRECI data set. If the FABGRECI DD statement is not specified, this is a predefined format member name.

RECORD ID
This is a record ID of the flat record. If the FABGRECI DD statement is specified, this is the value specified in the RECORD statement in the FABGRECI member. If the FABGRECI DD statement is not specified, this is the ID assigned by the predefined format.

RECORD TYPE
This is a record type of the flat record. If the FABGRECI DD statement is specified, this is the value specified in the RECORD statement in the FABGRECI member. If the FABGRECI DD statement is not specified, this is the record type assigned by the predefined format.

RECORD LENGTH
This is a data record length of the flat record. The flat record is a variable length record.

COUNTS of RECORDS
This is the number of flat records that are exported.

The subtotal number of flat records for the DBD is shown at the bottom of each DBD part.

TOTAL COUNTS OF RECORDS
This is the grand total number of flat records for all DBDs. It is shown on the last line of this report.

FABGEXPF data set (Flat File)

This section describes the FABGEXPF data set.

Overview

The FABGEXPF data set contains flat records that are generated by Export Utility. This data set is referred to as a flat file. This data set is required when TYPE=EXPORT is specified on the PROC statement in the HISTIN data set.

This data set is a sequential data set. The record is a variable length and the maximum length is 32,752 bytes.

There are two kinds of flat records:

- Predefined flat records
- User-defined flat records

Each of the records are described in the following sections. Restrictions and considerations are also described in this section.

Predefined flat records

If DUMMY is specified for the FABGRECI DD statement, or if the FABGRECI DD statement is not specified for the Export Utility FABGXEXP JCL, the flat records are generated in the predefined formats. These records are referred to as a predefined flat record.

The formats in Table 56 are prepared by Export Utility.

Table 56. Predefined flat records

Member name	Record type	Record ID	Contents
FABPR001	DB	A1	DBD information
FABPR002	DSG	A2	Data set group information -1
FABPR003	DSG	A3	Data set group information -2
FABPR004	SEGMENT	A4	Segment information -1
FABPR005	SEGMENT	A5	Segment information -2
FABPR006	RAP	A6	RAP information
FABPR007	POINTER	A7	Pointer information

To create the predefined flat record, specify the name to the MEMBER= on the PROC statement of the HISTIN data set. For an example of the HISTIN data set, see “Example 5: Creating the predefined flat records” on page 482.

The contents of the predefined members are stored in a sample library. They are described in the same syntax rules as the flat file record definitions. The syntax rule is described in "FABGRECI data set" on page 415.

The predefined formats cannot be changed. If you want to change the format, see "User-defined flat records."

User-defined flat records

When the FABGRECI data set is specified in the Export Utility FABGXEXP JCL, the flat records are generated in the user-defined formats. This record is referred to as a user-defined flat record.

For how to define the format, see "FABGRECI data set" on page 415.

Restrictions and considerations

The restrictions and considerations are common to the predefined flat record and the user-defined flat record.

Restrictions

The following items are not supported by Export Utility.

- Database
  - A primary index of HIDAM or PHIDAM database
  - A secondary index database
  - A partitioned secondary index (PSINDEX) of HALDB
  - An indirect list data set (ILDS) of HALDB
- Segment Type
  - A virtual logical child segment
- Pointer Type
  - Symbolic pointer
  - Index pointer
  - HISAM direct-address pointer
  - Counter field

  Note: The supported pointer types are follows:
  - Hierarchical Forward pointer (HF)
  - Hierarchical Backward pointer (HB)
  - Physical Twin Forward pointer (PTF)
  - Physical Twin Backward pointer (PTB)
  - Physical Parent pointer (PP)
  - Logical Twin Forward pointer (LTF)
  - Logical Twin Backward pointer (LTB)
  - Logical Parent pointer (LP)
  - Logical Child First pointer (LCF)
  - Logical Child Last pointer (LCL)
  - Physical Child First pointer (PCF)
  - Physical Child Last pointer (PCL)
  - Pointer to a data part of Variable Length Split segment (VLS)
  - RBA pointer to logical parent in an extended pointer set of HALDB (ELP)

   - RBA pointer to paired logical child for bidirectional logical relationships of HALDB (ELC)

Considerations

The sequence of flat records stored in the flat file has the following rules:

- The flat records will be sorted in the order of the database names in EBCDIC order. If there are multiple records of the same database name, the records will be in ascending order of the date.
- Within the same date, flat records will be in the order that they were specified for MEMBER= in the HISTIN control statement.
- If multiple flat records are generated from the same MEMBER, the order of the records will differ as follows according to the record type.
   - If TYPE=PART, in EBCDIC order of partition name
   - If TYPE=DSG, in ascending order of data set group
   - If TYPE=SEGMENT, in ascending order of segment code
   - If TYPE=POINTER, in the order that the pointers are within the segment prefix
- By using the OPTION IMSID= parameter of the HISTIN control statement, you can create flat records of a specific IMSID. Note that, however, flat records cannot be sorted in the order of IMSID. They will be in the order they were stored in the History record entries, that is the order HD Pointer Checker created the entries.

Example 1: Case of a Non-HALDB

Figure 158 shows a sample specification, for a non-HALDB, to generate a flat file.

```
//HISTIN  DD    *
PROC TYPE=EXPORT,
 MEMBER=(MEMDB,,MEMDSG,MEMSEG,MEMPTR),DBORG=HDAM
OPTION FROM=01012005,TO=12312005
DATABASE DB=HDAMDBA
DATABASE DB=HDAMDBB
/*

FABGRECI contains the following members:
MEMDB: RECORD TYPE=DB
MEMDSG: RECORD TYPE=DSG
MEMSEG: RECORD TYPE=SEGMENT
MEMPTR: RECORD TYPE=POINTER
```

Figure 158. Sample specification for a non-HALDB to generate a flat file

With the specification in Figure 158, flat records will be generated in the order shown in Figure 159 on page 445 in a flat file:

```
HDAMDBA,01/01/2005 flat record generated from MEMDB
HDAMDBA,01/01/2005 DSG A flat record generated from MEMDSG
HDAMDBA,01/01/2005 DSG B flat record generated from MEMDSG
HDAMDBA,01/01/2005 Segment Code=01 segment flat record generated from MEMSEG
HDAMDBA,01/01/2005 Segment Code=02 segment flat record generated from MEMSEG
HDAMDBA,01/01/2005 Segment Code=03 segment flat record generated from MEMSEG
HDAMDBA,01/01/2005 PTF pointer flat record generated from MEMPTR
HDAMDBA,01/01/2005 PTB pointer flat record generated from MEMPTR
HDAMDBA,01/01/2005 PCF pointer flat record generated from MEMSEG
HDAMDBA,01/01/2005 PCL pointer flat record generated from MEMSEG

HDAMDBA,01/02/2005 flat record generated from MEMDB
           :
           :
HDAMDBB,01/01/2005 flat record generated from MEMDB
            :
HDAMDBB,01/02/2005 flat record generated from MEMDB
            :
```

Figure 159. Flat file generated for a non-HALDB

Example 2: Case of a HALDB

Figure 160 shows a sample specification, for a HALDB, to generate a flat file.

For HALDB, the records of DSG, SEGMENT, and POINTER types will be sorted by the partition ID, and sorted by each order.

```
//HISTIN  DD    *
PROC TYPE=EXPORT,
 MEMBER=(MEMDB,MEMPART,MEMDSG,MEMSEG,MEMPTR),DBORG=PHDAM
OPTION FROM=01012005,TO=12312005
DATABASE DB=PHDAMA
DATABASE DB=PHDAMB
/*

FABGRECI contains the following members:
MEMDB: RECORD TYPE=DB
MEMPART: RECORD TYPE=PART
MEMDSG: RECORD TYPE=DSG
MEMSEG: RECORD TYPE=SEGMENT
MEMPTR: RECORD TYPE=POINTER
```

Figure 160. Sample specification for a HALDB to generate a flat file

With the specification in Figure 160, flat records will be generated in the order shown in Figure 161 on page 446 in a flat file:

```
PHDAMDBA,01/01/2005 flat record generated from MEMDB
PHDAMDBA,01/01/2005 partition id A flat record generated from MEMPART
PHDAMDBA,01/01/2005 partition id B flat record generated from MEMPART
PHDAMDBA,01/01/2005 partition id A DSG A flat record generated from MEMDSG
PHDAMDBA,01/01/2005 partition id A DSG B flat record generated from MEMDSG
PHDAMDBA,01/01/2005 partition id B DSG A flat record generated from MEMDSG
PHDAMDBA,01/01/2005 partition id B DSG B flat record generated from MEMDSG
PHDAMDBA,01/01/2005 partition id A Segment Code=01 segment flat record generated from MEMSEG
PHDAMDBA,01/01/2005 partition id A Segment Code=02 segment flat record generated from MEMSEG
PHDAMDBA,01/01/2005 partition id A Segment Code=03 segment flat record generated from MEMSEG
PHDAMDBA,01/01/2005 partition id B Segment Code=01 segment flat record generated from MEMSEG
PHDAMDBA,01/01/2005 partition id B Segment Code=02 segment flat record generated from MEMSEG
PHDAMDBA,01/01/2005 partition id B Segment Code=03 segment flat record generated from MEMSEG
PHDAMDBA,01/01/2005 partition id A PTF pointer flat record generated from MEMPTR
PHDAMDBA,01/01/2005 partition id A PTB pointer flat record generated from MEMPTR
PHDAMDBA,01/01/2005 partition id A PCF pointer flat record generated from MEMSEG
PHDAMDBA,01/01/2005 partition id A PCL pointer flat record generated from MEMSEG
PHDAMDBA,01/01/2005 partition id B PTF pointer flat record generated from MEMPTR
PHDAMDBA,01/01/2005 partition id B PTB pointer flat record generated from MEMPTR
PHDAMDBA,01/01/2005 partition id B PCF pointer flat record generated from MEMSEG
PHDAMDBA,01/01/2005 partition id B PCL pointer flat record generated from MEMSEG
PHDAMDBA,01/02/2005 flat record generated from MEMDB
            :
            :
PHDAMDBB,01/01/2005 flat record generated from MEMDB
            :
PHDAMDBB,01/02/2005 flat record generated from MEMDB
            :
```

Figure 161. Flat file generated for a HALDB

Chapter 22. Operating instructions for DB Historical Data Analyzer in the TSO/ISPF environment

In the TSO/ISPF environment, DB Historical Data Analyzer provides a function to create charts to show the historical trend of various aspects of IMS full-function database data set groups. The statistical data to be shown on the charts is obtained from the HISTORY data set produced by HD Pointer Checker and the Space Monitor graph record data set produced by Space Monitor.

The charts are displayed on your TSO terminal as the result of a dialog with DB Historical Data Analyzer and ICU through panels. The hard copies of the charts can also be obtained on printers that GDDM supports using GDDM provided printing utilities.

Using ICU standard functions, users can generate any of the nine types of chart: line graph, surface chart, histogram, bar chart, pie chart, venn diagram, polar chart, tower chart, and table chart. The charts can show database analysis information on the following major subjects:

- Database blocks
- Database free space
- Database segments
- Database root segments
- Database root segments and dependent segments
- Database records
- Space allocation

See "Panels and operations" on page 451 for the panels and their operations.

Topics:

- "Invoking DB Historical Data Analyzer"
- "Panels and operations" on page 451
- "Output (HD Analysis Graph)" on page 471
- "Customizing a graph chart through ICU" on page 472

Invoking DB Historical Data Analyzer

To run DB Historical Data Analyzer in the TSO/ISPF environment, the following two steps should be performed as the preparation for invoking DB Historical Data Analyzer. Refer to Chapter 24, "Customizing the DB Historical Data Analyzer," on page 485 for the detailed description.

1. Make sure that the TSO logon procedure is customized to contain GDDM program libraries and, optionally, the IMS HP Pointer Checker system library.
2. Allocate the required IMS HP Pointer Checker system libraries by issuing either a CLIST command or a TSO ALLOCATE command after the TSO session is established, if the required IMS HP Pointer Checker system library is not yet defined in the TSO logon procedure.

The following steps should be performed when you invoke DB Historical Data Analyzer:

3. Allocate the data sets that are required at run time using TSO ALLOCATE commands. (See "Run-time data set requirements" on page 448.)
4. Start a dialog with DB Historical Data Analyzer as an ISPF application. (See "Starting a dialog with DB Historical Data Analyzer" on page 449.)

Run-time data set requirements

To run DB Historical Data Analyzer, following data sets must be allocated in advance by using the appropriate TSO ALLOCATE commands.

GDDM data set requirements are shown in Table 57.

Table 57. Run-time data set requirements (GDDM data sets)

Data set	Optional or Required	Description
ADMSYMBL	Required	This is the GDDM symbol set data set used by ICU. DISP=SHR must be used to allocate the data set.
ADMCFORM	Optional	This is the GDDM chart format data set used by ICU. DISP=SHR or OLD can be used to allocate the data set.
ADMCDATA	Optional	This is the GDDM chart data data set used by ICU. DISP=SHR or OLD can be used to allocate the data set.
ADMGDF	Optional	This is the GDDM GDF (graphic data format) data set used by ICU. DISP=SHR or OLD can be used to allocate the data set.

For the description of the above data sets, see the documentations of GDDM products.

DB Historical Data Analyzer requires the data sets shown in Table 58.

Table 58. Run-time data set requirements (DB Historical Data Analyzer data sets)

Data set	Optional or Required	Description
HISTORY	Required	This is the HISTORY data set. DISP=SHR must be used to allocate the data set. For the description of the data set, refer to “HISTORY data set (HISTORY)” on page 367.
SPMNMBR	Optional	This is the control member data set. This data set is for the user of Space Monitor. It is required if you want to select a database data set by the member name of this data set. DISP=SHR must be used to allocate the data set. For the description of the data set, refer to “Control member data set (SPMNMBR)” on page 379.
SPMNSPDT	Optional	This is the Space Monitor graph record data set. It is required if you want to display a chart on space allocation. This data set is created and maintained by Space Monitor. DISP=SHR must be used to allocate the data set. For a description of the data set, refer to Part 5, “Space Monitor,” on page 489.

Starting a dialog with DB Historical Data Analyzer

DB Historical Data Analyzer is invoked as an ISPF application in the TSO/ISPF environment. The data sets described in “Run-time data set requirements” on page 448 must be allocated before starting a dialog with DB Historical Data Analyzer.

DB Historical Data Analyzer can be invoked in several different ways as follows:

- If ISPF is not started, issue the following command to start the dialog:

  ISPSTART PANEL(FABGP000)
- If the ISPF session is already started, run a command procedure (CLIST) that contains the following ISPEXEC command:

  ISPEXEC SELECT PANEL(FABGP000) NEWAPPL

You can also invoke DB Historical Data Analyzer in the following simple ways, if you customize DB Historical Data Analyzer:

- If you customize and install the sample FABGCMD0 CLIST (see Figure 162 on page 450) distributed with IMS HP Pointer Checker in your TSO CLIST library, execute **FABGCMD0**.
- If your ISPF/PDF Primary Option Menu panel is customized to invoke FABGCMD0, enter the selection code in the **OPTION** field and press **ENTER**.

Note: For more detailed information, see Chapter 24, “Customizing the DB Historical Data Analyzer,” on page 485.

When DB Historical Data Analyzer is invoked, the logo panel is displayed, and you can proceed to the “Historical Analysis Primary Menu” panel. For the description of the panels, refer to “Panels and operations” on page 451.

Sample TSO Command List (FABGCMD0)

A simple way to invoke DB Historical Data Analyzer is to use a CLIST. Figure 162 presents a sample CLIST that allocates all the required data sets and then invokes DB Historical Data Analyzer. Data set names shown in the CLIST may need to be modified to meet the requirements of your GDDM-ICU and IMS HP Pointer Checker installation.

```
/************************************************************/
/*                                                          */
/* SAMPLE CLIST FOR DB HISTORICAL DATA ANALYZER INVOCATION */
/*                                                          */
/************************************************************/
/*                                                          */
/* SYMBOL SETS ARE REQUIRED FOR GDDM BASE AND PGF.          */
  ALLOCATE FILE(ADMSYMBL) DATASET(GDDMSYM) SHR
/*                                                          */
/* ICU WHICH IS PART OF PGF.                                */
  ALLOCATE FILE(ADMCFORM) DATASET(ADMCFORM) SHR
/*                                                          */
/* ICU WHICH IS PART OF PGF.                                */
  ALLOCATE FILE(ADMCDATA) DATASET(ADMCDATA) SHR
/*                                                          */
/* GDF FILE IS FOR GDDM BASE AND THE ICU.                   */
  ALLOCATE FILE(ADMGDF) DATASET(ADMGDF) SHR
/*                                                          */
/* SPACE MONITOR CONTROL MEMBER DATA SET                    */
  ALLOCATE FILE(SPMNMBR) DATASET('SPMN.MEMBER') SHR
/*                                                          */
/* DB HISTORICAL DATA ANALYZER HISTORY DATA SET             */
  ALLOCATE FILE(HISTORY) DATASET('HIST.HISTORY') SHR
/*                                                          */
/* SPACE MONITOR GRAPH RECORD DATA SET                      */
  ALLOCATE FILE(SPMNSPDT) DATASET('SPMN.SPDT') SHR
/*                                                          */
/* INVOKING DB HISTORICAL DATA ANALYZER                     */
  IF &SYSISPF=ACTIVE THEN +
    ISPEXEC SELECT PANEL(FABGP000) NEWAPPL
  ELSE +
    ISPSTART PANEL(FABGP000)
/*                                                          */
  FREE FILE(ADMSYMBL)
  FREE FILE(ADMCFORM)
  FREE FILE(ADMCDATA)
  FREE FILE(ADMGDF)
  FREE FILE(SPMNMBR)
  FREE FILE(HISTORY)
  FREE FILE(SPMNSPDT)
/*                                                          */
```

Figure 162. Sample TSO Command List (FABGCMD0)

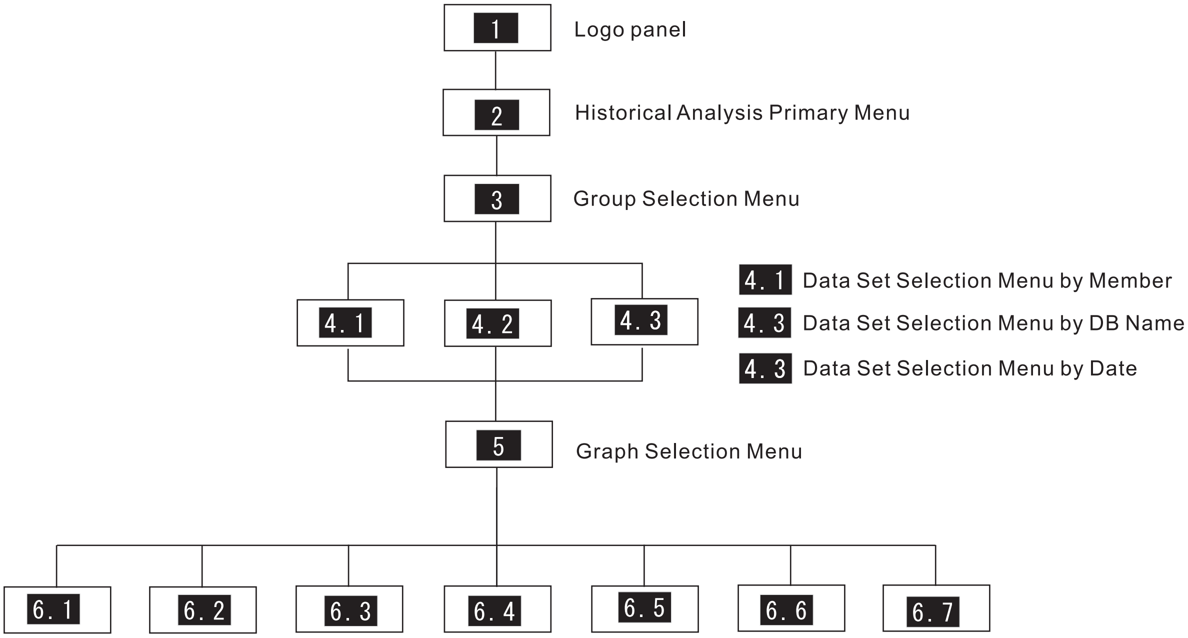
Panels and operations

This section describes the panels through which the user performs the interactive operations.

Figure 163 shows the overall panel structure. The numbers shown in the figure are used to identify the panels throughout this section.

Panel structure

This section shows the overall panel structure.


1 Logo panel
2 Historical Analysis Primary Menu
3 Group Selection Menu
4.1
4.2
4.3
4.1 Data Set Selection Menu by Member
4.3 Data Set Selection Menu by DB Name
4.3 Data Set Selection Menu by Date
5 Graph Selection Menu
6.1
6.2
6.3
6.4
6.5
6.6
6.7

6.1 Historical Analysis by DB Blocks

6.2 Historical Analysis by DB Free Space

6.3 Historical Analysis by DB Segments

6.4 Historical Analysis by Root Segments

6.5 Historical Analysis by DB Roots and Dependent Segments

6.6 Historical Analysis by DB Records

6.7 Historical Analysis by Space Allocation

Notes: 1) Panel 4.x is generically called “Data Set Selection Menu” in this manual.

2) Panel 6.x is generically called “Historical Analysis Menu” in this manual.

Figure 163. Panel structure

General operation procedure

Follow the steps shown below in order to obtain a graph chart through a dialog with DB Historical Data Analyzer.

Step 1: Selecting a way to obtain a database data set name list

On the “Group Selection Menu” panel, select one of the following ways to obtain a database data set name list, from which you can select one database data set.

- If you are using Space Monitor and have your own Space Monitor control member data set (SPMNMBR), you can use this data set to obtain the name list. In this case, select Group 1 on the “Group Selection Menu” panel, and specify a member name of the SPMNMBR data set.
- If you know the database name and/or ddname of the database data set you want to process, select Group 2 on the “Group Selection Menu” panel, and specify the database name and/or ddname.

  If you know only the database name, leave the ddname field blank. Then all the data sets of the specified database are listed.
- If you know only the date when the real databases were processed by HD Pointer Checker, you can obtain the name list using this date as the key. Select Group 3 on the “Group Selection Menu” panel, and specify the date.
- If you have no idea how to select the database data set, you can list all the database data sets in the HISTORY data set. To do this, select Group 2 on the “Group Selection Menu” panel, and leave the DB name and DD name fields blank. This way, you can obtain the name list of all database data sets in the HISTORY data set.

Step 2: Selecting a database data set

Select one database data set on one of the “Data Set Selection Menu” panels (panel 4.x in Figure 163 on page 451).

Step 3: Selecting a major database analysis item

On the “Graph Selection Menu” panel, select one major subject from the seven major database analysis items. For instance, if you want to see the historical trend of the database free space use of the specified database data set group, select *DB Free Space*.

- **DB Blocks**

  Select this item if you want to analyze the historical trend of the database data set group from the aspect of ***blocks (or CIs)***. This item applies to HDAM, HIDAM, PHDAM, and PHIDAM (except for the DSG-X) data set groups.
- **DB Free Space**

  Select this item if you want to analyze the historical trend of the database data set group from the aspect of ***free space***. This item applies to HDAM, HIDAM, PHDAM, and PHIDAM (except for the DSG-X) data set groups.
- **DB Segments**

  Select this item if you want to analyze the historical trend of the database data set group from the aspect of ***database segments***. This item applies to HISAM, HDAM, HIDAM, PHDAM, and PHIDAM (except for the DSG-X) data set groups.
- **DB Root Segments**

  Select this item if you want to analyze the historical trend of the database data set group from the aspect of ***database root segments***. This item applies to the primary data set groups of the HDAM and PHDAM databases.
- **DB Root and Dependent Segments**

Select this item if you want to analyze the historical trend of the database data set group from the aspect of ***database root segments and their dependent segments***. This item applies to the primary data set groups of HDAM, HIDAM, PHDAM, and PHIDAM databases.

- **DB Records**

  Select this item if you want to analyze the historical trend of the database data set group from the aspect of ***database records***. This item applies to the primary data set groups of HISAM, HDAM, HIDAM, PHDAM, and PHIDAM databases.

- **Space Allocation**

  Select this item if you want to analyze the historical trend of the database data set group from the aspect of ***space allocation***. This item applies to all database data sets (HISAM, HDAM, HIDAM, PHDAM, PHIDAM, and index databases) that have entries in the Space Monitor graph records data set (SPMNSPDT).

  Space allocation information is obtained from the Space Monitor graph record data set (SPMNSPDT), which is created and maintained by Space Monitor.

Table 59 shows the major database analysis items and the database types supported.

Table 59. Major database analysis items and supported database types

Major database analysis items	HISAM	HDAM	HIDAM	Index
1. DB Blocks	N	Y	Y	N
2. DB Free Space	N	Y	Y	N
3. DB Segments	Y (1)	Y (2)	Y (2)	N
4. DB Root Segments	N	Y (3)	N	N
5. DB Root and Dependent Segments	N	Y (3)	Y (3)	N
6. DB Records	Y (3)	Y (3)	Y (3)	N
7. Space Allocation	Y	Y	Y	Y

Note:

- (1) indicates that some detailed items in this major item group are not applicable or applicable only to the primary data set groups.
- (2) indicates that some detailed items in this major item group are applicable only to the primary data set groups.
- (3) indicates that this major item group is applicable only to the primary data set groups.
- PHDAM is included in the **HDAM** column header.
- PHIDAM DSG A-J is included in the **HIDAM** column header.
- PHIDAM DSG-X and PSINDEX are included in the **Index** column header.

Step 4: Selecting detailed database analysis items

On each major database analysis item, you can further select detailed items that are actually displayed on the graph chart. For example, if you choose *DB free space* as the major item, you can select one of the three detailed items: the total number of free space elements, total bytes of free space, or percentage of free space in the data set.

Select detailed database analysis items on one of the “Historical Analysis Menu” panels (panel 6.x in Figure 163 on page 451).

Step 5: Customizing a graph chart through ICU

Finally you get a chart (line graph) of the detailed database analysis items you selected. Through a dialog with GDDM-ICU, you can modify the chart to make it more suitable for your use. For the description of the data that is passed from DB Historical Data Analyzer to GDDM-ICU, see “Customizing a graph chart through ICU” on page 472.

Understanding DB Historical Data Analyzer panels

DB Historical Data Analyzer panels and operations are described in the following section. Before starting the operations, you should be aware of the following:

- ISPF commands are supported in DB Historical Data Analyzer panels. Since the HELP, END, RETURN, UP, and DOWN commands are used during the panel operations, it is recommended that you assign these commands to program function (PF) keys.
- The jump function "=X" is supported in any panels except for the Logo panel. "=X" is used to exit from DB Historical Data Analyzer.
- "Scroll==>" field can be set as any other ISPF panels.
- The ISPF PF key assignments are also effective during DB Historical Data Analyzer dialog. If you want to change the default assignments, use the ISPF **KEYS** command.

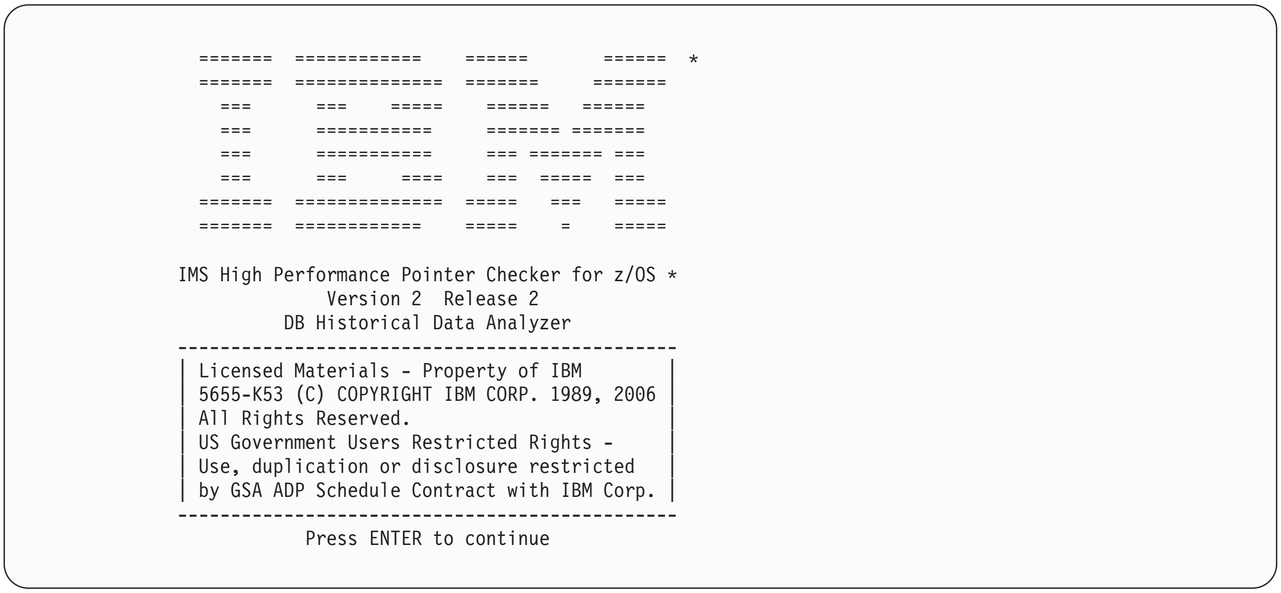
  Note: The **END** command must be assigned to a PF key before you start a dialog with DB Historical Data Analyzer.
- It is recommended that you display the PF key definitions at the bottom of the panel by using the ISPF **PFSHOW** command.
- The ISPF **HELP** command provides online help information of the DB Historical Data Analyzer panels. The HELP command is supported on any panels except for the Logo panel. Once in the HELP, press ENTER to scroll the panels, if necessary, and enter the END or RETURN command to return to the panel on which you entered the HELP command.

Descriptions of the panels

This section provides the descriptions of the panels.

Logo

Figure 164 shows the logo panel that is displayed when you start a dialog with DB Historical Data Analyzer.



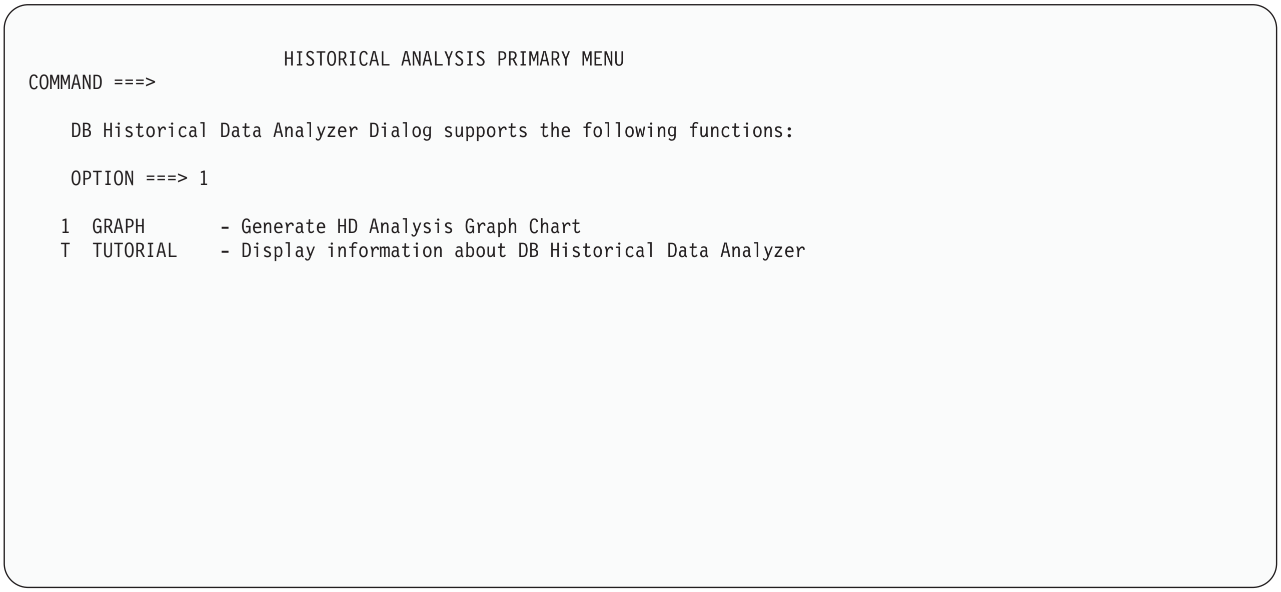
```
  =======  ============    ======       ======  *
  =======  ==============  =======     =======
    ===      ===    =====    ======   ======
    ===      ===========     ======= =======
    ===      ===========     === ======= ===
    ===      ===     ====    ===  =====  ===
  =======  ==============  =====   ===   =====
  =======  ============    =====    =    =====

IMS High Performance Pointer Checker for z/OS *
              Version 2  Release 2
          DB Historical Data Analyzer
-----------------------------------------------
| Licensed Materials - Property of IBM        |
| 5655-K53 (C) COPYRIGHT IBM CORP. 1989, 2006 |
| All Rights Reserved.                        |
| US Government Users Restricted Rights -     |
| Use, duplication or disclosure restricted   |
| by GSA ADP Schedule Contract with IBM Corp. |
-----------------------------------------------
            Press ENTER to continue
```

Figure 164. Logo

Press **ENTER** to proceed to the next panel, or **END** to cancel.

Historical Analysis Primary Menu

Figure 165 shows the primary menu panel that shows the function supported by DB Historical Data Analyzer.



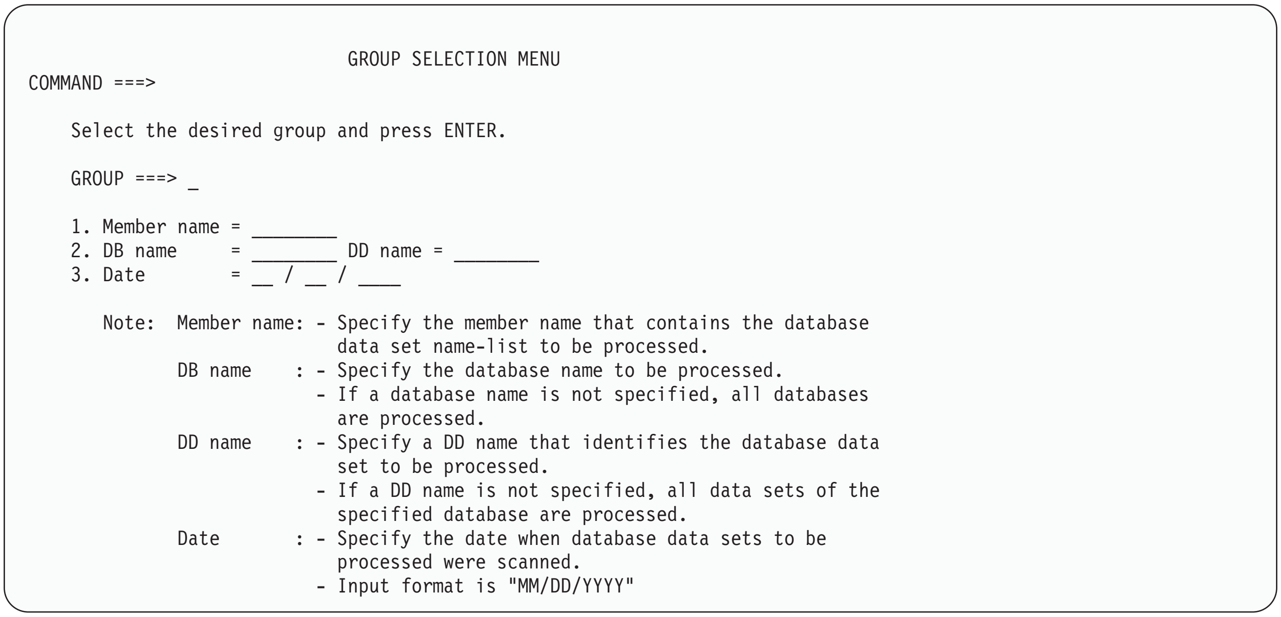
```
                        HISTORICAL ANALYSIS PRIMARY MENU
COMMAND ===>

    DB Historical Data Analyzer Dialog supports the following functions:

    OPTION ===> 1

   1  GRAPH       - Generate HD Analysis Graph Chart
   T  TUTORIAL    - Display information about DB Historical Data Analyzer
```

Figure 165. Historical Analysis Primary Menu

Select Option 1 (GRAPH) and press **ENTER** to proceed to the “Group Selection Menu” panel, or select Option T (TUTORIAL) to see the tutorial for panel operation. Or, enter **END** to return to the Logo panel.

Group Selection Menu

The panel shown in Figure 166 is displayed following the “Historical Analysis Primary Menu” panel. On this panel, you should select one of the three ways to obtain a database data set name list.



```
                              GROUP SELECTION MENU
COMMAND ===>

    Select the desired group and press ENTER.

    GROUP ===> _

    1. Member name = ________
    2. DB name     = ________ DD name = ________
    3. Date        = __ / __ / ____

       Note:  Member name: - Specify the member name that contains the database
                             data set name-list to be processed.
              DB name    : - Specify the database name to be processed.
                           - If a database name is not specified, all databases
                             are processed.
              DD name    : - Specify a DD name that identifies the database data
                             set to be processed.
                           - If a DD name is not specified, all data sets of the
                             specified database are processed.
              Date       : - Specify the date when database data sets to be
                             processed were scanned.
                           - Input format is "MM/DD/YYYY"
```

Figure 166. Group Selection Menu

Select a group number (1, 2, or 3), and specify the associated fields. Press **ENTER** to proceed to the next panel, or enter **END** to return to the "Historical Analysis Primary Menu" panel.

Group 1:

- Select this group if you have SPMNMBR data set, and want to list all the database data sets in a member of this data set. When you select Group 1, specify the following field:

  Member name =

  Specifies the member name in SPMNMBR data set. All the database data sets specified by the control statements in the member are listed on “Data Set Selection Menu by Member” panel (see “Data Set Selection Menu by Member” on page 457).

  Control statements that describe non-IMS data sets, or invalid control statements are ignored.

Group 2:

- Select this group if you want to list all the database data sets that have a particular database name and/or ddname. All the database data sets that have the specified database name and/or ddname are listed on “Data Set Selection Menu by DB Name” panel (see “Data Set Selection Menu by DB Name” on page 459). When you select Group 2, specify the following fields:

DB name =
Specifies the database name of the database data sets you want to list. For HALDB, the master database name should be used.

Note: If you choose this Group 2 and leave the DB name field blank, all the databases in the HISTORY data set are listed.

DD name =
Specifies the ddname of the database data sets you want to list within the specified database. For HALDB, use the ddname created by concatenating the partition name and the DSG suffix character (A-J, X). If no ddname is specified, all the data sets of the specified database are listed. If Group 2 is selected and only the ddname is specified, an error message is issued.

Group 3:

- Select this group if you want to list all the database data sets that were processed by HD Pointer Checker on a particular date (see note). All the database data sets that have HISTORY data set records of the specified date are listed on “Data Set Selection Menu by Date” panel (see “Data Set Selection Menu by Date” on page 460).

  Selecting this group is useful if you do not remember the database name but do know when the database was processed by HD Pointer Checker.

  Note: If an image copy data set is used, this is the date when the image copy data set was created.

  When you select Group 3, specify the following field:

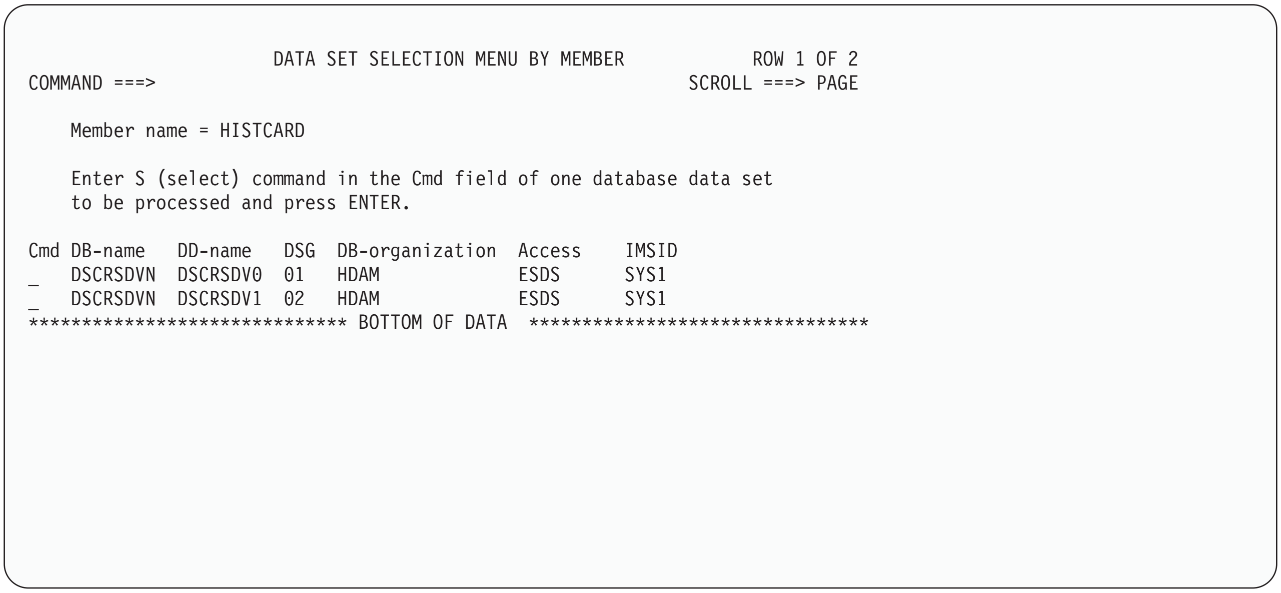
  Date =
  Specifies the date of the HISTORY data set record. The input format is MM/DD/YYYY, where MM is the month, DD is the date, and YYYY is the year.

Data Set Selection Menu by Member

This panel is displayed when you select "1" (Member name) on the “Group Selection Menu“ panel. On this panel, select one database data set to be processed from the database data set name list.

Note: Database data sets that are specified by the control statements but have no entries in the HISTORY data set are not shown on the panel.

The panel shown in Figure 167 on page 458 assumes that the user specified "HISTCARD" as the member name on the “Group Selection Menu“ panel. The member name is shown on the panel.



```
                       DATA SET SELECTION MENU BY MEMBER            ROW 1 OF 2
COMMAND ===>                                                  SCROLL ===> PAGE

    Member name = HISTCARD

    Enter S (select) command in the Cmd field of one database data set
    to be processed and press ENTER.

Cmd DB-name   DD-name   DSG  DB-organization  Access    IMSID
_   DSCRSDVN  DSCRSDV0  01   HDAM             ESDS      SYS1
_   DSCRSDVN  DSCRSDV1  02   HDAM             ESDS      SYS1
***************************** BOTTOM OF DATA  ********************************
```

Figure 167. Data Set Selection Menu by Member

Enter **UP** or **DOWN** to scroll up or down the database data set name list, if necessary. Enter the "S" (select) command in the Cmd field of the selected database data set, and press **ENTER**. The “Graph Selection Menu” panel is displayed (see “Graph Selection Menu” on page 461). Enter **END** to return to the “Group Selection Menu” panel.

The list of database data sets shown on this panel contains the following information:

DB-name
This is the name of the DBD as coded in the NAME= keyword of the DBD macro in the DBD that was processed by the HD Pointer Checker run.

DD-name
This is the ddname of the this data set group.

DSG
This is the data set group number. It is the ordinal number of the data set group. 1 to 10 is shown for non-HALDB, A to J and X for HALDB.

DB-organization
This is the IMS organization used for this database. One of the following is shown:

- SHISAM
- HISAM
- HISAM OFLW
- HDAM
- HIDAM
- HIDAM INDEX
- HIDAM INDEX OFLW
- 2NDARY INDX
- 2NDARY INDX OFLW
- SHR 2ND IDX
- SHR 2ND IDX OFLW
- PHDAM
- PHIDAM
- PHIDAM IDX
- PSINDEX

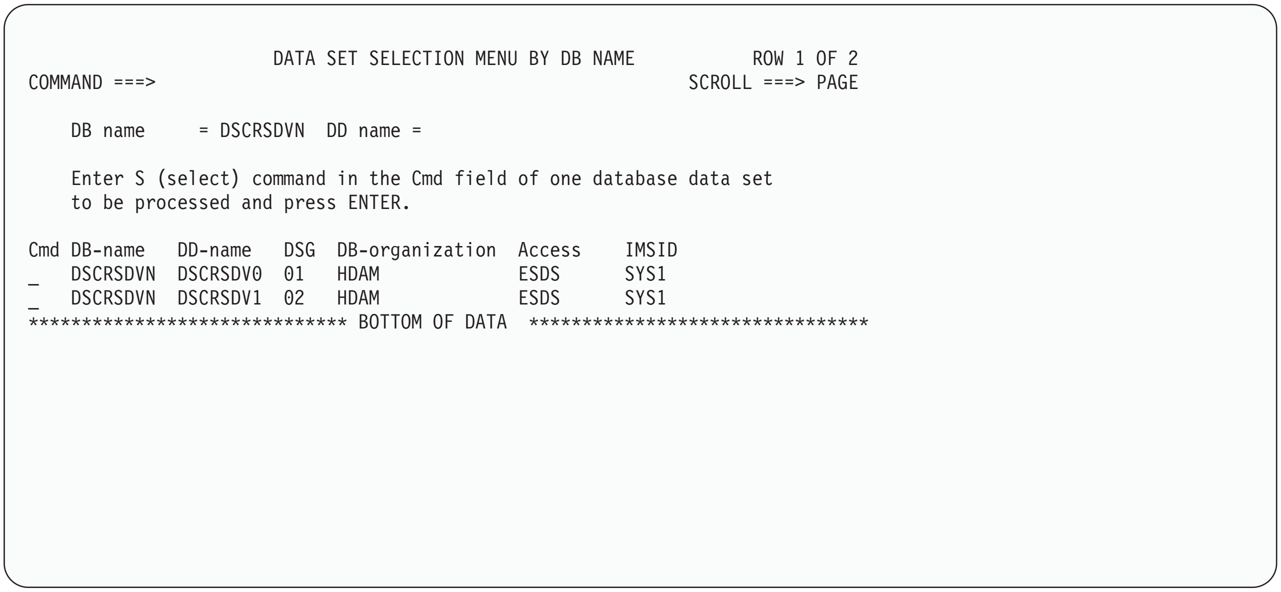
Access
This is the method used for access to this database data set (OSAM, ESDS, or KSDS).

IMSID
The IMSID of the database data set is displayed when the Multiple-IMSID option is enabled. Otherwise, no data is displayed.

Data Set Selection Menu by DB Name

The panel shown in Figure 168 is displayed when you select "2" (DB name, DD name) on the “Group Selection Menu” panel. On this panel, select one database data set from the database data set name list.

The following panel assumes that the user specified "DSCRSDVN" as the DB name on the “Group Selection Menu” panel. The DB name is shown on the panel.



```
                       DATA SET SELECTION MENU BY DB NAME           ROW 1 OF 2
COMMAND ===>                                                   SCROLL ===> PAGE

    DB name     = DSCRSDVN  DD name =

    Enter S (select) command in the Cmd field of one database data set
    to be processed and press ENTER.

Cmd DB-name   DD-name   DSG  DB-organization  Access    IMSID
_   DSCRSDVN  DSCRSDV0  01   HDAM             ESDS      SYS1
_   DSCRSDVN  DSCRSDV1  02   HDAM             ESDS      SYS1
***************************** BOTTOM OF DATA  ********************************
```

Figure 168. Data Set Selection Menu by DB Name

Enter **UP** or **DOWN** to scroll up or down the database data set name list, if necessary. Enter the “S” (select) command in the Cmd field of the selected database data set, and press **ENTER**. The “Graph Selection Menu” panel is displayed (see “Graph Selection Menu” on page 461). Enter **END** to return to the “Group Selection Menu” panel.

The list of database data sets shown on this panel contains the following information:

DB-name
This is the name of the DBD as coded in the NAME= keyword of the DBD macro in the DBD that was processed by the HD Pointer Checker run.

DD-name
This is the ddname of the this data set group.

DSG
This is the data set group number. It is the ordinal number of the data set group. 1 to 10 is shown for non-HALDB, A to J and X for HALDB.

DB-organization
This is the IMS organization used for this database. One of the following is shown:

- SHISAM
- HISAM
- HISAM OFLW
- HDAM
- HIDAM
- HIDAM INDEX
- HIDAM INDEX OFLW
- 2NDARY INDX
- 2NDARY INDX OFLW
- SHR 2ND IDX
- SHR 2ND IDX OFLW
- PHDAM
- PHIDAM
- PHIDAM IDX
- PSINDEX

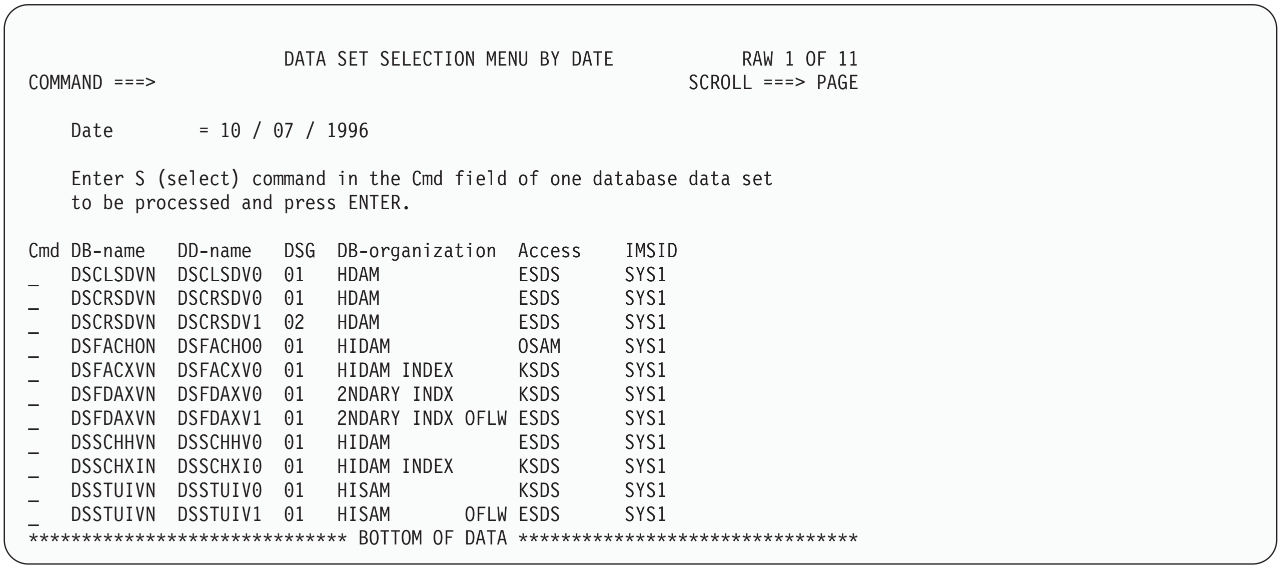
Access
This is the method used for access to this database data set (OSAM, ESDS, or KSDS).

IMSID
The IMSID of the database data set is displayed when the Multiple-IMSID option is enabled. Otherwise, no data is displayed.

Data Set Selection Menu by Date

The panel shown in Figure 169 is displayed when you select "3" (Date) on the “Group Selection Menu” panel. On this panel, select one database data set from the list of database data sets of the specified date.

The following panel assumes that the user specified "10/07/1996" as the date on the “Group Selection Menu” panel. The date is shown on the panel.



```
                        DATA SET SELECTION MENU BY DATE            RAW 1 OF 11
COMMAND ===>                                                  SCROLL ===> PAGE

    Date        = 10 / 07 / 1996

    Enter S (select) command in the Cmd field of one database data set
    to be processed and press ENTER.

Cmd DB-name   DD-name   DSG  DB-organization  Access    IMSID
_   DSCLSDVN  DSCLSDV0  01   HDAM             ESDS      SYS1
_   DSCRSDVN  DSCRSDV0  01   HDAM             ESDS      SYS1
_   DSCRSDVN  DSCRSDV1  02   HDAM             ESDS      SYS1
_   DSFACHON  DSFACHO0  01   HIDAM            OSAM      SYS1
_   DSFACXVN  DSFACXV0  01   HIDAM INDEX      KSDS      SYS1
_   DSFDAXVN  DSFDAXV0  01   2NDARY INDX      KSDS      SYS1
_   DSFDAXVN  DSFDAXV1  01   2NDARY INDX OFLW ESDS      SYS1
_   DSSCHHVN  DSSCHHV0  01   HIDAM            ESDS      SYS1
_   DSSCHXIN  DSSCHXI0  01   HIDAM INDEX      KSDS      SYS1
_   DSSTUIVN  DSSTUIV0  01   HISAM            KSDS      SYS1
_   DSSTUIVN  DSSTUIV1  01   HISAM       OFLW ESDS      SYS1
****************************** BOTTOM OF DATA ********************************
```

Figure 169. Data Set Selection Menu by Date

Enter **UP** or **DOWN** to scroll up or down the database data set name list, if necessary. Enter the "S" (select) command in the Cmd field of the selected database data set, and press **ENTER**. The “Graph Selection Menu” panel is displayed. Enter **END** to return to the “Group Selection Menu” panel.

The list of database data sets shown on this panel contains the following information:

DB-name
This is the name of the DBD as coded in the NAME= keyword of the DBD macro in the DBD that was processed by the HD Pointer Checker run.

DD-name
This is the ddname of the this data set group.

DSG
This is the data set group number. It is the ordinal number of the data set group. 1 to 10 is shown for non-HALDB, A to J and X for HALDB.

DB-organization
This is the IMS organization used for this database. One of the following is shown:

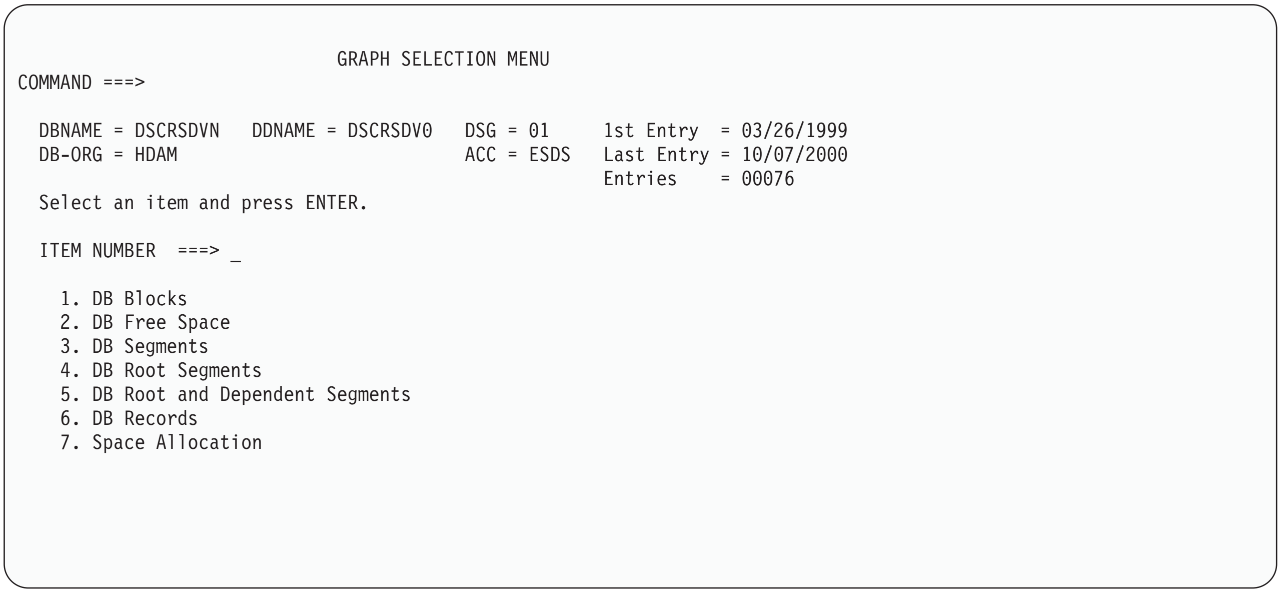
- SHISAM
- HISAM
- HISAM OFLW
- HDAM
- HIDAM
- HIDAM INDEX
- HIDAM INDEX OFLW
- 2NDARY INDX
- 2NDARY INDX OFLW
- SHR 2ND IDX
- SHR 2ND IDX OFLW
- PHDAM
- PHIDAM
- PHIDAM IDX
- PSINDEX

Access
This is the method used for access to this database data set (OSAM, ESDS, or KSDS).

IMSID
The IMSID of the database data set is displayed when the Multiple-IMSID option is enabled. Otherwise, no data is displayed.

Graph Selection Menu

The panel shown in Figure 170 is displayed when you select a database data set on the “Data Set Selection Menu by Member,” “Data Set Selection Menu by DB Name,” or “Data Set Selection Menu by Date” panel.



```
                              GRAPH SELECTION MENU
COMMAND ===>

  DBNAME = DSCRSDVN   DDNAME = DSCRSDV0   DSG = 01     1st Entry  = 03/26/1999
  DB-ORG = HDAM                           ACC = ESDS   Last Entry = 10/07/2000
                                                       Entries    = 00076
  Select an item and press ENTER.

  ITEM NUMBER  ===> _

    1. DB Blocks
    2. DB Free Space
    3. DB Segments
    4. DB Root Segments
    5. DB Root and Dependent Segments
    6. DB Records
    7. Space Allocation
```

Figure 170. Graph Selection Menu

Select an item and press **ENTER**, or enter **END** to return to the previous panel.

DB Historical Data Analyzer creates charts to see the trend of the IMS full-function database data set on the following seven items:

The entry information of a selected database data set shown on this panel contains the following information:

1st Entry
This is the key date entry of the oldest HISTORY data set records of the selected database data set that was processed by the HD Pointer Checker run.

Last Entry
This is the key date entry of the newest HISTORY data set records of the selected database data set that was processed by the HD Pointer Checker run.

Entries
This is the number of key date entries of the selected database data set.

Note: If an image copy data set is used by the HD Pointer Checker run, the date is when the image copy data set was created.

1. DB Blocks
Provides the historical trend of the database data set group from the aspect of blocks (or CIs). This item applies to HDAM, HIDAM, PHDAM, and PHIDAM (except for DSG-X) data set groups.

2. DB Free Space
Provides the historical trend of the database data set group from the aspect of free space. This item applies to HDAM, HIDAM, PHDAM, and PHIDAM (except for DSG-X) data set groups.

3. DB Segments
Provides the historical trend of the database data set group from the aspect of database segments. This item applies to HISAM, HDAM, HIDAM, PHDAM, and PHIDAM (except for DSG-X) data set groups.

4. DB Root Segments
Provides the historical trend of the database data set group from the aspect of database root segments. This item applies to the primary data set groups of the HDAM and PHDAM databases.

If you select this item, “Historical Analysis by Root Segments” panel is displayed (see “Historical Analysis by Root Segments” on page 466).

5. DB Root and Dependent Segments
Provides the historical trend of the database data set group from the aspect of database root segments and their dependent segments. This item applies to the primary data set groups of HDAM, HIDAM, PHDAM, and PHIDAM databases.

If you select this item, “Historical Analysis by DB Roots and Dependent Segments” panel is displayed (see “Historical Analysis by DB Roots and Dependent Segments” on page 468).

6. DB Records
Provides the historical trend of the database data set group from the aspect of database records. This item applies to the primary data set groups of HISAM, HDAM, HIDAM, PHDAM and PHIDAM databases.

If you select this item, “Historical Analysis by DB Records” panel is displayed. (see “Historical Analysis by DB Records” on page 469).

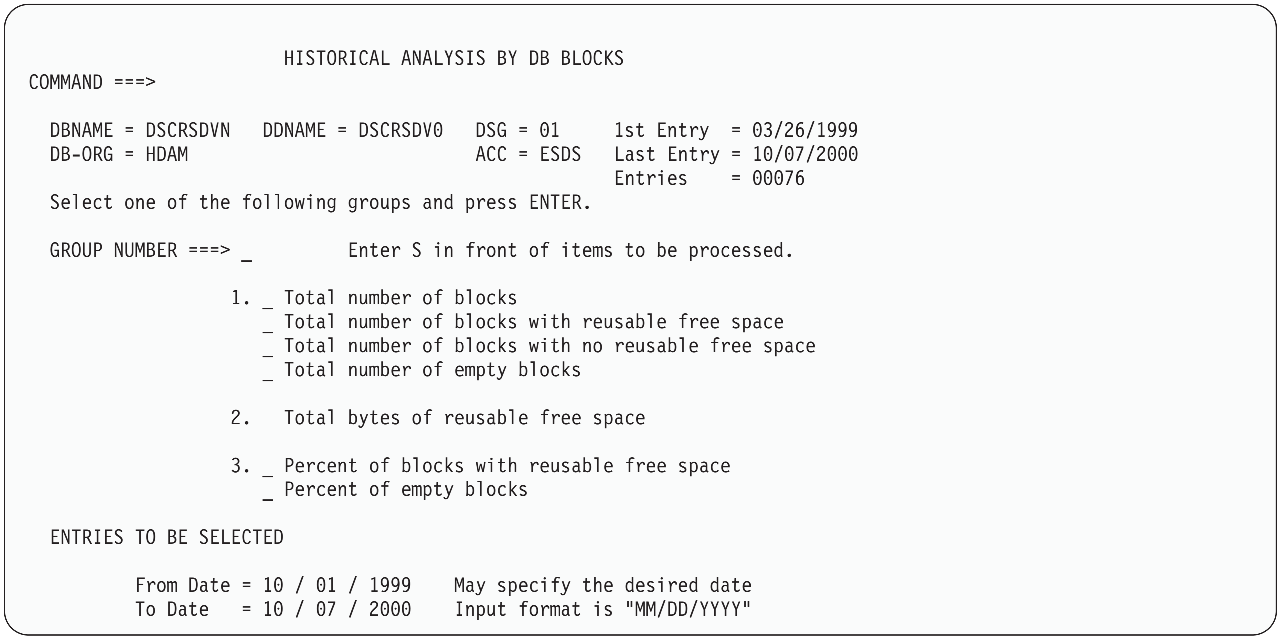
7. Space Allocation
Provides the historical trend of the database data set group from the aspect of space allocation. This item applies to all database data sets (HISAM, HDAM, HIDAM, PHDAM, PHIDAM, and index databases) that have entries in the Space Monitor graph records data set (SPMNSPDT).

Space allocation information is obtained from the Space Monitor graph record data set (SPMNSPDT), which is created and maintained by Space Monitor.

If you select this item, “Historical Analysis by Space Allocation” panel is displayed (see “Historical Analysis by Space Allocation” on page 470).

Historical Analysis by DB Blocks

On the panel that is shown in Figure 171, you can select detailed items of database data set blocks/CIs analysis information. DB Historical Data Analyzer will show the historical trend of the selected items for the database data set group identified by the DBNAME and DDNAME fields.



```
                        HISTORICAL ANALYSIS BY DB BLOCKS
COMMAND ===>

  DBNAME = DSCRSDVN   DDNAME = DSCRSDV0   DSG = 01     1st Entry  = 03/26/1999
  DB-ORG = HDAM                           ACC = ESDS   Last Entry = 10/07/2000
                                                       Entries    = 00076
  Select one of the following groups and press ENTER.

  GROUP NUMBER ===> _         Enter S in front of items to be processed.

                 1. _ Total number of blocks
                    _ Total number of blocks with reusable free space
                    _ Total number of blocks with no reusable free space
                    _ Total number of empty blocks

                 2.   Total bytes of reusable free space

                 3. _ Percent of blocks with reusable free space
                    _ Percent of empty blocks

  ENTRIES TO BE SELECTED

          From Date = 10 / 01 / 1999    May specify the desired date
          To Date   = 10 / 07 / 2000    Input format is "MM/DD/YYYY"
```

Figure 171. Historical Analysis by DB Blocks

Select a group number (1 to 3), and select one or more items under the selected group, if necessary. Press **ENTER** to pass the control to ICU, or enter **END** to return to the “Graph Selection Menu” panel.

The items shown on this panel are:

Group number 1:

- Total number of blocks/CIs.
- Total number of blocks/CIs that have reusable free space from the viewpoint of bit maps.
- Total number of blocks/CIs that have no reusable free space from the viewpoint of bit maps. Nonreusable free space is a free space element smaller than the longest segment in the data set (as defined in the DBD).
- Total number of empty blocks/CIs.

Group number 2:

- Total bytes of reusable free space.

Group number 3:

- The percentage of blocks/CIs that have reusable free space from the viewpoint of bit maps within the total number of blocks/CIs

- The percentage of empty blocks/CIs within the total number of blocks/CIs

The dates shown on this panel are:

From Date
This default date is the key date entry of the last one year old (actually 372 days) of HISTORY data set record or the key date entry of the first HISTORY data set record if the oldest record was created within a year.

Specify the desired date if necessary.

Note: The from date may be adjusted, because the range between “From Date” and “To Date” must be a multiple of 12.

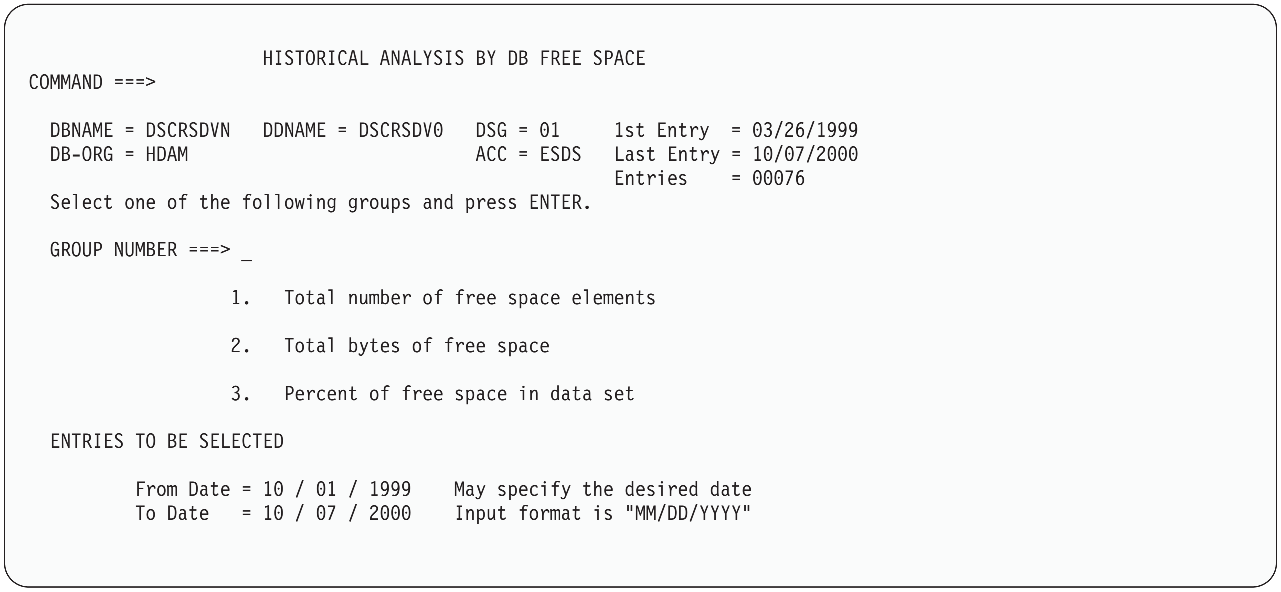
To Date
This default date is the key date entry of the latest HISTORY data set record of the selected database data set.

Specify the desired date if necessary.

Note: If an image copy data set is used by the HD Pointer Checker run, the date is when the image copy data set was created.

Historical Analysis by DB Free Space

On the panel that is shown in Figure 172, you can select detailed items of free space analysis information. DB Historical Data Analyzer will show the historical trend of the selected items for the database data set group identified by the DBNAME and DDNAME fields.



```
                      HISTORICAL ANALYSIS BY DB FREE SPACE
COMMAND ===>

  DBNAME = DSCRSDVN   DDNAME = DSCRSDV0   DSG = 01     1st Entry  = 03/26/1999
  DB-ORG = HDAM                           ACC = ESDS   Last Entry = 10/07/2000
                                                       Entries    = 00076
  Select one of the following groups and press ENTER.

  GROUP NUMBER ===> _

                 1.   Total number of free space elements

                 2.   Total bytes of free space

                 3.   Percent of free space in data set

  ENTRIES TO BE SELECTED

          From Date = 10 / 01 / 1999    May specify the desired date
          To Date   = 10 / 07 / 2000    Input format is "MM/DD/YYYY"
```

Figure 172. Historical Analysis by DB Free Space

Select a group number (1 to 3) and press **ENTER** to pass the control to ICU, or enter **END** to return to the “Graph Selection Menu” panel.

Group number 1:

- Total number of free space elements.

Group number 2:

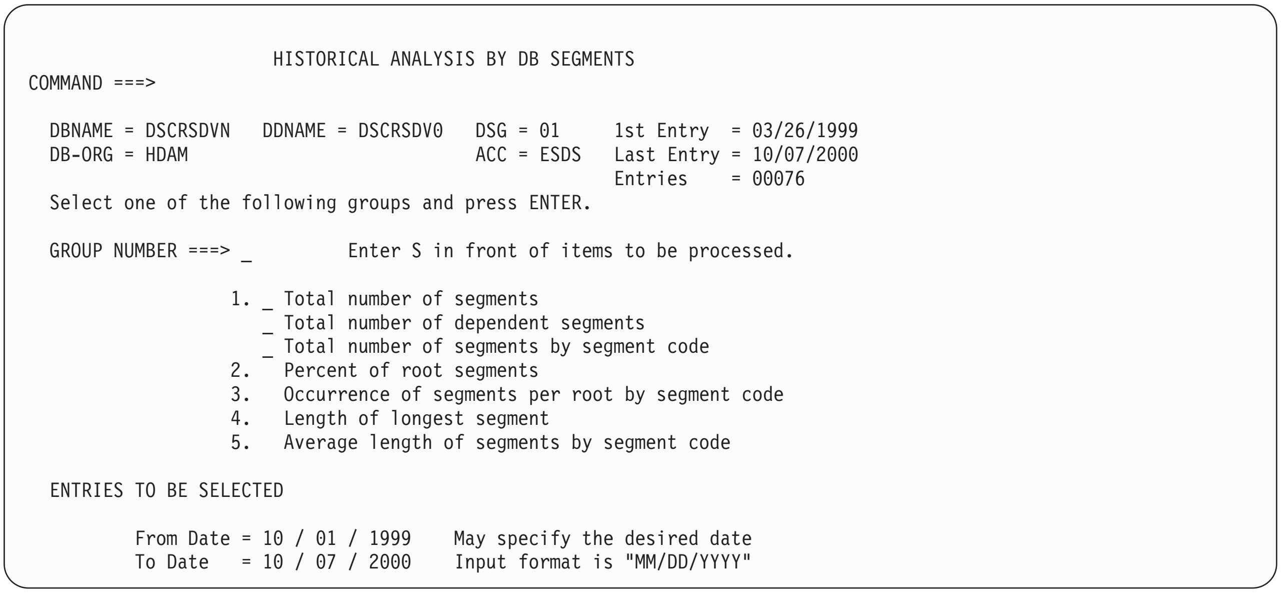
- Total bytes of free space.

Group number 3:

- The percentage of free space within the data set.

Historical Analysis by DB Segments

On the panel that is shown in Figure 173, you can select detailed items of database segment analysis information. DB Historical Data Analyzer will show the historical trend of the selected items for the database data set group identified by the DBNAME and DDNAME fields.



```
                       HISTORICAL ANALYSIS BY DB SEGMENTS
COMMAND ===>

  DBNAME = DSCRSDVN   DDNAME = DSCRSDV0   DSG = 01     1st Entry  = 03/26/1999
  DB-ORG = HDAM                           ACC = ESDS   Last Entry = 10/07/2000
                                                       Entries    = 00076
  Select one of the following groups and press ENTER.

  GROUP NUMBER ===> _         Enter S in front of items to be processed.

                  1. _ Total number of segments
                     _ Total number of dependent segments
                     _ Total number of segments by segment code
                  2.   Percent of root segments
                  3.   Occurrence of segments per root by segment code
                  4.   Length of longest segment
                  5.   Average length of segments by segment code

  ENTRIES TO BE SELECTED

          From Date = 10 / 01 / 1999    May specify the desired date
          To Date   = 10 / 07 / 2000    Input format is "MM/DD/YYYY"
```

Figure 173. Historical Analysis by DB Segments

Select a group number (1 to 5), and select one or more items under the selected group, if necessary. Press **ENTER** to pass the control to ICU, or enter **END** to return to the “Graph Selection Menu” panel.

The items shown on this panel are:

Group number 1:

- Total number of segments
- Total number of dependent segments
- Total number of segments by segment code.

These items are applicable to HISAM, HDAM, HIDAM, PHDAM, and PHIDAM DSG A-J.

Group number 2:

- Percentage of root segments within the total number of segments.

  This item applies to the primary data set groups of HISAM, HDAM, HIDAM, PHDAM, and PHIDAM databases.

Group number 3:

- Occurrence of segments per root by segment code.

  This item applies to the primary data set groups of HISAM, HDAM, HIDAM, PHDAM, and PHIDAM databases.

Group number 4:

- Length of the longest segment.

  This item applies to the primary data set groups of HDAM, HIDAM, PHDAM, and PHIDAM databases.

Group number 5:

- Average length of segments by segment code.

  These items are applicable to HISAM, HDAM, HIDAM, PHDAM, and PHIDAM DSG A-J.

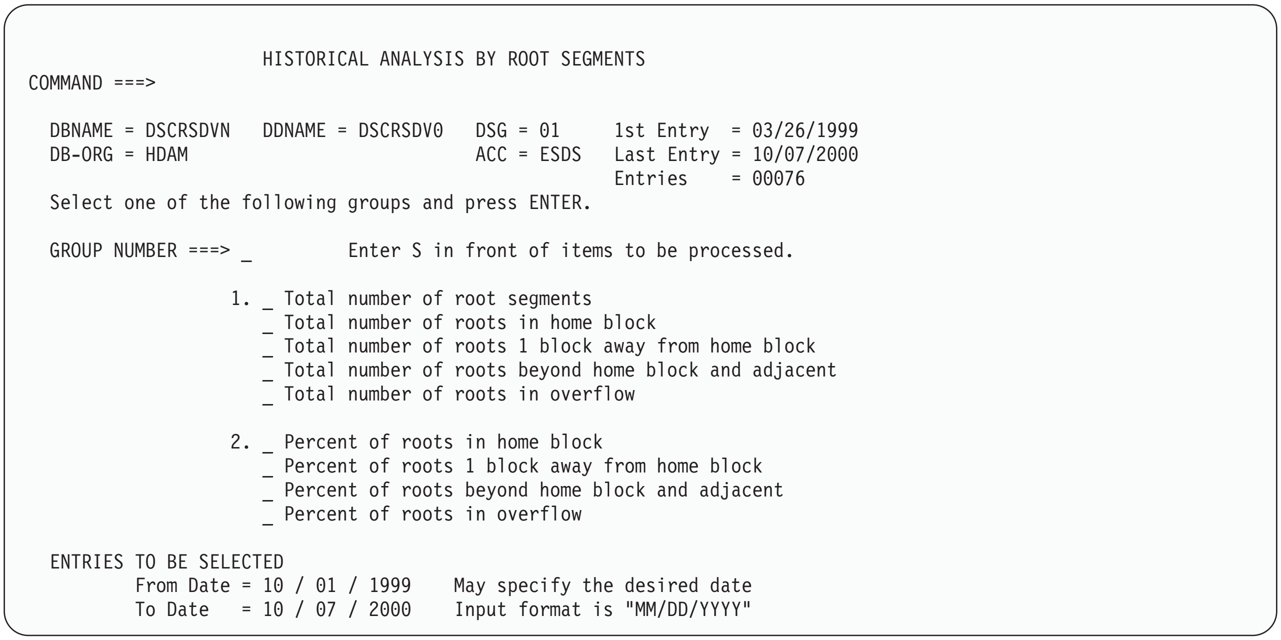
Note

If you select items related to segment code (for example, *total number of segments by segment code*), you may get a busy chart if the database data set group has many segment codes. In such a case, you should exclude an appropriate number of segment codes on the ICU panel 2.3 to make the chart more simple and clear.

Historical Analysis by Root Segments

On the panel that is shown in Figure 174, you can select detailed items of root segment analysis information. DB Historical Data Analyzer will show the historical trend of the selected items for the database data set group identified by the DBNAME and DDNAME fields.

This panel applies only to the primary data set groups of HDAM and PHDAM databases.



```
                      HISTORICAL ANALYSIS BY ROOT SEGMENTS
COMMAND ===>

  DBNAME = DSCRSDVN   DDNAME = DSCRSDV0   DSG = 01     1st Entry  = 03/26/1999
  DB-ORG = HDAM                           ACC = ESDS   Last Entry = 10/07/2000
                                                       Entries    = 00076
  Select one of the following groups and press ENTER.

  GROUP NUMBER ===> _         Enter S in front of items to be processed.

                   1. _ Total number of root segments
                      _ Total number of roots in home block
                      _ Total number of roots 1 block away from home block
                      _ Total number of roots beyond home block and adjacent
                      _ Total number of roots in overflow

                   2. _ Percent of roots in home block
                      _ Percent of roots 1 block away from home block
                      _ Percent of roots beyond home block and adjacent
                      _ Percent of roots in overflow

  ENTRIES TO BE SELECTED
          From Date = 10 / 01 / 1999    May specify the desired date
          To Date   = 10 / 07 / 2000    Input format is "MM/DD/YYYY"
```

Figure 174. Historical Analysis by Root Segments

Select a group number (1 or 2), and select one or more items under the selected group. Press **ENTER** to pass the control to ICU, or enter **END** to return to the “Graph Selection Menu” panel.

The items shown on this panel are:

Group number 1:

- Total number of root segments
- Total number of root segments that are stored in the same blocks as they are assigned by the randomizing routine
- Total number of root segments that are stored in the blocks that immediately precede or follow the blocks to which they are randomized
- Total number of root segments that are stored in the blocks that are neither adjacent to nor the same as the blocks to which they are randomized
- Total number of root segments that are stored in the overflow area (blocks that are not in the root addressable area of the database).

Group number 2:

- Percentage of root segments that are stored in the same blocks as they are assigned by the randomizing routine, within the total number of root segments
- Percentage of root segments that are stored in the blocks that immediately precede or follow the blocks to which they are randomized, within the total number of root segments
- Percentage of root segments that are stored in the blocks that are neither adjacent to nor the same as the blocks to which they are randomized, within the total number of root segments
- Percentage of root segments that are stored in the overflow area (blocks that are not in the root addressable area of the database), within the total number of root segments.

When the data set group is processed by IMS Image Copy Extensions (IMS ICE), the following values in this panel are shown as zero, because the HASH check function invoked by IMS Image Copy Extensions forces HOMECHK=NO:

Group number 1:

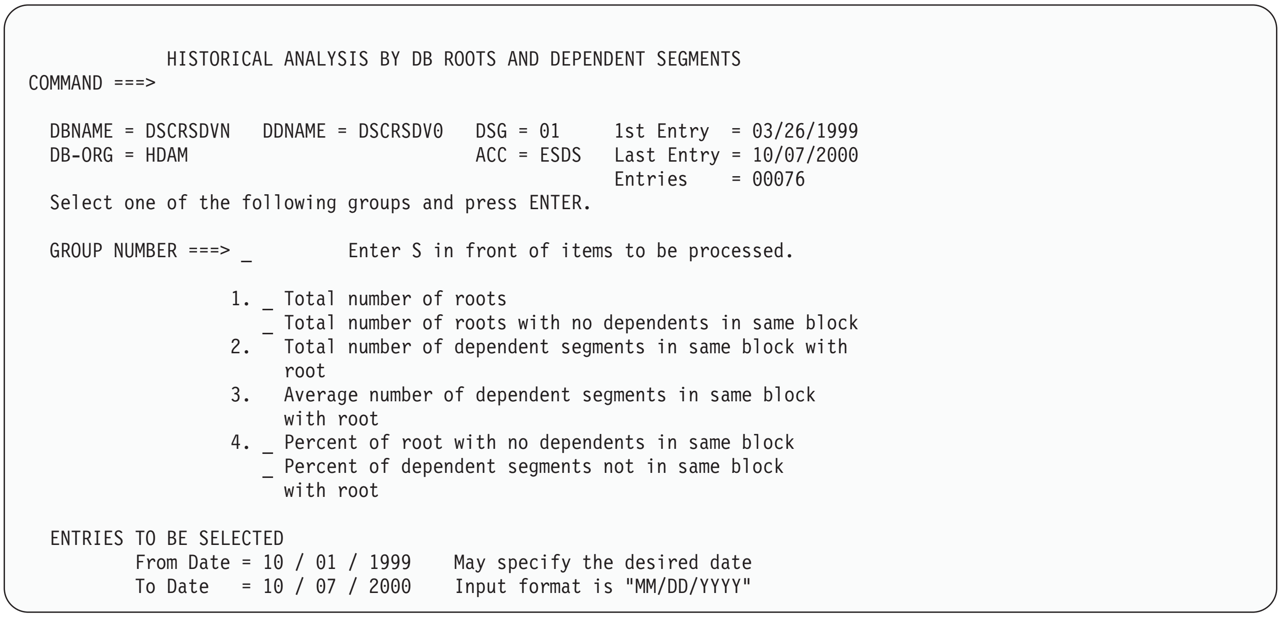
- Total number of roots in home block
- Total number of roots 1 block away from home block
- Total number of roots beyond home block and adjacent
- Total number of roots in overflow

Group number 2:

- Percent of roots in home block
- Percent of roots 1 block away from home block
- Percent of roots beyond home block and adjacent
- Percent of roots in overflow

Historical Analysis by DB Roots and Dependent Segments

On the panel that is shown in Figure 175, you can select detailed analysis items of root segments and their dependent segments. DB Historical Data Analyzer will show the historical trend of the selected items for the database data set group identified by the DBNAME and DDNAME fields.



```
             HISTORICAL ANALYSIS BY DB ROOTS AND DEPENDENT SEGMENTS
COMMAND ===>

  DBNAME = DSCRSDVN   DDNAME = DSCRSDV0   DSG = 01     1st Entry  = 03/26/1999
  DB-ORG = HDAM                           ACC = ESDS   Last Entry = 10/07/2000
                                                       Entries    = 00076
  Select one of the following groups and press ENTER.

  GROUP NUMBER ===> _         Enter S in front of items to be processed.

                   1. _ Total number of roots
                      _ Total number of roots with no dependents in same block
                   2.   Total number of dependent segments in same block with
                        root
                   3.   Average number of dependent segments in same block
                        with root
                   4. _ Percent of root with no dependents in same block
                      _ Percent of dependent segments not in same block
                        with root

  ENTRIES TO BE SELECTED
          From Date = 10 / 01 / 1999    May specify the desired date
          To Date   = 10 / 07 / 2000    Input format is "MM/DD/YYYY"
```

Figure 175. Historical Analysis by DB Roots and Dependent Segments

Select a group number (1 to 4), and select one or more items under the selected group, if necessary. Press **ENTER** to pass the control to ICU, or enter **END** to return to the “Graph Selection Menu” panel.

The items shown on this panel are:

Group number 1:

- Total number of root segments
- Total number of root segments that have no dependent segments in the same block as their root segments.

Group number 2:

- Total number of dependent segments that are in the same block as their root segments.

Group number 3:

- Average number of dependent segments that are in the same block as their root segments.

Group number 4:

- Percentage of root segments that have no dependent segments in the same block as their root segments, within the total number of root segments that have dependent segments.
- Percentage of dependent segments that are not in the same block as their root segments, within the total number of dependent segments.

When the data set group is processed by the HASH check function, the following values in this panel are shown as zero, because the HASH check function forces INCORE=NO:

Group number 1:

- Total number of roots with no dependents in same block

Group number 2:

- Total number of dependent segments in same block with root

Group number 3:

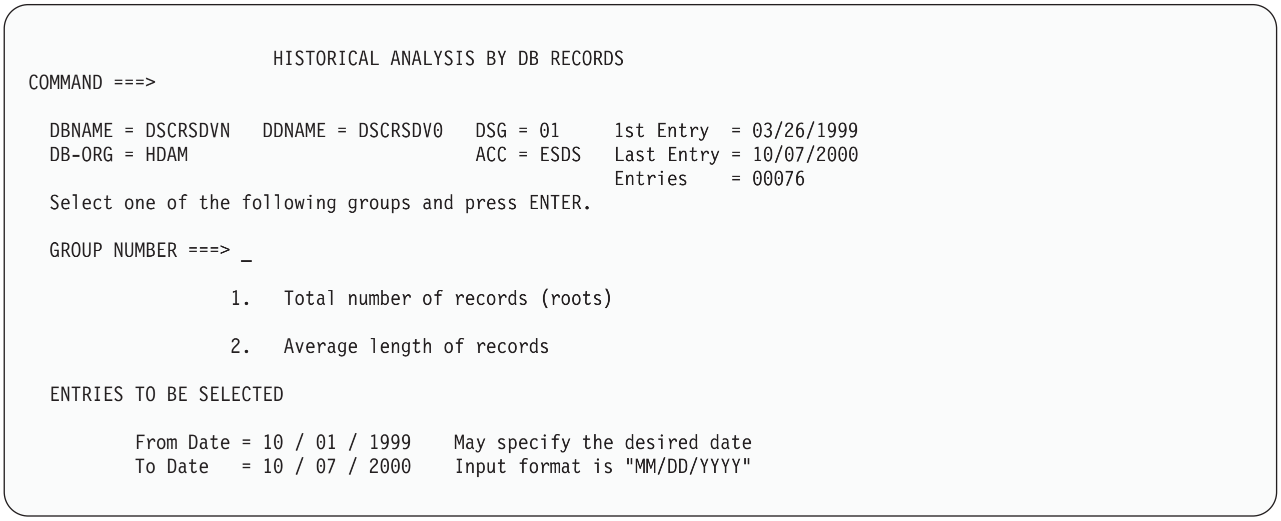
- Average number of dependent segments in same block with root

Group number 4:

- Percent of root with no dependents in same block
- Percent of dependent segments not in same block with root

Historical Analysis by DB Records

On the panel that is shown in Figure 176, you can select detailed items of database record analysis information. DB Historical Data Analyzer will show the historical trend of the selected items for the database data set group identified by the DBNAME and DDNAME fields.



```
                       HISTORICAL ANALYSIS BY DB RECORDS
COMMAND ===>

  DBNAME = DSCRSDVN   DDNAME = DSCRSDV0   DSG = 01     1st Entry  = 03/26/1999
  DB-ORG = HDAM                           ACC = ESDS   Last Entry = 10/07/2000
                                                       Entries    = 00076
  Select one of the following groups and press ENTER.

  GROUP NUMBER ===> _

                   1.   Total number of records (roots)

                   2.   Average length of records

  ENTRIES TO BE SELECTED

          From Date = 10 / 01 / 1999    May specify the desired date
          To Date   = 10 / 07 / 2000    Input format is "MM/DD/YYYY"
```

Figure 176. Historical Analysis by DB Records

Select a group number (1 or 2), then press **ENTER** to pass the control to ICU, or enter **END** to return to the “Graph Selection Menu” panel.

The items shown on this panel are:

Group number 1:

- Total number of database records (this is the same as the total number of root segments).

Group number 2:

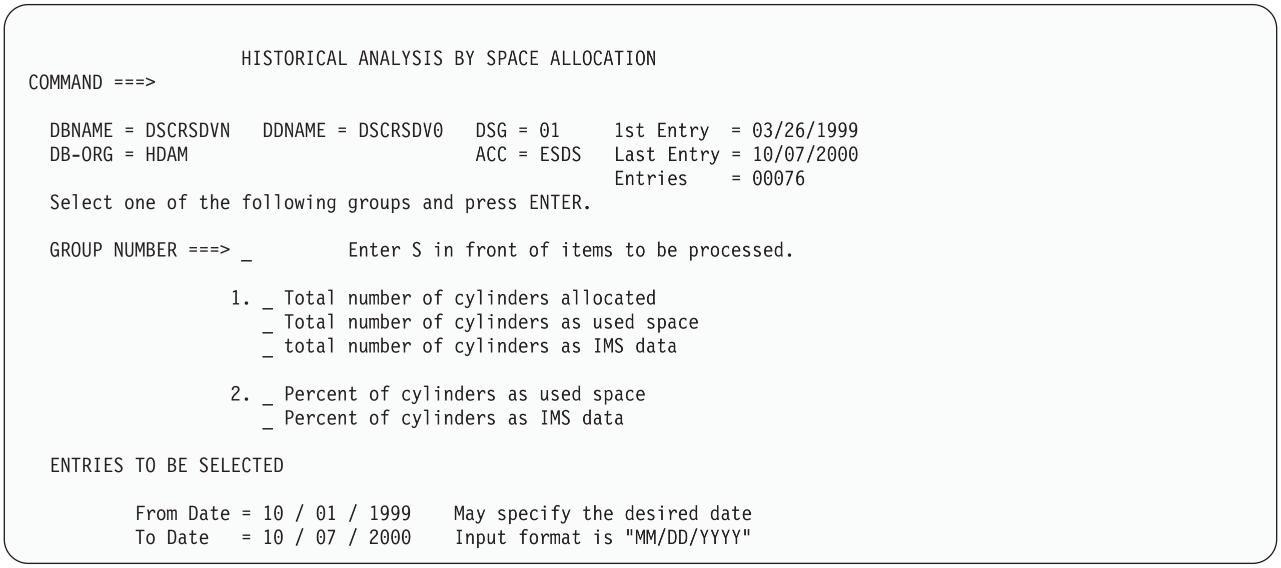
- Average length of database records, including prefix and data portions of all segments. (This does not include segments that are in the secondary data set group of the database.)

This item applies to the primary data set groups of HDAM, HIDAM, PHDAM, and PHIDAM databases.

Historical Analysis by Space Allocation

On the panel that is shown in Figure 177, you can select detailed items of space allocation analysis information. DB Historical Data Analyzer will show the historical trend of the selected items for the database data set identified by the DBNAME and DDNAME fields.

Space management information is obtained from the Space Monitor graph record data set (SPMNSPDT), which is created and maintained by Space Monitor.



```
                    HISTORICAL ANALYSIS BY SPACE ALLOCATION
COMMAND ===>

  DBNAME = DSCRSDVN   DDNAME = DSCRSDV0   DSG = 01     1st Entry  = 03/26/1999
  DB-ORG = HDAM                           ACC = ESDS   Last Entry = 10/07/2000
                                                       Entries    = 00076
  Select one of the following groups and press ENTER.

  GROUP NUMBER ===> _         Enter S in front of items to be processed.

                   1. _ Total number of cylinders allocated
                      _ Total number of cylinders as used space
                      _ total number of cylinders as IMS data

                   2. _ Percent of cylinders as used space
                      _ Percent of cylinders as IMS data

  ENTRIES TO BE SELECTED

          From Date = 10 / 01 / 1999    May specify the desired date
          To Date   = 10 / 07 / 2000    Input format is "MM/DD/YYYY"
```

Figure 177. Historical Analysis by Space Allocation

Select a group number (1 or 2), and select one or more items under the selected group. Press **ENTER** to pass the control to ICU, or enter **END** to return to the “Graph Selection Menu” panel.

The items shown on this panel are:

Group number 1:

- The space allocated for the database data set.
- The space used for the database data set.
- The space used as IMS data in the database data set.

Note: Since the unit of space is shown by cylinders, if the data set is allocated by tracks, the amount of space is rounded off to cylinders.

Group number 2:

- Percentage of used space within the total space allocated for the database data set
- Percentage of space used as IMS data in the database data set, within the total space allocated for the database data set.

Output (HD Analysis Graph)

This section describes the HD Analysis Graph.

As the result of a dialog with DB Historical Data Analyzer through panels, DB Historical Data Analyzer generates a line graph at first by passing control to GDDM Interactive Chart Utility (ICU). Then the user can generate other types of charts through ICU operation using the same data that DB Historical Data Analyzer provides. The user can also customize the chart using ICU—for example, change the title, color, or the size of the chart.

The X axis of the chart always shows intervals by key dates; when the default values of date fields are used, key dates of the last one year or from the date of the first entry to the date of the last entry if the oldest data was created within a year for the specified database data set group. The unit of the Y axis is “TOTAL NUMBER”, “PERCENT” or others depending on the items selected.

For the detailed description, refer to “Customizing a graph chart through ICU” on page 472.

Figure 178 shows the sample output of a line graph. This sample assumes that the user selected Group 1 on the “Historical Analysis by DB Blocks” panel, and selected *Total number of blocks* and *Total number of blocks with no reusable free space* as the detailed database analysis items.

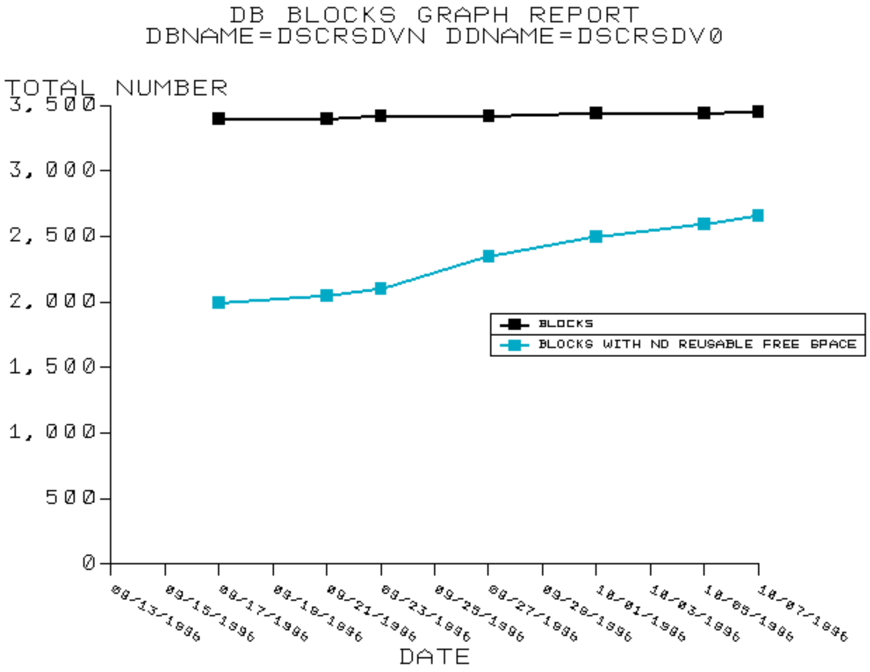


Figure 178. Sample output of HD Analysis graph

When you exit from ICU, you return to one of the “Historical Analysis Menu” panels (panel 6.x in Figure 163 on page 451) that you last made a dialog with.

Customizing a graph chart through ICU

After you select detailed database analysis items on one of the “Historical Analysis Menu” panels (panel 6.x in Figure 163 on page 451), DB Historical Data Analyzer passes the collected data and control to GDDM-ICU to generate a line graph chart.

Once the line graph chart is displayed, you can modify the graph to make it more suitable for your use through a dialog with GDDM-ICU.

Following sections present the graph data that is passed from DB Historical Data Analyzer to GDDM-ICU:

Chart heading (ICU Panel 5.1)

Heading text contains characters for two heading lines. The first line contains the major database analysis item name which was selected on the “Graph Selection Menu” panel. The second line contains the database name (DBNAME=) and ddname (DDNAME=), which were selected on one of the “Data Set Selection Menu” panels (panel 4.x in Figure 163 on page 451).

Data group names (ICU Panel 2.3)

The detailed database analysis item names selected on one of the “Historical Analysis Menu” panels (panel 6.x in Figure 163 on page 451) appear as a legend of the chart.

DB Historical Data Analyzer provides these item names as *long data group names* for the legend.

Attention

If you select items related to segment code (for example, *total number of segment by segment code*) on “Historical Analysis by DB Segments” panel, you may get a busy chart if the database data set group has many segment codes. Legend may contain too many items, and the ICU message

ADM0511 W NOT ENOUGH ROOM TO DRAW ALL KEYS.
ONE OR MORE WERE OMITTED

may be displayed as well.

In such a case, you should exclude an appropriate number of segment codes on the ICU panel 2.3 to make the chart more simple and clear.

Data manipulation (ICU Panel 2.2)

All the collected data for the detailed database analysis items are shown on the ICU panel 2.2 as Y values. If the collected data value is greater than 16,777,210, the value is expressed by a floating point number (such as 1.67772E+07).

Each X element represents the data collection date. The X value associated with each X element is calculated by DB Historical Data Analyzer from the data collection date, in order to draw a correct chart.

The first X element represents the oldest data collection date and its X value is the smallest. The last X element represents the latest data collection date and has the largest X value, which is same as the X axis range value.

X axis range is from zero to a multiple of 12, and one range value represents one day. (See X axis scale and range (ICU Panel 4.3)).

Example

In Figure 178 on page 471, the data collection period is from 09/17/1996 to 10/07/1996 (there are 21 days within the period). So the X axis range is determined as 0 to 24, because 24 is the value that is greater than 21 and a multiple of 12. X value 24 represents the latest data collection date; 10/07/1996. Consequently, X value 0 is determined to represent 09/13/1996, which is 24 days before 10/07/1996.

You can exclude or delete data of some dates in order to make the graph chart look neat. You should not, however, exclude or delete data of the *latest* collection date; otherwise, X axis label does not show the correct date position on the graph chart.

X labels represent actual data collection dates. These X labels are not used for X axis labels due to the difficulty to adjust X axis tick marks and X axis labels. For the description of the X axis labels, refer to X axis label values (ICU Panel 4.4.1).

X axis title (ICU Panel 4.1)

DB Historical Data Analyzer uses the characters **DATE** for the X axis title text.

X axis scale and range (ICU Panel 4.3)

DB Historical Data Analyzer provides Axis Scale Type = 1 (Linear).

X axis range is determined based on the period between the oldest data collection date and the latest data collection date. The range is adjusted to a multiple of 12 so that ***13 tick marks*** (there are 12 intervals in the range) can always be set on the X axis.

If the oldest data collection date is more than one year before the latest data collection date, the maximum period is adjusted to one year (actually 12 X 31 = 372 days) and the data that was collected more than one year before will be ignored when the default option is used.

X axis label values (ICU Panel 4.4.1)

DB Historical Data Analyzer provides X axis labels as your own labels. For every X axis tick mark, the corresponding data collection date is assigned as a label. The X axis labels are displayed in MM/DD/YYYY format.

For example, if data collection period is from 09/17/1996 to 10/07/1996, then X axis range is from 0 to 24 and X axis interval for 13 tick marks is 2. These 13 tick marks are set on X axis positions 0, 2, ..., 24, and the corresponding X axis labels that DB Historical Data Analyzer provides are 09/13/1996, 09/15/1996, ..., 10/05/1996, 10/07/1996. You can change these label texts but *should not* delete any of them, because each X axis label corresponds to each X axis tick mark.

X axis attribute (ICU Panel 4.4.2)

X axis character mode is set to 3 (scaled, exact positioning), X axis character width multiplier is set to 0.7, and X axis rotation angle is set to -30 degrees, in order to display 13 X axis labels at all times. These value should not be changed.

X axis markings (ICU Panel 4.5)

DB Historical Data Analyzer determines the X axis interval between major tick marks in order to show maximum 13 X axis major tick marks (X axis start position and end position are also treated as tick marks) on the graph chart.

Y axis title (ICU Panel 4.1)

One of the following Y axis titles are provided, depending on the selected detailed database analysis items group:

- TOTAL NUMBER
- TOTAL BYTES
- PERCENT
- LENGTH
- AVERAGE LENGTH
- AVERAGE NUMBER
- OCCURRENCE/ROOT

Y axis scale and range (ICU Panel 4.3)

DB Historical Data Analyzer provides SCALE = 1 for linear axes. Y axis range is determined by ICU based on the actual Y values.

Y axis label values (ICU Panel 4.4)

Label Type = 1 (multiplied by 1) is used so that Y axis labels generated are numeric values.

Y axis markings (ICU Panel 4.5)

Default values are applied to determine the interval between major tick marks for Y axis.

Legend position and format (ICU Panel 5.3)

The ICU default settings are used for the legend position and format. You can change these setting options in ICU panel 5.3 if you want to move the position of, or change the size of the legend. You can even suppress the display of the legend.

Chapter 23. JCL examples for DB Historical Data Analyzer

This chapter shows JCL examples used in DB Historical Data Analyzer.

Topics:

Example 1: Reorganizing a HISTORY data set

Figure 179 and Figure 180 on page 476 show sample JCLs for reorganizing the HISTORY data set. In this example, a backup copy of the HISTORY data set is created in Step 1, and then the data set is reorganized in Step 2. The HISTORY data set entries are listed out before and after the reorganization to make sure that the contents of the data set are not changed logically.

DISP=OLD must be used for the HISTORY DD statement.

Step 1: Making a backup copy of HISTORY data set

This section provides a sample JCL for making a backup copy of the HISTORY data set.

```
//*********************************************
//** DEFINE BACKUP HISTORY DATA SET          **
//*********************************************
//DEFINE   EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=A
//SYSIN    DD  *
  DELETE (HIST.HISTORY.BACKUP) CLUSTER
  SET MAXCC = 0
  DEFINE CLUSTER (NAME(HIST.HISTORY.BACKUP) -
                  INDEXED KEYS(23,0) UNIQUE SHAREOPTIONS(3 3) -
                  VOL(IMSDBT) CYLINDERS(2,1) -
                  RECORDSIZE(240,240) -
                  CONTROLINTERVALSIZE(4096)) -
         DATA    (NAME(HIST.HISTORYD.BACKUP)) -
         INDEX   (NAME(HIST.HISTORYI.BACKUP))
/*
//COPY     EXEC PGM=IDCAMS,COND=(0,LT)
//SYSPRINT DD  SYSOUT=A
//INFILE   DD  DISP=OLD,DSN=HIST.HISTORY
//OUTFILE  DD  DISP=OLD,DSN=HIST.HISTORY.BACKUP
//SYSIN    DD  *
  REPRO  INFILE(INFILE)  OUTFILE(OUTFILE)
/*
```

Figure 179. Example 1: Reorganizing a HISTORY data set (Step 1)

Step 2: Reorganizing a HISTORY data set

This section provides a sample JCL for reorganizing the HISTORY data set.

```
//*********************************************
//** REORGANIZE HISTORY DATA SET             **
//*********************************************
//REORG    EXEC  PGM=FABGHIST,COND=(0,LT)
//STEPLIB  DD DSN=HPS.SHPSLMD0,DISP=SHR
//HISTORY  DD DSN=HIST.HISTORY,DISP=OLD
//HISTPRT  DD SYSOUT=A
//HISTMSG  DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//HISTIN   DD  *
 PROC TYPE=LIST      ** LIST CONTENTS BEFORE REORG
 ENDPROC
 PROC TYPE=REORG     ** REORGANIZE HISTORY DS
 ENDPROC
 PROC TYPE=LIST      ** LIST CONTENTS AFTER REORG
 ENDPROC
/*
```

Figure 180. Example 1: Reorganizing a HISTORY data set (Step 2)

Example 2: Deleting entries from a HISTORY data set

Figure 181 shows a sample JCL for deleting entries from a the HISTORY data set. In this example, entries in the HISTORY data set are listed out before and after entries are deleted, to make sure that the specified entries are deleted.

The sample JCL specifies the deletion of:

- All entries of the database name DB01 from January 1 in 2000 to the current date
- Each entry of the ddname DB02DD01 and DB02DD02 of database DB02 which was created most recently
- All entries of the ddname DB03DD01 of database name DB03 created in March, 2000
- Entries of the ddname DB04DD01 of database name DB04 that are created more than 100 days ago

DISP=OLD must be used for the HISTORY DD statement.

```
//*********************************************
//** DELETING HISTORY DATA SET ENTRIES       **
//*********************************************
//DELETE   EXEC  PGM=FABGHIST
//STEPLIB  DD DSN=HPS.SHPSLMD0,DISP=SHR
//HISTORY  DD DSN=HIST.HISTORY,DISP=OLD
//HISTPRT  DD SYSOUT=A
//HISTMSG  DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//HISTIN   DD  *
 PROC TYPE=LIST      ** LIST CONTENTS BEFORE DELETION
 ENDPROC
 PROC TYPE=DELETE    ** DELETE HISTORY DS ENTRIES
   DATABASE DB=DB01,DD=ALL,FROM=01012000
   DATABASE DB=DB02,DD=DB02DD01
   DATABASE DB=DB02,DD=DB02DD02
   DATABASE DB=DB03,DD=DB03DD01,FROM=03012000,TO=03312000
   DATABASE DB=DB04,DD=DB04DD01,KEEP=100
 ENDPROC
 PROC TYPE=LIST      ** LIST CONTENTS AFTER DELETION
 ENDPROC
/*
```

Figure 181. Example 2: Deleting entries from a HISTORY data set

Example 3: Producing an HD Analysis report

Figure 182 on page 479 shows a sample JCL for running HD Pointer Checker and producing the HD Analysis reports in the same job. This method is very useful to reduce the output reports of HD Pointer Checker.

The first step is the HD Pointer Checker run; all the optional output reports are suppressed by the REPORT control statement.

The second step generates the HD Analysis reports for all database data sets that are scanned by HD Pointer Checker. In the DATABASE control statements for DB Historical Data Analyzer, all database data sets that are processed in the first step (by HD Pointer Checker) must be specified.

DISP=SHR must be used for the HISTORY DD statement.

```
//**********************************************************
//**  HD POINTER CHECKER RUN                              **
//**********************************************************
//PCRUN   EXEC FABPP,REGION=4000K,
//             PSB=PSBSMUAL,
//             HISTORY='HIST.HISTORY',
//             DBDLIB='IMS.DBDLIB',
//             PSBLIB='IMS.PSBLIB',
//             RESLIB='IMS.RESLIB',
//             DBTLIB='HPS.SHPSLMD0',
//             DBTSRC='FAB.SAMPLE(FABVSAMP)'
//*            --------------------------------------------------------
//HDPCPRO.PROCCTL  DD *
  PROC TYPE=ALL
   REPORT   RUNTM=NO,DBDIST=NO,BITMAP=NO,FSEMAP=NO,MAXFSD=NO,
            INTST=NO,INTFS=NO
   OPTION   HISTORY=YES
   DATABASE DB=DSCRSDVN,DD=DSCRSDV0,DATASET=IMAGECOPY
   DATABASE DB=DSCRSDVN,DD=DSCRSDV1,DATASET=IMAGECOPY
   DATABASE DB=DSCLSDVN,DD=DSCLSDV0,DATASET=IMAGECOPY
   END
/*
//HDPCPRO.DSCRSDV0 DD DSN=SAMPLE.DSCRSDV0.IMGCOPY,DISP=OLD
//HDPCPRO.DSCRSDV1 DD DSN=SAMPLE.DSCRSDV1.IMGCOPY,DISP=OLD
//HDPCPRO.DSCLSDV0 DD DSN=SAMPLE.DSCLSDV0.IMGCOPY,DISP=OLD
//*
//*********************************************************************
//*   DB HIST                                                         *
//*********************************************************************
//HIST1  EXEC PGM=FABGHIST
//STEPLIB  DD DSN=HPS.SHPSLMD0,DISP=SHR
//HISTORY  DD DSN=HIST.HISTORY,DISP=SHR
//HISTPRT  DD SYSOUT=A
//HISTMSG  DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//HISTIN  DD *
 PROC TYPE=SUMMARY
 ENDPROC
 PROC TYPE=ANALYSIS
   DATABASE DB=DSCRSDVN,DD=DSCRSDV0
   DATABASE DB=DSCRSDVN,DD=DSCRSDV1
   DATABASE DB=DSCLSDVN,DD=DSCLSDV0
 ENDPROC
/*
```

Figure 182. Example 3: Producing an HD Analysis report after HD Pointer Checker run

Example 4: Producing the HD Pointer Checker Summary report

Figure 183 and Figure 184 on page 481 show sample JCL for producing the HD Pointer Checker Summary report for the specified database data sets. In this example, the results of the last HD Pointer Checker run, for all the database data sets specified in the member APPL01, are summarized.

Note: DEDB data set specifications are ignored.

Step 1: Creating a member in the SPMNMBR data set

DB Historical Data Analyzer creates a member (APPL01) in the SPMNMBR data set.

```
//FPPROC  EXEC  PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD SYSOUT=A
//SYSUT2   DD DSN=SPMN.MEMBER,DISP=OLD
//SYSIN   DD  DATA,DLM=@@
./ ADD  NAME=APPL01,LIST=ALL
*        1         2         3         4         5         6         7
*...+....0....+....0....+....0....+....0....+....0....+....0....+....0..
* IMS DL/I DATABASE DATA SET GROUP FOR APPL01
*-DBD--* *-DD---* *-DATA SET NAME ---------------------------* AABBB CC
DSSCHHVN DSSCHHV0 SAMPLE.DSSCHHV0                              13 20  7
DSFACHON DSFACHO0 SAMPLE.DSFACHO0
DSCRSDVN DSCRSDV0 SAMPLE.DSCRSDV0                              13 20  7
DSCRSDVN DSCRSDV1 SAMPLE.DSCRSDV1
DSCLSDVN DSCLSDV0 SAMPLE.DSCLSDV0
*
*        1         2         3         4         5         6         7
*...+....0....+....0....+....0....+....0....+....0....+....0....+....0..
* IMS DEDB AREA DATA SET GROUP FOR APPL01
*-N/A--* *-N/A--* *-DATA SET NAME ---------------------------* AABBB CC
                  SAMPLE.DB01AR01
                  SAMPLE.DB01AR02
                  SAMPLE.DB02AR01
                  SAMPLE.DB02AR02
                  SAMPLE.DB02AR03
@@
//*
```

Figure 183. Example 4: Creating a member in the SPMNMBR data set (Step 1)

Step 2: Producing the HD Pointer Checker Summary report

Using the SPMNMBR data set created in “Step 1: Creating a member in the SPMNMBR data set” on page 480, DB Historical Data Analyzer produces the HD Pointer Checker Summary report as shown in Figure 184.

```
//***********************************************
//** PRODUCE HD POINTER CHECKER SUMMARY REPORT **
//***********************************************
//SUMM     EXEC  PGM=FABGHIST
//STEPLIB  DD DSN=HPS.SHPSLMD0,DISP=SHR
//HISTORY  DD DSN=HIST.HISTORY,DISP=SHR
//SPMNMBR  DD DSN=SPMN.MEMBER,DISP=SHR
//HISTPRT  DD SYSOUT=A
//HISTMSG  DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//HISTIN   DD  *
 PROC TYPE=SUMMARY
   DATABASE MEMBER=APPL01
 ENDPROC
  /*
```

Figure 184. Example 4: Producing the HD Pointer Checker Summary report (Step 2)

Example 5: Creating the predefined flat records

Figure 185 shows a sample JCL of Export Utility. It creates predefined flat records.

Do not specify the FABGRECI DD statement for using the predefined formats. Specify the predefined member name to MEMBER= in the HISTIN data set.

In this example, the format FABPR001 and FABPR002 are used as the predefined member name.

```
//*--------------------------------------------------------------------*
//*  EXPORT UTILITY                                                    *
//*--------------------------------------------------------------------*
//EXPORT   EXEC PGM=FABGXEXP
//STEPLIB  DD DISP=SHR,DSN=hppc.shpslmd0
//HISTORY  DD DISP=SHR,DSN=history.dataset
//FABGEXPF DD DISP=NEW,SPACE=(TRK,(1,1)),UNIT=SYSALLDA,
//         DSN=flat.file
//HISTMSG  DD SYSOUT=*
//HISTPRT  DD SYSOUT=*
//HISTIN   DD *
  PROC TYPE=EXPORT,MEMBER=(FABPR001,FABPR002)
/*
//SYSUDUMP DD SYSOUT=*
```

Figure 185. Example 5: Creating predefined flat records

Example 6: Creating the user-defined flat records

Figure 186 shows a sample JCL of Export Utility. It creates the user-defined flat records. Specify the FABGRECI DD statement for using the user-defined formats, and specify the member name to MEMBER= in the HISTIN data set. In this example, the format member01 and member02 are used.

```
//*-------------------------------------------------------------------*
//*  EXPORT UTILITY                                                   *
//*-------------------------------------------------------------------*
//EXPORT   EXEC PGM=FABGXEXP
//STEPLIB  DD DISP=SHR,DSN=hppc.shpslmd0
//HISTORY  DD DISP=SHR,DSN=history.dataset
//FABGRECI DD DISP=SHR,DSN=fabgreci.dataset
//FABGEXPF DD DISP=NEW,SPACE=(TRK,(1,1)),UNIT=SYSALLDA,
//         DSN=flat.file
//HISTMSG  DD SYSOUT=*
//HISTPRT  DD SYSOUT=*
//HISTIN   DD *
  PROC TYPE=EXPORT,MEMBER=(member01,member02)
/*
//SYSUDUMP DD SYSOUT=*
```

Figure 186. Example 6: Creating the user-defined flat records

Chapter 24. Customizing the DB Historical Data Analyzer

This chapter contains information about how to customize the DB Historical Data Analyzer.

Topics:

- "Modifying logon procedure"
- "Sample FABGCMD0 CLIST"
- "Modifying the ISPF/PDF Primary Option Menu panel" on page 486

Modifying logon procedure

This section describes how to modify the logon procedure.

GDDM program library

The GDDM library must be accessed as a PDS library by the TSO terminal user. Unless the GDDM target library is specified as a link library, you must change the existing, or define a new, TSO logon procedure to contain a STEPLIB DD statement that refers to the GDDM program library (GDDMLOAD).

For more information, see the documentations of GDDM products.

Libraries for panels, messages, and programs

IMS HP Pointer Checker data sets for panels and messages should be concatenated with the corresponding ISPF/PDF data sets. The IMS HP Pointer Checker data set for programs must be allocated in the ISPF link library (ddname ISPLLIB) in your TSO logon procedure. For example, specify as follows:

```
//ISPMLIB   DD DSN=HPS.SHPSMLIB,DISP=SHR
//             DSN=ISP.V4R2M0.SISPMENU,DISP=SHR
//ISPPLIB   DD DSN=HPS.SHPSPLIB,DISP=SHR
//             DSN=ISP.V4R2M0.SISPPENU,DISP=SHR
//ISPLLIB   DD DSN=HPS.SHPSLMD0,DISP=SHR
```

You can also allocate these data sets by coding appropriate TSO ALLOCATE commands. This way, you need not modify your TSO logon procedure. In this case, the allocations must be performed before you invoke ISPF.

If you use ISPPALT and ISPMALT for DBCS, allocate the IMS HP Pointer Checker data set to ISPPALT and ISPMALT.

For more information about ISPF, read *z/OS ISPF User's Guide, Volume 1* or a later version of the guide.

Sample FABGCMD0 CLIST

DB Historical Data Analyzer provides a sample command list (CLIST) FABGCMD0 to allocate data sets and to invoke DB Historical Data Analyzer (see Figure 162 on page 450).

This sample CLIST is distributed with IMS HP Pointer Checker. You may need to copy this sample CLIST to your command procedure data set, and modify it to meet your installation requirements.

Modifying the ISPF/PDF Primary Option Menu panel

You can modify the ISPF Primary Option Menu panel (ISR@PRIM) to add an entry so that you can invoke DB Historical Data Analyzer by a selection code. The following sample ISPF Primary Option Menu (see Figure 187 on page 487) has been modified to invoke DB Historical Data Analyzer by selecting option D. This option starts processing by invoking a command procedure. The FABGCMD0 CLIST that is distributed with IMS HP Pointer Checker can be used to start the dialog processing. You may need to modify the CLIST to meet your installation requirements.

```
%-----------------------  ISPF/PDF PRIMARY OPTION MENU  ------------------------
%OPTION  ===>_ZCMD                                                             +
%                                                           +USERID   - &ZUSER
%   0 +ISPF PARMS  - Specify terminal and user parameters   +TIME     - &ZTIME
%   1 +BROWSE      - Display source data or output listings +TERMINAL - &ZTERM
%   2 +EDIT        - Create or change source data           +PF KEYS  - &ZKEYS
%   3 +UTILITIES   - Perform utility functions
%   4 +FOREGROUND  - Invoke language processors in foreground
%   5 +BATCH       - Submit job for language processing
%   6 +COMMAND     - Enter TSO command or CLIST
%   7 +DIALOG TEST - Perform dialog testing
%   8 +LM UTILITIES- Perform library administrator utility functions
%   9 +IBM PRODUCTS- Additional IBM program development products
%   C +CHANGES     - Display summary of changes for this release
%   D +DBHDA       - Invoke DB Historical Data Analyzer dialog

%   T +TUTORIAL    - Display information about ISPF/PDF
%   X +EXIT        - Terminate ISPF using log and list defaults
%
+Enter%END+command to terminate ISPF.
%
)INIT
  .HELP = ISR00003
  &ZPRIM = YES        /* ALWAYS A PRIMARY OPTION MENU      */
  &ZHTOP = ISR00003   /* TUTORIAL TABLE OF CONTENTS        */
  &ZHINDEX = ISR91000 /* TUTORIAL INDEX - 1ST PAGE         */
  VPUT (ZHTOP,ZHINDEX) PROFILE
)PROC
&ZQ = &Z
  IF (&ZCMD ¬= ' ')
    &ZQ = TRUNC(&ZCMD,'.')
    IF (&ZQ = ' ')
      .MSG = ISRU000
  &ZSEL = TRANS( &ZQ
              0,'PANEL(ISPOPTA)'
              1,'PGM(ISRBRO) PARM(ISRBRO01)'
              2,'PGM(ISREDIT) PARM(P,ISREDM01)'
              3,'PANEL(ISRUTIL)'
              4,'PANEL(ISRFPA)'
              5,'PGM(ISRJB1) PARM(ISRJPA) NOCHECK'
              6,'PGM(ISRPTC)'
              7,'PGM(ISRYXDR) NOCHECK'
              8,'PANEL(ISRLPRIM)'
              9,'PANEL(ISRDIIS)'
              C,'PGM(ISPTUTOR) PARM(ISR00005)'
              D,'CMD(FABGCMD0)'
              T,'PGM(ISPTUTOR) PARM(ISR00000)'
            ' ',' '
              X,'EXIT'
              *,'?' )
  &ZTRAIL = .TRAIL
)END
```

Figure 187. Modifying ISPF/PDF Primary Option Menu panel

Part 5. Space Monitor

Chapter 25. Overview of Space Monitor

This chapter gives an overview of Space Monitor.

Topics:

Program functions

This section describes the program functions of Space Monitor.

Monitoring and logging data set space utilization

Space Monitor examines DASD VTOCs and VSAM catalog entries of user-specified data sets to monitor the space utilization of IMS full-function database data sets and/or OS data sets. For IMS full-function database data sets, Space Monitor also uses database analysis information collected by HD Pointer Checker to monitor the space utilization from the viewpoint of IMS data. IMS DEDB area data sets are treated simply as VSAM data sets. The collected data is logged into a data set as the space management information and is used to produce a graph report, which shows the most recent trend of space utilization. The collected data of the IMS full-function database data sets is also used by DB Historical Data Analyzer.

Describing space utilization

Space Monitor produces the following six types of reports to describe the space analysis data:

- Space Analysis by Data Set report
- Summary of Data Sets by Volume report
- Total DASD Utilization by Volume/Device-Type report
- Legend report
- Space Monitor Graph report
- Space Monitor Exception report.

Typical uses

The frequent (usually daily) use of this utility will prevent system abend conditions due to:

- More space is needed, but none is available.
- More space is needed, but the limit of the extent has been reached.

When this utility detects these conditions, it will notify them by return codes. Optionally, it will send a notification message to TSO user IDs.

Program structure

Space Monitor consists of one load module (FABKSPMN), which runs as a batch job.

Data flow

Figure 188 shows the data flow of this utility.

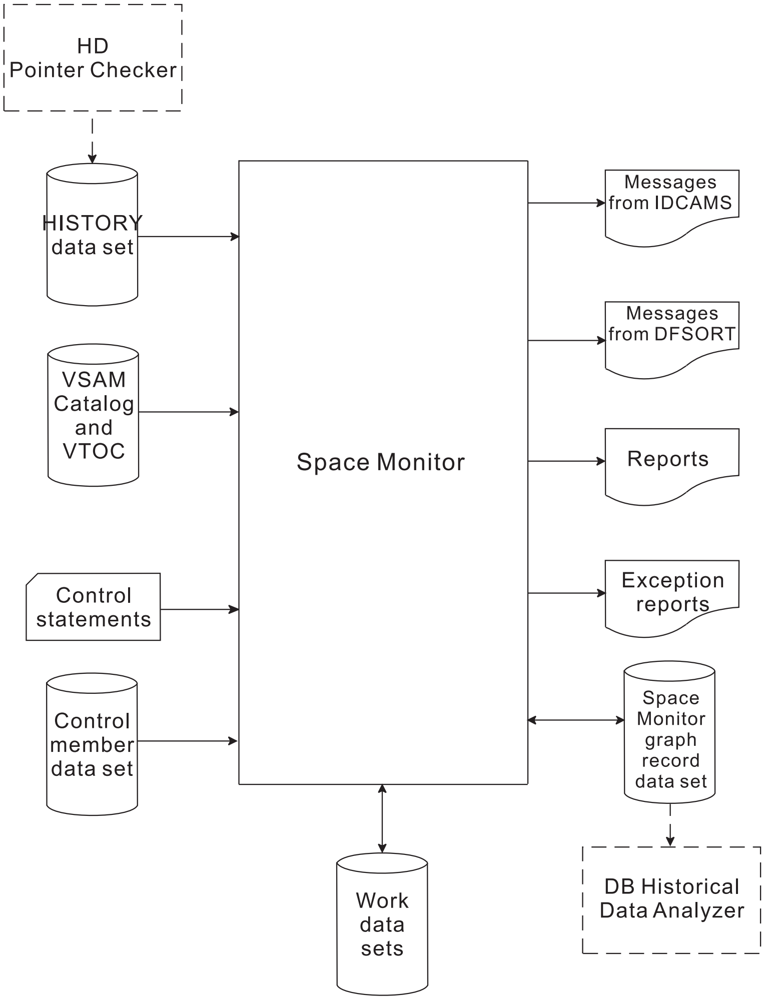


Figure 188. Data flow for Space Monitor

- If you want to obtain IMS-related information for the IMS full-function database data sets, HD Pointer Checker (with HISTORY=YES option) must be run in advance (before running Space Monitor) at least once for the database data sets to create entries in the HISTORY data set. Space Monitor uses the HISTORY data set as the input data set.
- The user must specify the database data sets and/or OS data sets to be monitored. They are specified by the control statements in the SPMNIN data set, or in the SPMNMBR data set. SPMNMBR data set (also called control member data set) is a partitioned data set whose members contain one or more control statements.
- Space Monitor monitors the DASD VTOCs and VSAM catalog entries for the user-specified data sets and logs the space management information in the Space Monitor graph record data set (SPMNSPDT).This data set maintains multiple entries of data for each data set to keep the Space Monitor information of the previous runs.

  This data set is also used by DB Historical Data Analyzer.

- Space Monitor produces various types of reports, such as Space Analysis by Data Set report, Summary of Data Sets by Volume report, Total DASD Utilization by Volume/Device-Type report, Space Monitor Exception report, and Legend report. In addition, Space Monitor uses the SPMNSPDT data set information to produce a Space Monitor Graph report, which is a scatter plot that shows the most recent trend of space utilization of the user-specified data sets.

Restrictions and considerations

Restrictions

- The following data sets are not supported:
  - Data set that resides on a tape
  - Multivolume data set that resides on more than one device type
  - OSAM multivolume data set whose block size or record length is not equal for all volumes
  - Extended sequential data set
- All the data sets you want to monitor must be cataloged, and the DASDs on which the data sets are allocated must be mounted in advance.
- In the case of partitioned data set extended (PDSE), *used space* is always treated as all, even if there is some free space.
- If IMS database data set is reloaded to a new data set whose device type is different from the old data set, space information of the old data set cannot be maintained correctly.
- The same data set cannot be processed more than once in one Space Monitor job step. Therefore, in the case of a shared index database, only one secondary index can be monitored in a job step.
- Space information of the VSAM KSDS index component is not monitored.
- The Indirect List Data Set (ILDS) of HALDB must be monitored as a non-IMS data set; leave the DB name and DD name columns in the Space Monitor control statement blank.

Considerations for using the HISTORY data set

- The following information are obtained from the HISTORY data set.
  - Reorganization date in the SPACE MONITOR EXCEPTION REPORT and SPACE ANALYSIS BY DATA SET REPORT
  - IMS data in the Space Monitor Graph report
- Though the reorganization date and IMS data are optional, HD Pointer Checker should be run before the Space Monitor run if they are to be monitored; in this case, HD Pointer Checker (with HISTORY=YES option) should be run in advance at least once for the database data set to create entries in the HISTORY data set. In order to obtain correct space analysis information, it is recommended that HD Pointer Checker is run regularly so that up-to-date history record entries are always maintained in the HISTORY data set.

  If the HISTORY data set is not specified, or if no entry exists for the database data set in the HISTORY data set, then the data set to be monitored will be treated simply as an OS data set.
- If the reorganization date is to be monitored, register the database to DBRC and run the HD Pointer Checker with DBRC=Y.
- Even if multiple database data set entries are stored per day, Space Monitor does not process the multiple entries. The last data of the day is shown in the reports.

Considerations for HALDB Online Reorganization (OLR)

- Space Monitor can run while a HALDB partition is in cursor-active status.
- For the HALDB which is Online Reorganization (OLR) capable, specifications to the statement in SPMNMBR or SPMNIN data set should be done as follows:
  - DD name: Specify either DD name of DSG ID=A through J, or M through V. Whichever suffix is specified, Space Monitor searches and processes both A and M sides of the history record entries. Therefore, the Reorganization date and IMS data show the side that was active when HD Pointer Checker created the history record entry.
  - Data set name: Specify the data set name of DSG ID=A through J, or M through V. Space Monitor takes space information for the specified data set. However, if the specified data set does not exist, Space Monitor searches the data set of the other side.

Chapter 26. Operating instructions for Space Monitor

This chapter describes how to use Space Monitor. In order to use Space Monitor, follow the steps described below:

1. If IMS database data sets are to be monitored, ensure that HD Pointer Checker (with HISTORY=YES option) is run against the database data sets in advance (at least once) to create entries in the HISTORY data set. Otherwise, the database data sets are treated as OS data sets.
2. Specify the data sets to be monitored in the control member data set (SPMNMBR) or in the SPMNIN data set.
3. Code the JCL. An IBM-supplied cataloged procedure FABKSPMP is provided.
4. Run the Space Monitor job.

Topics:

Job control language

The job control language (JCL) for Space Monitor is explained in this section.

FABKSPMN JCL

Space requirements for the various work data sets created by Space Monitor are summarized in Table 60.

Table 60. Space Monitor data set sizes

DDNAME	Space requirements (bytes)
SPMNCREC	(No. of data sets + No. of extents) × 160
SORTIN	(Same as SPMNCREC)
SPMNCVOL	(No. of volumes × 2 + No. of extents) × 100
SORTWK01 - 03	(Space for SPMNCREC) × 2 / 3

To run Space Monitor, supply an EXEC statement and the appropriate DD statements that define input and output data sets. Table 61 summarizes the DD statements.

Table 61. Space Monitor DD statements

DDNAME	Use	Format	Need
HISTORY	Input	KSDS	Optional
SPMNMBR	Input	PDS	Optional
SPMNIN	Input	LRECL=80	Optional
SPMNSPDT	Input/Output	Fixed Record Length	Required
SPMNCREC	Work data set	LRECL=160	Required
SPMNCVOL	Work data set	LRECL=160	Required
SORTIN	Work data set	LRECL=160	Required
SORTWK01-03	Work data set	LRECL=160	Required
SYSPRINT	Output	SYSOUT	Required

Table 61. Space Monitor DD statements (continued)

DDNAME	Use	Format	Need
SPMNANAL	Output	LRECL=133	Optional
SPMNSUMM	Output	LRECL=133	Optional
SPMNDASD	Output	LRECL=133	Optional
SPMNLGND	Output	LRECL=133	Optional
SPMNGRAF	Output	LRECL=133	Optional
SPMNPRTW	Output	LRECL=133	Optional
SPMNMSG	Output	LRECL=133	Required
SYSUDUMP	Output	SYSOUT	Optional
SPMNPRT	Output	LRECL=133	Optional

EXEC

This statement must be in the following format:

```
//      EXEC PGM=FABKSPMN,PARM=member-name
```

PARM= parameter specifies the member of the SPMNMBR data set. The member contains control statements for the data sets to be monitored by Space Monitor.

If the EXEC PARM= parameter is *not* specified, control statements must be specified in the data set defined by the SPMNIN DD statement.

HISTORY DD

This optional input data set defines the HISTORY data set (VSAM KSDS), which contains summary information of the HD Pointer Checker run.

For detailed information on this data set, refer to Part 4, “DB Historical Data Analyzer,” on page 355.

You must use DISP=SHR for this data set.

SPMNANAL DD

This optional output data set contains the Space Analysis by Data Set report.

SPMNSUMM DD

This optional output data set contains the Summary of Data Sets by Volume report.

SPMNDASD DD

This optional output data set contains the Total DASD Utilization by Volume/Device-Type report.

SPMNLGND DD

This optional output data set contains the Legend report.

SPMNGRAF DD

This optional output data set contains the Space Monitor Graph report.

These five data sets contain 133-byte fixed-length records. These data set can reside on a tape, a direct access device, or a printer; or they 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:

```
//SPMNANAL     DD SYSOUT=A
//SPMNSUMM     DD SYSOUT=A
//SPMNDASD     DD SYSOUT=A
//SPMNLGND     DD SYSOUT=A
//SPMNGRAF     DD SYSOUT=A
```

To suppress any of these reports, delete the corresponding DD statement or specify DUMMY for the DD statement.

SPMNMBR DD
This optional input partitioned data set (PDS) contains the member whose name is specified by the EXEC PARM= parameter. This DD statement is required only when the EXEC PARM= parameter is specified; otherwise, this DD statement is ignored even if it is coded.

SPMNIN DD
This optional input data set contains the control statements. This DD statement is required only when the EXEC PARM= parameter is *not* specified; otherwise, this DD statement is ignored even if it is coded.

SPMNSPDT DD
This required input/output sequential data set is the Space Monitor graph record data set. It contains one Space Monitor record for each data set to be monitored. Each record contains multiple entries of data to keep the Space Monitor information of the previous runs. The number of entries to be created is defined by the logical record length of the data set, which is specified by the user.

This data set should reside on a direct access device, and must have fixed-length records. You should use DISP=(MOD,KEEP,KEEP).

Refer to “Space Monitor Graph Record data set (SPMNSPDT)” on page 508 for more details.

SPMNCREC DD
This required work data set contains data set space allocation information records that are collected while analyzing DASD VTOC and VSAM catalog.

SPMNCVOL DD
This required work data set contains volume information records that are generated while analyzing DASD VTOC and VSAM catalog.

SORTIN DD
This required work data set contains data set space allocation information records that are produced while analyzing DASD VTOC and VSAM catalog. This data set is used as the input to DFSORT (Data Facility Sort), which is called internally by Space Monitor.

SORTWK01/02/03 DD
These are intermediate storage data sets used by DFSORT (Data Facility Sort), which is called internally by Space Monitor. See *DFSORT Application Programming Guide* for more information on how to code SORTWKnn DD statements.

SYSOUT DD
This required output data set contains the messages produced by DFSORT (Data Facility Sort), which is called internally by Space Monitor.

SYSPRINT DD
This required output data set contains the messages produced by Access Method Services (IDCAMS), which is called internally by Space Monitor.

SPMNPRT DD

This optional output data set contains the following reports produced by Space Monitor:

- Space Analysis by Data Set report
- Summary of Data Sets by Volume report
- Total DASD Utilization by Volume/Device-Type report
- Legend report
- Space Monitor Graph report

This data set contains 133-byte fixed-length records. This data set can reside on a tape, a direct access device, or a 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:

```
//SPMNPRT      DD SYSOUT=A
```

If you want to get all the Space Monitor reports, use this data set. If you want to get selective reports, use a combination of the SPMNANAL, SPMNSUMM, SPMNDASD, SPMNLGND, and SPMNGRAF data sets, instead. If you use both of the SPMNPRT data set and some of the others, you will get duplicate reports.

SPMNPRTW DD

This optional output data set contains the Space Monitor Exception report. The data set contains 133-byte fixed-length records. It can reside on a tape, direct access device, or printer, or 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:

```
//SPMNPRTW     DD SYSOUT=A
```

SPMNMSG DD

This required output data set contains messages issued by Space Monitor. The data set contains 133-byte fixed-length records. This data set can reside on a tape, direct access device, or printer, or 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:

```
//SPMNMSG     DD SYSOUT=A
```

SYSUDUMP DD

This optional output data set defines an output from a system abend dump routine. It is used only when a dump is required. Although optional, it is recommended that you include this data set.

The data sets specified by DD statements in Table 62 on page 499 are allocated dynamically, if they are not specified on the JCL statement.

The UNIT and SPACE parameter values are described in Table 62 on page 499. If space is not available in the specified UNIT, code the DD statements in the JCL.

If DUMMY is specified on the DD statement, the report is not generated. Do not specify DUMMY on the DD statement, unless you want to suppress generation of the report.

Table 62. DDs for which dynamic allocation is available

DDNAME	Dynamic allocation parameters
SPMNCREC	UNIT=SYSALLDA,SPACE=(CYL,(10,1))
SPMNCVOL	UNIT=SYSALLDA,SPACE=(CYL,(10,1))
SORTIN	UNIT=SYSALLDA,SPACE=(CYL,(10,1))
SORTWK01-03	UNIT=SYSALLDA,SPACE=(CYL,(10,1)) for each
SPMNPRT	SYSOUT=*
SPMNANAL	SYSOUT=*
SPMNSUMM	SYSOUT=*
SPMNDASD	SYSOUT=*
SPMNLGND	SYSOUT=*
SPMNGRAF	SYSOUT=*
SPMNPRTW	SYSOUT=*
SPMNMSG	SYSOUT=*
SYSOUT	SYSOUT=*

JCL procedure

To run Space Monitor, use the IBM-supplied cataloged procedure shown in Figure 189 on page 500, or prepare a similar procedure of your own. Chapter 27, “JCL examples for Space Monitor,” on page 527 assumes that this IBM-supplied cataloged procedure is used.

```
//********************************************************************  00010000
//*   Licensed Materials - Property of IBM                            *  00020000
//*                                                                   *  00030000
//*   5655-K53                                                        *  00040000
//*                                                                   *  00050000
//*   (c) Copyright IBM Corporation 2000, 2006. All rights reserved. *  00060000
//*                                                                   *  00070000
//*   US Government Users Restricted Rights - Use,                    *  00080000
//*   duplication or disclosure restricted by GSA ADP                 *  00090000
//*   Schedule Contract with IBM Corp.                                *  00100000
//*                                                                   *  00110000
//********************************************************************  00120000
//         PROC MEMBER=,                       FOR MEMBER NAME          00130000
//             STEPLIB='HPS.SHPSLMD0',         FOR STEPLIB DS           00140000
//             HISTORY='HPS.HIST.HISTORY',     FOR HISTORY DS           00150000
//             SPMNSPD='HPS.SPMN.SPDT',        FOR SPMNSPDT DS          00160000
//             SPMNMBR='NULLFILE'              FOR SPMNMBR DS           00170000
//S        EXEC PGM=FABKSPMN,PARM='&MEMBER'                             00180000
//STEPLIB  DD  DSN=&STEPLIB,DISP=SHR                                    00190000
//HISTORY  DD  DSN=&HISTORY,DISP=SHR                                    00200000
//SPMNSPDT DD  DSN=&SPMNSPD,DISP=(MOD,KEEP,KEEP)                        00210000
//SPMNMBR  DD  DSN=&SPMNMBR,DISP=SHR                                    00220000
//SPMNCREC DD  UNIT=SYSDA,SPACE=(CYL,(10,1)),                           00230000
//             DCB=(RECFM=FBA,LRECL=160,BLKSIZE=12800)                  00240000
//SPMNCVOL DD  UNIT=SYSDA,SPACE=(CYL,(10,1)),                           00250000
//             DCB=(RECFM=FB,LRECL=100,BLKSIZE=13000)                   00260000
//SORTIN   DD  UNIT=SYSDA,SPACE=(CYL,(10,1)),                           00270000
//             DCB=(RECFM=FB,LRECL=160,BLKSIZE=12800)                   00280000
//SORTWK01 DD  UNIT=SYSDA,SPACE=(CYL,(10,1))                            00290000
//SORTWK02 DD  UNIT=SYSDA,SPACE=(CYL,(10,1))                            00300000
//SORTWK03 DD  UNIT=SYSDA,SPACE=(CYL,(10,1))                            00310000
//SYSOUT   DD  SYSOUT=A                                                 00320000
//SYSPRINT DD  SYSOUT=A                                                 00330000
//SPMNPRT  DD  SYSOUT=A                                                 00340000
//SPMNPRTW DD  SYSOUT=A                                                 00350000
//SPMNMSG  DD  SYSOUT=A                                                 00360000
//*-------------------------------------------------------------        00370000
```

Figure 189. Space Monitor JCL procedure (FABKSPMP)

Parameters in the JCL procedure are described below:

MEMBER=
Specifies the member name of the SPMNMBR data set that contains control statements. This parameter is optional. If this parameter is omitted or no member name is specified, control statements must be specified in the data set defined by the SPMNIN DD statement.

STEPLIB=
Specifies the name of the IMS HP Pointer Checker load module library. This parameter is optional. The default is 'HPS.SHPSLMD0'.

HISTORY=
Specifies the name of the HISTORY data set. This parameter is optional. If 'NULLFILE' is specified, the data sets to be monitored are treated as OS data sets. The default is 'HIST.HISTORY'.

SPMNSPD=
Specifies the name of the SPMNSPDT data set. This parameter is optional. The default is 'SPMN.SPDT'.

SPMNMBR=
Specifies the name of the control member data set. This parameter is optional.

This parameter is required only when the MEMBER= parameter is specified. Otherwise, this data set is ignored even if it is coded. The default is 'NULLFILE' (member is not specified).

Input

Space Monitor uses one of the two input data sets (SPMNMBR or SPMNIN), in which the user specifies control statements. This section describes these data sets and the control statement format.

Control member data set (SPMNMBR)

This section describes the Control Member data set (SPMNMBR).

Function

SPMNMBR data set is used to specify the data sets to be monitored. The members of this partitioned data set contain control statements. This data set is required only when the PARM= parameter of the EXEC statement is specified.

This data set is also used by DB Historical Data Analyzer.

Format

This data set should be defined as a partitioned data set. It must contain 80-byte fixed-length records. BLKSIZE, if coded, must be a multiple of 80.

```
%USER    USER001 USER002 USER003 USER004 USER005 USER006 USER007
%USER    USER008
*** IMS DATABASE DATA SETS ***
%IMSID   SYS1
DSSTUIVN DSSCHX   SAMPLE.DSSTUIVN                         12 20 10
DSSCHHVN DSSCHH   SAMPLE.DSSCHHVN                         13 30  7
DSFACHON DSFACH   SAMPLE.DSFACHON
DSCRSDVN DSCRS1   SAMPLE.DSCRSDV1                         13 20  7
DSCRSDVN DSCRS2   SAMPLE.DSCRSDV2
```

Record format

Each member contains one or more control statements up to 2003 (excluding comment statements). Following is the record format of control statements:

```
0.........1.........2.........3.........4.........5.........6.........7.........8
012345678901234567890123456789012345678901234567890123456789012345678901234567890
 %USER    user001 user002 user003 user004 user005 user006 user007
 %USER    user008 user009 user010 user011 user012 user013 user014
 %USER    user015 user016 user017 user018 user019 user020
 %IMSID   ims1
 dbname   ddname   dsname                                       aabbb cc
 -        pp q rrr s ttt uuu vvv
```

The control statement is an 80-byte card image record that specifies the IMS database data set or OS data set to be monitored by this utility.

There are two types of control statements; one is a user control statement, another is a data set control statement.

User control statement

User control statement for specifying TSO user IDs. It is an optional control statement. Space Monitor will send notification messages to the TSO users, when it detects threshold exceptions in the monitoring data sets. Specify the user control statements so that they precede the data set control statements. You can specify up to 3 statements.

Note: The user control statements are effective in Space Monitor, but not effective in DB Historical Data Analyzer. If they are specified to DB Historical Data Analyzer, they will be ignored.

IMSID control statement

The IMSID control statement specifies the IMSID of the database data set that is specified by the data set control statement. Only one IMSID can be specified. This statement is required only when the Multiple-IMSID option is enabled and should be placed before any of the data set control statements.

Data set control statement

Data set control statement specifies the data set that Space Monitor monitors. At least one data set control statement is required.

If the database is composed of more than one physical data set (for example, an HISAM primary data set and overflow data set), a set of one data set control statements must be specified for each physical data set, which is represented by the DD1, DD2, or OVFLW parameter in DBD.

The set of data set control statements consists of two statements. The first one contains the database and data set information and some threshold control fields. The second one contains more threshold control fields. If the second statement is omitted, the default values are taken. You can specify up to 2000 statements.

Format of the user control statement

Specify TSO user IDs. If Space Monitor detects threshold exceptions when monitoring databases, it will send notification messages.

Position	Description
1 - 5	Specify, left-justified, a word %USER. This word indicates that this statement is a user control statement.
6 - 9	Blanks.
10 - 16	Specify, left-justified, TSO user ID.
17	Blank.
18 - 64	Repeat TSO user ID (7 columns) and one blank.

Maximum of three statements and 20 user IDs can be specified. Space Monitor will send the messages to all of the TSO user IDs. If some of TSO users are not logged on or are disconnect from their terminals, the messages to the TSO users will be discarded.

You can also specify the special value *JOBUSR as one of these user IDs. This value will be converted to the user ID of the submitter of a JOB when Space Monitor runs.

If you need not send notification to TSO user IDs, do not specify this statement or specify *NO in positions 10 to 12.

Format of the IMSID control statement

Specify the IMSID of the database data set that is specified by the data set control statement. Only one IMSID can be specified. This statement is required only when the Multiple-IMSID option is enabled and should be placed before any of the data set control statements. When this file is specified by the MEMEBER= parameter of the DATABASE statement of the HISTIN data set of DBHDA, IMSID control statement is ignored.

Position **Description**

1 - 6 Specify, left-justified, a word %IMSID. This word indicates that this statement is an IMSID control statement.

7 - 9 Blanks.

10 - 13 Specify the IMSID for the database data sets.

14 - 80 Blanks.

Format of the data set control statement

Specify a data set name and threshold values. Then, Space Monitor will monitor the data set and detects any threshold exceptions. This section describes the syntax of the first and second statements of the data set control statement.

First statement:

Position **Description**

1 - 8 Specify, left-justified, the name of the DBD to be processed. It is required only for an IMS full-function database data set. For HALDB, specify the master database name. For a non-IMS data set and DEDB, if used, it must be left blank.

An asterisk (*) in column 1 indicates a comment statement.

10 - 17 Specify, left-justified, the ddname to be processed. It is required only for an IMS full-function database data set. For HALDB, specify the ddname created with the concatenation of the partition name and the DSG suffix character, A through J or X. You cannot specify the ddname of the Indirect List Data Set (ILDS).

For the HALDB, which is Online Reorganization (OLR) capable, you can specify the DSG suffix character M through V or Y, as well as A through J or X.

Whichever suffix is specified, Space Monitor searches and processes history record entries of both A and M sides. Therefore, HD Pointer Checker writes to a report the statistics of the side that was active when the History record entry was created.

19 - 62 Specify, left-justified, the data set name. For an IMS full-function database data set, this name must correspond to the ddname to be processed.

For the HALDB, which is Online Reorganization (OLR) capable, if the specified database data set does not exist, Space Monitor searches for the other of database data set, (A-J&X) or (M-V&Y).

64 - 65 Specify the warning threshold value for the number of extents. If the number of extents in the data set reaches or passes this threshold, a warning message for this data set is shown in the Space Monitor Exception report.

The value must be right-justified. A leading blank or leading zero can be used. The minimum value is 0, and the maximum value is 99.

To turn off the threshold analysis for the number of extents, enter two asterisks ('**').

The default value is **10**.

66 - 68 Specify the warning threshold value for the percentage of free space. If the percentage of free space in the data set reaches or drops below this threshold, a warning message for this data set is shown in the Space Monitor Exception report.

The value must be right-justified. Leading blank(s) or leading zero(s) can be used. The minimum value is 0, and the maximum value is 100.

To turn off the threshold analysis for the percentage of free space, enter three asterisks ('***').

The default value is **10**.

70 - 71
Specify the warning threshold value for the number of days without an HD Pointer Checker run with HISTORY=YES option. HD Pointer Checker creates an entry in the HISTORY data set when HISTORY=YES option is specified.

If the most recent HD Pointer Checker run with HISTORY=YES option for the database data set was done earlier than this threshold period, a warning message for this database data set is shown in the Space Monitor Exception report.

The value must be right-justified. A leading blank or leading zero can be used.

This field applies only to the IMS full-function database data sets. The minimum value is 0, and the maximum value is 99.

To turn off the threshold analysis for the number of days without running HD Pointer Checker with HISTORY=YES option, enter two asterisks ('**').

The default value is **14**.

Second statement:

Position **Description**

1
This field indicates whether this is the second statement. If this is the second statement, specify '-'. The first statement must precede this one.

10 - 11
Specify the warning threshold value for the number of available extents. If the number of available extents in the data set reaches or drops below this threshold, a warning message for this data set is shown in the Space Monitor Exception report.

The value must be right-justified. A leading blank or leading zero can be used. The minimum value is 0, and the maximum value is 50.

To turn off the threshold analysis for the number of extents, enter two asterisks ('**').

The default value is **10**.

For the details of the available extents, read Chapter 28, “Available data set extents and last space,” on page 531.

13
Specify 'Y' or '*'. If 'Y' is specified and the data set uses the last extent, a warning message for this data set is shown in the Space Monitor Exception report.

To turn off the threshold analysis for the last extension, enter one asterisk ('*').

The default value is **'Y'**.

For the details of the last extent, read Chapter 28, “Available data set extents and last space,” on page 531.

15 - 17 Specify the warning threshold value for the percentage of space used. The value indicates the ratio of the used space to the allocated space in the data set on the volume. If the percentage of space used reaches or passes this threshold, a warning message for this data set is shown in the Space Monitor Exception report.

The value must be right-justified. Leading blank(s) or leading zero(s) can be used. The minimum value is 0, and the maximum value is 100.

To analyze this threshold, the database data set entries must exist in the HISTORY data set.

To turn off the threshold analysis for the space use, enter three asterisks ('***').

The default value is **90**.

19 Specify 'Y' or '*'. If 'Y' is specified and not enough space is left on the DASD volume for the data set to extend, a warning message for this data set is shown in the Space Monitor Exception report.

For the details of the last space, read Chapter 28, “Available data set extents and last space,” on page 531.

To turn off the threshold analysis for the last space, enter one asterisk ('*').

The default value is **'Y'**.

21 - 23 Specify the warning threshold value for the percentage of CA splits. The value indicates the ratio of the number of CA splits to the used control areas. If the percentage of CA splits of the data set reaches or passes this threshold, a warning message for this data set is shown in the Space Monitor Exception report.

The value must be right-justified. Leading blank(s) or leading zero(s) can be used. The minimum value is 0, and the maximum value is 100.

To analyze this threshold, the database data set must be a KSDS and its entries must exist in the HISTORY data set.

To turn off the threshold analysis for the percentage of CA split, enter three asterisks ('***').

The default value is **50**.

25 - 27 Specify the warning threshold value for the percentage of CI splits. The value indicates the ratio of the number of CI splits to the used control intervals. If the percentage of CI splits of the data set reaches or passes this threshold, a warning message for this data set is show in the Space Monitor Exception report.

The value must be right-justified. Leading blank(s) or leading zero(s) can be used. The minimum value is 0, and the maximum value is 100.

To analyze this threshold, the database data set must be a KSDS and its entries must exist in the HISTORY data set.

To turn off the threshold analysis for CI split, enter three asterisks ('***').

The default value is **50**.

29 - 31

Specify the warning threshold value for the number of days that can pass without a database reorganization.

If the number of days since most recent reorganization exceeds this threshold, a warning message for this database data set is shown in the Space Monitor Exception report.

To analyze this threshold, HD Pointer Checker needs to create entries in the HISTORY data set with DBRC=Y parameter in the EXEC statement and HISTORY=YES option.

The value must be right-justified. Leading blank(s) or leading zero(s) can be used.

This field applies only to the IMS full-function database data set. The minimum value is 0, and the maximum value is 999.

To turn off the threshold analysis for the number of days that can pass without the reorganization, enter three asterisks ('***').

The default value is "do not monitor the threshold analysis."

33-35

Specify the warning threshold value for the percentage of the data set used space within the maximum size. The value must be the ratio of the used space to the maximum size of the IMS database data set.

The maximum size is as follows:

- When OSAM data set of non-HALDB HD database of which block size is an even number, the maximum size is 8GB.
- For other database data set than above, the maximum size is 4GB. (1 GB is 1,024 MB)

If the percentage to the data set size limit reaches or exceeds this threshold, a warning message for this data set is shown in the Space Monitor Exception report.

The value must be right-justified. Leading blank(s) or leading zero(s) can be used. The minimum value is 0, and the maximum value is 100.

This threshold is effective for a IMS full-function database. To analyze this threshold, the database data set entries must exist in the HISTORY data set.

To turn off the threshold analysis for the data set limit, specify three asterisks ('***').

The default value is 90.

37-40

Specify the warning threshold value for the data set size. The value must be the MB size of the data set used space. (1 MB is 1,024 KB = 1,048,576 byte)

If the data set size reaches or exceeds this threshold, a warning message for this data set is shown in the Space Monitor Exception report.

The value must be right-justified. Leading blank(s) or leading zero(s) can be used. The minimum value is 0, and the maximum value is 9999.

To turn off the threshold analysis for the data set size, specify four asterisks ('****').

The default value is that no analyzing is done for the threshold, so leaving the field blanks has the same effect as specifying four asterisk.

SPMNIN data set

This section describes the SPMNIN data set.

Function

This optional input data set contains control statements. This data set is required only when the PARM= parameter of the EXEC statement is not specified.

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.

```
//SPMNIN   DD *
DSSTUIVN DSSCHX   SAMPDS                                  12020 10
/*
```

Record format

The record format of this data set is exactly the same as that of the control member data set (SPMNMBR). Refer to “Control member data set (SPMNMBR)” on page 502.

Output

This section describes the output of Space Monitor.

Following reports are stored in the IMS Tools KB Output repository when PROC ITKBSRVR=*servername* is specified and Space Monitor is called by setting SPMN=YES on a DATABASE statement:

- Space Analysis by Data Set report
- Summary of Data Sets by Volume report
- Space Monitor Exception report
- Space Monitor Messages

Space Monitor Graph Record data set (SPMNSPDT)

This section describes the Space Monitor Graph Record data set (SPMNSPDT).

Function

This is an OS sequential data set that contains Space Monitor graph records. One record is created for each data set, and each record consists of a prefix, multiple entries of data (called ***buckets***), and a suffix.

For HALDB partitions, the data for any (M-V&Y) data set is merged into the record for the corresponding (A-J&X) data set.

One bucket contains one day's space management information, such as the number of cylinders allocated, the number of cylinders used from the viewpoint of an OS data set, the number of cylinders used for IMS data (applicable only to the IMS database data sets), and the date when the data was collected.

This data set is used as an input data set for producing graph reports.

Note: You should allocate enough space for this data set to maintain space allocation information for all the IMS database data sets and/or OS data sets you want to monitor in your system environment.

This data set should reside on a direct access device, and must have fixed-length records. If blocked, the block size must be a multiple of the logical record length.

You should use DISP=(MOD,KEEP,KEEP) for this data set. If this data set is to be allocated and used for the first time, the SPACE= parameter must be specified.

Format

Each Space Monitor record consists of a prefix, multiple entries of data, and a suffix.

Figure 190 shows the format of Space Monitor Record.

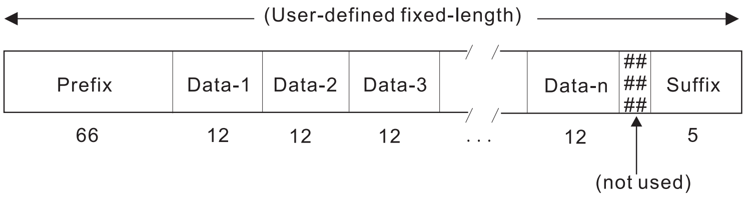


Figure 190. Format of the Space Monitor record

Total number of buckets in a record is calculated as shown in Figure 191.

$$\text{Number of buckets} = \frac{\text{(Logical record length) - (Prefix length) - (Suffix length)}}{\text{(Bucket length)}}$$

Note: Prefix length = 65
Suffix length = 5
Bucket length = 12

Figure 191. Formula to calculate the total number of buckets in a record—1

Therefore, if you want to maintain 90 days' space management information for each data set to be monitored, the required logical record length of the SPMNSPDT data set is calculated as shown in Figure 192.

$$\frac{\text{LRECL} - 66 - 5}{12} = 90 \longrightarrow \text{LRECL} = 1151$$

Figure 192. Formula to calculate the total number of buckets in a record—2

SPMNPRT data set

This output data set contains the following five types of reports produced by Space Monitor:

- Space Analysis by Data Set report
- Summary of Data Sets by Volume report
- Total DASD Utilization by Volume/Device-Type report
- Legend report
- Space Monitor Graph report

Space Analysis by Data Set report

This report is used to analyze the space utilization of data sets specified by the user. Entries are sorted in alphabetical order of the database name (DBNAME) and data set ddname (DDNAME). Figure 193 on page 511 shows a sample of the Space Analysis by Data Set report.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - SPMN          "SPACE ANALYSIS BY DATA SET REPORT"                        PAGE:     1
5655-K53                                                   DATE: 07/11/2006  TIME: 11.19.55                         FABKSPMN - V2.R2

 MEMBER NAME : N/A
 ----------------------

 DBNAME         DDNAME      DSNAME                                         DBORG ACCM     CISP  CASP  UNIT    REORGDATE    HDPCDATE
 TYP    PRI    SEC EXT AEXT     ALLOC %FSP %NRUS    TOTBLK BLKSZ LRECL  MXSEG ACTMX      ROOTS   TOTALSEG VOLSER EXT     ALLOC %USE
        DATASET SIZE    %SZ
 --- ------ ------ ---  --- --------- ---- ----- --------- ----- ----- ------ -----  ---------  --------- ------ --- --------- ----

                            TESTDS.PUBLIC.SAMPLE.VSAM01                                    N/A   N/A  3390-3        N/A         N/A
 CYL      3      2   1  237         3  100  N/A         0  4096   240   N/A    N/A        N/A        N/A SYS001   1         3    0
 CYL      3      2                                                                                         SYS002   0         0    0

                   0  *****

 HDAMDB2        HDAMDS4     TESTDS.PUBLIC.SAMPLE.HDAMDS4                    HDAM ES-U      N/A   N/A  3390-3 07/06/2006  07/10/2006
 CYL     50     50   1  118        50   91    7      2475  1024  1017   122    122         70      18087 SYS004   1        50   10

            2,534,400   0.1

 HISAMDB1       HISAMDS1    TESTDS.PUBLIC.SAMPLE.HISAMDS1                  HISAM KS-U        0     0  3390-3 07/06/2006  07/10/2006
 CYL     50     50   1  118        50   98  N/A        90  8192   510   N/A    N/A        130        584 SYS004   1        50    2

              737,280   0.0

 HISAMDB1       HISAMDS2    TESTDS.PUBLIC.SAMPLE.HISAMDS2                  HISAM ES-U      N/A   N/A  3390-3 07/06/2006  07/10/2006
 CYL     50     20   1  118        50   68  N/A      1419  8192   512   N/A    N/A          0     106017 SYS004   1        50   32

           11,624,448   0.3

 TPFOH1         TPFOH1AA    TESTDS.PUBLIC.SAMPLE.TPFOH1.A00001             PHDAM ES-U      N/A   N/A  3390-3 07/06/2006  07/10/2006
 CYL     50     50   1  118        50   47    7     19845   512   505   246    246      11000      80037 SYS004   1        50   54

           10,160,640   0.2

 TPFOH2         TPFOH2AA    TESTDS.PUBLIC.SAMPLE.TPFOH2.A00001            PHIDAM ES-U      N/A   N/A  3390-3 07/06/2006  07/10/2006
 CYL     50     50   1  118        50   89    7      4410   512   505   122    122       9000      18178 SYS02D   1        50   12

            2,257,920   0.1

 TPFOH2         TPFOH2AX    TESTDS.PUBLIC.SAMPLE.TPFOH2.X00001            PHINDX KS-U        0     0  3390-3     NONE    07/10/2006
 CYL     20     10   1  118        20   98  N/A       735   512    14   N/A    N/A        N/A       9001 SYS02E   1        20    5

              376,320   0.0

 TPFOH3         TPFOH3AA    TESTDS.PUBLIC.SAMPLE.TPFOH3.A00001             PHDAM ES-U      N/A   N/A  3390-3 07/06/2006  07/10/2006
 CYL     50     20   1  118        50   97    7      1470   512   505   150    150       5000       5220 SYS02F   1        50    4

              752,640   0.0

 TPFOX1         TPFOX1AA    TESTDS.PUBLIC.SAMPLE.TPFOX1.A00001            PSINDX KS-U        0     0  3390-3 07/06/2006  07/10/2006
 CYL     10     10   1  118        10   87  N/A      1470   512    54   N/A    N/A        N/A       9178 SYS02F   1        10   20

              752,640   0.0
```

Figure 193. SPMNPRT—Space Analysis by Data Set report

This report contains the following information for each database data set and/or OS data set specified by the control statements.

Note

In the following description, (G) indicates that the data is general information, (H) indicates that it is HD Pointer Checker analysis data (this data is not available for OS data sets), and (S) indicates that it is space analysis data.

For an OS data set, the IMS-related information is not applicable.

MEMBER NAME
This is the name of the member specified by the EXEC PARM= parameter. If no member name is specified by the EXEC PARM= parameter (that is, control statements are specified in the SPMNIN data set), N/A is shown. (G)

DBNAME
This is the name of the database. This field is left blank in the case of an OS data set. (G)

DDNAME
This is the ddname of the data set. This field is left blank in the case of an OS data set. (G)

DSNAME
This is the name of the data set. (G)

DBORG
This is the database organization. (H)

One of the following is shown:

HDAM, HIDAM, HISAM, SHISAM, PINDX (HIDAM index), SINDX (Secondary index), PHDAM, PHIDAM, PHINDX (PHIDAM index), PSINDX

This field is left blank in the case of an OS data set.

ACCM
This is the access method. (H)

One of the following is shown:

OSAM, KS-U, KS-S, ES-U, ES-S (KS=VSAM KSDS, ES=VSAM ESDS, U=UNIQUE, S=SUBALLOCATION)

This field is left blank in the case of an OS data set.

UNIT
This is the DASD device type. (G)

TYP
This is the space allocation type: CYL (cylinders), TRK (tracks) or BLK (blocks). (S)

PRI
This is the primary allocation (cylinders, tracks or blocks). (S)

SEC
This is the secondary allocation (cylinders, tracks or blocks). (S)

EXT
This is the total number of extents on the data set. (S)

AEXT
This is the number of available extents. It shows the smaller of the two values: that per volume or that for the entire data set. (S)

For the details, read Chapter 28, “Available data set extents and last space,” on page 531.

ALLOC
This is the total space allocated on the data set (cylinders, tracks or blocks). (S)

For OSAM multivolume data set, if the space allocation type is different for each volume, Space Monitor selects the unit for reporting the total allocation (cylinder, track or block). The unit is shown in parentheses just below the allocation value.

%FSP

This is the percentage of the usable free space within the total allocated space. (G, H)

For HDAM, HIDAM, PHDAM, and PHIDAM database data sets, usable free space is calculated as shown in Figure 194.

```
Usable free space = (remaining space between the last block and the end of extent)
                    the end of extent)
                    +(usable imbedded free space according
                    to the bit map)
```

Figure 194. Formula to calculate usable free space (HDAM, HIDAM, PHDAM, and PHIDAM)

For index database data sets, usable free space is calculated as shown in Figure 195.

```
Usable free space = allocated space - (total number of segments
                    x logical record length)
```

Figure 195. Formula to calculate usable free space (index database data set)

Note: Total number of segments for a shared index database is the sum of segments only for the specified DBNAME on the control statement.

For HISAM database data sets and OS data sets, usable free space is calculated as shown in Figure 196.

```
Usable free space = (remaining space between the last block
                    and the end of extent)
```

Figure 196. Formula to calculate usable free space (HISAM database data set and OS data set)

An asterisk (*) after %FSP indicates that the imbedded free space is not accurate, because the HD Pointer Checker run with the HISTORY=YES option is obsolete (older than the warning threshold value specified in the SPMNMBR or SPMNIN data set).

%NRUS

This is the percentage of nonreusable free space according to the bit map. (G, H)

This field applies to HDAM, HIDAM, PHDAM, and PHIDAM database data sets.

An asterisk (*) after %NRUS indicates that the nonreusable free space is not accurate, because the HD Pointer Checker run with the HISTORY=YES option is obsolete (older than the warning threshold value specified in the SPMNMBR or SPMNIN data set).

TOTBLK

This is the total number of blocks in the data set. (G)

BLKSZ

This is the block size or control interval (CI) size. (G)

LRECL

This is the logical record length. (G)

MXSEG

This is the longest segment length in the data set, including the segment prefix as defined in DBD. (H)

This field applies to HDAM, HIDAM, PHDAM, and PHIDAM database data sets.

ACTMX

This is the size of the longest segment detected by HD Pointer Checker in this data set. (H)

This field applies to HDAM, HIDAM, PHDAM, and PHIDAM database data sets.

CISP

This is the number of VSAM control interval (CI) splits. This field applies only to the data set with ACCM=KS-U or KS-S. (H)

CASP

This is the number of VSAM control area (CA) splits. This field applies only to the data set with ACCM=KS-U or KS-S. (H)

ROOTS

This is the number of root segments in this database data set. (H)

TOTALSEG

This is the total number of segments in this database data set, including roots. (H)

REORGDATE

This is the date of the most recent reorganization. (H)

If the reorganization date does not exist in the HISTORY data set, this field shows 'NONE'.

An asterisk (*) after REORGDATE indicates that the imbedded reorganization date might not be accurate, because the HD Pointer Checker run with the HISTORY=YES option is obsolete (older than the warning threshold value specified in the SPMNMBR or SPMNIN data set).

HDPCDATE

This is the date of the most recent HD Pointer Checker run (with HISTORY=YES option) that created an entry in the HISTORY data set. (H)

If HD Pointer Checker run date does not exist in the HISTORY data set, this field shows 'NONE'.

An asterisk (*) after the HDPCDATE indicates that the date of the HD Pointer Checker run with HISTORY=YES option is obsolete (older than the warning threshold value specified in the SPMNMBR or SPMNIN data set).

For the following fields (VOLSER, EXT, ALLOC, and %USE), the information is shown for each volume, in the case of a multivolume data set.

VOLSER

This is the volume serial number of the data set. (S)

EXT

This is the number of extents on the volume. (S)

ALLOC

This is the allocated space on the volume. (S)

%USE
This is the percentage of used space within the total allocated space on the volume. (S)

DATASET SIZE
The space, that is used for the data set in bytes. This is the block or CI size multiplied by the number of the total blocks or CIs in the data set.

%SZ
The percentage of the space that is used in the data set to the maximum database data set size.

The following error messages are included in the report if the data set encounters associated error conditions:

***** **DASD NOT MOUNTED** *****
The DASD which should contain the data set is not mounted, and the space allocation information for the data set is not available.

***** **NO VTOC ENTRY ON ANY VOLUME** *****
The data set is cataloged but no VTOC entry is found on any volume for the data set. Space allocation information for the data set is not available.

***** **DATA SET NOT CATALOGED** *****
The data set is not cataloged and the space allocation information for the data set is not available.

***** **DATA SET ON TAPE** *****
The data set resides on tape and space allocation information for the data set is not available.

***** **DATA SET MIGRATED** *****
The data set is cataloged but it is migrated. Space allocation information for the data set is not available.

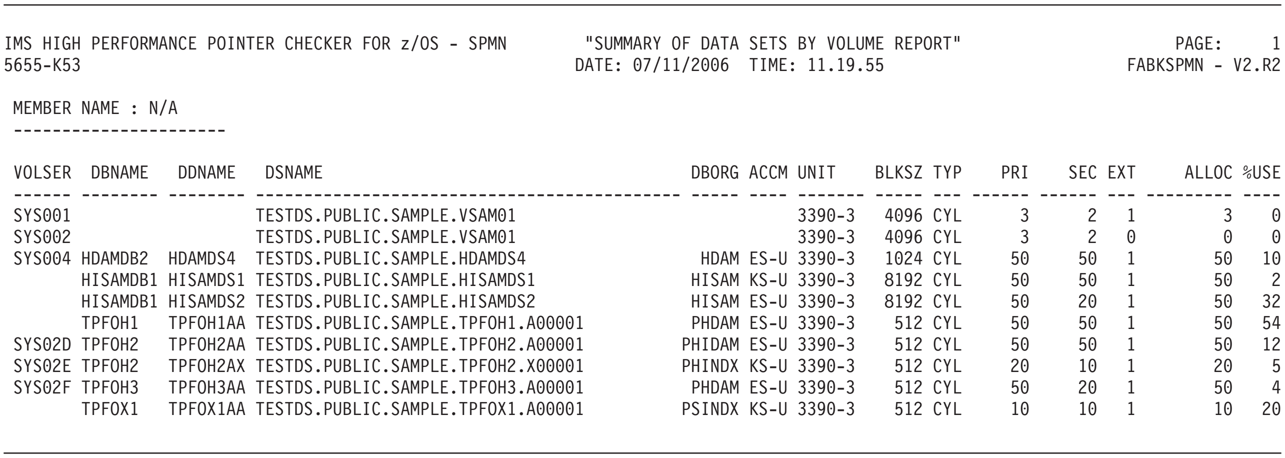
***** **DSCB BLOCK SIZE=0** *****
The block size information in DSCB for the data set is zero. Space allocation information for the data set is not available.

***** **DATA SET ON UNSUPPORTED DEVICE** *****
The data set resides on an unsupported device and space allocation information for the data set is not available.

Summary of Data Sets by Volume report

This report contains a list of data sets by volume. Entries are sorted in alphabetical order of the volume serial number (VOLSER), database name (DBNAME), and data set ddname (DDNAME).

Figure 197 on page 516 shows a sample of the Summary of Data Sets by Volume report.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - SPMN        "SUMMARY OF DATA SETS BY VOLUME REPORT"                     PAGE:     1
5655-K53                                                   DATE: 07/11/2006  TIME: 11.19.55                          FABKSPMN - V2.R2

 MEMBER NAME : N/A
 ----------------------

 VOLSER  DBNAME   DDNAME   DSNAME                                      DBORG ACCM UNIT    BLKSZ TYP    PRI    SEC EXT     ALLOC %USE
 ------ -------- -------- -------------------------------------------- ----- ---- ------- ----- --- ------ ------ --- --------- ----
 SYS001                   TESTDS.PUBLIC.SAMPLE.VSAM01                             3390-3   4096 CYL      3      2  1          3    0
 SYS002                   TESTDS.PUBLIC.SAMPLE.VSAM01                             3390-3   4096 CYL      3      2  0          0    0
 SYS004 HDAMDB2  HDAMDS4  TESTDS.PUBLIC.SAMPLE.HDAMDS4                  HDAM ES-U 3390-3   1024 CYL     50     50  1         50   10
        HISAMDB1 HISAMDS1 TESTDS.PUBLIC.SAMPLE.HISAMDS1                HISAM KS-U 3390-3   8192 CYL     50     50  1         50    2
        HISAMDB1 HISAMDS2 TESTDS.PUBLIC.SAMPLE.HISAMDS2                HISAM ES-U 3390-3   8192 CYL     50     20  1         50   32
        TPFOH1   TPFOH1AA TESTDS.PUBLIC.SAMPLE.TPFOH1.A00001           PHDAM ES-U 3390-3    512 CYL     50     50  1         50   54
 SYS02D TPFOH2   TPFOH2AA TESTDS.PUBLIC.SAMPLE.TPFOH2.A00001          PHIDAM ES-U 3390-3    512 CYL     50     50  1         50   12
 SYS02E TPFOH2   TPFOH2AX TESTDS.PUBLIC.SAMPLE.TPFOH2.X00001          PHINDX KS-U 3390-3    512 CYL     20     10  1         20    5
 SYS02F TPFOH3   TPFOH3AA TESTDS.PUBLIC.SAMPLE.TPFOH3.A00001           PHDAM ES-U 3390-3    512 CYL     50     20  1         50    4
        TPFOX1   TPFOX1AA TESTDS.PUBLIC.SAMPLE.TPFOX1.A00001          PSINDX KS-U 3390-3    512 CYL     10     10  1         10   20
```

Figure 197. SPMNPRT—Summary of Data Sets by Volume report

This report contains the following information:

MEMBER NAME
This is the name of the member specified by the EXEC PARM= parameter. If no member name is specified by the EXEC PARM= parameter (that is, control statements are specified in the SPMNIN data set), N/A is shown.

VOLSER
This is the volume serial number.

DBNAME
This is the database name. This field is left blank in the case of an OS data set.

DDNAME
This is the data set ddname. This field is left blank in the case of an OS data set.

DSNAME
This is the data set name.

DBORG
This is the database organization. (H)

One of the following is shown:

HDAM, HIDAM, HISAM, SHISAM, PINDX (HIDAM index), SINDX (Secondary index), PHDAM, PHIDAM, PHINDX (PHIDAM index), PSINDX

For an OS data set, this field is left blank.

ACCM
This is the access method. (H)

One of the following is shown:

OSAM, KS-U, KS-S, ES-U, ES-S (KS=VSAM KSDS, ES=VSAM ESDS, U=UNIQUE, S=SUBALLOCATION)

This field is left blank in the case of an OS data set.

UNIT
This is the DASD device type.

BLKSZ
This is the block size or control interval (CI) size.

TYP
This is the space allocation type: CYL (cylinders), TRK (tracks) or BLK (blocks).

PRI
This is the primary allocation (cylinders, tracks or blocks).

SEC
This is the secondary allocation (cylinders, tracks or blocks).

EXT
This is the number of extents on the volume.

ALLOC
This is the allocated space on the volume (cylinders, tracks or blocks).

%USE
This is the percentage of used space within the total allocated space on the volume.

Total DASD Utilization by Volume/Device-Type report

This report contains information on DASD utilization by volume and device type.

Figure 198 shows a sample of the Total DASD Utilization by Volume/Device-Type report.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - SPMN    "TOTAL DASD UTILIZATION BY VOLUME/DEVICE-TYPE REPORT"          PAGE:     1
5655-K53                                                      DATE: 07/11/2006  TIME: 11.19.55                       FABKSPMN - V2.R2

 MEMBER NAME : N/A
 ----------------------

                 SPECIFIED DATA SETS    OTHER        FREE SPACE ON VOLUME
                 ------------------- --------- ------------------------------
 DEVICE   VOLSER     ALLOC      USED     ALLOC     EMPTY EXTNTS CONTIG  CONTIG
 TYPE                CYLS       CYLS     CYLS      CYLS          CYLS    TRKS
 ------   ------     -----      -----    -----     -----  -----  -----  ------

  3390-3  SYS001         3         0       247      3089     21   2366   35490
          SYS002         0         0         5      3334      1   3334   50012
          SYS004       200        49       552      2587      3   2587   38805
          SYS02D        50         6        14      3275      3   3274   49110
          SYS02E        20         1         4      3315      3   3314   49710
          SYS02F        60         4         4      3275      3   3274   49110

          ------ --------- --------- --------- ---------  -----  -----  ------

  (TOTAL)      6       333        60       826     18875
```

Figure 198. SPMNPRT—Total DASD Utilization by Volume/Device-Type report

MEMBER NAME
This is the name of the member specified by the EXEC PARM= parameter. If no member name is specified by the EXEC PARM= parameter (that is, control statements are specified in the SPMNIN data set), N/A is shown.

DEVICE TYPE
This is the DASD device type of the volume.

VOLSER
This is the volume serial number.

SPECIFIED DATA SETS
ALLOC CYLS shows the total number of cylinders allocated for the data sets that are specified by the control statements.

USED CYLS shows the total number of cylinders used for the data sets that are specified by the control statements.

OTHER
ALLOC CYLS shows the total number of cylinders allocated for the data sets on the volume that are ***not*** specified by the control statements.

FREE SPACE ON VOLUME
EMPTY CYLS shows the number of empty cylinders on the volume.

EXTNTS shows the number of free space extents on the volume.

CONTIG CYLS shows, in number of cylinders, the largest free space extent on the volume.

CONTIG TRKS shows, in number of tracks, the largest free space extent on the volume.

Legend report

This report contains the legend for Space Analysis by Data Set report.

Figure 199 shows a sample of the Legend report.

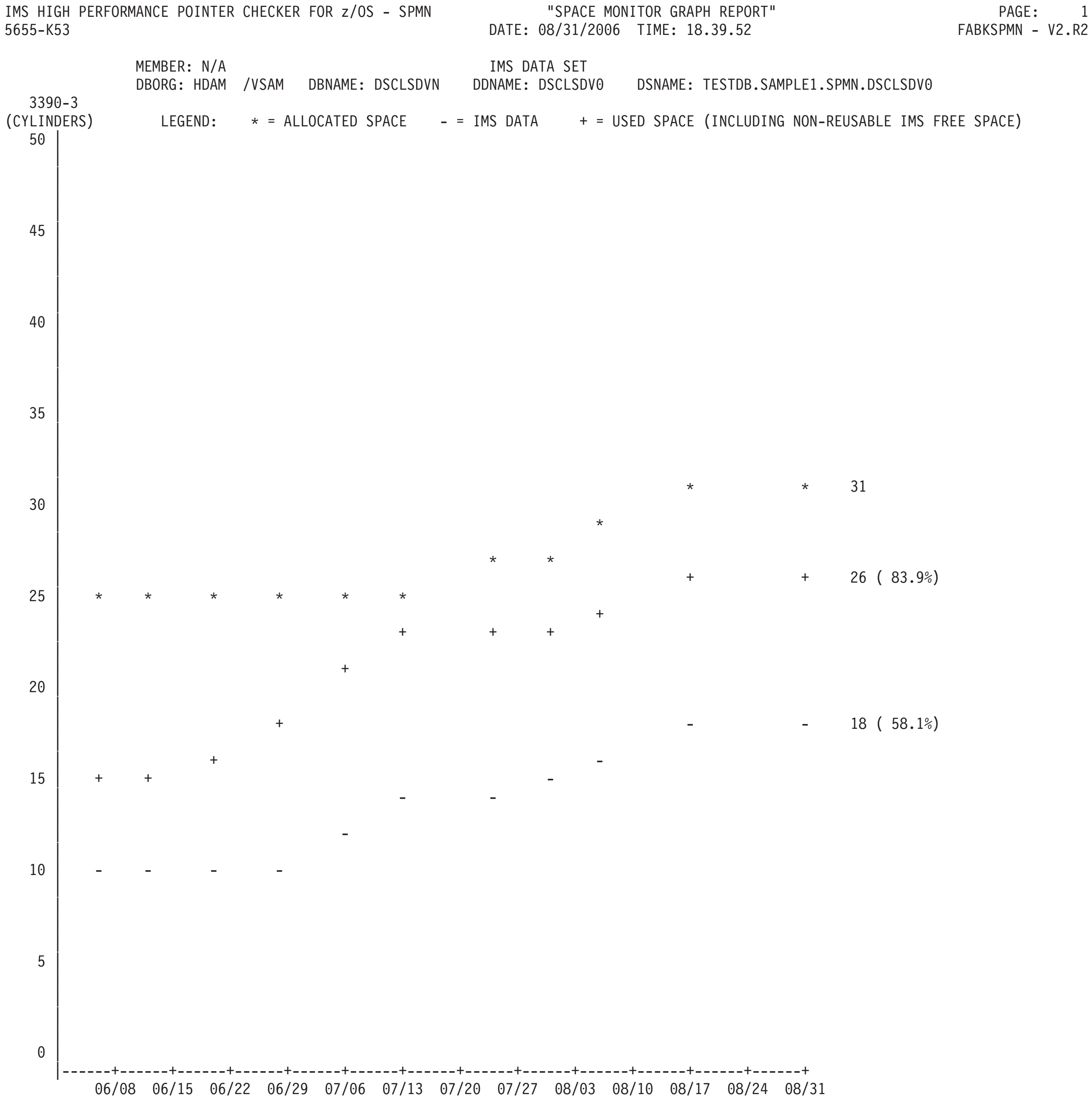
```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - SPMN                    "LEGEND REPORT"                                  PAGE:     1
5655-K53                                                   DATE: 07/11/2006  TIME: 11.19.55                         FABKSPMN - V2.R2


 DBNAME     - DATABASE NAME
 DDNAME     - DATA SET DDNAME
 DSNAME     - DATA SET NAME
 DBORG      - DATABASE ORGANIZATION:
              HDAM, HIDAM, HISAM, SHISAM, PINDX(HIDAM INDEX), SINDX(SECONDARY INDEX)
              PHDAM, PHIDAM, PHINDX(PHIDAM INDEX), PSINDX(SECONDARY INDEX)

 ACCM       - ACCESS METHOD:
              OSAM, KS-U, KS-S, ES-U, ES-S  (KS=VSAM KSDS, ES=VSAM ESDS, U=UNIQUE, S=SUBALLOCATION)
 UNIT       - DASD DEVICE TYPE

 TYP        - SPACE ALLOCATION TYPE:
              CYL(CYLINDERS), TRK(TRACKS), BLK(BLOCKS)
 PRI        - PRIMARY ALLOCATION IN CYLINDERS, TRACKS, OR BLOCKS
 SEC        - SECONDARY ALLOCATION IN CYLINDERS, TRACKS, OR BLOCKS
 EXT        - NUMBER OF EXTENTS
 AEXT       - COUNT OF AVAILABLE EXTENTS
 ALLOC      - TOTAL ALLOCATED SPACE (CYLINDERS, TRACKS OR BLOCKS)
 %FSP       - PERCENTAGE OF USABLE FREE SPACE RELATIVE TO TOTAL ALLOCATED SPACE:
              USABLE FREE SPACE (FOR HDAM AND HIDAM DATA SETS) =
                REMAINING SPACE BETWEEN LAST BLOCK AND END OF EXTENT
               + USABLE IMBEDDED FREE SPACE ACCORDING TO BIT MAP
              USABLE FREE SPACE (FOR INDEX DB DATA SETS) = ALLOCATED SPACE - (TOTAL NUMBER OF SEGMENTS  X  LOGICAL RECORD LENGTH)
              USABLE FREE SPACE (FOR OTHER DATA SETS) = REMAINING SPACE BETWEEN LAST BLOCK AND END OF EXTENT
              AN * AFTER %FSP MEANS IMBEDDED FREE SPACE NOT COMPUTED BECAUSE OF AN
              OBSOLETE HD POINTER CHECKER RUN WITH HISTORY=YES OPTION (OLDER THAN SPECIFIED LIMIT)
 %NRUS      - PERCENTAGE OF NON-REUSABLE FREE SPACE ACCORDING TO BIT MAP
 TOTBLK     - TOTAL BLOCKS IN DATA SET
 BLKSZ      - BLOCK OR CI SIZE
 LRECL      - LOGICAL RECORD LENGTH
 MXSEG      - LONGEST SEGMENT LENGTH IN DATA SET INCLUDING SEGMENT PREFIX
              AS DEFINED IN DBD
 ACTMX      - SIZE OF THE LONGEST SEGMENT DETECTED BY HD POINTER CHECKER IN THIS DATA SET
 CISP       - VSAM CI SPLITS
 CASP       - VSAM CA SPLITS
 ROOTS      - NUMBER OF ROOT SEGMENTS IN DATABASE
 TOTALSEG   - TOTAL SEGMENTS IN DATA SET INCLUDING ROOTS IF ANY
 REORGDATE  - REORGANIZATION DATE OF THE DATABASE
              AN * AFTER REORGDATE MEANS REORGDATE IS NOT ACCURATE BECAUSE OF AN
              OBSOLETE HD POINTER CHECKER RUN WITH HISTORY=YES OPTION (OLDER THAN SPECIFIED LIMIT)
 HDPCDATE   - DATE OF THE MOST RECENT HD POINTER CHECKER RUN WITH HISTORY=YES OPTION
              AN * AFTER HDPCDATE MEANS AN HD POINTER CHECKER RUN WITH HISTORY=YES OPTION OLDER THAN SPECIFIED LIMIT
 VOLSER     - SERIAL NUMBER OF VOLUME(S) CONTAINING DATA SET
 EXT        - COUNT OF EXTENTS ON VOLUME
 ALLOC      - ALLOCATED SPACE ON VOLUME
 %USE       - PERCENTAGE OF ALLOCATED SPACE USED
 DATASET SIZE - DATA SET SIZE = BLOCK OR CI SIZE X TOTAL BLOCKS IN DATA SET
 %SZ        - PERCENTAGE OF DATA SET USED WITHIN THE MAXIMUM SIZE LIMIT OF THE DATABASE DATA SET
```

Figure 199. SPMNPRT—Legend report

Space Monitor Graph report

This report is a scatter plot that shows space allocation and use information.

Figure 200 shows a sample of the Space Monitor Graph report.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - SPMN              "SPACE MONITOR GRAPH REPORT"                           PAGE:     1
5655-K53                                                   DATE: 08/31/2006  TIME: 18.39.52                         FABKSPMN - V2.R2

                MEMBER: N/A                                IMS DATA SET
                DBORG: HDAM  /VSAM   DBNAME: DSCLSDVN    DDNAME: DSCLSDV0    DSNAME: TESTDB.SAMPLE1.SPMN.DSCLSDV0
   3390-3
(CYLINDERS)        LEGEND:    * = ALLOCATED SPACE    - = IMS DATA     + = USED SPACE (INCLUDING NON-REUSABLE IMS FREE SPACE)
   50 |
      |
      |
      |
      |
   45 |
      |
      |
      |
      |
   40 |
      |
      |
      |
      |
   35 |
      |
      |
      |
      |                                                                            *             *     31
   30 |
      |                                                                 *
      |
      |                                                    *      *
      |                                                                            +             +     26 ( 83.9%)
   25 |    *     *       *       *       *      *
      |                                                                 +
      |                                         +          +      +
      |
      |                                  *
   20 |
      |
      |                          +                                                 -             -     18 ( 58.1%)
      |
      |                  +                                              -
   15 |    +     +                                                -
      |                                         -          -
      |
      |                                  -
      |
   10 |    -     -       -       -
      |
      |
      |
      |
    5 |
      |
      |
      |
      |
    0 |
      |------+------+------+------+------+------+------+------+------+------+------+------+------+
           06/08  06/15  06/22  06/29  07/06  07/13  07/20  07/27  08/03  08/10  08/17  08/24  08/31
```

Figure 200. SPMNPRT—Space Monitor Graph report

The scatter plot shows the following data for each data set:

- Allocated space
- Used space
- IMS data (space used for IMS data)

  IMS data is produced only for the IMS full-function database data sets that have entries in the HISTORY data set.

This report contains the following information:

MEMBER
This is the name of the member specified by the EXEC PARM= parameter. If no member name is specified by the EXEC PARM= parameter (that is, control statements are specified in the SPMNIN data set), N/A is shown.

DBORG
This is the database organization. The format is:

```
xxxxx/yyyy
```

Where:

xxxxx
HDAM, HIDAM, HISAM, SHISAM, PINDX (HIDAM index), SINDX (secondary index), PHDAM, PHIDAM, PHINDX (PHIDAM index).

yyyy
VSAM or OSAM

In the case of an OS data set, only 'VSAM' or 'NON-VSAM' is shown.

DBNAME
This is the database name. This field is left blank in the case of an OS data set.

DDNAME
This is the ddname of the data set. This field is left blank in the case of an OS data set.

DSNAME
This is the data set name.

The following is the description of the scatter plot data:

ALLOCATED SPACE
The space allocated for this data set is shown as the asterisk (*) marks.

IMS DATA
The space used as IMS data in this data set is shown as minus (-) marks. For an OS data set, this value is always shown as zero.

The space used as IMS data is as follows, depending on the database type:

- For HDAM and HIDAM databases, it is the total space used by the IMS segments in the data set.
- For index databases, the value is calculated as follows:

  ```
  (Total number of segments in the data set) x (Logical record length)
  ```

  Note: In the case of shared index databases, the space information is reported for one secondary index portion only.
- For HISAM databases, the value is calculated as follows:

  ```
  (Total number of segments in the data set) x (Average segment length)
  ```

  Note: The average segment length is the average for the entire database, including both the primary data set and overflow data set.

USED SPACE
The space used for this data set is shown as plus (+) marks. For VSAM data sets, the space includes VSAM CI splits and CA splits.

For IMS full-function database data sets (except HISAM and index database data sets), the used space includes nonreusable IMS free space.

Note

On the report, where the asterisk (*) mark intersects other marks (+ or -), the asterisk mark overrides others. Where the plus (+) mark intersects the minus (-) mark, plus mark is shown.

X-axis shows the date. A unit of scale corresponds to one day. The recent 90 days of space information is shown on the graph up to the date of this Space Monitor run.

Y-axis shows the amount of space in cylinders. The maximum and minimum scale values are determined by the maximum and minimum space values.

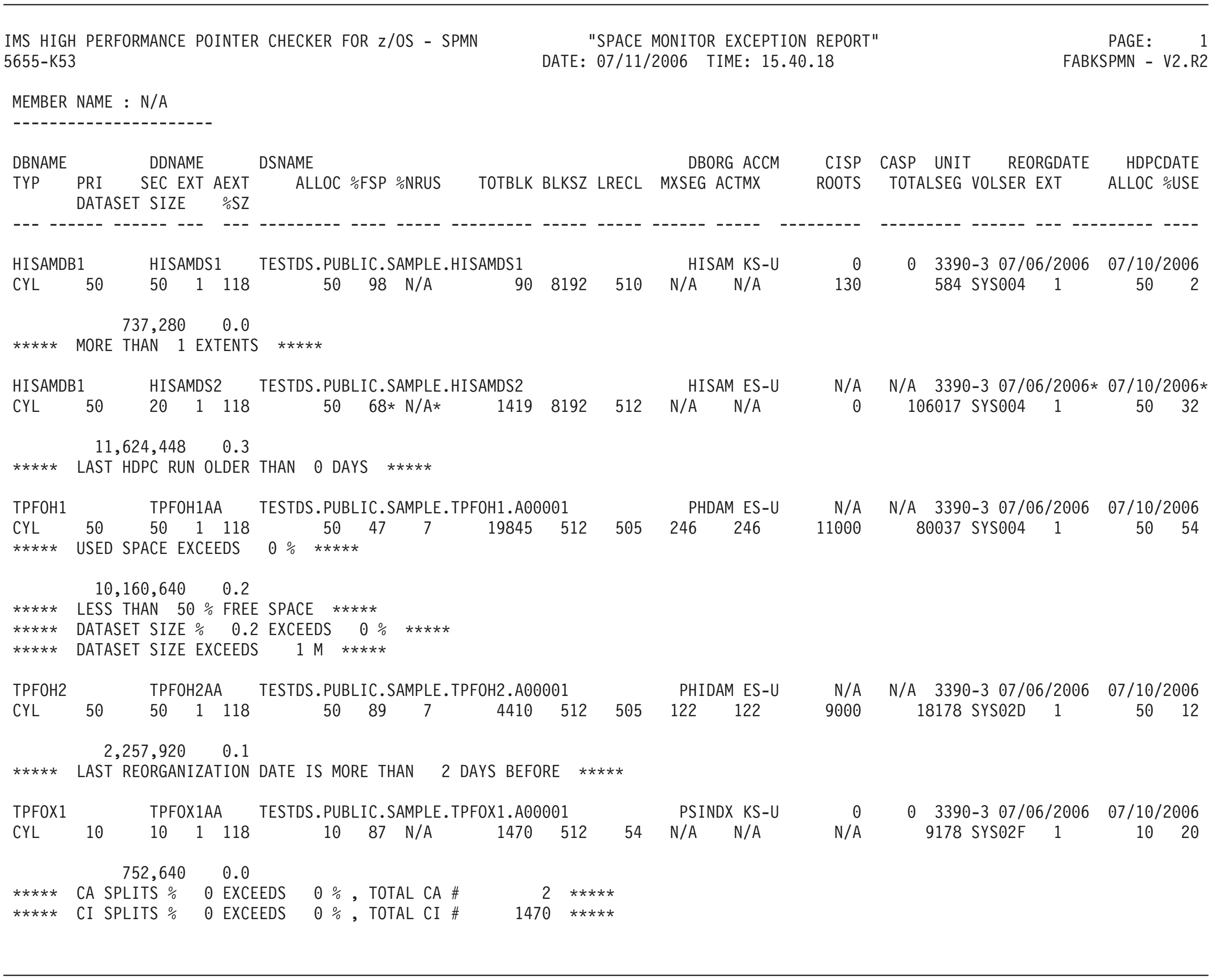
SPMNPRTW data set

This output data set contains the Space Monitor Exception report. This report describes the analyzed data set information and the associated threshold warning messages for any of the database data sets and/or OS data sets.

Entries are sorted in alphabetical order of the database name (DBNAME) and data set ddname (DDNAME).

Space Monitor Exception report

Figure 201 on page 523 shows a sample of the Space Monitor Exception report.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - SPMN            "SPACE MONITOR EXCEPTION REPORT"                          PAGE:      1
5655-K53                                                   DATE: 07/11/2006  TIME: 15.40.18                          FABKSPMN - V2.R2

 MEMBER NAME : N/A
 ----------------------

 DBNAME         DDNAME      DSNAME                                         DBORG ACCM     CISP  CASP  UNIT    REORGDATE    HDPCDATE
 TYP    PRI    SEC EXT AEXT     ALLOC %FSP %NRUS    TOTBLK BLKSZ LRECL  MXSEG ACTMX      ROOTS   TOTALSEG VOLSER EXT     ALLOC %USE
        DATASET SIZE    %SZ
 --- ------ ------ ---  --- --------- ---- ----- --------- ----- ----- ------ -----  ---------  --------- ------ --- --------- ----

 HISAMDB1       HISAMDS1    TESTDS.PUBLIC.SAMPLE.HISAMDS1                  HISAM KS-U        0     0  3390-3 07/06/2006  07/10/2006
 CYL     50     50   1  118        50   98  N/A         90  8192   510   N/A    N/A        130        584 SYS004   1        50    2

             737,280    0.0
 *****  MORE THAN  1 EXTENTS  *****

 HISAMDB1       HISAMDS2    TESTDS.PUBLIC.SAMPLE.HISAMDS2                  HISAM ES-U      N/A   N/A  3390-3 07/06/2006* 07/10/2006*
 CYL     50     20   1  118        50   68* N/A*      1419  8192   512   N/A    N/A          0     106017 SYS004   1        50   32

          11,624,448    0.3
 *****  LAST HDPC RUN OLDER THAN  0 DAYS  *****

 TPFOH1         TPFOH1AA    TESTDS.PUBLIC.SAMPLE.TPFOH1.A00001             PHDAM ES-U      N/A   N/A  3390-3 07/06/2006  07/10/2006
 CYL     50     50   1  118        50   47    7      19845   512   505   246    246      11000      80037 SYS004   1        50   54
 *****  USED SPACE EXCEEDS   0 %  *****

          10,160,640    0.2
 *****  LESS THAN  50 % FREE SPACE  *****
 *****  DATASET SIZE %   0.2 EXCEEDS   0 %  *****
 *****  DATASET SIZE EXCEEDS    1 M  *****

 TPFOH2         TPFOH2AA    TESTDS.PUBLIC.SAMPLE.TPFOH2.A00001            PHIDAM ES-U      N/A   N/A  3390-3 07/06/2006  07/10/2006
 CYL     50     50   1  118        50   89    7       4410   512   505   122    122       9000      18178 SYS02D   1        50   12

           2,257,920    0.1
 *****  LAST REORGANIZATION DATE IS MORE THAN   2 DAYS BEFORE  *****

 TPFOX1         TPFOX1AA    TESTDS.PUBLIC.SAMPLE.TPFOX1.A00001            PSINDX KS-U        0     0  3390-3 07/06/2006  07/10/2006
 CYL     10     10   1  118        10   87  N/A       1470   512    54   N/A    N/A        N/A       9178 SYS02F   1        10   20

             752,640    0.0
 *****  CA SPLITS %   0 EXCEEDS   0 % , TOTAL CA #         2  *****
 *****  CI SPLITS %   0 EXCEEDS   0 % , TOTAL CI #      1470  *****
```

Figure 201. SPMNPRTW—Space Monitor Exception report

For the description of all the fields in this report, see “Space Analysis by Data Set report” on page 510.

The following messages are included in the report:

******* LARGER THAN 4 GIGABYTES *******
: The total space of the data set is equal to or larger than 4 gigabytes for a VSAM data set and an odd block size for a non-VSAM data set. Each field of ALLOC, %FSP, %NRUS, TOTBLK, and %USED is calculated by the total space in the data set, which is treated as X’FFFFFFFF’ (4 gigabytes - 1 byte).

******* LARGER THAN 8 GIGABYTES *******
: The total space of the data set is equal to or larger than 8 gigabytes. Each field on ALLOC, %FSP, %NRUS, TOTBLK, and %USED is calculated by the total space in the data set, which is treated as X’1FFFFFFFE’ (8 gigabytes - 2 bytes).

******* LAST HDPC RUN OLDER THAN*cc* DAYS *******
: The most recent HD Pointer Checker run with HISTORY=YES option for the database data set was done earlier than the warning threshold specified in columns 70-71 of the first control statement for the database data set.

An * is shown after %FSP, %NRUS, and REORGDATE value, showing that these values are not accurate, because the HD Pointer Checker run with the HISTORY=YES option is obsolete (older than the specified limit). An * is also shown after the date on the HDPCDATE field showing that the HD Pointer Checker run date is older than the specified limit.

******* LESS THAN *bbb* % FREE SPACE *******
The percentage of free space in the data set is equal to or less than the warning threshold value specified in columns 66-68 of the first control statement for the data set.

******* MORE THAN *aa* EXTENTS *******
The number of extents of the data set is equal to or greater than the warning threshold value specified in columns 64-65 of the first control statement for the data set.

******* NONE *******
No exceptional conditions were detected for any of the database data sets and/or OS data sets processed.

******* AVAILABLE EXTENTS LESS THAN *ppp* EXTENTS (VOLUME LIMIT) ***** or ***** AVAILABLE EXTENTS LESS THAN *ppp* EXTENTS (DATASET LIMIT) *******
The number of available extents in the data set is equal to or less than the warning threshold value specified in columns 10 and 11 of the second control statement for the data set. For either of the following kinds of data set, however, the message is not issued, because the extent number used (AEXT) is always zero:

- VSAM data set whose secondary allocation is zero and for which only one volume is defined.
- Non-VSAM data set whose secondary allocation is zero.

******* CA SPLITS %*xxx* EXCEEDS *ttt* %, TOTAL CA *#yyyyyyyy* *******
The percentage *xxx*—that is, the ratio of the area where CA split is occurring to the total control areas *yyyyyyyy*—is equal to or more than the warning threshold value *ttt* specified in columns 21-23 of the second control statement for the data set.

******* CI SPLITS %*xxx* EXCEEDS *uuu* %, TOTAL CI *#yyyyyyyy* *******
The percentage *xxx*—that is, the ratio of the CI (control interval) where CI split is occurring to the total CIs *yyyyyyyy*— is equal to or more than the warning threshold value *uuu* specified in columns 25-27 of the second control statement for the data set. The total CI is the high-used RBA.

******* USING LAST EXTENT (VOLUME LIMIT) ***** or ***** USING LAST EXTENT (DATASET LIMIT) *******
The last extent has been used. To display this message, specify Y in column 13 of the second control statement for the data set; to suppress it, specify * there. For either of the following kinds of data set, however, the message is not issued, because the extent number used (AEXT) is always zero:

- VSAM data set whose secondary allocation is zero and to which only one volume is defined.
- Non-VSAM data set whose secondary allocation is zero.

******* USED SPACE EXCEEDS*rrr* % *******
The percentage of space used on the volume has reached or exceeded the warning threshold value specified in columns 15-17 of the second control statement for the data set.

******* LAST REORGANIZATION DATE IS MORE THAN *vvv* DAYS BEFORE *******

The most recent reorganization date of the database data set is earlier than the warning threshold specified in columns 29-31 of the second control statement for the database data set.

******* NO LAST REORGANIZATION DATE *******

The warning threshold value for the reorganization data was specified in columns 29-31 of the second control statement, but the most recent reorganization data does not exist in the HISTORY data set.

******* NO AVAILABLE SPACE *******

The number of extents available is more than zero, but there is no space for extension. This threshold value is controlled in column 19 of the second control statement for the data set. For either of the following kinds of data set, however, the message is not issued, because the extent number used (AEXT) is always zero:

- VSAM data set whose secondary allocation is zero and for which only one volume is defined.
- Non-VSAM data set whose secondary allocation is zero.

******* NEXT VOLUME NOT YET SELECTED BY SMS *******

The warning threshold value for the last available space was specified in column 19 of the second control statement, but the available space cannot be calculated, because the volume cannot be identified.

This message is issued when no space exists on the current volume and the next volume names is '*'—that is, when the volume name has not been selected by DFSMS/MVS™.

******* DATASET SIZE % xxx.x EXCEEDS yyy % *******

The percentage *xxx.x*—that is, the ratio of the space that is used in the data set to the maximum size of the IMS database data set—is equal to or more than the warning threshold value *yyy* that is specified in column 33-35 of the second control statement for the data set.

******* DATASET SIZE EXCEEDS xxxx M *******

The data set size is equal to or more than the warning threshold value *xxxx* MB that is specified in column 37-40 of the second control statement for the data set.

SPMNMSG data set

This output data set contains the Space Monitor Messages report produced by Space Monitor.

Space Monitor Messages report

This report (see Figure 202 on page 526) shows all control statements and messages.

These control statements have been specified in the control member data set (SPMNMBR) or the SPMNIN data set. Messages issued from Space Monitor are also shown in the report.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - SPMN                "SPACE MONITOR MESSAGES"                                  PAGE:     1
5655-K53                                                    DATE: 07/11/2006  TIME: 11.19.55                             FABKSPMN - V2.R2

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

HISAMDB1 HISAMDS1 TESTDS.PUBLIC.SAMPLE.HISAMDS1
HISAMDB1 HISAMDS2 TESTDS.PUBLIC.SAMPLE.HISAMDS2
HDAMDB2  HDAMDS4  TESTDS.PUBLIC.SAMPLE.HDAMDS4
TPFOH1   TPFOH1AA TESTDS.PUBLIC.SAMPLE.TPFOH1.A00001
TPFOH2   TPFOH2AA TESTDS.PUBLIC.SAMPLE.TPFOH2.A00001
TPFOH2   TPFOH2AX TESTDS.PUBLIC.SAMPLE.TPFOH2.X00001
TPFOH3   TPFOH3AA TESTDS.PUBLIC.SAMPLE.TPFOH3.A00001
TPFOX1   TPFOX1AA TESTDS.PUBLIC.SAMPLE.TPFOX1.A00001
                  TESTDS.PUBLIC.SAMPLE.VSAM01

FABK0001I FABKSPMN ENDED NORMALLY
```

Figure 202. SPMNMSG—Space Monitor Messages report

Chapter 27. JCL examples for Space Monitor

This chapter shows JCL examples used in Space Monitor.

<u>**Topics:**</u>

Example 1: Using the SPMNIN data set to monitor space

Figure 203 on page 528 presents an example of a job in which the SPMNIN input data set is used to monitor several IMS full-function database data sets and DEDB data sets.

The first and third control statements for the IMS full-function database data sets specify three threshold values explicitly. The warning threshold for the number of extents is 13, the warning threshold for the percentage of free space is 20, and the warning threshold for the number of days without HD Pointer Checker run (with HISTORY=YES option) is 7 for both of the data sets. For all other IMS full-function database data sets, the default values 10, 10, and 14 are applied for these three threshold values.

DEDB data sets are treated simply as VSAM data sets. Threshold values are meaningless for DEDB data sets for the following reasons:

- Extents are allocated at the DEDB initialization time by the IMS DEDB Initialization utility and no extents are dynamically obtained after the initialization.
- The IMS DEDB Initialization utility initializes all space allocated and no OS free space exists.
- HD Pointer Checker does not support DEDB data sets.

```
//SPMN     EXEC SPMN
//S.SPMNIN  DD  *
*        1         2         3         4         5         6         7
*...+....0....+....0....+....0....+....0....+....0....+....0....+....0..
* IMS DL/I DATABASE DATA SET GROUP FOR APPL01
*-DBD--* *-DD---* *-DATA SET NAME ---------------------------* AABBB CC
DSSCHHVN DSSCHHV0 SAMPLE.DSSCHHV0                              13 20  7
DSFACHON DSFACHO0 SAMPLE.DSFACHO0
DSCRSDVN DSCRSDV0 SAMPLE.DSCRSDV0                              13 20  7
DSCRSDVN DSCRSDV1 SAMPLE.DSCRSDV1
DSCLSDVN DSCLSDV0 SAMPLE.DSCLSDV0
*
*        1         2         3         4         5         6         7
*...+....0....+....0....+....0....+....0....+....0....+....0....+....0..
* IMS DEDB AREA DATA SET GROUP FOR APPL01
*-N/A--* *-N/A--* *-DATA SET NAME ---------------------------* AABBB CC
                  SAMPLE.DB01AR01
                  SAMPLE.DB01AR02
                  SAMPLE.DB02AR01
                  SAMPLE.DB02AR02
                  SAMPLE.DB02AR03
/*
//*
```

Figure 203. Example 1: Using the SPMNIN data set to monitor space

Example 2: Using the SPMNMBR data set to monitor space

Figure 205 presents an example of a job in which the SPMNMBR input data set is used to monitor several IMS full-function database data sets and DEDB data sets.

The input control statements are the same as in “Example 1: Using the SPMNIN data set to monitor space” on page 527, and they are specified in the member APPL01 of the SPMN.MEMBER partitioned data set. Figure 204 presents an example of an IEBUPDTE utility job in which the member APPL01 is added to the data set SPMN.MEMBER.

```
//FPPROC  EXEC  PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD SYSOUT=A
//SYSUT2   DD DSN=SPMN.MEMBER,DISP=OLD
//SYSIN   DD  DATA,DLM=@@
./ ADD  NAME=APPL01,LIST=ALL
*        1         2         3         4         5         6         7
*...+....0....+....0....+....0....+....0....+....0....+....0....+....0..
* IMS DL/I DATABASE DATA SET GROUP FOR APPL01
*-DBD--* *-DD---* *-DATA SET NAME ---------------------------* AABBB CC
DSSCHHVN DSSCHHV0 SAMPLE.DSSCHHV0                              13 20  7
DSFACHON DSFACHO0 SAMPLE.DSFACHO0
DSCRSDVN DSCRSDV0 SAMPLE.DSCRSDV0                              13 20  7
DSCRSDVN DSCRSDV1 SAMPLE.DSCRSDV1
DSCLSDVN DSCLSDV0 SAMPLE.DSCLSDV0
*
*        1         2         3         4         5         6         7
*...+....0....+....0....+....0....+....0....+....0....+....0....+....0..
* IMS DEDB AREA DATA SET GROUP FOR APPL01
*-N/A--* *-N/A--* *-DATA SET NAME ---------------------------* AABBB CC
                  SAMPLE.DB01AR01
                  SAMPLE.DB01AR02
                  SAMPLE.DB02AR01
                  SAMPLE.DB02AR02
                  SAMPLE.DB02AR03
@@
//*
```

Figure 204. Example 2: Creating a member on SPMNMBR data set

```
//SPMN     EXEC SPMN,
//             MEMBER=APPL01,                  FOR MEMBER NAME
//             SPMNMBR='SPMN.MEMBER'           FOR SPMNMBR DS
//*
```

Figure 205. Example 2: Using the SPMNMBR data set to monitor space

Example 3: Increasing buckets on Space Monitor graph record

Figure 208 presents an example of a job to increase the number of buckets on the Space Monitor graph record.

Suppose the logical record length of the original Space Monitor graph record (SPMN.SPDT) is 440, and this length is to be increased to 812. The original record has 30 buckets, as calculated by the formula show in Figure 206.

$$\text{Number of buckets} = \frac{440 - 66 - 5}{12} = 30 \qquad \text{(remainder 9)}$$

Figure 206. Sample for calculating the number of buckets in a Space Monitor graph record with original record length

The new Space Monitor graph record (SPMN.SPDT) will contain 61 buckets, calculated as shown in Figure 207.

$$\text{Number of buckets} = \frac{812 - 66 - 5}{12} = 61 \qquad \text{(remainder 9)}$$

Figure 207. Sample for calculating the number of buckets in a Space Monitor graph record with increased record length

The new record contains the space allocation information copied from the old record. At this time, the prefix, suffix, and the added buckets are not yet reformatted. When Space Monitor accesses the new record, it automatically reformats these parts.

```
//EXPAND    EXEC PGM=IEBGENER
//SYSPRINT  DD   SYSOUT=A
//SYSUT2    DD DSN=SPMN.SPDTN,DISP=(NEW,KEEP),
//             UNIT=SYSDA,VOL=SER=IMSDBT,SPACE=(TRK,(10,5)),
//             DCB=(RECFM=FB,LRECL=812,BLKSIZE=3248)
//SYSUT1    DD DSN=SPMN.SPDT,DISP=OLD
//SYSIN     DD *
  GENERATE  MAXFLDS=1
    RECORD  FIELD=(440,1,,1)
/*
```

Figure 208. Example 3: Increasing buckets on Space Monitor graph record

Chapter 28. Available data set extents and last space

This chapter describes how the Space Monitor utility checks the following:

- The number of available extents for the data set
- Whether there is enough space left or not on the DASD volume

Topics:

- “Available data set extents”
- “Last space” on page 534

Available data set extents

The number of available extents can be calculated as shown in Figure 209.

```
  (the number of allocatable extents) - (the number of extents allocated)
```

Figure 209. Formula to calculate available extents

The number of allocatable extents is taken as either the number of allocatable extents for the usable volumes or the number of allocatable extents for a data set, whichever is smaller. They can be calculated as shown in Figure 210.

```
A. The number of allocatable extents for the usable volumes[1] =
    (Number of Extents per Volume[2]
  × the number of volumes on that the data set to be allocated)
  + (Maximum Number of Extents per Volume - the number of extents on the current volume)
  + (the number of extents allocated)

B  The number of allocatable extents for a data set =
   Maximum Number of Extents per Data set[3]
```

Figure 210. Formula to calculate the number of allocatable extents for the usable volumes and for a data set

Notes:

1. ’usable volumes’ refers to the volumes on that the data set are already allocated and to be allocated.
2. ’Maximum Number of Extents per Volume’ is shown in Table 63 on page 532.
3. ’Maximum Number of Extents per Data set’ is shown in Table 63 on page 532.

Table 63 shows the maximum number of extents that Space Monitor utility uses to calculate the number of available extents.

Table 63. Maximum number of extents that Space Monitor utility uses to calculate the number of available extents

Data set type	Maximum number of extents per volume	Maximum number of extents per data set
VSAM	119	119 (251[1])
OSAM	16	52 to 60 (62[2])
Others	16	16 × 59

Notes:

1. 119 for DFSMS/MVS up to Version 1.3, 251 for Versions over 1.4.
2. The maximum number of extents per OSAM data set is determined as follows:
   a. If there is no rotational position sensing (RPS) device, the maximum is 62.
   b. If there is a rotational position sensing (RPS) device, the maximum depends on the number of blocks per track, as follows:
      1) The length of OSAM data extent block (DEB)

```
Length of DEB = Length of appendage sector table (5 doublewords fixed)
+ Length of basic DEB (4 doublewords fixed)
+ Length of access method dependent section (2 doublewords fixed)
+ Length of subroutine name section (1 doubleword fixed)
+ Length of standard DEB extents (2 doublewords * number of extents)
+ Length of OSAM extent data (2 doublewords * number of extents)
+ Length of sector table (1 doubleword fixed + [number of blocks per track / 8]
  (round up decimals) doublewords)
```

      2) The maximum number of extents is the number which gives the largest size of DEB within 255 doublewords.
      3) Examples:
         - For the minimum-sized sector table (7 blocks per track), the number of extents is 60.
         - For the maximum-sized sector table (255 blocks per track), the number of extents is 52.

   For the details of OSAM, see *IMS Administration Guide: Database Manager*.

Note: To check whether there is an RPS, UCBSCAN is used and UCB is referred to.

Table 64 on page 533 shows the examples of how to calculate the number of allocatable extents from the number of extents currently allocated. Allocatable extents means the space that can be allocated under the current status of allocation.

Table 64. Examples of calculating the number of allocatable extents from the number of extents allocated

Ex am ple	Data set type	Number of usable volumes	Number of extents allocated					Number of allocatable extents calculated	How the number of allocatable extents is calculated				
No.			vol1	vol2	vol3	vol4	Total		vol1	vol2	vol3	vol4	Total
1	VSAM	2	119	100	-	-	219	The allocatable extents is 238, because the value of A ((119 - 100) + 219 = 238) is smaller than the value of B (251).	119	119	-	-	238
2	VSAM	3	119	100	-	-	219	The allocatable extents is 251, because the value of A (119 × 1 + (119 - 100) +219=357) is greater than the value of B (251).	119	119	13	-	251
3	OSAM	3	16	16	10	-	42	The allocatable extents is 48, because the value of A ((16 - 10) + 42 = 48) is smaller than the value of B (62).	16	16	16	-	48
4	OSAM	4	16	16	10	-	42	The allocatable extents is 62, because the value of A (16 × 1 + (16 - 10) + 42 = 64) is greater than the value of B (62).	16	16	16	14	62
5	OSAM	4	1	1	1	-	3	In this case, one extent is allocated on volumes 1 through 3 and no free space exists on volumes 1 and 2. The allocatable extents is 34, because the value of A (16 × 1 + (16 - 1) + 3 = 34) is smaller than the value of B (62).	1	1	16	16	34

Table 64. Examples of calculating the number of allocatable extents from the number of extents allocated (continued)

Example	Data set type	Number of usable volumes	Number of extents allocated					Number of allocatable extents calculated	How the number of allocatable extents is calculated				
No.			vol1	vol2	vol3	vol4	Total		vol1	vol2	vol3	vol4	Total
6	OSAM	4	1	1	1 [eof]	1*	4	In this case, OSAM Preallocated function allocated volumes 1 through 4, but data exists only on volumes 1 through 3. volume 4 (marked *) is empty. No more extent is allocated on volume 3. The allocatable extents is 19, because the value of A ((16 - 1) + 4 = 19) is smaller than the value of B (62).	1	1	1	16	19

Note:
- DFSMS/MVS V1.4
- No RPS

When the secondary allocation size is zero, the following values are shown:
- VSAM data set: AEXT is equivalent to the number of candidate volumes, because one extension is allowed for each volume.
- Non-VSAM data set: AEXT is always zero.

The Space Monitor utility monitors whether the last extent has been reached. This is determined in the same way as the available extents. When the remaining extent becomes zero, the last extent has been reached.

For how to specify whether to monitor the available extents and the last extent, see “Control member data set (SPMNMBR)” on page 502.

Last space

Whether there is enough space for extension on a volume is determined as follows:
- When there is contiguous free space that is larger than the size of extension on the volume, it is determined as there is enough space.

  The size of the extension is as follows:
  - For OSAM, the secondary amount.
  - For VSAM, the primary amount if it is the first extent on the volume, the secondary amount if it is the second or after.

- The volumes searched are as follows:
  - For VSAM, the last volume with extents and candidate volumes are searched.
  - For OSAM, the last volume with extents and the first candidate volume are searched.
  - For VSAM and OSAM undefined volumes (that is, candidate volumes are searched by IDCAMS but no volume has been selected by SMS and so is marked as * on the catalog), a warning message is written in a report.

Chapter 29. Operating instructions for Space Monitor Site Default Generation utility

The Space Monitor Site Default Generation utility (SPMN Site Default Generation utility) enables you to use your own default value for the Space Monitor control statements. It runs as a batch job. For the description of the control statements, see “Control member data set (SPMNMBR)” on page 379.

This chapter describes how to generate and use the site default table of Space Monitor, which is referred to as the SPMN site default table.

Topics:

Functions

The SPMN Site Default Generation utility has two functions; generating and reporting the SPMN site default table.

Generating the SPMN site default table

The SPMN Site Default Generation utility analyzes the Space Monitor control statements and generates a source code for the SPMN site default table. If you use the site default, the library for the SPMN site default table module (FABKCTL0) must be concatenated to the STEPLIB DD of FABKSPMN runtime JCL.

When FABKSPMN finds a load module with the name FABKCTL0 in the STEPLIB libraries, FABKSPMN loads it and uses the values in the FABKCTL0 as defaults of the control statements.

If you specify a value in the control statement in the SPMNMBR data set or the SPMNIN data set in the FABKSPMN JCL, you can override the site default value at run time.

To use the SPMN site default, do as follows:

1. Run the Space Monitor Site Default Generation utility (FABKTGEN) to create a source code of the SPMN site default table (FABKCTL0)
2. Assemble and link the FABKCTL0 source code
3. Concatenate the load module library in which FABKCTL0 resides to the STEPLIB of the Space Monitor FABKSPMN JCL

The SPMN site default table is also effective when Space Monitor is called in the HD Pointer Checker FABPMAIN job or is called in the HP Image Copy FABJMAIN job.

Figure 211 on page 538 shows the process flow of how to use the SPMN site default table.

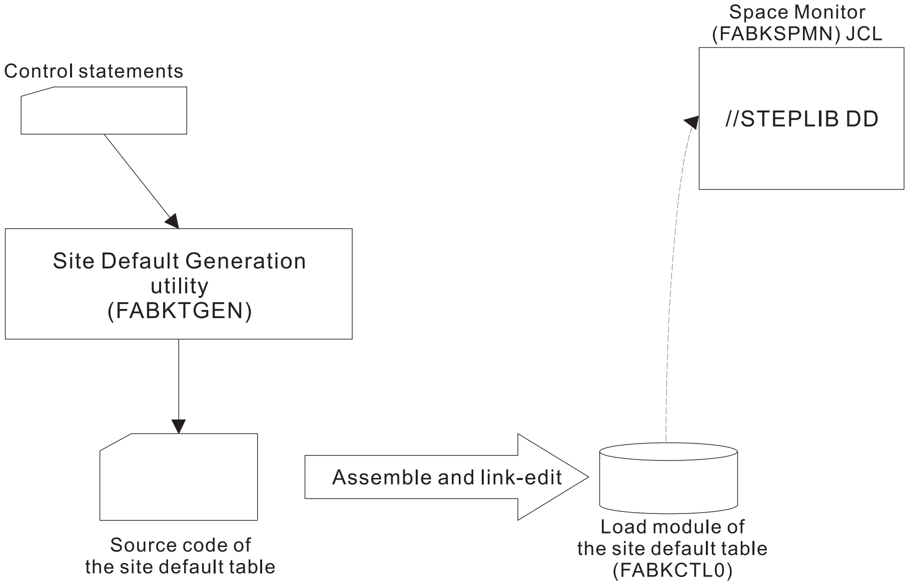

Control statements
Space Monitor (FABKSPMN) JCL
//STEPLIB DD
Site Default Generation utility (FABKTGEN)
Assemble and link-edit
Source code of the site default table
Load module of the site default table (FABKCTL0)

Figure 211. Process flow of SPMN site default table creation

Reporting the SPMN site default table

The SPMN Site Default Generation utility reads the SPMN site default table and prints the site default values that are set in the reports.

Job control language

This section describes the requirements for using the SPMN Site Default Generation utility (FABKTGEN).

FABKTGEN JCL

To run the SPMN Site Default Generation utility (FABKTGEN), supply an EXEC statement with the PARM parameters and appropriate DD statements as shown in Table 65.

Table 65 summarizes the DD statements.

Table 65. FABKTGEN DD statements

DDNAME	Use: Input (In), Output (Out)	Format	EXEC PARM= Required (R) or optional (O) Header	
			PARM='GEN'	PARM='REPORT'
STEPLIB	In	PDS	R	R
SPMNIN	In	LRECL=80	R	
SYSPUNCH	Out	LRECL=80	R	
SYSPRINT	Out	LRECL=133	R	R

Table 65. FABKTGEN DD statements (continued)

DDNAME	Use: Input (In), Output (Out)	Format	EXEC PARM= Required (R) or optional (O) Header	
			PARM='GEN'	PARM='REPORT'
SYSABEND or SYSUDUMP	Out	LRECL=133	O	O

EXEC

This statement must be in the following format:

```
//     EXEC PGM=FABKTGEN,PARM='parm'
```

Specify GEN or REPORT for *parm*.

GEN

Specifies to generate a site default table. This is the default.

REPORT

Specifies to print the site default values stored in the site default table.

Sample JCLs that run the FABKTGEN program with PARM='GEN' and PARM='REPORT' are provided in the SHPSSAMP data set that is provided by the product. The member names are FABKDFL1 and FABKDFL2.

STEPLIB DD

This required input data set contains the IMS HP Pointer Checker load module library. When PARM='REPORT' is specified in the EXEC statement, you must specify the data set that includes the site default table module member (FABKCTL0).

SPMNIN DD

This is a required data set, when PARM='GEN' is specified in the EXEC statement. The format is the same as the SPMNIN data set statement. Specify this input data set to include your own default values for the control statements.

SYSPUNCH DD

This is a required output data set when PARM='GEN' is specified in the EXEC statement. An assembler source code of the site default table will be produced in this data set. The following DCB parameters must be specified:

RECFM=F or FB

LRECL=80

BLKSIZE=80 or multiple of 80

SYSPRINT DD

This is a required output data set. The Space Monitor Messages report will be printed in this data set. You can specify SYSOUT=* (or JES output class name) instead of a data set name.

When FABKTGEN generates the SPMN site default table, the Space Monitor Messages report contains exactly the same control statements that you specified in the SPMNIN data set. When FABKTGEN reports site default values, the Space Monitor Messages report contains the site default values that is stored in the SPMN site default table.

SYSUDUMP DD (or SYSABEND)

This defines the output for a system ABEND dump routine. It is used only when a you want to create a dump.

FABKTGEN SPMNIN data set

The SPMNIN data set is required to generate the SPMNIN site default table.

You can specify the USER control statements, the data set control statements, or both to the SPMNIN data set. Specify your own value to the column that you want to change the default values from the Space Monitor's system default value. The SPMN Site Default Generation utility analyzes the control statements, and stores the site default values in the SPMN site default table (FABKCTL0).

For description of each statement, see "Control member data set (SPMNMBR)" on page 379.

USER control statement
You can specify up to 20 user IDs. If you specify this control statement, code it before the data set statements.

Data set control statements
You can specify the first statement, or a set of the first and the second statements.

The positions where you can specify your own default value for the site default are described in Table 66 and Table 67. Specify your own default value in each position marked 'Yes.' If values are omitted, the Space Monitor system default values will be used.

Table 66. The positions available for site default in the first statement

Position	System default value of Space Monitor	Can specify site default value?	Description
1-8	(None)	No (Ignored if specified)	DBD name.
10-17	(None)		DD name.
19-62	(None)		Data set name.
64-65	10	Yes	Warning threshold value for the number of data set extents.
66-68	10	Yes	Warning threshold value for the percentage of data set free space.
70-71	14	Yes	Warning threshold value for the number of days without an HD Pointer Checker run.

Table 67. The positions available for site default in the second statement

Position	System default value of Space Monitor	Can specify site default value?	Description
01	-	-	″-″ (Indicates that it is the second statement. It is required.)

Table 67. The positions available for site default in the second statement (continued)

Position	System default value of Space Monitor	Can specify site default value?	Description
10-11	10	Yes	The warning threshold value for the number of available extents.
13	Y		Whether to monitor the data set that uses the last extent.
15-17	90		The warning threshold value for the percentage of space used.
19	Y		Whether to monitor if there is enough space left on the DASD volume for the data set to extend.
21-23	50		The warning threshold value for the percentage of CA splits.
25-27	50		The warning threshold value for the percentage of CI splits.
29-31	***		The warning threshold value for the number of days that can pass without a database reorganization.
33-35	90		The warning threshold value for the percentage of data set that is used within the maximum size limit of the database data set.
37-40	****		The warning threshold value for the data set size in megabytes.

Examples

The part underlined in Figure 212 show the effective statements and columns for the site default. For the blank column, the Space Monitor system default value will be used.

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

%USER    USER001
DB000001 DS000001                                              12080 10
-        01 N 080 Y             080 1000
```

Figure 212. Example of the specification for the site default

Assembling and link-editing FABKCTL0

To create the site default table module FABKCTL0, assemble and link-edit the SYSPUNCH that is generated by FABKTGEN.

For SYSIN of the assemble job step, specify the SYSPUNCH data set that is generated in the FABKTGEN. In the link-edit job step, it is recommended that you use AMODE=31, RMODE=ANY instead of the default values AMODE=24, RMODE=24 by adding MODE=31 and RMDE=ANY to the EXEC statement PARM list.

Figure 213 on page 542 shows a JCL is the sample for creating FABKCTL0.

```
//FABKDFL1 JOB
//********************************************************************
//*   Licensed Materials - Property of IBM                           *
//*                                                                  *
//*   5655-K53                                                       *
//*                                                                  *
//*   (c) Copyright IBM Corp. 2006 All Rights Reserved.              *
//*                                                                  *
//*   US Government Users Restricted Rights - Use,                   *
//*   duplication or disclosure restricted by GSA ADP                *
//*   Schedule Contract with IBM Corp.                               *
//*                                                                  *
//********************************************************************
//* FABKDFL1:                                                        *
//*   Space Monitor Site Default Generation Utility Sample JCL       *
//*   (PARM='GEN')                                                   *
//*                                                                  *
//*   This is a Sample JCL for running Site Default Utility.         *
//*   It generates the site default table module Space Monitor.      *
//*                                                                  *
//*   This JCL consists of the following steps:                      *
//*     1) Generate the site default table source code               *
//*     2) Assemble the site default table                           *
//*     3) Link-Edit the site default table module                   *
//********************************************************************
//*
//*********************************************************************
//*  FABKTGEN - SPMN SITE DEFAULT GENARATION UTILITY
//*  ( PARM='GEN' SAMPLE PROCEDURE )
//*********************************************************************
//SPMNTGEN PROC HLQ='HPS'
//*--------------------------------------------------------------------
//*  CREATE SOURCE CODE OF SITE DEFAULT TABLE
//*--------------------------------------------------------------------
//G        EXEC PGM=FABKTGEN,PARM='GEN'
//STEPLIB  DD   DISP=SHR,DSN=&HLQ..SHPSLMD0
//SYSPUNCH DD   DISP=(NEW,PASS,DELETE),DSN=&&SOURCE,
//             DCB=(RECFM=FB,BLKSIZE=800),SPACE=(TRK,(1,1)),UNIT=SYSDA
//SYSPRINT DD   SYSOUT=*
//SYSUDUMP DD   DUMMY
```

Figure 213. Example JCL for creating FABKCTL0 (Part 1 of 2)

```
//*-------------------------------------------------------------------
//*  ASSEMBLE & LINK ==> SITE DEFAULT TABLE MODULE (FABKCTL0)
//*-------------------------------------------------------------------
//ASM      EXEC PGM=ASMA90,COND=(4,LT,G),
//             PARM='OBJECT,NODECK,LIST,XREF(SHORT)'
//SYSLIN   DD   DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(5,5,0)),
//             DCB=(BLKSIZE=400),DSN=&&OBJECT
//SYSUT1   DD   DISP=(,DELETE),UNIT=SYSDA,SPACE=(CYL,(10,5))
//SYSPUNCH DD   DUMMY
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DISP=(OLD,DELETE,DELETE),DSN=&&SOURCE
//*
//L        EXEC PGM=IEWL,COND=(4,LT,ASM),REGION=4096K,
//             PARM='LIST,REFR,REUS,AMODE=31,RMODE=ANY'
//SYSPRINT DD   SYSOUT=*
//SYSLIN   DD   DISP=(OLD,DELETE,DELETE),DSN=&OBJECT
//*
//         PEND
//*
//*------------------------------------------------------------------*
//*  FABKTGEN (PARM='GEN') - SPMN SITE DEFAULT GENARATION UTILITY    *
//*------------------------------------------------------------------*
//GO       EXEC SPMNTGEN,HLQ=HPS
//*------------------------------------*
//* SPECIFY SITE DEFAULT VALUES        *
//*------------------------------------*
//G.SPMNIN DD  *
%USER    user001 user002 user003 user004 user005 user006 user007
dbname   ddname   dsname                                       aabbb cc
-        pp q rrr s ttt uuu vvv www xxxx
/*
//L.SYSLMOD DD  DISP=SHR,DSN=HPS.TABLELIB(FABKCTL0)
//*
//
```

Figure 213. Example JCL for creating FABKCTL0 (Part 2 of 2)

Part 6. DB Segment Restructure

Chapter 30. Overview of DB Segment Restructure

This chapter gives an overview of DB Segment Restructure.

Topics:

Program functions

The DB Segment Restructure utility performs the following tasks:

- Initializing fields with specific values
- Moving data to different positions in a segment
- Combining segments
- Splitting segments

  Note: To split segments, user exit routines may also be used. For details, see Chapter 33, "User exit routines," on page 585.
- Deleting segments
- Creating a test database using part of a larger database
- Changing database hierarchy
- Converting an HDAM database to HIDAM
- Converting an HIDAM and HISAM databases to HDAM
- Converting an HSAM database to HISAM
- Converting HDAM, HIDAM, and HISAM databases to DEDB

Typical uses

DB Segment Restructure is used to restructure databases by unloading and then reloading them. DB Segment Restructure can also be used to convert databases between various database organizations.

Program structure

DB Segment Restructure is made up of two IMS application programs:

- FABRUNLD is the unload program. It creates a sequential data set to hold all or part of one or more IMS databases that you specify in either of the following two types of record formats:
  - HD unload record format (same as the one used in the IMS HD Reorganization Unload utility)
  - DB Segment Restructure unique unload record format
- FABRRELD is the reload program. It creates new IMS databases from the sequential data set that FABRUNLD created. FABRRELD can reload databases from the input data set created either in the HD unload record format or in the DB Segment Restructure unique unload record format.

Both programs are invoked in AMODE=31 via the DLIBATCH procedure supplied by IMS. The DLIBATCH procedure is explained in detail in *IMS Installation Volume 2: System Definition and Tailoring.*

However, FABRRELD must be executed under IMS BMP region when it reloads the databases to DEDB. (See Chapter 34, “Conversion to DEDB,” on page 591.)

Database changes are defined with standard control statements. Changes that cannot be made with control statements can sometimes be made with a user exit routine. (One example would be when you want to make changes that are a function of the segment data itself.)

Data flow

Figure 214 shows the data flow for the DB Segment Restructure programs. In addition to supplying input through your requested SYSIN data sets, you will also use PSBs and DBDs to control the operation of FABRUNLD and FABRRELD.

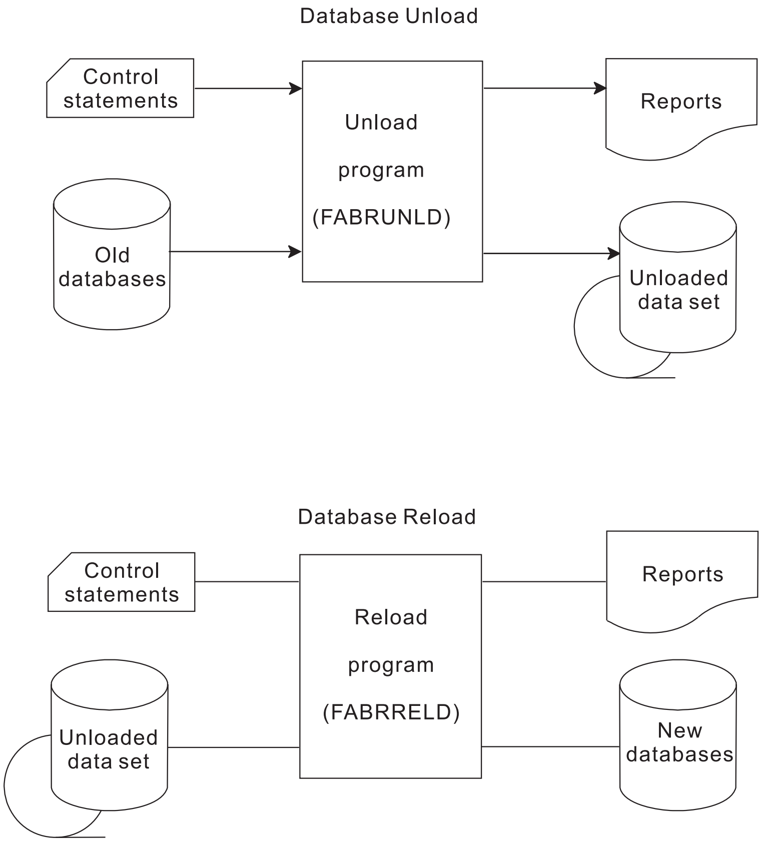


Figure 214. Data flow for DB Segment Restructure

Restrictions and considerations

DB Segment Restructure has the following restrictions:

1. DB Segment Restructure is an initial load program. Under some conditions, it can also be used as a database update program. However, DB Segment Restructure is *not* a general purpose reorganization program.

   The data set, which FABRUNLD has unloaded without specifying the HD record format option, cannot be used for the input data set to the IMS HD Reorganization Reload utility (DFSURGL0).
2. Logical relationships require special handling.

For the IMS Database Prefix Update utility (DFSURGP0) to work correctly in restructuring a database with logical relationships:

- DB Segment Restructure *must* be run as an initial load program.
- *All* logically related databases must be processed.

All logical pointers are maintained by IMS in behalf of DB Segment Restructure, in the manner which is functionally equivalent to the maintenance performed during the IMS HD Reorganization Reload utility (DFSURGL0).

3. When converting an HDAM, HIDAM, or HISAM database to DEDB, the DB Segment Restructure reload program FABRRELD must be executed under IMS BMP region. (See Chapter 34, "Conversion to DEDB," on page 591.)
4. When converting an HDAM, HIDAM, or HISAM database to DEDB, the all the segment in the database must be variable-length. (See Chapter 34, "Conversion to DEDB," on page 591.)
5. A DB Segment Restructure user exit routine is invoked in AMODE=31. (See Chapter 33, "User exit routines," on page 585.)
6. FABRRELD does not support the Data Capture exit routine when reloading a DL/I database. That is, any Data Capture exit routine which is defined to any segment of the DL/I database to be reloaded is not called during the run of FABRRELD.

   FABRRELD supports the Data Capture exit routine when reloading a DEDB database. That is, FABRRELD runs as a pure BMP region application program, and any Data Capture exit routine which is defined to any segment of the DEDB database is called under the control of the IMS DB Manager.
7. DB Segment Restructure supports only a HALDB without logical relationships. A HALDB indexed by Partitioned Secondary Indexes (PSINDEX) is supported as long as it has no logical relationships.
8. DB Segment Restructure unloads and reloads the entire HALDB database at once. Partition processing is not supported.
9. You must use the DB Segment Restructure unique (DBSR-unique) unload record format to process HALDB. The HD unloaded data set format is not supported.
10. Neither migration unload nor fallback unload for a HALDB is supported.

Considerations for HALDB Online Reorganization

1. If HALDB Online Reorganization (OLR) is not completed for the specified partition, DB Segment Restructure abnormally terminates due to the IMS batch region user abend 3303.
2. For the HALDB partition which is OLR capable, when FABRRELD runs as an initial load program (by using the PSB which contains PROCOPT=L or PROCOPT=LS on the PCB statement), it always loads the (A-J&X) side. When FABRRELD runs as an update program (by using the PSB which contains PROCOPT=A on the PCB statement), the update is on the active data sets, either (A-J&X) or (M-V&Y).

Considerations for data set compatibility among related utilities

DB Segment Restructure provides the following compatibilities for the unloaded data set with the related utilities:

- With the HD unload record format option specified, the DB Segment Restructure unload program (FABRUNLD) can unload the sequential database data set that will be reloaded by:
  - The DB Segment Restructure reload program (FABRRELD)

  - The IMS HD Reorganization Reload utility, or a functionally equivalent utility such as IMS High Performance Load
- The DB Segment Restructure reload program (FABRRELD) reloads the sequential database data set that was unloaded by:
  - The DB Segment Restructure unload program (FABRUNLD)
  - The IMS HD Reorganization Unload utility, or a functionally equivalent utility such as IMS High Performance Unload
- Physical Sequence Sort for Reload (PSSR) of IMS High Performance Load for z/OS sorts the sequential database data set that is unloaded by DB Segment Restructure with the HD unload record format option specified.
- The DB Segment Restructure programs (FABRUNLD and FABRRELD) do not support the PHD unloaded record format that is used for the unloaded data set to be reloaded into a HALDB. For HALDBs, use the DB Segment Restructure unique unload record format, and reload the data set by FABRRELD.

Note: For the specifications of the HD record format option, see “FABRUNLD SYSIN data set” on page 557.

Figure 215 shows the relations among the utilities:

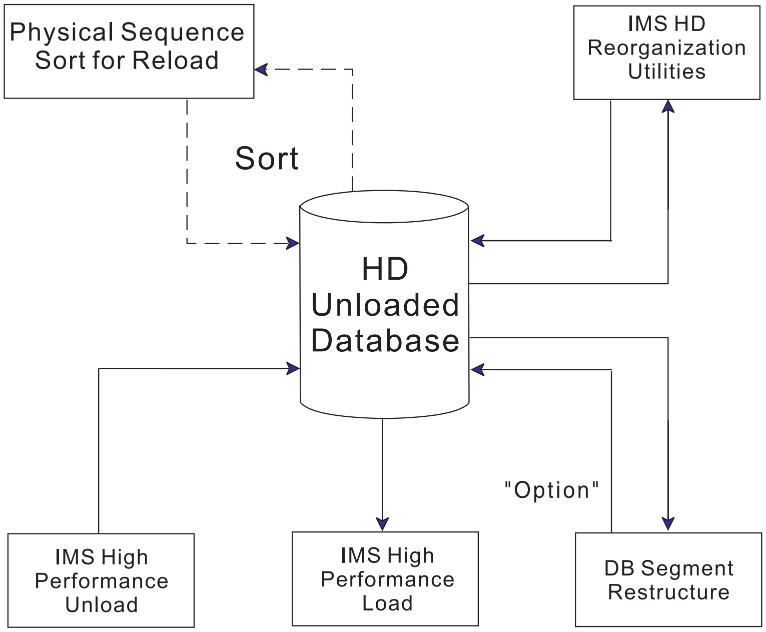

Physical Sequence Sort for Reload
IMS HD Reorganization Utilities
Sort
HD Unloaded Database
"Option"
IMS High Performance Unload
IMS High Performance Load
DB Segment Restructure

Figure 215. Compatibility of data sets

Chapter 31. Operating instructions for DB Segment Restructure

In order to use DB Segment Restructure, you must also use the IMS utilities. The following information describes the use of IMS utilities:

- *IMS Administration Guide: Database Manager*
- *IMS Utilities Reference: Database Manager*

For the job steps and job control language (JCL) for converting databases to DEDB, refer to Chapter 34, "Conversion to DEDB," on page 591.

This chapter describes how to use the two DB Segment Restructure programs.

To run DB Segment Restructure, you must complete the following tasks:

- Select the correct IMS utilities to run with DB Segment Restructure.
- Code the JCL for the IMS utilities and for both DB Segment Restructure programs.
- Code the input data for the IMS utilities and both DB Segment Restructure programs.
- Run the DB Segment Restructure job.
- Interpret the output reports to verify that the process completed successfully.

Topics:

Preparing steps for DB segment restructuring

You must run several programs in order to use DB Segment Restructure. The programs can be run in a single job or in several jobs. The steps in a DB Segment Restructure job stream vary, depending on the changes that you are making, and on the logical relationships involved.

Chapter 32, "JCL examples for DB Segment Restructure," on page 567 shows you the job steps required for several typical changes. See also "Considerations for data set compatibility among related utilities" on page 549.

Below is a list of the job steps required in order for you to use DB Segment Restructure. A typical job stream may contain some or all of these steps.

1. **DFSUDMP0:** The IMS Database Image Copy utility creates a copy of each of the old databases. This step is usually included as a safeguard. *This step should be considered mandatory.* If a problem occurs during the DB Segment Restructure process, you will have a "backup" database copy.
2. **FABRUNLD:** The DB Segment Restructure unload program creates a sequential data set that contains the old unloaded databases. *This step is mandatory.*
3. **DFSURPR0:** The IMS Database Prereorganization utility creates the DFSURCDS control data set for use by the other IMS logical relationship resolution utilities. It also shows any databases that the IMS Database Scan utility would scan during a normal database reorganization.

Note: Do *not* run the IMS Database Scan utility (DFSURGS0) in a DB Segment Restructure job. If DFSURPR0 shows some databases that should be scanned, unload and reload those databases with the DB Segment Restructure programs. Databases that have logical relationships with the databases that FABRRELD is reloading must be reloaded with FABRRELD, too. (See "Restrictions and considerations" on page 548.)

4. **DBDGEN:** This IMS procedure creates the DBDs (database descriptions) for newly restructured databases. Assemble and link-edit the new DBDs.
5. **IDCAMS:** This procedure deletes the old database data sets and allocates the new database data sets. This step *must* be included.
6. **FABRRELD:** The DB Segment Restructure reload program creates new restructured databases. This step *must* be included.
7. **DFSURG10:** The IMS Database Prefix Resolution utility collects information generated on the work data sets DFSURCDS and DFSURWF1. It produces the DFSURWF3 output data set, which contains the prefix information needed to complete the logical relationships defined for the databases. It also produces the DFSURIDX output data set, which contains the information necessary to create secondary index databases.
8. **DFSURUL0:** The IMS HISAM Reorganization Unload utility formats the DFSURIDX index work data set so that the HISAM Reload utility can use it.
9. **DFSURRL0:** The IMS HISAM Reorganization Reload utility takes the reorganized output data sets from the HISAM Unload utility and creates secondary index databases.
10. **DFSURGP0:** The IMS Database Prefix Update utility takes the DFSURWF3 output from the Database Prefix Resolution utility and updates the prefix of each segment affected by the FABRRELD process.
11. **DFSUDMP0:** The IMS Database Image Copy utility creates a copy of each new database. This is the first backup of the new databases. *Like job step 1, this "safeguard" step should be considered mandatory.*

Note: In the job steps 1 and 11, you can also use the IMS Online Database Image Copy utility instead of the IMS Database Image Copy utility.

The job steps listed above applies to non-HALDBs. The job steps listed below applies to HALDB.

1. **DFSUDUMP0:** The IMS Database Image Copy utility creates a copy of each old database. This step is mandatory.
2. **FABRUNLD:** The DB Segment Restructure unload program creates a sequential data set that contains the old unloaded databases. For the unloaded data set, specify the DB Segment Restructure unique unload record format. This step is mandatory.
3. **DBDGEN:** The IMS procedure creates the DBDs for newly restructured databases. Assemble and link-edit the new DBDs.
4. **Partition Definition Utility:** The IMS ISPF-base utility defines partition definitions, if any partition definition change is needed.
5. **IDCAMS:** This procedure deletes the old database data sets and allocates the new ones. If the partitioned secondary indexes point to the new databases, this procedure must also delete the old indexes and allocate the new PSINDEXes. This step is mandatory.
6. **DFSURPR0:** The IMS Database Prereorganization utility performs required partition initialization for the newly defined partition definitions.

7. **FABRRELD:** The DB Segment Restructure reload program creates new restructured databases. This step is mandatory. If secondary indexes point to the new restructured databases, they are also created in this step.
8. **DFSUDUMP0:** The IMS Database Image Copy utility creates a copy of each new database. This is the first backup of the new databases. This step should be considered mandatory.

Note: The IMS Prefix Resolution utility, the IMS Prefix Update utility, the IMS HISAM Reorganization Unload utility, and the IMS HISAM Reorganization Reload utility are not used for HALDB.

Job control language

Both DB Segment Restructure programs, FABRUNLD and FABRRELD, run in an offline DL/I batch processing region, using PSB and DBD libraries. Use the DLIBATCH cataloged procedure to run the programs. For a description of the DLIBATCH procedure, see *IMS Installation Volume 2: System Definition and Tailoring*.

FABRUNLD JCL

To run FABRUNLD, you must add some additional DD statements and run the DLIBATCH procedure. The FABRUNLD JCL requirements are shown in Table 68.

Table 68. DD statements for FABRUNLD JCL

DDNAME	Use	Format	Need
IEFRDER	Not used	Dummy	Required
IEFRDER2	Not used	Dummy	Required
SYSPRINT	Output	LRECL=133	Required
DFSVSAMP	Input		Required
SYSIN	Input	LRECL=80	Required
dataout	Output		Required
database	Input		Required
RECONx DD	Input/Output		Optional for Non-HALDB Required for HALDB

EXEC

Code this statement as:

```
//     EXEC DLIBATCH,MBR=FABRUNLD,PSB=psbname
```

The PSB must have these characteristics:

- It is sensitive only to the segments to be unloaded.
- It contains PROCOPT=G on the PCB statement.
- It contains LANG=ASSEM (or LANG=COBOL) on the PSBGEN statement.

The user may optionally turn DBRC off by specifying DBRC=N, if it is allowed by the installation standards, or if the database data set is not registered with DBRC.

DBRC=Y is required for HALDB unload.

IEFRDER DD

Code the primary system log data set as DUMMY.

IEFRDER2 DD

Code the secondary system log data set as DUMMY.

SYSPRINT DD

This output data set contains the reports produced by FABRUNLD. BLKSIZE, if coded on the DD statement, must be a multiple of 133.

DFSVSAMP DD

This input data set contains the buffer information that the DL/I buffer handler uses.

SYSIN DD
This input data set contains your description of the processing to be done by FABRUNLD. It describes the data to be unloaded. BLKSIZE, if coded on the DD statement, must be a multiple of 80.

dataout DD
This output data set is a sequential data set containing an unloaded database. There is one of these DD statements for each database being unloaded. You can use any ddname for this data set.

DCB information, if coded on the DD statement, should include *both* LRECL and BLKSIZE. LRECL must be at least 16 bytes larger than the longest segment in your database if the HD unload format option is not specified. When the HD unload format option is specified, LRECL must be at least 39 bytes larger than the longest segment. BLKSIZE must be at least 4 bytes larger than LRECL.

FABRUNLD uses the following block size for this data set if block size is not specified on the dataout DD statement:

- For 3380, the default block size is 23K.
- For 3390, the default block size is 28K.
- For 9345, the default block size is 22K.
- For all the other devices, the default is the maximum device capacity.

database DD
This input data set is an IMS database data set. There is one of these DD statements for each database data set being unloaded. Use the ddname specified in the DBD.

RECON1 DD
Defines the first Data Base Recovery Control (DBRC) RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

Note: If you use dynamic allocation, these RECON data set ddnames should not be used.

FABRRELD JCL

To run FABRRELD, you must provide some additional DD statements and run the DLIBATCH procedure. However, when FABRRELD reloads databases to DEDB, it must be executed under BMP region. The FABRRELD JCL requirements are as shown in Table 69.

Table 69. DD statements for FABRRELD JCL

DDNAME	Use	Format	Need
IEFRDER	Not used	Dummy	Required
IEFRDER2	Not used	Dummy	Required
SYSPRINT	Output	LRECL=133	Required
DFSVSAMP	Input		Required
SYSIN	Input	LRECL=80	Required
USEREXIT	Input		Optional

Table 69. DD statements for FABRRELD JCL (continued)

DDNAME	Use	Format	Need
datain	Input		Required
database	Output		Required
RECONx DD	Input/Output		Optional for Non-HALDB Required for HALDB

EXEC

To run FABRRELD via the DLIBATCH procedure, code this statement as:

```
//     EXEC DLIBATCH,MBR=FABRRELD,PSB=psbname
```

To reload databases to DEDB under BMP region, code this statement as:

```
//     EXEC IMSBATCH,MBR=FABRRELD,PSB=psbname
```

The PSB must have these characteristics:

- It contains PROCOPT=L or PROCOPT=LS (on the PCB statement) if you are using FABRRELD as an initial load program.
- It contains PROCOPT=A (on the PCB statement) if you are using FABRRELD as an update program.
- It contains LANG=ASSEM (or LANG=COBOL) on the PSBGEN statement.

The user may optionally turn DBRC off by specifying DBRC=N, if it is allowed by the installation standards, or if the database data set is not registered with DBRC.

DBRC=Y is required for HALDB unload.

When you load a HALDB indexed by PSINDEXes, the index pointer segments related to the indexed segment are also loaded into the PSINDEXes at the same time even if you are using PROCOPT=L or LS for the load program run.

IEFRDER DD

Code the primary system log data set as DUMMY. This DD statement is not necessary when reloading databases to DEDB.

IEFRDER2 DD

Code the secondary system log data set as DUMMY. This DD statement is not necessary when reloading databases to DEDB.

SYSPRINT DD

This output data set contains the reports produced by FABRRELD. BLKSIZE, if coded on the DD statement, must be a multiple of 133.

DFSVSAMP DD

This input data set contains the buffer information that the DL/I buffer handler uses. This DD statement is not necessary when reloading databases to DEDB.

SYSIN DD

This input data set contains your description of the processing to be done by FABRRELD. It describes the data to be reloaded. BLKSIZE, if coded on the DD statement, must be a multiple of 80.

USEREXIT DD

This input-partitioned data set contains the user exit routine load modules.

datain DD
This input data set is a sequential data set containing an unloaded database. There is one of these DD statements for each database to be reloaded. You can use any ddname for this data set.

database DD
This output data set is an IMS database data set. There is one of these DD statements for each database data set to be reloaded. Use the ddname specified in the DBD.

RECON1 DD
Defines the first Data Base Recovery Control (DBRC) RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

Note: If you use dynamic allocation, these RECON data set ddnames should not be used.

Input

Both programs of DB Segment Restructure use a data set of control statements as input. This section describes those data sets.

FABRUNLD SYSIN data set

This section describes the FABRUNLD SYSIN data set.

Function

A single FABRUNLD step can unload one or more databases. The FABRUNLD SYSIN data set specifies which databases are to be unloaded. It also specifies the subset to be extracted from each database.

The user can specify either of the following record formats for unloaded databases:

- HD unload record format (same as the one used in the IMS HD Reorganization Unload utility)
- DB Segment Restructure unique unload record format.

Format

This data set contains 80-byte fixed-length records, and block size must be a multiple of 80. The data set can be a sequential data set or a member of a partitioned data set.

Control statements

This data set has two types of control statements: a primary request control statement and a continuation request control statement.

Important note: *All* fields of control statements must be left-justified.

Primary request control statement: There must be one primary request control statement for each database to be unloaded by FABRUNLD.

```
0.........1.........2.........3.........4.........5.....//....7.........8
01234567890123456789012345678901234567890123456789012345//678901234567890

                    *ssa------------------------------------------------>
 dbdname   dataout HCssa------------------------------------------------>
                     ssa                           'kfb'
```

Position
Description

1
This required field is the left-justified name of the DBD for the database to be unloaded.

11
This required field is the left-justified ddname of the sequential data set to contain the unloaded database. Use any ddname you like.

19
This field is a control byte used for the HD record format option as follows:

H
FABRUNLD unloads the databases to the data sets in the HD unload record format. This record format is compatible with the one of the IMS HD Reorganization Unload utility.

Blank field
FABRUNLD unloads the databases to the data sets in the DB Segment Restructure unique unload record format. This record format is compatible with the one of the IMS DBT Version 1 DB Segment Restructure utility.

20
This field is a control byte that may contain one of these three codes:

C
The key feedback compare string is specified in a continuation request control statement (which immediately follows the primary request control statement). The segment search argument (SSA) may extend beyond column 50.

*
There is no key feedback compare string. The SSA may extend beyond column 50.

Blank field
Both the SSA and the key feedback compare string (if any) are on the primary request control statement.

21
This field contains a left-justified SSA used in an initial get unique call to begin the unload process. If the control byte is blank, this field is contained in columns 21-50. If the control byte is **C** or *, this field is contained in columns 21-80.

If the field is blank, the unload process begins with the first segment in the database.

51
This field must be enclosed in single quotes. It contains a left-justified character string that determines when the unload process ends. The string is compared to the key feedback returned after each database call. If the key feedback data is greater (with a higher collating sequence) than the string, the unload process stops for that database.

If the field is blank and the control byte is not **C**, the unload process continues until reaching the end of the database.

Continuation request control statement: One continuation request control statement may immediately follow each primary request control statement. Use this if you have a **C** in column 20 of your primary request control statement.

```
0.........1.........2.........3.........4.........5.........6.........7.........8
012345678901234567890123456789012345678901234567890123456789012345678901234567890

 'kfb'                                  comments
```

Position **Description**

1 *This field must be enclosed in single quotes*. It contains a left-justified character string that determines when the unload process ends. The string is compared to the key feedback returned after each database call. If the key feedback data is greater (with a higher collating sequence) than the string, the unload process ends for that database.

Use this control statement only when the control byte is **C**.

41 FABRUNLD does not use this field. You can use it for comments.

FABRRELD SYSIN data set

This section describes the FABRRELD SYSIN data set.

Function

A single FABRRELD step can reload several databases. The FABRRELD SYSIN data set describes all of the databases to be reloaded. It specifies segments to be restructured, and describes the restructuring to be performed.

Format

This data set contains 80-byte fixed-length records, and block size must be a multiple of 80. The data set can be a sequential data set or a member of a partitioned data set.

Control statements

This data set has three types of control statements: a database request (DB) control statement, a segment request (SEG) control statement, and a change request (CHG) control statement. There must be one DB control statement for each database being reloaded. There must be a SEG control statement for each segment type requiring change. Describe each change with a CHG control statement.

Each DB control statement is followed by any SEG control statement that apply to the database the DB control statement describes. Each SEG control statement is followed by any CHG control statements that apply to the segment that the SEG control statement describes.

Important Note: *All* control statement fields must be left-justified.

DB control statement: A DB control statement defines the database to be reloaded. It also defines the data set containing the unloaded database.

```
0.........1.........2.........3.........4.........5.........6.........7.........8
012345678901234567890123456789012345678901234567890123456789012345678901234567890

 DB       dbdname   datain    comments
```

Position
Description

1
Code **DB** in the first two bytes to identify this as a DB control statement. This field is required.

11
This left-justified field is the name of the DBD for the database to be reloaded. It is required.

21
This left-justified field is the ddname of the sequential data set containing the unloaded database. Use any ddname you like. This field is required.

31
FABRRELD does not use this field. This optional field can be used for comments.

SEG control statement: A SEG control statement defines segments that need changes before being loaded into an output database. You may specify up to 50 SEG control statements for each DB control statement.

```
0.........1.........2.........3.........4.........5.........6.........7.........8
012345678901234567890123456789012345678901234567890123456789012345678901234567890

 SEG       oldname   newname   userexit  comments
                               CANCEL
```

Position
Description

1
Code **SEG** in the first three bytes to identify this as a SEG control statement. There must be a SEG control statement for each segment type that requires change before the segment type can be loaded into the new database. This field is required.

11
This required, left-justified field is the name of the segment in the old DBD.

21
This is the left-justified name of the segment in the new DBD. If this field is blank, the segment name in the old DBD (column 11) is used.

31
This field will contain one of the following codes:

userexit
This code is the left-justified name of the user exit routine that gets control before inserting each occurrence of this segment type.

Blank field
Leave the field blank if a user exit routine is not used for this segment type.

CANCEL
Use this code if this segment type is not to be reloaded into the new database.

41
FABRRELD does not use this field. You can use it for comments.

CHG control statement: In addition to the changes performed in a user exit routine, there are two types of standard changes that FABRRELD can do:

- Move data from the old segment to different locations within the new segment
- Move literal data to the new segment

Use one CHG control statement for each standard change made to a segment type. CHG control statements do not have to be in a special sequence, but they must immediately follow the affected SEG control statement. You may specify up to 150 CHG control statements for each DB control statement.

Output positions not altered by a CHG control statement contain the corresponding data from the old segment.

```
0.........1.........2.........3.........4.........5.........6.........7.........8
012345678901234567890123456789012345678901234567890123456789012345678901234567890

 CHG       outpos    inpos     length
                     X'xxxxxx...xx'
                     C'character literal...'
```

Position **Description**

1
Code **CHG** in the first three bytes to identify this as a CHG control statement. A CHG control statement is used for each change in the structure of the segment type of the previous SEG control statement. This field is required.

11
This required field is the position in the output segment where data is to be moved. To code this field, you must include five decimal digits. Use leading zeros if necessary. The smallest allowable value is **00001**. (This is the first byte in the data portion of the segment.)

21
This required field may contain either a literal or a five-digit number. It defines the data that is moved into the appropriate output segment.

5-digit number
This number is the position in the input segment from which data is moved. To code this field, you must include five decimal digits. The smallest allowable value is **00001**. This value represents the first byte in the data part of the segment.

literal
The field can also be a left-justified character or hex literal. It is moved into each appropriate output segment.

Enclose the literal in single quotes. It can use up to 60 bytes. FABRRELD computes the actual length of the literal. Do not specify input position or length. For each DB control statement, the total number of literal bytes cannot exceed 1500.

Hex literals are specified as **X'xxxx...'** with an even number of hex digits (0-9, A-F) between the single quotes.

Character literals are specified as either **C'char...'** or **'char...'**. If a quote is needed as part of a character literal, specify it as two consecutive quotes.

31
This entry defines the number of bytes to be moved from the input

segment to the output segment. Code the field with five decimal digits, using leading zeros, if necessary. **00001** is the smallest allowable value. Do *not* code this field when you are using a literal.

Output

DB Segment Restructure produces two types of output:
- Printed reports
- Sequential data sets containing unloaded databases.

FABRUNLD SYSPRINT data set

This section describes the FABRUNLD SYSPRINT data set.

Function

This data set contains the following two reports:
- Control Card Format report
- Control Cards and Messages report

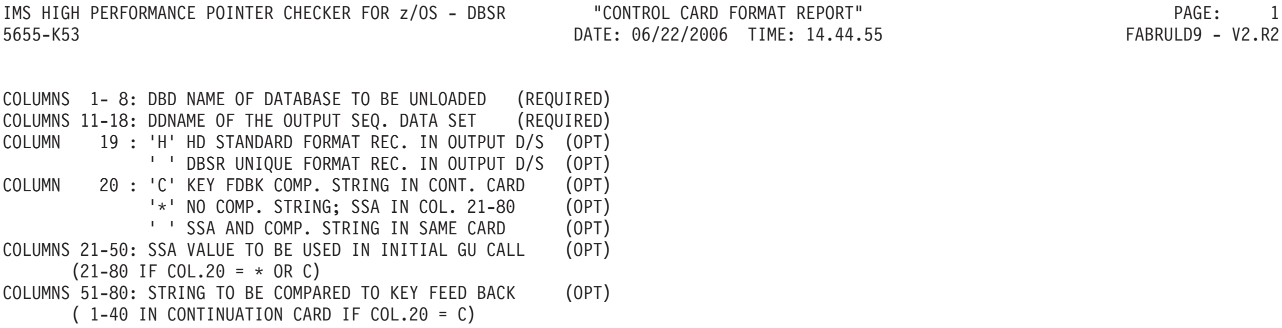
Format

The format is 133-byte fixed-length records. Block size, if coded in your JCL, must be a multiple of 133. The blocking factor is insignificant, because the data set usually contains only two printed pages. Code your DD statement:

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

Control Card Format report

This printed report (see Figure 216) describes the fields in the records contained in the FABRUNLD SYSIN data set.



```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBSR         "CONTROL CARD FORMAT REPORT"                              PAGE:     1
5655-K53                                                   DATE: 06/22/2006  TIME: 14.44.55                          FABRULD9 - V2.R2


COLUMNS  1- 8: DBD NAME OF DATABASE TO BE UNLOADED   (REQUIRED)
COLUMNS 11-18: DDNAME OF THE OUTPUT SEQ. DATA SET    (REQUIRED)
COLUMN    19 : 'H' HD STANDARD FORMAT REC. IN OUTPUT D/S  (OPT)
               ' ' DBSR UNIQUE FORMAT REC. IN OUTPUT D/S  (OPT)
COLUMN    20 : 'C' KEY FDBK COMP. STRING IN CONT. CARD    (OPT)
               '*' NO COMP. STRING; SSA IN COL. 21-80     (OPT)
               ' ' SSA AND COMP. STRING IN SAME CARD      (OPT)
COLUMNS 21-50: SSA VALUE TO BE USED IN INITIAL GU CALL    (OPT)
      (21-80 IF COL.20 = * OR C)
COLUMNS 51-80: STRING TO BE COMPARED TO KEY FEED BACK     (OPT)
       ( 1-40 IN CONTINUATION CARD IF COL.20 = C)
```

Figure 216. FABRUNLD SYSPRINT—Control Card Format report (Unload)

Control Cards and Messages report

This report shows:

- A printed copy of the complete FABRUNLD SYSIN data set
- The number of segments unloaded from each database
- Error messages produced by FABRUNLD

Figure 217 shows a sample of the Control Cards and Messages report (Unload).

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBSR    "CONTROL CARDS AND MESSAGES REPORT"                             PAGE:      1
5655-K53                                                  DATE: 06/22/2006  TIME: 14.44.55                          FABRULD9 - V2.R2

 <-------------------------------- CARD COLUMNS ---------------------------------> <------------------- MESSAGES -------------------->
 0........1.........2.........3.........4.........5.........6.........7.........8
 12345678901234567890123456789012345678901234567890123456789012345678901234567890

 HDAMDB1   UNLD1
                                                                                  FABR3524I           30 SEGMENTS UNLOADED
```

Figure 217. FABRUNLD SYSPRINT—Control Cards and Messages report (Unload)

FABRRELD SYSPRINT data set

This section describes the FABRRELD SYSPRINT data set.

Function

This data set contains the following two reports:

- Control Card Formats report
- Control Cards and Messages report

Format

The format is 133-byte fixed-length records. Block size, if coded in your JCL, must be a multiple of 133. The blocking factor is insignificant because the data set usually contains only two printed pages. Code your DD statement as follows:

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

Control Card Format report

This printed report (see Figure 218 on page 564) describes the fields in the records contained in the FABRRELD SYSIN data set.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBSR        "CONTROL CARD FORMAT REPORT"                            PAGE:     1
5655-K53                                                  DATE: 06/22/2006  TIME: 14.36.08                       FABRRLD8 - V2.R2


DB CARD FORMAT

COLUMNS  1- 2: DB SPECIFIES A DATABASE RELOAD       (REQUIRED)
COLUMNS 11-18: DBD NAME OF DATABASE TO BE RELOADED   (REQUIRED)
COLUMNS 21-28: DDNAME OF DATA SET CREATED BY UNLOAD  (REQUIRED)

SEG CARD FORMAT

COLUMNS  1- 3: SEG  SPECIFIES SEGMENT TO BE CHANGED  (REQUIRED)
COLUMNS 11-18: OLD SEGMENT NAME (IN OLD DATABASE)    (OPTIONAL)
COLUMNS 21-28: NEW SEGMENT NAME                      (OPTIONAL)
COLUMNS 31-38: USER EXIT NAME OR CANCEL              (OPTIONAL)

CHG CARD FORMAT

COLUMNS  1- 3: CHG  SPECIFIES CHANGE SEG FORMAT      (REQUIRED)
COLUMNS 11-15: POSITION IN OUTPUT SEG TO MOVE DATA   (REQUIRED)
COLUMNS 21-25: POSITION IN INPUT SEG TO OBTAIN DATA     (*)
COLUMNS 31-35: LENGTH OF DATA TO BE MOVED               (*)
     OR
COLUMNS 21-80: LITERAL TO BE MAPPED INTO OUTPUT POSITION
               SPECIFIED IN COLUMNS 11-15

  (*) SPECIFICATION OF LITERAL IN COLUMNS 21-80 REPLACES THE
      INFO REQUIRED IN COLUMNS 21-25 AND 31-35
```

Figure 218. FABRRELD SYSPRINT—Control Card Format report (Reload)

Control Cards and Messages report

This report shows:

- A printed copy of the complete FABRRELD SYSIN data set
- The number of segments reloaded into each database
- Error messages produced by FABRRELD

Figure 219 is an example of a Control Cards and Messages report.

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBSR    "CONTROL CARDS AND MESSAGES REPORT"                         PAGE:     1
5655-K53                                                  DATE: 06/22/2006  TIME: 14.36.08                       FABRRLD8 - V2.R2


 <-------------------------------- CARD COLUMNS --------------------------------> <------------------- MESSAGES ------------------->
 0........1.........2.........3.........4.........5.........6.........7.........8
 12345678901234567890123456789012345678901234567890123456789012345678901234567890


 DB        HISMDB1   RELD1
                                                                                  FABR3523I           6 SEGMENTS RELOADED
```

Figure 219. FABRRELD SYSPRINT—Control Cards and Messages report (Reload)

Unloaded database

This sequential data set contains the IMS database that was unloaded by FABRUNLD. FABRRELD uses the data set as input.

FABRUNLD unloads databases into a data set in the following two types of the record formats:

HD Unload record format

With the HD record format option specified on the primary request control statement as input for the FABRUNLD SYSIN data set, FABRUNLD unloads databases into a data set in the HD unload record format. This HD unload record format of the data set is the same as the record format of the data set unloaded by the IMS HD Reorganization Unload utility and its equivalent programs. This data set can be used as input for the IMS HD Reorganization Reload utility and its equivalent programs.

DB Segment Restructure Unique Unload record format

Without the HD record format option specification, FABRUNLD unloads databases into a data set in the DB Segment Restructure unique unload record format. This data set cannot be used as input for the IMS HD Reorganization Reload utility and its equivalent programs.

Chapter 32. JCL examples for DB Segment Restructure

There are many ways to use the DB Segment Restructure utility. These examples show some of the typical tasks that DB Segment Restructure can perform. By studying these examples and using them as models, you can use these same techniques to restructure your own databases.

Topics:

- “Example 1: How to restructure physical databases”
- “Example 2: How to restructure databases with logical relationships” on page 572
- “Example 3: How to change the hierarchy of a database” on page 578
- “Example 4: How to convert an HDAM database to HIDAM” on page 581
- “Example 5: How to split database segments” on page 582

Example 1: How to restructure physical databases

This example shows how to restructure physical databases. The two databases in the example do not have logical relationships. The example shows how to:

- Use the control statements for FABRUNLD and FABRRELD
- Copy data from one position to another within a segment
- Initialize fields with specific values
- Use user exit routines
- Delete all occurrences of a specific segment type.

Figure 220 and Figure 221 on page 568 describe a HIDAM database (ORDHIDD) that is unloaded. Substantial changes are made to four of its segment types during the reload step.

Figure 222 on page 568 describes the HDAM database (ORDHDAM) that is unloaded. Changes made by the reload step modify the key field in each root segment.

The IMS utilities that handle logical relationships do not need to be used in this example; these databases have no logical relationships. Follow these steps to restructure physical databases:

1. Create an image copy of the old database.
2. Unload the databases with FABRUNLD.
3. Create a new HIDAM DBD and a new load PSB.
4. Delete the old database clusters and define new ones.
5. Reload the databases with FABRRELD.
6. Create an image copy of the new databases.

```
DBD       NAME=ORDHIDX,ACCESS=(INDEX,VSAM,,DOSCOMP)
DATASET   DD1=KSDSINDX,DEVICE=3330
SEGM      NAME=ORDNDX,BYTES=10,FREQ=100
FIELD     NAME=(NDXKEY,SEQ),BYTES=10,START=1,TYPE=C
LCHILD    NAME=(ORDER,ORDHIDD),INDEX=ORDKEY
DBDGEN
FINISH
END
```

Figure 220. Example 1: HIDAM index DBD

```
DBD       NAME=ORDHIDD,ACCESS=(HIDAM,VSAM)
DATASET   DD1=ESDSDATA,DEVICE=3330,FRSPC=(6,25)
SEGM      NAME=ORDER,PARENT=0,BYTES=50,PTR=TB
FIELD     NAME=(ORDKEY,SEQ,U),BYTES=10,START=1,TYPE=C
LCHILD    NAME=(ORDNDX,ORDHIDX),PTR=INDX
FIELD     NAME=ORDATE,BYTES=6,START=41,TYPE=C
SEGM      NAME=ORDART,PARENT=((ORDER,SNGL)),BYTES=75,             X
           PTR=T,FREQ=3.5
FIELD     NAME=(ARTKEY,SEQ,M),BYTES=8,START=1,TYPE=C
SEGM      NAME=DELIVER,PARENT=((ORDART,SNGL)),BYTES=50,           X
           PTR=T,FREQ=0.7
FIELD     NAME=(DELDAT,SEQ),BYTES=6,START=1,TYPE=C
SEGM      NAME=SCHEDULE,PARENT=((ORDART,SNGL)),BYTES=50,          X
           PTR=T,FREQ=0.25
FIELD     NAME=(SCHEDAT,SEQ),BYTES=6,START=1,TYPE=C
SEGM      NAME=HISTORY,PARENT=((ORDART,SNGL)),BYTES=50,           X
           PTR=T,FREQ=0.3
DBDGEN
FINISH
END
```

Figure 221. Example 1: HIDAM DBD

```
DBD       NAME=ORDHDAM,ACCESS=(HDAM,VSAM),                        X
           RMNAME=(DFSHDC40,2,23,2048)
DATASET   DD1=DATAESDS,DEVICE=3330
SEGM      NAME=REDRO,PARENT=0,BYTES=50,PTR=TB
FIELD     NAME=(ORDKEY,SEQ,U),BYTES=10,START=1,TYPE=C
FIELD     NAME=ORDATE,BYTES=6,START=41,TYPE=C
SEGM      NAME=TRADRO,PARENT=((REDRO,DBLE)),BYTES=75,             X
           PTR=TB,FREQ=3.5
FIELD     NAME=(ARTKEY,SEQ,M),BYTES=8,START=1,TYPE=C
SEGM      NAME=REVILED,PARENT=((TRADRO,DBLE)),BYTES=50,           X
           PTR=TB,FREQ=0.7
FIELD     NAME=(DELDAT,SEQ),BYTES=6,START=1,TYPE=C
SEGM      NAME=ELUDEHCS,PARENT=((TRADRO,DBLE)),BYTES=50,          X
           PTR=TB,FREQ=0.25
FIELD     NAME=(SCHEDAT,SEQ),BYTES=6,START=1,TYPE=C
SEGM      NAME=YROTSIH,PARENT=((TRADRO,DBLE)),BYTES=50,           X
           PTR=TB,FREQ=0.3
DBDGEN
FINISH
END
```

Figure 222. Example 1: HDAM DBD

Creating an image copy of the old database

The standard image-copy job is run. This safeguard avoids the loss of data if an error is made.

Unloading the databases with FABRUNLD

This step unloads two databases.

Figure 223 shows a sample of unloading two databases.

```
//UNLOAD  EXEC DLIBATCH,MBR=FABRUNLD,PSB=ORDHIDAA,
//             DBRC=N,IRLM=N
//IEFRDER   DD DUMMY
//IEFRDER2  DD DUMMY
//DFSVSAMP  DD DSN=HPS.TEST.SOURCE(DFSVSAMP),DISP=SHR
//SYSPRINT  DD SYSOUT=A
//DATAESDS  DD DISP=OLD,DSN=HPS.ESDSHDAM
//ESDSDATA  DD DISP=OLD,DSN=HPS.HIDADATA
//KSDSINDX  DD DISP=OLD,DSN=HPS.HIDAINDX
//UNLD1     DD DSN=HPS.ORDHIDAM.SRU,DISP=(NEW,CATLG,DELETE),
//             UNIT=SYSDA,VOL=SER=TS0013,SPACE=(TRK,(10,10),RLSE),
//             DCB=(LRECL=13000,BLKSIZE=13004)
//UNLD2     DD DSN=HPS.ORDHIDAM.SRU,DISP=(MOD,CATLG,DELETE),
//             UNIT=SYSDA,VOL=SER=TS0013,
//             DCB=(LRECL=13000,BLKSIZE=13004)
//UNLD3     DD DSN=HPS.ORDHDAM.SRU,DISP=(NEW,CATLG,DELETE),
//             UNIT=SYSDA,VOL=SER=TS0013,SPACE=(TRK,(10,10),RLSE),
//             DCB=(LRECL=13000,BLKSIZE=13004)
//SYSIN     DD *
ORDHDAM   UNLD3   H
ORDHIDD   UNLD1                                 '0000000200'
ORDHIDD   UNLD2     ORDER   (ORDKEY   =0000000400)
/*
```

Figure 223. Example 1: FABRUNLD JCL

Database DD statements

A DD statement is required for each database that is being unloaded. In this example, the databases being unloaded are DATAESDS and ESDSDATA. A DD statement for the HIDAM index database KSDSINDX is also required.

Unloaded database DD statements

Each primary request control statement (in the SYSIN data set) requires a corresponding DD statement. The ddnames UNLD1, UNLD2, and UNLD3 are used here, but you can use any legal ddnames. UNLD1 and UNLD2 refer to the same data set. Note the use of "DISP=(MOD..." on the UNLD2 data set. This is necessary because the same database is being unloaded with two primary request control statements. Even though there are actually only two output data sets that contain unloaded databases, three ddnames are required for those data sets in the unload step.

Block size of 13004 is an example only. Use the block size you need. In this example, the UNLD1 data set and the UNLD2 data set are unloaded in the DB Segment Restructure unique unload record format, but the UNLD3 data set is unloaded in the HD unload record format.

SYSIN data set

The first control statement (in the SYSIN data set) causes the entire ORDHDAM database to be unloaded. This is the predefined format for a primary request control statement when you want to unload the complete database.

The second and third control statements unload the ORDHIDD database. The second control statement unloads all database records of which root segment key is less than or equal to 0000000200. The third control statement unloads all database records of which root segment key is greater than or equal to 0000000400. All database records except those with a root key between 0000000200 and 0000000400 are unloaded.

To initially load ORDHIDD, the segments must be inserted in hierarchical sequence (because it is an HIDAM database). The two control statements for ORDHIDD are in the order shown so as to be sure that the database to be unloaded is in hierarchical sequence.

Creating a new HIDAM DBD and a new load PSB

In this example, we create a new DBD for the ORDHIDD database and a new load PSB. Two changes are made to the DBD as follows:

- The DELIVER segment is changed from 50 to 70 bytes in length
- The HISTORY segment is removed

The new PSB is required because the new (restructured) database no longer contains a HISTORY segment.

Deleting the old database clusters and defining new ones

Three databases are deleted and then defined: ORDHDAM, ORDHIDD, and the HIDAM index for ORDHIDD.

Reloading the databases with FABRRELD

This step reloads two databases (see Figure 224 on page 571).

Database DD statements

There is a DD statement for each database (DATAESDS and ESDSDATA) to reload, and there is a DD statement for the HIDAM index database (KSDSINDX).

Unloaded database DD statements

Each DB control statement (in the SYSIN data set) requires a corresponding DD statement. Ddnames RELD1 and RELD2 are only for the example. Any legal ddnames can be used.

SYSIN data set

The first three records refer to the ORDHDAM database. The DB control statement loads the ORDHDAM database with data from the RELD1 unloaded database data set. The SEG control statement shows that all occurrences of the REDRO segment type are to be restructured. The CHG control statement replaces byte 1 with byte 8. Changes are not made to any other bytes. Byte 1 is part of the root key field, so this restructuring probably changes the hierarchical sequence of the root segments. This does not affect the reload process of an HDAM database.

The next ten control statements refer to the ORDHIDD database. The DB control statement loads the ORDHIDD database with data from the RELD2 unloaded database data set. The first SEG control statement shows that restructuring occurs for all occurrences of the ORDER segment type. The next two CHG control statements define that restructuring. Data that was originally in bytes 41-47 is moved to bytes 17-23. Data that was originally in bytes 1-10 is moved to bytes 29-38. Bytes 1-16, 24-28, and 38-50 contain their original data, because they do not have CHG control statements.

The next SEG control statement appears because the database to be unloaded contains HISTORY segments that do not need to be reloaded. "CANCEL" in the user exit field tells FABRRELD to skip all HISTORY records. (Another way to do this is to make the unload PSB (ORDHIDAA) insensitive to the HISTORY segment by omitting the SENSEG statement for the HISTORY statement. Since the unloaded database would not contain any history segments, this SEG control statement would be unnecessary.)

The next SEG control statement defines a user exit routine (EXITASM) that processes each SCHEDULE segment before reloading into the new database.

The last SEG control statement defines the restructuring for each DELIVER segment. Bytes 51-60 contain the character string "NEW FIELD." Bytes 61-64 contain binary zeros. Bytes 65-70 contain blanks. Since they are not mapped by a CHG control statement, bytes 1-50 contain their original data.

```
//RELOAD  EXEC DLIBATCH,MBR=FABRRELD,PSB=ORDHIDLA,
//             DBRC=N,IRLM=N
//IEFRDER   DD DUMMY
//IEFRDER2  DD DUMMY
//DFSVSAMP  DD DSN=USER.TEST.SOURCE(DFSVSAMP),DISP=SHR
//SYSPRINT  DD SYSOUT=A
//USEREXIT  DD DISP=SHR,DSN=HPS.LOAD
//DATAESDS  DD DISP=OLD,DSN=HPS.ESDSHDAM
//ESDSDATA  DD DISP=OLD,DSN=HPS.HIDADATA
//KSDSINDX  DD DISP=OLD,DSN=HPS.HIDAINDX
//RELD1     DD DISP=OLD,DSN=HPS.ORDHDAM.SRU
//RELD2     DD DISP=OLD,DSN=HPS.ORDHIDAM.SRU
//SYSIN     DD *
DB        ORDHDAM   RELD1
SEG       REDRO
CHG       00001     00008     00001     CHANGE THE KEY FIELD
DB        ORDHIDD   RELD2
SEG       ORDER                         MOVE DATA WITHIN SEGMENT
CHG       00017     00041     00007
CHG       00029     00001     00010
SEG       HISTORY             CANCEL    PREVENT LOAD OF THIS SEGMENT
SEG       SCHEDULE            EXITASM   USER EXIT ROUTINE
SEG       DELIVER                       ADD LITERAL DATA
CHG       00061     X'00000000'
CHG       00051     C'NEW FIELD '
CHG       00065     '      '
/*
```

Figure 224. Example 1: FABRRELD step

Creating an image copy of the new databases

The standard image-copy job is run. This is the first backup of the new databases.

Example 2: How to restructure databases with logical relationships

In this example, we restructure two logically related (unidirectional) databases. This example illustrates using the IMS logical relationship utilities with DB Segment Restructure.

Two HDAM databases (ORDHDAM1 and ORDHDAM2 - see Figure 225 and Figure 226 on page 573) are unloaded. The reload step changes the structure of the segments involved in the logical relationship.

Do the following steps:

1. Create an image copy of the old databases.
2. Run the IMS Database Prereorganization utility (DFSURPR0).
3. Unload the databases with FABRUNLD.
4. Delete the old database clusters and define new ones.
5. Reload the databases with FABRRELD.
6. Run the IMS Database Prefix Resolution utility (DFSURG10).
7. Run the IMS Database Prefix Update utility (DFSURGP0).
8. Create an image copy of the new databases.

```
DBD       NAME=ORDHDAM1,ACCESS=(HDAM,VSAM),                       X
           RMNAME=(DFSHDC40,2,50,450)
DATASET   DD1=ESDSDAT1,DEVICE=3350
SEGM      NAME=ORDER,PARENT=0,BYTES=50,PTR=TB
FIELD     NAME=(ORDKEY,SEQ,U),BYTES=10,START=1,TYPE=C
FIELD     NAME=ORDATE,BYTES=6,START=41,TYPE=C
SEGM      NAME=ORDART,PARENT=((ORDER,DBLE)),BYTES=75,             X
           PTR=TB,FREQ=3.5
FIELD     NAME=(ARTKEY,SEQ,M),BYTES=8,START=1,TYPE=C
LCHILD    NAME=(REVILED,ORDHDAM2),PTR=NONE
SEGM      NAME=DELIVER,PARENT=((ORDART,DBLE)),BYTES=50,           X
           PTR=TB,FREQ=1
FIELD     NAME=(DELDAT,SEQ),BYTES=6,START=1,TYPE=C
SEGM      NAME=SCHEDULE,PARENT=((ORDART,DBLE)),BYTES=50,          X
           PTR=TB,FREQ=1
FIELD     NAME=(SCHEDAT,SEQ),BYTES=6,START=1,TYPE=C
SEGM      NAME=HISTORY,PARENT=((ORDART,DBLE)),BYTES=50,           X
           PTR=TB,FREQ=1
DBDGEN
FINISH
END
```

Figure 225. Example 2: Logical parent HDAM DBD

```
DBD       NAME=ORDHDAM2,ACCESS=(HDAM,VSAM),                      X
           RMNAME=(DFSHDC40,2,23,2048)
DATASET   DD1=ESDSDAT2,DEVICE=3350
SEGM      NAME=REDRO,PARENT=0,BYTES=50,PTR=TB
FIELD     NAME=(ORDKEY,SEQ,U),BYTES=10,START=1,TYPE=C
FIELD     NAME=ORDATE,BYTES=6,START=41,TYPE=C
SEGM      NAME=TRADRO,PARENT=((REDRO,DBLE)),BYTES=75,            X
           PTR=TB,FREQ=3.5
FIELD     NAME=(ARTKEY,SEQ,M),BYTES=8,START=1,TYPE=C
SEGM      NAME=REVILED,                                          X
           PARENT=((TRADRO,DBLE),(ORDART,PHYSICAL,ORDHDAM1)),    X
           BYTES=50,PTR=(TB,LT,LP),FREQ=1
FIELD     NAME=(DELDAT,SEQ),BYTES=18,START=1,TYPE=C
SEGM      NAME=ELUDEHCS,PARENT=((TRADRO,DBLE)),BYTES=50,         X
           PTR=TB,FREQ=1
FIELD     NAME=(SCHEDAT,SEQ),BYTES=6,START=1,TYPE=C
SEGM      NAME=YROTSIH,PARENT=((TRADRO,DBLE)),BYTES=50,          X
           PTR=TB,FREQ=1
DBDGEN
FINISH
END
```

Figure 226. Example 2: Logical child HDAM DBD

Creating an image copy of the old databases

The standard image-copy job was run. This is the backup copy of the old databases.

Running the IMS Database Prereorganization utility (DFSURPR0)

Since the databases being unloaded and reloaded are connected by a logical relationship, this is a mandatory step. The Database Prereorganization utility produces the DFSURCDS control data set, which contains information about pointers that need to be resolved later (because of the logical relationship). The DFSURCDS control data set is used as input for:

- The FABRRELD initial load
- The IMS Database Prefix Resolution utility, after loading the databases.

The IMS Database Prereorganization utility also produces a list of any databases *not* initially loaded that have segments logically related to initially loaded databases. There should be *no* such databases. When DB Segment Restructure restructures a logically related database, all logically related databases *must* be processed by DB Segment Restructure.

Figure 227 on page 574 shows IMS Database Prereorganization JCL.

```
//PRERE   EXEC PGM=DFSRRC00,PARM='ULU,DFSURPR0'
//IMS       DD DISP=SHR,DSN=HPS.TEST.DBDLIB
//DFSRESLB  DD DISP=SHR,DSN=IMSVS.RESLIB
//SYSPRINT  DD SYSOUT=A,DCB=BLKSIZE=120
//IEFRDER   DD DUMMY
//DFSURCDS  DD DSN=&&PRERECDS,DISP=(NEW,PASS),
//             UNIT=SYSDA,SPACE=(CYL,(10,1)),
//             DCB=BLKSIZE=1600
//SYSIN     DD *
DBIL=ORDHDAM1
DBIL=ORDHDAM2
/*
```

Figure 227. Example 2: Prereorganization JCL

Unloading the databases with FABRUNLD

This step *unloads* two databases (see Figure 228).

```
//DBUNLOAD EXEC DLIBATCH,MBR=FABRUNLD,PSB=ORDHDAMA,
//             DBRC=N,IRLM=N
//IEFRDER   DD DUMMY,UNIT=SYSDA
//IEFRDER2  DD DUMMY,UNIT=SYSDA
//ESDSDAT1  DD DISP=OLD,DSN=HPS.ESDSHDM1
//ESDSDAT2  DD DISP=OLD,DSN=HPS.ESDSHDM2
//UNLD1     DD DSN=HPS.ESDSHDM1.RESTRUCT,DISP=(NEW,CATLG,DELETE),
//             UNIT=SYSDA,VOL=SER=TS0010,SPACE=(TRK,(10,10),RLSE),
//             DCB=(RECFM=VB,LRECL=1024,BLKSIZE=13004,DSORG=PS)
//UNLD2     DD DSN=HPS.ESDSHDM2.RESTRUCT,DISP=(NEW,CATLG,DELETE),
//             UNIT=SYSDA,VOL=SER=TS0010,SPACE=(TRK,(10,10),RLSE),
//             DCB=(RECFM=VB,LRECL=1024,BLKSIZE=13004,DSORG=PS)
//DFSVSAMP  DD DSN=HPS.TEST.SOURCE(DFSVSAMP),DISP=SHR
//SYSPRINT  DD SYSOUT=A
//SYSIN     DD *
ORDHDAM1  UNLD1
ORDHDAM2  UNLD2   H
/*
```

Figure 228. Example 2: FABRUNLD JCL

Database DD statements

A DD statement appears for each database to be unloaded (ESDSDAT1 and ESDSDAT2).

Unloaded database DD statements

Each primary request control statement (in the SYSIN data set) requires a corresponding DD statement. The ddnames UNLD1 and UNLD2 are used here, but you can use any legal ddnames.

Block size of 13004 is an example only. Use the block size you need.

In this example, the UNLD1 data set will be unloaded in the DB Segment Restructure unique unload record format and the UNLD2 data set will be unloaded in the HD unload record format.

SYSIN data set

The first control statement (in the SYSIN data set) unloads the complete ORDHDAM1 database. This is the predefined format for a primary request control statement when you want to unload the complete database. The second control statement unloads the complete ORDHDAM2 database.

Deleting the old database clusters and defining new ones

This step deletes and defines two databases: ORDHDAM1 and ORDHDAM2 (see Figure 229).

```
//ALLOCAT2 EXEC  PGM=IDCAMS
//DDESDS   DD  VOL=SER=TS0010,UNIT=3350,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//SYSIN DD *
 DELETE (HPS.ESDSHDM1)                CLUSTER  ERASE PURGE
 DELETE (HPS.ESDSHDM2)                CLUSTER  ERASE PURGE
 DEFINE  CLUSTER -
               (NAME(HPS.ESDSHDM1) -
                FILE(DDESDS) -
                VOL(TS0010) -
                NONINDEXED -
                RECSZ (2041,2041) -
                CISZ(2048)-
                CYL(2))
 DEFINE  CLUSTER -
               (NAME(HPS.ESDSHDM2) -
                FILE(DDESDS) -
                VOL(TS0010) -
                NONINDEXED -
                RECSZ (2041,2041) -
                CISZ(2048)-
                CYL(2))
/*
```

Figure 229. Example 2: IDCAMS JCL

Reloading the databases with FABRRELD

This step reloads two databases (see Figure 230 on page 576).

Database DD statements

A DD statement is required for each database to be reloaded (ESDSDAT1 and ESDSDAT2).

Unloaded database DD statements

Each DB control statement (in the SYSIN data set) requires a corresponding DD statement. Ddnames RELD1 and RELD2 are arbitrary. Any legal ddnames can be used.

SYSIN data set

The first three records refer to the ORDHDAM1 database. The DB control statement loads the ORDHDAM1 database with data from the RELD1 unloaded database data set. The SEG control statement restructures all occurrences of the ORDART segment type. The CHG control statement replaces bytes 9 through 25 with the character string "CHARACTER LITERAL." No other bytes are changed.

The next three records refer to the ORDHDAM2 database. The DB control statement loads the ORDHDAM2 database with data from the RELD2 unloaded database data set. The SEG control statement restructures all occurrences of the REVILED segment type. The CHG control statement replaces bytes 19 through 34 with the hexadecimal digits "000102030405060708090A0B0C0D0E0F." No other bytes are changed.

```
//DBREL   EXEC DLIBATCH,MBR=FABRRELD,PSB=ORDHDAML,
//             DBRC=N,IRLM=N
//IEFRDER   DD DUMMY,UNIT=SYSDA
//IEFRDER2  DD DUMMY,UNIT=SYSDA
//USEREXIT  DD DISP=SHR,DSN=USER.LOAD
//ESDSDAT1  DD DISP=OLD,DSN=HPS.ESDSHDM1
//ESDSDAT2  DD DISP=OLD,DSN=HPS.ESDSHDM2
//RELD1     DD DISP=OLD,DSN=HPS.ESDSHDM1.RESTRUCT
//RELD2     DD DISP=OLD,DSN=HPS.ESDSHDM2.RESTRUCT
//DFSURCDS  DD DSN=&&PRERECDS,DISP=(OLD,PASS)
//DFSURWF1  DD DSN=&&DBRELWF1,DISP=(NEW,PASS),
//             UNIT=SYSDA,SPACE=(CYL,(5,1)),
//             DCB=(RECFM=VB,LRECL=300,BLKSIZE=1200)
//DFSVSAMP  DD DSN=HPS.TEST.SOURCE(DFSVSAMP),DISP=SHR
//SYSPRINT  DD SYSOUT=A
//SYSIN     DD *
DB        ORDHDAM1  RELD1
SEG       ORDART    ORDART
CHG       0009      C'CHARACTER LITERAL'
DB        ORDHDAM2  RELD2
SEG       REVILED
CHG       0019      X'000102030405060708090A0B0C0D0E0F'
/*
```

Figure 230. Example 2: FABRRELD JCL

Running the IMS Database Prefix Resolution utility (DFSURG10)

The Database Prefix Resolution utility (see Figure 231) collects and sorts the information put on the DFSURWF1 work data set by the initial load step. The resulting DFSURWF3 output data set contains the sorted prefix information needed to resolve logical relationships.

```
//PREFRES EXEC PGM=DFSURG10
//SORTLIB   DD DISP=SHR,DSN=SYS1.SORTLIB
//SYSPRINT  DD SYSOUT=A,DCB=BLKSIZE=121
//SYSOUT    DD SYSOUT=A
//SORTWK01  DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//SORTWK02  DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//SORTWK03  DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//DFSURCDS  DD DSN=&&PRERECDS,DISP=(OLD,DELETE)
//DFSURWF2  DD UNIT=SYSDA,SPACE=(CYL,(5,1)),
//             DCB=(RECFM=VB,LRECL=300,BLKSIZE=1200)
//DFSURWF3  DD DSN=&&PRFRSWF3,DISP=(NEW,PASS),
//             UNIT=SYSDA,SPACE=(CYL,(5,1)),
//             DCB=(RECFM=VB,LRECL=300,BLKSIZE=1200)
//DFSURIDX  DD DSN=&&DFSURIDX,DISP=(NEW,PASS),
//             UNIT=SYSDA,SPACE=(CYL,(5,1)),
//             DCB=(RECFM=VB,LRECL=300,BLKSIZE=1200)
//SORTIN    DD DSN=&&DBRELWF1,DISP=(OLD,DELETE)
```

Figure 231. Example 2: Prefix Resolution JCL

Running the IMS Database Prefix Update utility (DFSURGP0)

The Database Prefix Update utility updates the prefix of each segment affected by initial loading of the databases (see Figure 232). It updates logical parent and logical twin forward pointer fields and counter fields. (Because the relationship is unidirectional, there are no logical child pointers.)

```
//PRFUPDT EXEC PGM=DFSRRC00,PARM='ULU,DFSURGP0'
//IMS       DD DISP=SHR,DSN=HPS.TEST.DBDLIB
//DFSRESLB  DD DISP=SHR,DSN=IMSVS.RESLIB
//DFSURWF3  DD DSN=&&PRFRSWF3,DISP=(OLD,DELETE)
//IEFRDER   DD DUMMY
//SYSPRINT  DD SYSOUT=A,DCB=BLKSIZE=120
//DFSVSAMP  DD DISP=SHR,DSN=HPS.TEST.SOURCE(DFSVSAMP)
//ESDSDAT1  DD DSN=HPS.ESDSHDM1,DISP=OLD
//ESDSDAT2  DD DSN=HPS.ESDSHDM2,DISP=OLD
```

Figure 232. Example 2: Prefix Update JCL

Creating an image copy of the new databases

The standard image-copy job is run. This is the first backup of the new database.

Example 3: How to change the hierarchy of a database

This example restructures an HDAM database. The example shows how to change the hierarchical structure of a database. Figure 233 and Figure 234 show the original and changed structures. The position in the hierarchy of the HISTORY segment changes. This causes all the level-3 dependent segments to have different segment codes in the new database.

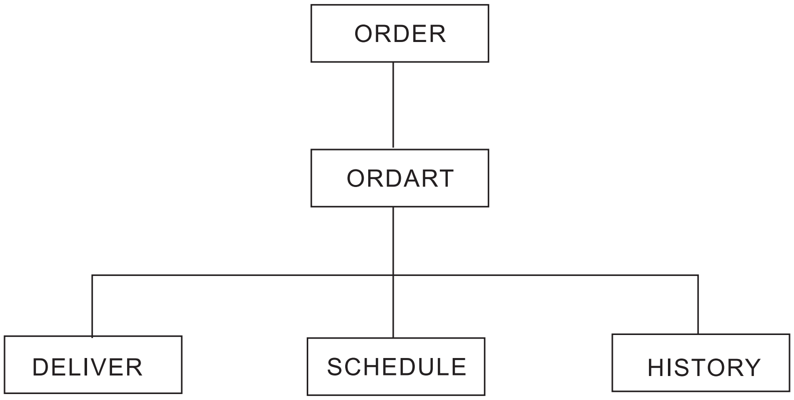


Figure 233. Original hierarchical structure

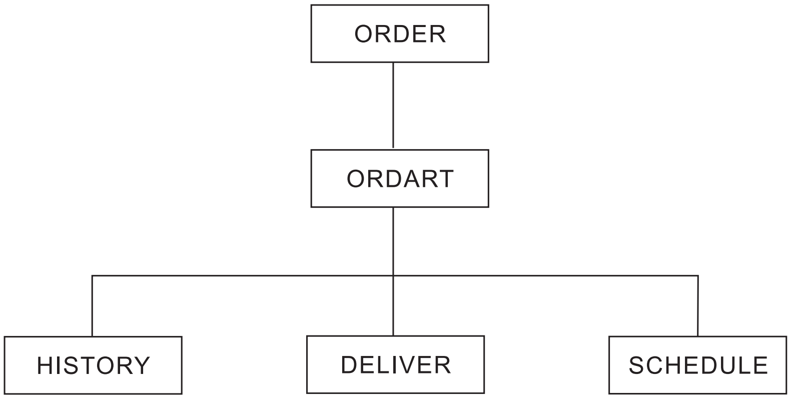


Figure 234. New hierarchical structure

Since there are no logical relationships in the database, there is no need to execute any of the IMS utilities that handle logical relationships. Do the following steps:

1. Create an image copy of the old database.
2. Unload the database with FABRUNLD.
3. Create a new DBD and new load and update PSBs.
4. Delete the old database cluster and define a new one.
5. Load the database with a dummy segment.
6. Update the database with FABRRELD.
7. Delete the dummy segment.
8. Create an image copy of the new database.

Creating an image copy of the old database

The standard image-copy job is run. This is the first backup of the new database.

Unloading the database with FABRUNLD

This step unloads one database (see Figure 235).

Database DD statement

A DD statement is required for the database to be unloaded (ESDSDATA).

Unloaded database DD statements

The primary request control statement (in the SYSIN data set) requires a corresponding DD statement. The ddname UNLD1 is arbitrary; any legal ddname can be used.

Since no DCB information is coded on the DD statement, LRECL and BLKSIZE default to the maximum limit allowable for the device.

SYSIN data set

The control statement (in the SYSIN data set) unloads the complete ORDHDAM database. This is the predefined format for a primary request control statement when you want to unload the complete database.

```
//UNLOAD  EXEC DLIBATCH,MBR=FABRUNLD,PSB=ORDHDAMA,
//             DBRC=N,IRLM=N
//IEFRDER   DD DUMMY
//IEFRDER2  DD DUMMY
//DFSVSAMP  DD DSN=HPS.TEST.SOURCE(DFSVSAMP),DISP=SHR
//SYSPRINT  DD SYSOUT=A
//ESDSDATA  DD DISP=OLD,DSN=HPS.ESDSHDAM
//UNLD1     DD DSN=HPS.ORDHDAM.SRU,DISP=(NEW,CATLG,DELETE),
//             UNIT=SYSDA,VOL=SER=TS0013,SPACE=(TRK,(10,10),RLSE)
//SYSIN     DD *
ORDHDAM   UNLD1
/*
```

Figure 235. Example 3: FABRUNLD step

Creating a new DBD and new load and update PSBs

To change the hierarchy of the database, we create a new DBD and PSBs. The change moves the HISTORY segment's SEGM statement and its corresponding FIELD statements. It is moved to immediately follow the ORDART segment.

Deleting the old database cluster and defining a new one

One database (ORDHDAM) is deleted and defined.

Loading the database with a dummy segment

Segments must be inserted in hierarchical sequence during an initial load. The unloaded database contains the segments in their original hierarchical sequence. Therefore, they are in the wrong order for an initial load of the new database because the new hierarchical sequence is different.

Segments can be inserted in any order by an update program. You can use FABRRELD as an update program, if your database was previously initially loaded. In this example, the database is initially loaded with a single "dummy" root segment. The key of this root segment is different from any of those in the original database.

An easy way to do this is to use the IMS Test Program (DFSDDLT0), using a load PSB (PROCOPT=L). (For more information, see *IMS Application Programming: Database Manager*.)

Updating the database with FABRRELD

One database is reloaded in this step (see Figure 236), using an update PSB (PROCOPT=A).

When "FIRST" or "HERE" operand of RULES= keyword is specified on the new DBD with no sequence field or with a nonunique sequence field, updating the database with FABRRELD will reverse the order of segment occurrences which is in the unloaded data set.

Database DD statement

The reloaded database (ESDSDATA) requires a DD statement.

Unloaded database DD statements

The DB control statement (in the SYSIN data set) requires a corresponding DD statement. The ddname of RELD1 is arbitrary; you can use any legal ddname.

SYSIN data set

The DB control statement loads the ORDHDAM database with data from the RELD1 unloaded database data set. There are no SEG or CHG statements, because there are no changes to the segment data.

```
//RELOAD  EXEC DLIBATCH,MBR=FABRRELD,PSB=ORDHDAMX,
//             DBRC=N,IRLM=N
//IEFRDER   DD DUMMY
//IEFRDER2  DD DUMMY
//DFSVSAMP  DD DSN=HPS.TEST.SOURCE(DFSVSAMP),DISP=SHR
//SYSPRINT  DD SYSOUT=A
//USEREXIT  DD DUMMY
//ESDSDATA  DD DISP=OLD,DSN=HPS.ESDSHDAM
//RELD1     DD DISP=OLD,DSN=HPS.ORDHDAM.SRU
//SYSIN     DD *
DB        ORDHDAM   RELD1
/*
```

Figure 236. Example 3: FABRRELD step

Deleting the dummy segment

The "dummy" root segment inserted during the initial load of the database is now deleted. An easy way to do this is to use the IMS Test Program (DFSDDLT0), using an update PSB (PROCOPT=A). (For more information, see *IMS Application Programming: Database Manager*.)

Creating an image copy of the new database

The standard image-copy job is run. This is the first backup of the new database.

Example 4: How to convert an HDAM database to HIDAM

In this example, an HDAM database is restructured and converted to a HIDAM database. When an HDAM database is converted to HIDAM, the hierarchical structure changes. (This is because HIDAM root segments are stored in key sequence, while HDAM root segments are stored in randomized sequence.)

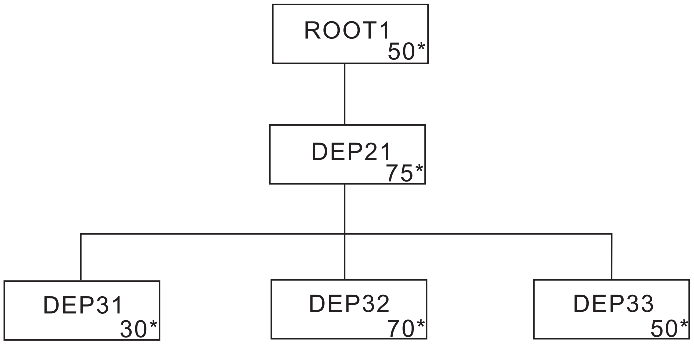
There are no logical relationships in the database, so IMS utilities for logical relationships do not have to be run. The following steps are required:

1. Create an image copy of the old database.
2. Unload the database with FABRUNLD.
3. Create a new DBD and new load and update PSBs.
4. Delete the old database cluster and define a new one.
5. Load the database with a dummy segment.
6. Update the database with FABRRELD.
7. Delete the dummy segment.
8. Create an image copy of the new database.

These steps are the same as those of "Example 3: How to change the hierarchy of a database" on page 578. All JCL and control statements for this example are the same as those for "Example 3: How to change the hierarchy of a database" on page 578. Refer to "Example 3: How to change the hierarchy of a database" on page 578 for instructions.

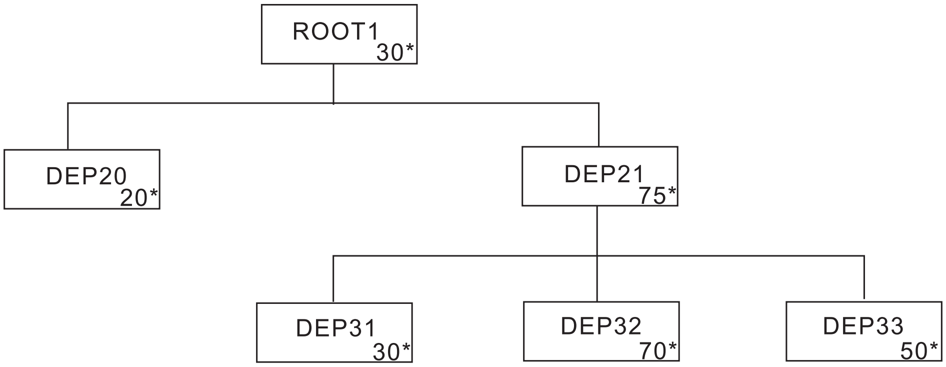
Example 5: How to split database segments

This example restructures an HDAM database. The example shows how to split database segments using the SEG and CHG control statements. Figure 237 and Figure 238 show a simplified example of the original structure and the split structure.



* : segment length

Figure 237. Original hierarchical structure (DBD1DB)



* : segment length

Figure 238. New hierarchical structure (DBD2DB)

In this case, the root segment (ROOT1) is split into 2 segments as follows:

```
ROOT1  30 Bytes,  Containing bytes 01 - 30 of the original ROOT1
DEP20  20 Bytes,  Containing bytes 31 - 50 of the original ROOT1
```

You must use the DBD1 to unload the database with FABRUNLD, and DBD2 to reload it with FABRRELD.

Since there are no logical relationships in the database, there is no need to execute any of the IMS utilities that handle logical relationships. Do the following steps:
1. Create an image copy of the old database.
2. Unload the database with FABRUNLD.
3. Create a new DBD and new load and update PSBs.
4. Delete the old database cluster and define a new one.

5. Load the database with a dummy segment.
6. Update the database with FABRRELD.
7. Delete the dummy segment.
8. Create an image copy of the new database.

Except for Step 3 and Step 6, these steps are the same as those in “Example 3: How to change the hierarchy of a database” on page 578. JCL and control statements for the steps are the same as those in “Example 3: How to change the hierarchy of a database” on page 578. Refer to “Example 3: How to change the hierarchy of a database” on page 578 for instructions. Instructions for Step 3 and Step 6 are described below.

Creating a new DBD and new load and update PSBs

To split the database segment, you have to create a new DBD (DBD2) and PSBs. Length of ROOT1 is changed to 30 bytes and the new segment DEP20 must have a 20-byte length following the ROOT1 segment.

Updating the database with FABRRELD

One database is reloaded in this step (see Figure 239 on page 584) using an update PSB (PROCOPT=A).

Database DD statement

The reloaded database (ESDSDATA) requires a DD statement.

Unloaded database DD statements

The DB control statement (in the SYSIN data set) requires a corresponding DD statement. Ddname of RELD1 is arbitrary; you can use any legal ddname.

SYSIN data set

The DB control statement loads the DBD2DB database with data from the RELD1 unloaded database data set. The first pair of SEG and CHG control statements is for the ROOT1 segments and the next pair for the DEP20 segment.

To split a segment by control statements, use multiple SEG control statements for the intended segment to be split. Each SEG control statement should be followed by its related CHG statement(s). The SEG and CHG control statements for all the split parts (new segments) must follow each other. Also, you have to make sure that you do not violate IMS rules (that is, try to insert a dependent segment prior to its parent being inserted).

```
//RELOAD  EXEC DLIBATCH,MBR=FABRRELD,PSB=DBD2DBX,
//             DBRC=N,IRLM=N
//IEFRDER   DD DUMMY
//IEFRDER2  DD DUMMY
//DFSVSAMP  DD DSN=HPS.TEST.SOURCE(DFSVSAMP),DISP=SHR
//SYSPRINT  DD SYSOUT=A
//USEREXIT  DD DUMMY
//ESDSDATA  DD DISP=OLD,DSN=HPS.ESDSHDAM
//RELD1     DD DISP=OLD,DSN=HPS.DBD2DB.SRU
//SYSIN     DD *
DB        DBD2      RELD1
SEG       ROOT1
CHG       00001     00001     00030
SEG       ROOT1     DEP20
CHG       00001     00031     00020
/*
```

Figure 239. Example 5: FABRRELD step

Another way to split a segment using DB Segment Restructure is to use a user exit routine. For more information on coding user exit routines, see Chapter 33, “User exit routines,” on page 585.

Chapter 33. User exit routines

This chapter describes product-sensitive programming interface information. See "Programming interface information" on page 732 to understand the restrictions associated with this type of material.

Product-Sensitive Programming Interface

You can use user exit routines with the DB Segment Restructure reload program FABRRELD for data type conversion, parallel processing of another user data set, data-dependent structure changes, and any algorithm change that cannot be performed via a standard CHG. Each segment type named on a "SEG" record can have one exit routine.

End of Product-Sensitive Programming Interface

Topics:

- "Techniques"
- "Interface" on page 586

Techniques

Product-Sensitive Programming Interface

When FABRRELD begins execution, each exit routine is loaded. After being loaded once, the same copy of the exit routine is invoked one time for each occurrence of its segment type. Because of this, an exit routine's data areas are maintained throughout the execution of FABRRELD. This allows the exit routine to set switches on one call and to use those switches on another call. The exit routine does not have to be re-enterable.

If more than one exit routine is being used, each must have its own unique name. To use the same routine for more than one segment type, copy the original exit routine and change the CSECT name in the copy.

Before each segment is written to the new database, FABRRELD calls the exit routine. It also calls the exit routine, once when the reload is completed, to let the exit routine do whatever cleanup is necessary. For example, all DCBs opened by the exit routine must be closed.

A FABRRELD exit routine can be coded in Assembler, COBOL, or PL/I. The FABRRELD exit routine is invoked in AMODE=31.

End of Product-Sensitive Programming Interface

Interface

This section describes the interface of the user exit routines.

Registers and input parameter lists

Product-Sensitive Programming Interface

Registers must be saved when the exit routine is entered. There is a save area address in register 13 for this purpose. The exit routine must observe standard MVS conventions.

DB Segment Restructure sets these registers before calling the exit routine:

Register	Meaning or Content
R1	Address of input parameter list
R13	Save area address
R14	Return to FABRRELD address
R15	Entry point address of the exit routine.

On return to FABRRELD, restore registers 0 through 14 to their original contents. Register 15 must contain one of the following return codes:

Value	Meaning
0	Insert the segment into the new database.
4	Do not insert the segment into the new database.
8	End processing of this database.

Input is passed to the exit routine via a list of addresses. Figure 240 shows the list of addresses for a standard call (before writing a segment). Figure 241 shows the list of addresses for the final call (when the reload is complete).

	0	1	2	3
0	Address of database PCB			
4	Address of new segment name			
8	Address of old segment			
12	1	Address of new segment		

Figure 240. Input parameter list for standard call

	0	1	2	3
0	1	Address of database PCB		

Figure 241. Input parameter list for final call

End of Product-Sensitive Programming Interface

FABRRELD to PL/I exit routine interface

Product-Sensitive Programming Interface

The following assembler routine is offered as a sample interface routine to be used in conjunction with a PL/I exit routine. When using PL/I, a special environment is established at each invocation unless the PL/I environment is maintained by the invoking program. The function of this interface routine is to maintain that PL/I environment. Furthermore, FABRRELD expects a return code in register 15 to indicate disposition of the segment just processed by the exit routine. PL/I has no facility to set a return code in register 15 except at the termination of the PL/I environment which in this case is the end of the job. Therefore, this interface provides the return code from PL/I as set by the PL/I built-in function "PLIRETC."

The following rules must be observed when using this interface:

1. The PL/I routine must be compiled as an external procedure (no "OPTIONS" on the procedure statement).
2. The name on the procedure statement must be "RELDEXIT."
3. The parameters in the PL/I routine must be declared as pointer variables.
4. Based variables should be used to access the data passed by FABRRELD to the exit routine.
5. The built-in function "PLIRETC" should be used to set the return code expected by FABRRELD.
6. The interface routine must be link-edited with the exit routine using the following link-edit control statement:

```
INCLUDE exitlib(RELDEXIT)     RELDEXIT - NORMALLY ON SYSLIN
INCLUDE somelib(RLDTOPLI)     INTERFACE ROUTINE
ENTRY RLDTOPLI                ENTRY POINT IN INTERFACE
NAME whatever                 ANY NAME THE USER CHOOSES
```

7. The module name (whatever) is the exit routine name specified in the PSB control statement.

A sample PL/I exit routine follows the interface routine.

Figure 242 on page 588 shows FABRRELD to PL/I exit routine interface.

```
RLDTOPLI  TITILE 'FABRRELD TO PL/I EXIT ROUTINE INTERFACE'
RLDTOPLI  CSECT
R0        EQU    0
R1        EQU    1
R2        EQU    2
R3        EQU    3
R4        EQU    4
R5        EQU    5
R6        EQU    6
R7        EQU    7
R8        EQU    8
R9        EQU    9
R10       EQU    10
R11       EQU    11
R12       EQU    12
R13       EQU    13
R14       EQU    14
R15       EQU    15
          EJECT
          STM    R14,R12,12(R13)       SAVE FABRRELD REGISTERS
          LR     R11,R15               SET UP BASE REG
          USING  RLDTOPLI,R11
          LA     R2,SAVE1              POINT TO NEW SAVEAREA
          ST     R2,8(R13)             SET FORWARD CHAIN
          ST     R13,SAVE1+4           SET BACKWARD CHAIN
          LR     R13,R2                POINT TO NEW SAVEAREA
          GET    FIRSTSW,X'00'         IS THIS FIRST TIME THRU?
          BE     FIRST                 YSE, BR
          L      R13,SAVE2+4           RESTORE PLICALLA REGISTERS
          LM     R14,R12,12(R13)       SAVE BY RLDMAIN
          BR     R15                   AND REENTER RLDMAIN BYPASSING
*                                      THE PL/I INITIALIZATION
          SPACE  2
FIRST     MVI    FIRSTSW,X'FF'         SET FIRST TIME SWITCH
          L      R2,=V(PLIMAIN)        IN SURE CSECT IS INVOKED FROM
          LA     R3,RLDMAIN            PLICALLA
          ST     R3,0(R2)
          L      R15,=V(PLICALLA)      GO TO PL/I TO SET UP PL/I ENV
          BALR   R14,R15
          ABEND  500,DUMP              CONTROL SHOULD NEVER RETURN
          TITLE  'PLIMAIN ROUTINE'
          DS     0D
          DC     C'RLDMAIN',AL1(7)
          ENTRY  RLDMAIN
          SPACE
```

Figure 242. FABRRELD to PL/I exit routine interface (Part 1 of 2)

```
RLDMAIN   DS     0H
          STM    R14,R12,12(R13)       SAVE PL/I REGISTERS
          LR     R2,R15                SET UP BASE REG
          USING  RLDMAIN,R2
          LA     R3,SAVE2              GET ADDRESS OF SAVEAREA
          ST     R3,8(R13)             SET FORWARD CHAIN
          ST     R13,SAVE2+4           SET BACKWARD CHAIN
          MVC    SAVE2+72(8),72(R13)   COPY THE PL/I SLOTS
          LR     R13,R3                SET UP POINTER TO OUR SA
          SPACE
*            CREATE INTERMEDIATE PARM LIST TO SIMULATE PL/I
*            INVOCATION FOR POINTER PARAMETER
          SPACE
          LA     R4,0                  ZERO REG USED TO INDEX MYPARMS
          LA     R5,6                  SET LOOP CONTROL - MAX 6 PARMS
STORLOOP  ST     R1,MYPARMS(R4)        STORE ADDR OF NTH PARM
          TM     0(R1),X'80'           LAST PARM?
          BO     ENDPARMS              YES, BR
          LA     R1,4(R1)              BUMP TO NEXT PARM
          LA     R4,4(R4)              BUMP TO NEXT SLOT IN MYPARMS
          BCT    R5,STORLOOP
          SPACE
          WTO    'MORE THAN 4 PARMS PASSED TO RLDTOPLI'
          ABEND  600,DUMP
          SPACE
ENDPARMS  DS     0H
          LA     R6,MYPARMS(R4)        GET ADDR OF LAST PARM IN LIST
          OI     0(R6),X'80'           INDICATE LAST PARM IN LIST
          LA     R1,MYPARMS            SET R1 -> NEW PARM LIST
          SR     R5,R5                 CLEAR R5 FOR PL/I INVOCATION
          SPACE
          L      R15,=V(RELDEXIT)
          BALR   R14,R15               INVOKE THE PL/I EXIT SUBROUTINE
          SPACE
          L      R13,SAVE1+4           GET ADDR OF FSU II SAVE AREA
          L      R14,12(R13)           RESTORE R14
          LH     R15,X'46'(R12)        PLACE RETURN CODE CREATED IN
*                                      PL/I TCA+X'46' BY THE PLIRETC
*                                      BUILT-IN FUNCTION INTO REG 15
          LM     R0,R12,20(R13)        RESTORE R0 THRU R12
          BR     R14                   RETURN TO FAST SCAN BYPASSING
*                                      THE TERMINAION OF PL/I ENV
          SPACE  3
FIRSTSW   DC     XL1'00'
MYPARMS   DC     6A(0)
SAVE1     DC     20F'0'
SAVE2     DC     20F'0'
          LTORG
          SPACE  3
PLIMAIN   CSECT                        STD FORMAT PLIMAIN CSECT
          DC     V(RLDMAIN)
          DC     F'0'
          END
```

Figure 242. FABRRELD to PL/I exit routine interface (Part 2 of 2)

Figure 243 on page 590 shows PL/I exit routine using interface.

```
RELDEXIT:PROC(SEG_PREFIX_PTR,SEG_DATA_PTR,SEG_TABLE_PTR,CON_KEY_PTR)
DCL (SEG_PREFIX_PTR,SEG_DATA_PTR,SEG_TABLE_PTR,CON_KEY_PTR) POINTER;
DCL RETURN_CODE BIN FIXED(31,0) STATIC INIT(0);
DCL (ADDR,PLIRETC) BUILTIN;
DCL 1 SEGTABEL BASED(SEG_TABLE_PTR),
      2 SEGNAME CHAR(8),
      2 SEGCODE BIT(8),
      2 PARCODE BIT(8),
      2 SEGLEVEL BIT(8);
    IF SEGCODE = '00000011'B      /* TEST IF SEGMENT CODE IS X'03' */
       THEN RETURN_CODE = 4;      /*   IF YES BYPASS THIS SEGMENT  */
       ELSE RETURN_CODE = 0;      /*   ELSE ACCEPT THE SEGMENT     */
    CALL PLIRETC(RETURN_CODE);
    END;
```

Figure 243. PL/I exit routine using interface

End of Product-Sensitive Programming Interface

Chapter 34. Conversion to DEDB

When an HDAM, HIDAM, or HISAM database is converted to an IMS Fast Path data entry database (DEDB), the following conditions must be provided:

- The segment of the full-function database must be variable-length.
- DBD, PSB, and ACB must be generated before running FABRRELD to reload DEDB.
- VSAM data sets for the reloaded DEDB areas must be defined by Access Method Services (IDCAMS), and initialized by the IMS DEDB Initialization utility (DBFUMIN0).
- DB Segment Restructure reload program FABRRELD must run under IMS BMP region.

Because FABRRELD issues a SYNC call internally to reuse NBA/OBA buffers, the performance of the reload processing depends on how frequently the SYNC call is issued. The more number of NBA buffers are specified, the less number of SYNC calls are issued, that is, performance is improved. But note that this improved performance may offset the performance of other applications concurrently running under other dependent regions.

Fast Path HSSP can be applied to elicit good performance for reloading DEDB. This can be achieved by specifying PROCOPT=H in a PCB statement of this reload program PSB.

Topics:

Job steps

You must run several programs in order to use DB Segment Restructure. The programs can be run in a single job or in several jobs. The steps in a DB Segment Restructure job stream vary, depending on the changes that you are making, and on the logical relationships involved.

Below is a list of the job steps required to use DB Segment Restructure. A typical job stream may contain some or all of these steps. Since there are no logical relationships in the database in this job step, there is no need to execute any of the IMS utilities that handle logical relationships.

1. **DFSUDMP0:** The IMS Database Image Copy utility creates a copy of each of the old databases. This step is usually included as a safeguard. *This step should be considered mandatory.* If a problem occurs during the DB Segment Restructure process, you will have a "backup" database copy.
2. **FABRUNLD:** The DB Segment Restructure unload program creates a sequential data set that contains the old unloaded databases with current DBDs and PSBs of the full-function database being unloaded. *This step is mandatory.*
3. **DBDGEN, PSBGEN, and ACBGEN:** This IMS procedure creates the DBDs, the PSBs, and the ACB for new DEDB to be reloaded.
4. **IDCAMS:** This procedure deletes the old database data sets and allocates the new DEDB database data sets. This step *must* be included.
5. **DBFUMIN0:** The IMS DEDB Initialization utility initializes the new DEDB database. This step *must* be included.

6. **FABRRELD:** The DB Segment Restructure reload program creates new restructured databases. This program must run under IMS BMP region. This step *must* be included.
7. **DFSUDMP0:** The IMS Database Image Copy utility creates a copy of each new database. This is the first backup of the new databases. *Like job step 1, this "safeguard" step should be considered mandatory.*

FABRRELD JCL

To convert any full-function database to DEDB, the DB Segment Restructure reload program FABRRELD must run under IMS BMP region.

To run FABRRELD, you must provide some additional DD statements. These are the FABRRELD JCL requirements:

EXEC
Code this statement as:

```
//     EXEC IMSBATCH,
//          MBR=FABRRELD,
//          PSB=psbname
```

The PSB must have these characteristics:

- It contains PROCOPT=I or PROCOPT=A (on the PCB statement) if you are using FABRRELD as an initial load program.
- It contains PROCOPT=A (on the PCB statement) if you are using FABRRELD as an update program.
- It contains LANG=ASSEM on the PSBGEN statement.

SYSPRINT DD
This output data set contains the reports produced by FABRRELD. BLKSIZE, if coded on the DD statement, must be a multiple of 133.

SYSIN DD
This input data set contains your description of the processing to be done by FABRRELD. It describes the data to be reloaded. BLKSIZE, if coded on the DD statement, must be a multiple of 80.

USEREXIT DD
This input partitioned data set contains the user exit routine load modules.

datain DD
This input data set is a sequential file containing an unloaded database. You can use any ddname for this data set.

database DD
This input data set is an IMS database data set. Use the ddname specified in the DBD.

Figure 244 on page 593 shows a sample of FABRRELD JCL to convert full-function database to DEDB.

```
//RELD    EXEC IMSBATCH,
//             MBR=FABRRELD,
//             PSB=psbname
//SYSPRINT DD  SYSOUT=A
//USEREXIT DD  DUMMY
//DATADEDB DD  DISP=OLD,DSN=HPS.DEDB.DATABASE
//RELD1    DD  DISP=OLD,DSN=HPS.UNLOADED.DATASET
//SYSIN    DD  *
DB        dbdname   RELD1
/*
```

Figure 244. FABRRELD JCL (Converting to DEDB)

Appendix A. Messages and codes

This appendix describes the abend codes, return codes, and messages issued by the five utilities of IMS HP Pointer Checker.

Topics:

HD Pointer Checker

This section describes the abend codes, return codes, and messages issued during the execution of the HD Pointer Checker programs.

Abend codes

Every 3*nnn* abend code is accompanied by an FABP3*nnn*E message. (3*nnn* is a four-digit identification number of the abend code and message.) See the associating FABP3*nnn*E message description for the 3*nnn* abend code.

Return codes

The HD Pointer Checker return codes for each module, and their meanings, are described below.

FABPMAIN

The following list shows the return codes set by FABPMAIN.

Code	Meaning
0	Successfully completed. Your database data sets were successfully processed by FABPMAIN. It may **not** mean that the database is error-free. Your database is valid from an IMS standpoint when **TYPE=ALL** or **CHECK** is specified.
2	Successfully completed; defined unknown data (T2) was detected. One of the HD Pointer Checker programs detected an unknown data error which is defined by T2NUM and T2LEN. This return code is issued only when unknown data and no other errors are detected.
4	Successfully completed; database errors were detected. One of the HD Pointer Checker programs detected a database error.
8	Unsuccessfully completed; control statement errors were detected. HD Pointer Checker ends the job.

The return codes are original values, and they are converted to the value specified in the HPSRETCD control statement.

Note: When you specify the HPSRETCD statement, you can change the return code. For each return code, see "HPSRETCD Statements report" on page 120.

FABPCHRO

The following list shows the return codes set by FABPCHRO.

Code	Meaning
0	Successfully completed.

FABPAUTH

The following list shows the return codes set by FABPAUTH.

Code	Meaning
0	The environmental setting for the HASH pointer checking has completed successfully.
16	Severe errors; Syntax errors were detected in the HD Pointer Checker PROCCTL control statement, which was internally generated by the IMS Database Recovery Facility program. For more details, see HD Pointer Checker ″PROCCTL Statement Report″, which is printed in the IMS Database Recovery Facility master address space.

FABPTGEN

The following list shows the return codes that are set by FABPTGEN.

Code	Meaning
0	Successfully completed. A report or the default table source is generated.
4	Successfully completed but with warnings. The default table source is generated but some warning messages are issued.
8	Unsuccessfully completed; control statement errors or other errors were detected. For the details, see the error messages.

Messages

This section explains the messages that are issued by HD Pointer Checker.

Message format

All the messages have the following format:

```
FABPnnnnx text
```

where:

nnnn Is a four-digit message identification number.

x Indicates the severity of the message as follows:

- **I** Information message. This message usually requires no programmer action.
- **W** Warning message. This message is a warning to alert the programmer of a possible error condition.
- **E** Error message. This message requires programmer action.

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 may represent such things as:

- The name of a module
- A return code
- A condition code
- A command keyword
- A name or value provided by the user

Messages

FABP0001I HD POINTER CHECKER ENDED NORMALLY

Explanation: The HD Pointer Checker job ended normally. Check the HD Pointer Checker Summary report for the detected pointer errors.

System action: This is the normal end of the HD Pointer Checker job with RC=0.

Programmer response: None.

Problem determination: None.

FABP0003E HD POINTER CHECKER ENDED WITH ERRORS

Explanation: HD Pointer Checker detected an error when analyzing an input control statement.

System action: HD Pointer Checker ends the job with RC=8.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Check the error message in the PROCCTL Statement report.

FABP0004E HD POINTER CHECKER EVALUATION IS NOT PROCESSED

Explanation: HD Pointer Checker did not complete pointer checking, because some error or user abend occurred in the HD Pointer Checker process. This message is issued when HD Pointer Checker is invoked from another product.

System action: Processing continues.

Programmer response: Correct the error, and rerun the job.

Problem determination: Check the error message that is generated by the scan or evaluation process.

FABP0005I BLOCK MAP PROCESS ENDED NORMALLY

Explanation: Block Map process ended normally.

System action: This is the normal end of the HD Pointer Checker job with RC=0.

Programmer response: None.

Problem determination: None.

FABP0007I SPACE MONITOR ENDED NORMALLY

Explanation: This message is generated when Space Monitor ends successfully.

System action: HD Pointer Checker ends the job with RC=0.

Programmer response: None.

Problem determination: None.

FABP0008W SPACE MONITOR ENDED WITH WARNINGS

Explanation: Either Space Monitor encountered minor error conditions or one or more threshold warning messages were generated in the Space Monitor Exception report.

System action: HD Pointer Checker ends the job with RC=4.

Programmer response: See the other message generated by Space Monitor to determine the nature and causes of the errors detected. If necessary, correct the problem and rerun the job.

Problem determination: None.

FABP0009E SPACE MONITOR ENDED WITH ERRORS

Explanation: Either Space Monitor encountered major error conditions or one or more error messages were generated in the Space Analysis by the Data Set Report. This message is shown in the HPSRETCD statement report.

System action: HD Pointer Checker ends the job with RC=8.

Programmer response: See the other message generated by Space Monitor to determine the nature and causes of the errors detected. Correct the problem and rerun the job.

Problem determination: None.

FABP0010I #EVALUATION ERRORS DETECTED = *nnn*

Explanation: If *nnn* is not zero, then the database is damaged. At least one segment is pointed to by an incorrect combination of pointers.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0020I #VALIDATION ERRORS DETECTED = *nnn*

Explanation: If *nnn* is not zero, then the database is damaged. At least one pointer or free space element is damaged.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0030E BAD FREE SPACE ELEMENT

Explanation: The database is damaged. The free space element chain is damaged.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0035E BAD FREE SPACE ANCHOR POINT

Explanation: The database is damaged. The free space anchor point (the first 4 bytes in the database block) is incorrect.

System action: Processing continues.

Programmer response: Repair the database and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0040E COUNTER VALUE < NUMBER OF LCHILD

Explanation: The database is probably damaged. The actual number of logical child segments that point to this target segment is greater than its counter value. This could result in the target segment being deleted before all of its logical child segments are deleted.

System action: Processing continues.

Programmer response: Repair the database and rerun the HD Pointer Checker job.

If there is a deleted logical child segment in HISAM overflow that do not have the delete flag set, the HD Pointer Checker treats the deleted segment as "active" and issues the reported message. If the HISAM database is logically connected to an HD database, reorganize only the HISAM database and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0045W CANNOT VALIDATE # OF LCHILD > 32K

Explanation: HD Pointer Checker cannot validate the actual number of logical child segments that point to this target segment that is greater than 32K.

System action: Processing continues.

Programmer response: HD Pointer Checker cannot validate the number of logical segments with the counter value of target segment that is greater than 32K.

Problem determination: None.

FABP0050E COUNTER VALUE > NUMBER OF LCHILD

Explanation: The database is damaged. The actual number of logical child segments that point to this target segment is less than its counter value. The target segment cannot be physically removed from the database, even if it and all its dependent segments are deleted.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0051E THE SUM OF THE CTR FIELDS (*count1*) IN LP SEGMENT *segname1* DOES NOT EQUAL TO THE NUMBER OF LC (*count2*)

Explanation: The sum of CTR values in logical parent segments (*count1*) is not the same as the number of logical child segment occurrences (*count2*).

System action: Processing continues.

Programmer response: The database may be corrupted. Repair the database.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0060W SUM OF COUNTERS > 2,147,483,647

Explanation: The database is probably damaged. While calculating the sum of all counter fields for this segment type, the running sum exceeded 2,147,483,647.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. If there is an error, the message FABP0050E or FABP0040E will indicate which segment has an incorrect counter field.

FABP0070E DISPLAY OF VALIDATION ERROR MESSAGES LIMITED TO 100

Explanation: The database is damaged. More than one hundred error messages were generated. ERRLIMIT=YES was specified on the OPTION statement.

System action: Processing continues. No more error messages are printed, but all errors are counted.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271.

FABP0075E DISPLAY OF EVALUATION ERROR MESSAGES LIMITED TO 100

Explanation: More than one hundred error or warning messages were generated. ERRLIMIT=YES was specified on the OPTION statement.

System action: Processing continues. No more messages are printed, but all errors are counted.

Programmer response: See the messages listed before this message.

Problem determination: If any error messages were generated, see Chapter 10, "Database repair guidelines," on page 271 to repair the database.

FABP0090E LCF & (LTF OR LP) TO SAME TARGET

Explanation: The database is damaged. The target of a logical child first pointer is also the target of a logical twin forward or a logical parent pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. The LCF pointers point to the first logical twin segment in the twin chain. The LTF pointers point only to logical twins after the first one. Therefore, it is illegal to have LCF and LTF pointers to the same target. Since a logical child segment cannot also be a logical parent segment, it is illegal to have LCF and LP pointers to the same target.

FABP0100E LCF & LP POINT TO SAME TARGET

Explanation: The database is damaged. The target of a logical child first pointer is also the target of a logical parent pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. Since a logical child segment cannot also be a logical parent segment, it is illegal to have the LCF and LP pointers pointing to the same target.

FABP0110E LCF & LTF POINT TO SAME TARGET

Explanation: The database is damaged. The target of a logical child first pointer is also the target of a logical twin forward pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. The LCF pointers point to the first logical twin segment in the twin chain. The LTF pointers point only to the logical twins after the first one. Therefore, it is illegal to have LCF and LTF pointers to the same target.

FABP0120E LCF IS ZERO & LCL IS NON-ZERO

Explanation: The database is damaged. A segment contains a nonzero logical child last pointer, while its logical child first pointer is zero.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. Since the LC pointers define the start and end of a logical twin chain, they must either both be zero or both be nonzero.

FABP0130E LCL & (LTB OR LP) TO SAME TARGET

Explanation: The database is damaged. The target of a logical child last pointer is also the target of a logical twin backward or a logical parent pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. The LCL pointers point to the last logical twin segment in the twin chain. The LTB pointers point only to the logical twins before the last one. Therefore, it is illegal to have LCL and LTB pointers to the same target. Since a logical child segment cannot also be a logical parent segment, it is illegal to have LCL and LP pointers to the same target.

FABP0140E LCL & LP POINT TO SAME TARGET

Explanation: The database is damaged. The target of a logical child last pointer is also the target of a logical parent pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. Since a logical child segment cannot also be a logical parent segment, it is illegal to have the LCL and LP pointers pointing to the same target.

FABP0150E LCL & LTB POINT TO SAME TARGET

Explanation: The database is damaged. The target of a logical child last pointer is also the target of a logical twin backward pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. The LCL pointers point to the last logical twin segment in the twin chain. The LTB pointers point only to the logical twins before the last one. Therefore, it is illegal to have LCL and LTB pointers to the same target.

FABP0160E LCL IS ZERO & LCF IS NON-ZERO

Explanation: The database is damaged. A segment contains a nonzero logical child first pointer, while its logical child last pointer is zero.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. Since the LC pointers define the start and end of a logical twin chain, they must either both be zero or both be nonzero.

FABP0170E LP & (LTF OR LCF) TO SAME TARGET

Explanation: The database is damaged. The target of a logical parent pointer is also the target of a logical twin forward or a logical child first pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. Since a logical child segment cannot also be a logical parent segment, it is illegal to have LCF and LP pointers or LTF and LP pointers to the same target.

FABP0180E LP & LCF POINT TO SAME TARGET

Explanation: The database is damaged. The target of a logical child first pointer is also the target of a logical parent pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. Since a logical child segment cannot also be a logical parent segment, it is illegal to have LP and LCF pointers to the same target.

FABP0190E LP & LTF POINT TO SAME TARGET

Explanation: The database is damaged. The target of a logical twin forward pointer is also the target of a logical parent pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. Since a logical child segment cannot also be a logical parent segment, it is illegal to have LP and LTF pointers to the same target.

FABP0195E LP & PAIRED LC POINT TO SAME TARGET

Explanation: The database is damaged. The target of a Paired Logical Child pointer is also the target of a logical parent pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. Since a logical child segment cannot also be a logical parent segment, it is illegal to have LP and Paired LC pointers to the same target.

FABP0200E LP POINTER VALUE IS ZERO

Explanation: The database is damaged. A logical child segment contains zeros in its logical parent pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271.

FABP0210E LTB & (LCL OR LP) TO SAME TARGET

Explanation: The database is damaged. The target of a logical twin backward pointer is also the target of a logical child first or a logical parent pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. The LCL pointers point to the last logical twin segment in the twin chain. The LTB pointers point only to the logical twins before the last one. Therefore, it is illegal to have LCL and LTB pointers to the same target. Since a logical child segment cannot also be a logical parent segment, it is illegal to have LTB and LP pointers to the same target.

FABP0220E LTB & LCL POINT TO SAME TARGET

Explanation: The database is damaged. The target of a logical twin backward pointer is also the target of a logical child first pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. The LCL pointers point to the last logical twin segment in the twin chain. The LTB pointers point only to the logical twins before the last one. Therefore, it is illegal to have LCL and LTB pointers to the same target.

FABP0230E LTB & LP POINT TO SAME TARGET

Explanation: The database is damaged. The target of a logical twin backward pointer is also the target of a logical parent pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. Since a logical child segment cannot also be a logical parent segment, it is illegal to have the LTB and LP pointers point to the same target.

FABP0240E LTB WITH NO CORRESPONDING LTF

Explanation: The database is damaged. A logical twin backward pointer points to a segment, but there is no logical twin forward pointer to that segment. HD Pointer Checker determined that this segment was not the first logical twin segment in the twin chain.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0250E LTF & (LCF OR LP) TO SAME TARGET

Explanation: The database is damaged. The target of a logical twin forward pointer is also the target of a logical child first or a logical parent pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. The LCF pointers point to the first logical twin segment in the twin chain. The LTF pointers point only to the logical twins after the first one. Therefore, it is illegal to have LCF and LTF pointers to the same target. Since a logical child segment cannot also be a logical parent segment, it is illegal to have LP and LTF pointers to the same target.

FABP0260E LTF & LCF POINT TO SAME TARGET

Explanation: The database is damaged. The target of a logical child first pointer is also the target of a logical twin forward pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. The LCF pointers point to the first logical twin segment in the twin chain. The LTF pointers point only to the logical twins after the first one. Therefore, it is illegal to have LCF and LTF pointers to the same target.

FABP0270E LTF & LP POINT TO SAME TARGET

Explanation: The database is damaged. The target of a logical twin forward pointer is also the target of a logical parent pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. Since a logical child segment cannot also be a logical parent segment, it is illegal to have the LP and LTF pointers to the same target.

FABP0275E PAIRED LC & LP POINT TO SAME TARGET

Explanation: The database is damaged. The target of a Paired Logical Child pointer is also the target of a logical parent pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. Since a logical child segment cannot also be a logical parent segment, it is illegal to have the LP and LTF pointers to the same target.

FABP0280E LTF WITH NO CORRESPONDING LTB

Explanation: The database is damaged. A logical twin forward pointer points to a segment, but there is no logical twin backward pointer to that segment. HD Pointer Checker determined that this segment was not the last logical twin segment in the twin chain.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271.

FABP0290E MORE THAN 1 LCF TO SAME TARGET

Explanation: The database is damaged. More than one logical child first pointer points to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. Since a segment cannot be a logical child in more than one logical relationship, and each segment must have a unique parent, it is illegal to have multiple LCF pointers to a segment.

FABP0300E MORE THAN 1 LCL TO SAME TARGET

Explanation: The database is damaged. More than one logical child last pointer points to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. Since a segment cannot be a logical child in more than one logical relationship, and since each segment must have a unique parent, it is illegal to have multiple LCL pointers to a segment.

FABP0310E MORE THAN 1 LTB TO SAME TARGET

Explanation: The database is damaged. More than one logical twin backward pointer points to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. The logical twin chain is crossed or otherwise damaged.

FABP0315E MORE THAN ONE OF TO SAME TARGET

Explanation: The database is damaged. More than one pointer in the overflow (OSAM) part of a HIDAM index database points to the same target.

System action: Processing continues.

Programmer response: Repair the database and rerun the HD Pointer Checker job. If only the index database is damaged, you can resolve the problem by reorganizing the HIDAM database; thereby rebuilding the HIDAM index database at the same time.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. The correspondence between the HIDAM root segments and the HISAM Index segments must be "bijective". (The term *bijective* means that there must be exactly the same number of root segments as there are index segments, and each root segment must correspond to a unique index segment.)

FABP0320E MORE THAN 1 LTF TO SAME TARGET

Explanation: The database is damaged. More than one logical twin forward pointer points to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. The logical twin chain is crossed or otherwise damaged.

FABP0324I NUMBER OF INDEX SOURCE (SC: *nn*) NOT COMPARED WITH NUMBER OF INDEX POINTER (DB#: *nn*)

Explanation: The HD Pointer Checker does not compare the number of index source segments with that of index pointer segments. This message is issued when index pointer segment suppressed by sparse indexing is split and deleted.

System action: Processing continues.

Programmer response: The database may be damaged. If the database is damaged, repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271.

FABP0325E MORE THAN 1 SX TO SAME TARGET

Explanation: The database is damaged. The secondary index source and target segments (as defined in the DBDs) are the same, and two secondary index (KSDS) pointers point to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271.

FABP0326E FABP0326E NUMBER OF INDEX SOURCE (SC: ***xx*****) < NUMBER OF INDEX POINTER (DB#:** ***nnn*****)**

Explanation: The number of the index source segments is less than that of the index pointer segments is. NUMBER OF INDEX SOURCE means the number of index source segments except the number of segments meeting the conditions of suppressing index pointer segment.

The secondary index database maintenance exit routine cannot be validated by the HD Pointer Checker. If, for example, the secondary index database maintenance exit has some logic that depend on the date and time logic that the segment inserted, HD Pointer Checker cannot reproduce the logic because it depends on the current data and time. Check whether the customer's secondary index database maintenance exit has this kind of logic, and if it has, specify SPIXCHK=NO for the OPTION statement in the PROCCTL data set.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271.

FABP0327E NUMBER OF INDEX SOURCE (SC: ***xxx*****) > NUMBER OF INDEX POINTER (DB#:** ***xxx*****)**

Explanation: The number of the index source segments is greater than that of the index pointer segments. 'NUMBER OF INDEX SOURCE' means the number of index source segments other than those segments meeting the conditions of suppressing index pointer segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271.

FABP0330E MORE THAN 1 PCF TO SAME TARGET

Explanation: The database is damaged. More than one physical child first pointer points to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. Each dependent segment must have a unique physical parent. Therefore, there can be at most one physical child first pointer to any segment.

FABP0335E MORE THAN 1 SXO TO SAME TARGET

Explanation: The database is damaged. The secondary index source and target segments (as defined in the DBDs) are the same, and two secondary index overflow (ESDS) pointers point to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271.

FABP0340E MORE THAN 1 PCL TO SAME TARGET

Explanation: The database is damaged. More than one physical child last pointer points to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. Each dependent segment must have a unique physical parent. Therefore, there can be at most one physical child last pointer to any segment.

FABP0350E MORE THAN 1 HB TO SAME TARGET

Explanation: The database is damaged. More than one hierarchical backward pointer points to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. The hierarchical pointer chain is crossed or otherwise damaged.

FABP0360E MORE THAN 1 HF TO SAME TARGET

Explanation: The database is damaged. More than than one hierarchical forward pointer is pointing to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. The hierarchical pointer chain is crossed or otherwise damaged.

FABP0370E MORE THAN 1 PTB TO SAME TARGET

Explanation: The database is damaged. More than one physical twin backward pointer points to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. The physical twin pointer chain is crossed or otherwise damaged.

FABP0380E MORE THAN 1 PTF TO SAME TARGET

Explanation: The database is damaged. More than one physical twin forward pointer points to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. The physical twin pointer chain is crossed or otherwise damaged.

FABP0390E MORE THAN 1 RAP TO SAME TARGET

Explanation: The database is damaged. More than one root anchor pointer points to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job. Consider changing all but one of the duplicate RAPs to zero, followed by a database reorganization.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0400E MORE THAN 1 VLS TO SAME TARGET

Explanation: The database is damaged. The VLS pointers from two or more split segments point to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0410E *xxxx* (HEX) BYTES OF UNKNOWN DATA

Explanation: The database may be damaged.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271 and Chapter 11, “Reported by HD Pointer Checker slack bytes, unknown data, and T2 errors,” on page 283.

FABP0411E SEGMENT LENGTH FIELD IS INVALID

Explanation: The length field of a variable length segment contains a value outside of the range defined for the segment.

System action: Processing continues. A FABP0410E message indicating T2 error and one or more FABP0960E messages indicating T5 errors will follow in the report.

Programmer response: Refer to the FABP0410E message.

Problem determination: None.

FABP0412E SEGMENT BEYOND THE SIZE LIMIT

Explanation: The database is probably damaged. A segment data is located in the block or CI that exceeds the data set size limit. The maximum size of database data set is:

- 8 giga bytes when it is an OSAM data set of non-HALDB with an even number of block size.
- 4 giga bytes for a HALDB, a VSAM data set of non-HALDB, or an OSAM data set of non-HALDB with an odd number of block size.

System action: Processing continues.

Programmer response: The pointers pointing to those segments are incorrect. Repair the database and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0413W BYPASS SPACE BEYOND THE *n*GB SIZE LIMIT

Explanation: The database data set is allocated beyond the *n* GB size limit. For an OSAM database data set that has an even-numbered block size, *n* is 8; otherwise *n* is 4. The HD Pointer Checker bypasses checking of the blocks that exist over *n* GB. If other messages appears with FABP0413W, check the description of the messages. If no message other than FABP0413W appears, the database is not damaged. Then, this message means that the number of blocks allocated to this database exceeds the limit, and no valid segment data exists beyond that limit. This status occurs when some segments were attempted to be inserted beyond the limit.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP0415E ROOT KEY IS OUT OF RANGE

Explanation: The HIDAM database is damaged. A root key that is out of range for the partition was found during the HIDAM SCAN process.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0420I N/A FOR IMAGE COPY

Explanation: Disk address cannot be calculated if input database is image copy.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP0430E NO H/T/PC TO DEPENDENT

Explanation: The database is damaged. A dependent segment exists and is not the target of any hierarchical, twin, or physical child pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0440E NO INDEX POINTER TO THIS ROOT

Explanation: The database is damaged. A root segment in a HIDAM database exists, and there is no segment in the primary index database that points to that root segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job. You should consider reorganizing the HIDAM database, rebuilding the index database at the same time.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0450E NO LTF OR LCF POINTER TO LC

Explanation: The database is damaged. A virtually-paired logical child segment exists that is not the target of any logical twin forward or logical child first pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0460E NO HF OR PTF POINTER TO ROOT

Explanation: The database is damaged. Two or more HIDAM root segments that are not the target of any hierarchical forward or physical twin forward pointer were detected. (The first such occurrence is assumed to be the first root in the HIDAM database. Any other occurrence is assumed to be an error.)

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0470E NO POINTERS TO THIS SEGMENT

Explanation: The database is damaged. A valid segment exists that is the target of no pointer. Such a segment, and its dependents, cannot be accessed by IMS.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job. To keep the segment, connect it to the correct pointer chains.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. If you reorganize the database, this segment will disappear.

FABP0480E NO RAP/H/T TO RT

Explanation: The database is damaged. An HDAM root segment was found that is not the target of any root anchor point, hierarchical, or physical twin pointer. Such a segment, and its dependents, cannot be accessed by IMS.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job. To keep the segment, connect it to the correct pointer chains.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. If you reorganize the database, this segment will disappear.

FABP0490E NO. OF RECORDS WRITTEN FOR POINTER RECONSTRUCTION LIMITED TO 100

Explanation: The database is damaged.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0500E NON-ZERO LTF AT END OF CHAIN

Explanation: The database is damaged. A segment that is not the target of a logical twin backward pointer has a nonzero logical twin forward pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. Only the last twin in the chain is not the target of a logical twin backward pointer. Such a segment must have a zero logical twin forward pointer.

FABP0510E NON-ZERO PTF AT END OF CHAIN

Explanation: The database is damaged. A segment that is not the target of a physical twin backward pointer has a nonzero physical twin forward pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job. Only the last twin in the chain is not the target of a physical twin backward pointer. Such a segment must have a zero physical twin forward pointer.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0530E PAIRED LOG. CHILD < PHYS. CHILD

Explanation: The database is damaged. The number of physically paired segments does not match. Fewer logical child segments were detected than physical child segments.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0540E PAIRED LOG. CHILD > PHYS. CHILD

Explanation: The database is damaged. The number of physically paired segments does not match. Fewer physical child segments were detected than logical child segments.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0541E THE OCCURRENCES OF THE PHYSICALLY PAIRED SEGMENT ARE DIFFERENT. *segname1* **(***count1***) :** *segname2* **(***count2***)**

Explanation: The numbers of logical child segment occurrences and paired logical child segment occurrences are not the same among the Bidirectional Physically Paired Logical Relationship.

System action: Processing continues.

Programmer response: The database may be corrupted. Repair the database.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0550E PCF & (HF, PTF, RAP, OR VL)

Explanation: The database is damaged. An incorrect combination of pointers was detected. In addition to a physical child first pointer, a hierarchical forward, physical twin forward, root anchor point, or variable-length split pointer was detected to this segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0560E PCF & HF POINT TO SAME TARGET

Explanation: The database is damaged. Both a physical child first and a hierarchical forward pointer point to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0570E PCF & PTF POINT TO SAME TARGET

Explanation: The database is damaged. Both a physical child first and a physical twin forward pointer point to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. The physical child first pointer identifies this segment as the first twin segment in the twin chain. Twin forward pointers cannot point to the first twin in the chain.

FABP0580E PCF & RAP POINT TO SAME TARGET

Explanation: The database is damaged. Both a physical child first pointer and a root anchor pointer point to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. The target of a physical child first pointer must be a dependent segment. The target of a root anchor pointer must be a root segment. Therefore, this combination is illegal.

FABP0590E PCF & VLS POINT TO SAME TARGET

Explanation: The database is damaged. Both a physical child first pointer and a variable-length split pointer point to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. The pointer to the data part of a split variable-length segment is the only pointer allowed. All other pointers to a split segment must point to the prefix part.

FABP0600E PCF IS ZERO & PCL IS NON-ZERO

Explanation: The database is damaged. This segment contains a nonzero physical child last pointer, and its physical child first pointer is zero.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. The physical child’s first and last pointers must both be zero or both be nonzero.

FABP0602E PCF WITH NO CORRESPONDING PTB

Explanation: The database is damaged. A physical child first pointer to this segment was detected, but no corresponding physical twin backward pointer was detected. HD Pointer Checker determined that this segment was not the last in the twin chain.

System action: Processing continues.

Programmer response: Repair the database and rerun the HD Pointer Checker.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0604E PCF WITH NO CORRESPONDING HB

Explanation: The database is damaged. A physical child first pointer to this statement was detected, but no corresponding physical hierarchical backward pointer was detected. HD Pointer Checker determined that this segment was not the last in the hierarchical chain.

System action: Processing continues.

Programmer response: Repair the database and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0610E PCL & (HB OR PTB) TO SAME TARGET

Explanation: The database is damaged. A physical child last pointer and either a hierarchical backward pointer or a physical twin backward pointer point to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0620E PCL & HB POINT TO SAME TARGET

Explanation: The database is damaged. A physical child last pointer and a hierarchical backward pointer point to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0630E PCL & PTB POINT TO SAME TARGET

Explanation: The database is damaged. A physical child last pointer and a physical twin backward pointer point to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0640E PCL IS ZERO & PCF IS NON-ZERO

Explanation: The database is damaged. This segment contains a nonzero physical child first pointer, and its physical child last pointer is zero.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. The physical child’s first and last pointers must both be zero or both be nonzero.

FABP0650E HB & (PTB OR PCL) TO SAME TARGET

Explanation: The database is damaged. A physical hierarchical backward pointer and either a physical twin backward pointer or a physical child last pointer point to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0660E HB & PCL POINT TO SAME TARGET

Explanation: The database is damaged. A physical hierarchical backward pointer and a physical child last pointer point to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0670E HB & PTB POINT TO SAME TARGET

Explanation: The database is damaged. A physical hierarchical backward pointer and a physical twin backward pointer point to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0680E HB WITH NO CORRESPONDING HF

Explanation: The database is damaged. A hierarchical backward pointer points to a segment, but there is no hierarchical forward pointer to that segment. HD Pointer Checker determined that this segment was not the first segment in the hierarchical chain.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0690E HF & (PTF, PCF, RAP, OR VL)

Explanation: The database is damaged. An incorrect combination of pointers was detected. In addition to a physical hierarchical forward pointer, physical twin forward, physical child first, root anchor point, or variable-length split pointer was detected to this segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0700E HF & PCF POINT TO SAME TARGET

Explanation: The database is damaged. A hierarchical forward pointer and a physical child first pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0710E HF & PTF POINT TO SAME TARGET

Explanation: The database is damaged. A hierarchical forward pointer and a physical twin forward pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0720E HF & RAP POINT TO SAME TARGET

Explanation: The database is damaged. A hierarchical forward pointer and a root anchor pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0730E HF & VLS POINT TO SAME TARGET

Explanation: The database is damaged. A hierarchical forward pointer and a variable-length split pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0740E HF OR PTF PTR IS ZERO

Explanation: The database is damaged. A HIDAM root segment that is not the end of a twin chain has its hierarchical or physical twin pointer set to zero.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0750E HF WITH NO CORRESPONDING HB

Explanation: The database is damaged. A hierarchical forward pointer points to a segment, but there is no hierarchical backward pointer to that segment. HD Pointer Checker determined that this segment was not the last segment in the hierarchical chain.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0760E PP POINTER VALUE IS ZERO

Explanation: The database is damaged. A dependent segment contains zeros in its physical parent pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0770E PTB & (HB OR PCL) TO SAME TARGET

Explanation: The database is damaged. A physical twin backward pointer and either a hierarchical backward or a physical child last pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0780E PTB & PCL POINT TO SAME TARGET

Explanation: The database is damaged. A physical twin backward pointer and a physical child last pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0785E PTB/HB POINT TO END OF CHAIN

Explanation: The database is damaged. A physical twin backward pointer or a hierarchical backward pointer points to a segment that has zero value in a physical twin forward or a hierarchical forward pointer.

System action: Processing Continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0790E PTB & HB POINT TO SAME TARGET

Explanation: The database is damaged. A physical twin backward pointer and a hierarchical backward pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0800E PTB WITH NO CORRESPONDING PTF

Explanation: The database is damaged. A physical twin backward pointer was detected to this segment, but no corresponding physical twin forward pointer was detected. HD Pointer Checker determined that this segment was not the first in the twin chain.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0810E PTF & (HF, PCF, RAP, OR VL)

Explanation: The database is damaged. A physical twin forward pointer and either a hierarchical forward pointer, physical child first pointer, root anchor point, or variable-length split pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0820E PTF & PCF POINT TO SAME TARGET

Explanation: The database is damaged. A physical twin forward pointer and a physical child first pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0830E PTF & HF POINT TO SAME TARGET

Explanation: The database is damaged. A physical twin forward pointer and a hierarchical forward pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0840E PTF & RAP POINT TO SAME TARGET

Explanation: The database is damaged. A physical twin forward pointer and a root anchor pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0850E PTF & VLS POINT TO SAME TARGET

Explanation: The database is damaged. A physical twin forward pointer and a variable-length split pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0860E PTF WITH NO CORRESPONDING PTB

Explanation: The database is damaged. A physical twin forward pointer was detected to this segment, but no corresponding physical twin backward pointer was detected. HD Pointer Checker determined that this segment was not the last in the twin chain.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0870E RAP & (HF, PTF, PCF, OR VL)

Explanation: The database is damaged. A root anchor pointer and either a hierarchical forward, physical twin forward, physical child first, or variable-length split pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0880E RAP & PCF POINT TO SAME TARGET

Explanation: The database is damaged. A root anchor pointer and a physical child first pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0890E RAP & HF POINT TO SAME TARGET

Explanation: The database is damaged. A root anchor pointer and a hierarchical forward pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0900E RAP & PTF POINT TO SAME TARGET

Explanation: The database is damaged. A root anchor pointer and a physical twin forward pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0910E RAP & VLS POINT TO SAME TARGET

Explanation: The database is damaged. A root anchor pointer and a variable-length split pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0920E SX & SXO POINT TO SAME TARGET

Explanation: The database is damaged. The secondary index source and target segment (as defined in the DBDs) are the same, and a secondary index (KSDS) pointer and a secondary index overflow (ESDS) pointer point to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0930E SXO & SX POINT TO SAME TARGET

Explanation: The database is damaged. The secondary index source and target segment (as defined in the DBDs) are the same, and both a secondary index (KSDS) pointer and a secondary index overflow (ESDS) pointer point to the same target.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0940E TARGET IS AT THE END OF THE BLOCK

Explanation: The database is damaged. The target of this pointer is an impossible byte near the end of the database block.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0950E TARGET IS IN FREE SPACE

Explanation: The database is damaged. This pointer contains an address that is in the range of a valid free space element.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. This segment is likely to be overwritten if IMS tries to insert a segment into this block. It is important to fix this problem immediately.

FABP0960E TARGET IS NOT A VALID SEGMENT

Explanation: The database is damaged.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0970E TARGET IS THE WRONG SEGMENT TYPE

Explanation: The database is damaged. The target of this pointer is a valid segment, but it has the wrong segment code for this particular pointer.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0971E TARGET ISN’T A VALID LOGICAL RECORD

Explanation: The HISAM database is damaged. The logical record corresponding to the direct-address pointer does not exist in the overflow data set. Note that this message can be issued for a normal database if some segments have been deleted from the database.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0972E MORE THAN 1 POINTER TO THE SAME RECORD

Explanation: The HISAM database is damaged. More than one direct-address pointer points to the same logical record in the overflow data set. Note that this message can be issued for a normal database if some segments have been deleted from the database.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0973W NO POINTER TO THIS OVERFLOW LOGICAL RECORD

Explanation: No direct-address pointer points to the logical record in the overflow data set.

If an application program deleted segment of the HISAM database, the delete flag is not set by DL/I and the space of the deleted segment remains in the database. Therefore, this message is issued for the normal database, and the message can be ignored. However, it is suggested to reorganize the database if a lot of the messages are issued.

If no application program deletes a segment of the HISAM database, the HISAM database is damaged.

System action: Processing continues.

Programmer response: If it is damaged, repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0980E VL & (HF, PTF, PCF, OR RAP)

Explanation: The database is damaged. A pointer to the data part of a split variable-length segment and either a hierarchical forward pointer, physical twin forward pointer, physical child first pointer, or root anchor pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP0990E VLS & PCF POINT TO SAME TARGET

Explanation: The database is damaged. A pointer to the data part of a split variable-length segment and a physical child first pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. It is incorrect for any pointer other than the VLS pointer to point to the data part of a split variable-length segment.

FABP1000E VLS & HF POINT TO SAME TARGET

Explanation: The database is damaged. A pointer to the data part of a split variable-length segment and a hierarchical forward pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. It is incorrect for any pointer other than the VLS pointer to point to the data part of a split variable-length segment.

FABP1010E VLS & PTF POINT TO SAME TARGET

Explanation: The database is damaged. A pointer to the data part of a split variable-length segment and a physical twin forward pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. It is incorrect for any pointer other than the VLS pointer to point to the data part of a split variable-length segment.

FABP1020E VLS & RAP POINT TO SAME TARGET

Explanation: The database is damaged. A pointer to the data part of a split variable-length segment and a root anchor pointer point to the same target segment.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271. It is incorrect for any pointer other than the VLS pointer to point to the data part of a split variable-length segment.

FABP1030E HIGH KEY NOT FOUND IN THIS PARTITION

Explanation: The partition of the HIDAM database is damaged. The root segment with the high key was not found for the partition during the HIDAM SCAN process.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1040I NO ERRORS DETECTED

Explanation: No damage was detected in this database.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1050I #LP SEGMENTS WITH ZERO CTR FIELD = ***nnnnnnnn***

Explanation: *nnnnnnnn* is the number of logical parent segments with a counter field of zero. This is not an error.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1055W SEGMENT WITH ZERO COUNTER FIELD

Explanation: The logical parent segment has a counter field with zero value.

System action: Processing continues.

Programmer response: If the counter field value of the segment should not be zero, the database is probably damaged, repair the database. Ignore this message in other cases.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1060E TARGET SEGMENT CODE INVALID

Explanation: The database is damaged. This pointer points to a valid segment, but the segment code is the wrong value for this particular pointer.

System action: Processing continues.

Programmer response: Repair the database and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1070W EMPTY DATA SET

Explanation: Either the overflow part of your index or HISAM database is empty, or the HDAM, HIDAM, PHDAM, or PHIDAM database is empty.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1080I COUNT OF LCHILD SEGS (UNI-DIRECTNL) WITH SYMB LP PTRS FROM DATA SET INTO OTHER DB(s) = ***xxx***

Explanation: *xxx* is the number of logical child segments (in a unidirectional logical relationship) in this data set that have symbolic logical parent pointers.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1082E TARGET ***dsg*** **IS POINTED BY ODD RBA**

Explanation: The pointer contains an odd value for RBA, but the active DBDS of the target partition is A to J. The pointer should contain an even value.

System action: Processing continues.

Programmer response: The database might be corrupted. Repair the database.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1084E TARGET *dsg* IS POINTED BY EVEN RBA

Explanation: The pointer contains an even value for RBA, but the active DBDS of the target partition is M to V. The pointer should contain an odd value.

System action: Processing continues.

Programmer response: The database might be corrupted. Repair the database.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1086E POINTER CONTAINS AN ODD VALUE

Explanation: The pointer contains an odd value for RBA, though the database is not capable of online reorganization. The pointer should contain an even value.

System action: Processing continues.

Programmer response: The database might be corrupted. Repair the database.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1090W NO RECORDS IN THIS DATABASE

Explanation: Your database has no root segment.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1092W NO ROOT RECORDS FOUND IN SCAN PROCESS

Explanation: The primary data set group was not scanned with this data set group in the SCAN process. HD Pointer Checker is not able to correct the root segments information. "OCC/ROOT" and related fields are shown as "N/AVAIL" on the "Database Statistics Report" or the "Partition Statistics Report".

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1094W RMOD *rndmname* RETURNED; RC4= *nnnnnnnnnn* RC8=*mmmmmmmm* DURING HOME BLOCK CHECK PROCESS

Explanation: The HD Pointer Checker received the indicated number of return code 4/8 from the randomizer routine for the root segments.

System action: Processing continues.

Programmer response: The application program will receive the FM status code and/or abend U812 for the root segment keys by using the specified randomizer routine.

Problem determination: None.

FABP1095W FP RMOD *rndmname* RETURNED INVALID PART#/RAP# = *nnnnnnnnnn* DURING HOME BLOCK CHECK PROCESS

Explanation: During the Home Block Check process, HD Pointer Checker detected incorrect partition numbers or RAP numbers from the indicated FP randomizer. *nnnnnnnn* indicates the number of segments for which the incorrect value was returned.

System action: Processing continues.

Programmer response: An incorrect randomizer was probably used. Check if the correct one was used, and if not, correct the error and rerun the HD Pointer Checker job.

Problem determination: None.

FABP1096E PARTITION ID *(nnnn)* IN BITMAP BLOCK IS INCORRECT

Explanation: Partition ID in FSEAP of the first bitmap is different from RECON.

System action: Processing continue.

Programmer response: Repair the databases, and rerun the HD Pointer Checker job.

Problem determination: FSEAP of the first bit map block in the data set group A must have a correct partition ID and reorganization number.

FABP1097E REORG# IN DBDS: *mmmmm* IS NOT EQUAL TO REORG# IN RECON: *nnnnn*. ILDS REBUILD IS RECOMMENDED

Explanation: The reorganization number in the partition data set is lower than the reorganization number in the HALDB Partition Database record in RECON. There is an inconsistency between the HALDB (PHDAM or PHIDAM database) and the Indirect List Data Set (ILDS).

System action: Processing continues.

Programmer response: If the database is damaged, rebuild the Indirect List Data Set (ILDS), and rerun the HD Pointer Checker job. If the reorganization number in the image copy data set is obsolete, specify ICRG#CHK=NO in the PROC statement in the PROCCTL data set.

Problem determination: If this message is issued for a real database data set, the database is damaged.

If this message is issued for an image copy data set, the following are possible causes:

- The database was reorganized after the image copy is taken. The reorganization number in the image copy data set is obsolete. In this case, the reorganization number cannot be validated. Do not specify ICRG#CHK=YES.
- If the database was not reorganized after the image copy is taken, the database is damaged.

FABP1098E INCORRECT PARTITION ID RETURNED FROM PARTITION SELECTION: *nnnnn*

Explanation: Partitioned ID returned from a partition selection routine is different from RECON.

System action: Processing continues.

Programmer response: Repair the database or partition selection exit, and rerun the HD Pointer Checker job.

Problem determination: The partitioned ID returned from the partition selection routine must be the same partition ID as that in RECON.

FABP1099E PARTITION SELECTION FAILED RC: *rc* RSN: *rsn*

Explanation: During the partition selection process, HD Pointer Checker received an error return code. *rc* is the return code and *rsn* is the reason code from partition selection.

System action: Processing continues.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: An incorrect partition high key or a string is defined in RECON, an incorrect partition selection exit is used, or the database is broken.

FABP1101I WORK DATA SET *name* DYNAMIC ALLOCATION SPACE=(*unit*,(*primary*,*secondary*))

Explanation: The work data set *name* has been dynamically allocated the space specified.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1110W THE SYMBOLIC INDEX POINTERS HAVE NOT BEEN CHECKED

Explanation: The symbolic index pointers in the secondary index were not checked. The reason is one of the following:

- The target is not a root segment.
- SYMIXCHK=YES is not specified in the PROCCTL statement.
- TYPE=ALL is not specified in the PROCCTL statement.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1140I FIRST RECORD OF HIDAM INDEX

Explanation: Information about the first record in your HIDAM index database is printed. This is not an error message.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1145I LAST RECORD OF HIDAM INDEX

Explanation: This is a message accompanying the message FABP1175E and shows the last record of your HIDAM index.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1150I FIRST RECORD OF SECONDARY INDEX

Explanation: Information about the first record in your secondary index database is printed. This is not an error message.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1160I FIRST RECORD OF SHARED SECONDARY INDEX

Explanation: Information about the first record in your shared secondary index database is printed. This is not an error message.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1170E BAD DELETE FLAG

Explanation: The database is damaged. A segment that has an incorrect delete byte was found by SCAN processor in an index database.

System action: Processing continues.

Programmer response: Repair the database and rerun the HD Pointer Checker job. If only the index database is damaged, you can repair the problem by reorganizing the prime database; thereby rebuilding the index database at the same time.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1171E INVALID PART NUMBER

Explanation: The index is damaged. This message is issued when one of the following occurred:

- The partition number in the index segment is larger than the maximum partition number in the DBD of the target database.
- The partition number in the index segment is zero, when the target database has more than one partition.
- The partition number in the HIDAM primary index database is out of ascending order.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job. Consider reorganizing the HIDAM database and rebuilding the index database at the same time.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1172I PARTITION ID = X’01’

Explanation: Although the target database has one partition, the partition number in the index segment is X'01',

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1175E LAST INDEX RECORD DOES NOT HAVE A KEY OF ALL X’FF’S

Explanation: The database is damaged. The last record of a HIDAM primary index does not have a key of all X'FF's. Message FABP1145I will accompany this message to show the last record of the HIDAM index.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1176E HIGH KEY NOT FOUND FOR PARTITION#: *nn*

Explanation: The index is damaged. The index segment with the high key was not found for the partition indicated by the number *nn* during the index SCAN process. This message can be issued when partition numbers in the index database are out of ascending order.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job. Consider reorganizing the HIDAM database and rebuilding the index database at the same time.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1177E INVALID INDEX KEY (OUT OF RANGE)

Explanation: The index is damaged. The index key value in the index segment pointing a partition is out of range for the target partition.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job. Consider reorganizing the HIDAM database and rebuilding the index database at the same time.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1180I SCAN COMPLETED

Explanation: Your index database data set was processed by SCAN processor.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1185W EMPTY DATA SET

Explanation: An empty index database data set was specified in the DATABASE statement.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1190W NO ACTIVE SEG'S, ALL LOGICALLY DLTD

Explanation: All segments found in the index database data set are flagged (in their delete bytes) as deleted.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1200I COUNT OF LCHILD SEGS (PHYS. PAIRED) WITH SYMB LP PTRS FROM DATA SET INTO OTHER DB(S) = ***nnn***

Explanation: *nnn* is the number of logical child segments (in a physically-paired, bidirectional logical relationship) in this data set that have symbolic logical parent pointers.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1210I EXPECTED COUNT OF SYMBOLIC LP POINTERS THAT POINT TO THE DATA SET IN THE SAME DB AND/OR OTHER DB(S) = ***nnn***

Explanation: *nnn* is the number of logical child segments that have symbolic logical parent pointers that point to segments checked by the HD Pointer Checker.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1220E PREVIOUS OCCURRENCE(S) OF FOLLOWING POINTER TO THIS SEGMENT DETECTED DURING IN-CORE POINTER CHECKING

Explanation: The database is damaged. There are incorrect duplicate pointers to the same segment. At least one of the incorrect pointers is not printed on any report.

System action: Processing continues.

Programmer response: Rerun the HD Pointer Checker job, specifying INCORE=NO on the OPTION statement. This causes all error messages of this kind to be printed. Then repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. Refer to the specific error messages for more information.

FABP1230I ILLEGAL COMBINATION OF POINTERS TO FOLLOWING SEGMENT DETECTED DURING IN-CORE POINTER CHECKING

Explanation: The database is damaged. There is an incorrect combination of pointers to the same segment. At least one of the incorrect pointers is not printed on any reports.

System action: Processing continues.

Programmer response: Rerun the HD Pointer Checker job, specifying INCORE=NO on the OPTION statement. This causes all error messages of this kind to be printed. Then repair the database and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. Refer to the specific error messages for more information.

FABP1240W DB: ***xxx*** **DSG:** ***yy*** **WAS NOT SCANNED (SEE DMB DIRECTORY)**

Explanation: The indicated HDAM or HIDAM database *xxx* (data set group:*yy*) was not scanned in the SCAN process.

System action: Processing continues.

Programmer response: Add the control statements and JCL for all missing databases, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. Always process all related databases in a single HD Pointer Checker job. If you omit a database, you are taking a risk that pointer errors will be left undetected.

FABP1250W ***xxx*** **TARGET POINTERS CANNOT BE VALIDATED. ERRONEOUS EVALUATION MESSAGES MAY BE PRODUCED.**

Explanation: There are *xxx* pointers whose targets are in a database that was not processed by this HD Pointer Checker run.

System action: Processing continues.

Programmer response: Add control statements and JCL for all missing databases, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271. Always process all related databases in a single HD Pointer Checker job. If you omit a related database, you are taking a risk that pointer errors will go undetected.

FABP1260I DATABASE NUMBER NOT IN INPUT, OR CONTROL RECORD NOT IN SEQUENCE

Explanation: The specified input record in your JRM or BLKMPIN data set is an address in a database with no type T0, T1, or T2 record (in your CHECKREC data set).

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1265I PARTITION ID NOT IN INPUT, OR CONTROL RECORD NOT IN SEQUENCE

Explanation: The specified input record in your JRM or BLKMPIN data set is an address in a database with no type T0, T1, or T2 record (in your CHECKREC data set).

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1270I NO MORE RECORDS FOR SPECIFIED DATA SET GROUP

Explanation: There are no more records (in the CHECKREC data set) that refer to the address requested on the JRM or BLKMAPIN data set.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1280I NO MORE RECORDS FOR SPECIFIED RBA

Explanation: All records on the CHECKREC data set for this RBA (and data set) were processed.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1290I END OF FILE ON CONTROL DATA SET

Explanation: All records on the JRM or BLKMAPIN data set were processed by BLOCKMAP processor.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1300I END OF FILE ON CHECKREC DATA SET

Explanation: All records on the CHECKREC data set were processed by BLOCKMAP processor.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1310I INPUT RECORD FROM BLKMAPIN FILE

Explanation: The information on this report line was read by BLOCKMAP processor from the BLKMAPIN data set.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1320I INPUT RECORD FROM JRM FILE

Explanation: The information on this report line was read by BLOCKMAP processor from the JRM data set.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1330I ADDRESS FOUND IN WORK DATA SET

Explanation: The address on this report line was requested on your BLOCKMAP process input (see message FABP1310I or FABP1320I) and was found in the CHECKREC data set.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1340I SEGMENT POINTS TO ABOVE INPUT ADDRESS

Explanation: The segment identified on this report line contains a direct pointer that points to the address requested on your BLOCKMAP process input (see message FABP1310I or FABP1320I).

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1345I UP TO 300 RECORDS ARE PRINTED

Explanation: The number of records having the same RBA as the target and generated in the pointer chain reconstruction exceeded 300. The records over 300 are not generated.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1350W REQUESTED RBA (*xxxxxxxx*) ALREADY SNAPPED ... SKIPPING FOR NEXT CONTROL STATEMENT

Explanation: The RBA (*xxxxxxxx*) specified by BLOCKDUMP parameter on the DATABASE statement is in a block that was already printed.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1360W SPECIFIED RBA (*xxxxxxxx*) BEYOND DATABASE... SKIPPING FOR NEXT CONTROL STATEMENT

Explanation: The RBA (*xxxxxxxx*) specified by BLOCKDUMP parameter on the DATABASE statement is outside the range of the database.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1365I INPUT IS IC2. DUMPED DATASET NAME: *dsname*

Explanation: This information message indicates that the input is an image copy taken with the Database Image Copy 2 utility.

This message shows the data set name of the dumped data set in image copy so that you can verify your data set specification when an error occurs.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1370I SCAN OF DB: *dbdname* **DSG:** *xx* **IN PROGRESS TIME=***hh.mm.ss*

Explanation: SCAN processor has begun the processing of data set group *xx* of HISAM database *dbdname*. The processing started at *hh.mm.ss*.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1380I SCAN OF DB: *dbdname* **DSG:** *xx* **COMPLETED TIME=***hh.mm.ss*

Explanation: SCAN processor has completed the processing of data set group *xx* of HISAM database *dbdname*. The data set is empty. No errors were detected by SCAN processor. The processing completed at *hh.mm.ss*.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1390I SCAN OF DB: *dbdname* **DSG:** *xx* **EMPTY DATA SET TIME=***hh.mm.ss*

Explanation: The processing of data set group *xx* of HISAM database *dbdname* by SCAN processor has completed. The data set is empty. The processing completed at *hh.mm.ss*.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1400E SCAN OF DB: *dbdname* **DSG:** *xx* **COMPLETED TIME=***hh.mm.ss* *nnnnnnnnn* **ERRORS DETECTED**

Explanation: SCAN processor has completed the processing of data set group *xx* of HISAM database *dbdname*. The processing completed at *hh.mm.ss*. *nnnnnnnnn* error messages were generated.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1410I SCAN OF DB: *dbdname* **PID:** *xxxxx* **DSG:** *xx* **IN PROGRESS TIME=***hh.mm.ss*

Explanation: The processing of data set group *xx* of partition ID *xxxxx* of index database *dbdname* by SCAN processor has begun. The processing started at *hh.mm.ss.*

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1420I SCAN OF DB: *dbdname* **PID:** *xxxxx* **DSG:** *xx* **COMPLETED TIME=***hh.mm.ss*

Explanation: The scan processor has completed the processing of data set group *xx* of partition ID *xxxxx* of index database *dbdname*, and detected no errors. The processing was completed at *hh.mm.ss.*

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1425E SCAN OF DB: *dbdname* **PID:** *xxxxx* **DSG:** *xx* **COMPLETED WITH ERRORS TIME=***hh.mm.ss*

Explanation: The scan processor has completed the processing of the data set group of partition ID *xxxxx* of index database *dbdname*, and some errors were detected. The processing was completed at *hh.mm.ss.*

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1430I SCAN OF DB: *dbdname* **PID:** *xxxxx* **DSG:** *xx* **EMPTY DATA SET TIME=***hh.mm.ss*

Explanation: The scan processor has completed the processing of data set group *xx* of partition ID *xxxxx* of index database *dbdname*. The data set is empty. The processing was completed at *hh.mm.ss.*

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1440I SCAN OF DB: *dbdname* **PID:** *xxxxx* **DSG:** *xx* **IN PROGRESS @ BLOCK** *xxx* **TIME=***hh.mm.ss*

Explanation: The processing of data set group *xx* of partition ID of HDAM or HIDAM database *dbdname* by the SCAN processor has reached block number *xxx* at *hh.mm.ss.* This message is a status report to help avoid an inappropriate operator cancel of the HD Pointer Checker job.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1450I SCAN OF DB: *dbdname* **PID:** *xxxxx* **DSG:** *xx* **COMPLETED @ BLOCK** *xxx* **TIME=***hh.mm.ss*

Explanation: The processing of data set group *xx* of partition ID *xxxxx* of HDAM or HIDAM database *dbdname* by SCAN processor has reached block number *xxx*, and is complete. No errors were detected by the SCAN processor. The processing was completed at *hh.mm.ss.*

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1460I SCAN OF DB: *dbdname* **PID:** *xxxxx* **DSG:** *xx* **EMPTY DATA SET TIME=***hh.mm.ss*

Explanation: The scan processor has completed the processing of data set group *xx* of partition ID *xxxxx* of HDAM or HISAM database *dbdname*. The data set is empty. The processing was completed at *hh.mm.ss*.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1470E SCAN OF DB: *dbdname* **PID:** *xxxxx* **DSG:** *xx* **COMPLETED @ BLOCK** *xxx* **TIME=***hh.mm.ss nnnnnnnnn* **ERRORS DETECTED**

Explanation: The processing of data set group *xx* of partition ID *xxxxx* of HDAM or HIDAM database *dbdname* by SCAN processor has reached block number *xxx*, and is complete. The processing was completed at *hh.mm.ss*. *nnnnnnnnn* error messages were generated.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1475I DB: *dbdname* **PART:** *partname* **IS NOT PROCESSED**

Explanation: The *partname* partition of the *dbdname* database is not processed by IMS Parallel Reorganization.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1480I EVAL OF DB: *dbdname* **PID:** *xxxxx* **DSG:** *xx* **IN PROGRESS TIME=***hh.mm.ss*

Explanation: The processing of data set group *xx* of partition ID *xxxxx* of database *dbdname* by CHECK processor has begun. The processing started at *hh.mm.ss*.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1490I EVAL OF DB: *dbdname* **PID:** *xxxxx* **DSG:** *xx* **COMPLETED TIME=***hh.mm.ss*

Explanation: The CHECK processor has completed the processing of data set group *xx* of partition ID *xxxxx* of database *dbdname*. No errors were detected. The processing completed at *hh.mm.ss*.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1500I EVAL OF DB: *dbdname* **PID:** *xxxxx* **DSG:** *xx* **FAILED (DB#:** *nnn***) TIME=***hh.mm.ss* **(SEE DMB DIRECTORY)**

Explanation: The CHECK processor has completed the processing of data set group *xx* of partition ID *xxxxx* of database *dbdname* (database number: *nnn*). HD Pointer Checker found pointers to a HDAM or HIDAM database that was not scanned in the SCAN process. The processing ended at *hh.mm.ss*.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1503E *nnnnn* **ERRORS FOUND IN HASH CHECK OF DB:** *dbdname* **PID:** *xxxxx* **DSG:** *xx* **TIME=***hh.mm.ss*

Explanation: The HASH Check process of data set group *xx* in partition ID *xxxxx*, database *dbdname* has completed at *hh.mm.ss*. and *nnn* errors were found. The *nnnnn* errors are issued during the HASH evaluation process. Errors issued during scan processes are reported in separate messages.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1504E *nnnnnnn* **ERRORS FOUND IN EPS CHECK OF DB:** *dbname* **PID:** *xxxxx* **TIME=***hh.mm.ss*

Explanation: The EPS healing process of partition ID *xxxxx* in database *dbdname* completed at *hh.mm.ss*. and *nnnnnnn* errors were reported.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1510E EVAL OF DB: *dbdname* **PID:** *xxxxx* **DSG:** *xx* **COMPLETED TIME=***hh.mm.ss* *nnnnnnnnn* **ERRORS DETECTED**

Explanation: The CHECK processor has completed the processing of data set group *xx* of partition ID *xxxxx* of database *dbdname*. The processing was completed at *hh.mm.ss*. *nnnnnnnnn* error messages were generated.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1520W { T2 | POINTER } ERRORS WERE DETECTED DURING POINTER-CHECKER EXECUTION

Explanation: If message text shows ″T2 ERRORS″, one or more databases contain unknown data (T2) more than the threshold values, which are specified by T2NUM and T2LEN. In this case, pointer error is not detected.

If message text shows ″POINTER ERRORS″, one or more databases have pointer errors. The databases are probably damaged. In some cases, T2 errors are also detected.

Error messages are printed in some of the HD Pointer Checker reports.

System action: Processing continues with RC= 2 or 4. Return code 2 is returned only when T2 errors are detected. Return code 4 is returned when Pointer errors are detected, or both Pointer and T2 errors are detected.

Programmer response: If the database has T2 errors, consider reorganizing the database to remove the unknown data. If the database has pointer errors, determine the causes of the error messages. If necessary, repair the database.

After the database is reorganized or repaired, rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1530W DISTRIBUTION OF DEPENDENT SEGMENTS IN ROOT BLOCK AND DISTRIBUTION OF DEPENDENT SEGMENTS BY SEGMENT CODE ARE NOT AVAILABLE.

Explanation: DISTRIBUTION OF DEPENDENT SEGMENTS IN ROOT BLOCK and DISTRIBUTION OF DEPENDENT SEGMENTS BY SEGMENT CODE are not generated because the NO-INCORE check option or the HASH check option was selected.

System action: Processing continues.

Programmer response: To print the report, specify HASH=NO on the PROC statement and INCORE=YES on the OPTION statement, and rerun the HD Pointer Checker job. The report is never printed if you run the HASH Check function in jobs other than the HD Pointer Checker FABPMAIN job.

Problem determination: None.

FABP1531W { DISTRIBUTION OF ROOT SEGMENTS | DISTRIBUTION OF RAP CHAIN LENGTHS } CANNOT BE PRINTED BECAUSE HIGH BLOCK NUMBER IS NOT DEFINED

Explanation: Indicated report cannot be printed, because number of blocks or CIs of root addressable area is not defined in DBDGEN.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP1955E INDEX KEY OF INDEX SOURCE SEGMENTS (DB: *dbname* **SEGMENT:** *segment***) AND INDEX POINTER SEGMENTS ARE NOT THE SAME**

FABP1955E INDEX KEY OF ROOT SEGMENTS (DB: *dbname* **SEGMENT:** *segment***) AND INDEX POINTER SEGMENTS ARE NOT THE SAME**

Explanation: The database is damaged. The HASH total of index keys in source segments or ROOT segments does not equal to the HASH total of the index keys in the index pointer segments.

System action: Processing continues.

Programmer response: Correct the database, and rerun the HD Pointer Checker job.

Problem determination: None.

FABP1956E THE NUMBER OF ROOT SEGMENTS: *segname* **IS NOT EQUAL TO THE NUMBER OF PRIMARY INDEX:** *indexdb* **POINTERS**

Explanation: The PHIDAM or HIDAM database is damaged. The number of the *segname* root segment occurrences in the database should be equal to the number of index pointers in the primary index *indexdb*; however, they are different. The root segment occurrences, primary index pointers, or both are damaged.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1957E MISMATCH BETWEEN PRIMARY INDEX: *indexdb* **POINTERS TO ROOT SEGMENT:** *segname* **AND TARGET RBA VALUES**

Explanation: The PHIDAM or HIDAM database is damaged. The index pointers in the primary index *indexdb* do not point correctly to the RBA values of the *segname* root segment occurrences. The root segments, primary index segments, or both are damaged.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1958E THE NUMBER OF ROOT SEGMENTS: *segname* **IS NOT EQUAL TO THE NUMBER OF RAPS & PTF POINTERS**

Explanation: The database is damaged. The number of the *segname* root segment occurrences should be equal to the number of Root Anchor Points (RAPs) and Physical Twin Forward (PTF) pointers; however, they are different. The root segments, RAPs, PTF pointers, or all of them are damaged.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1959E MISMATCH BETWEEN RAPS & PTF POINTERS TO ROOT SEGMENT: *segname* **AND TARGET RBA VALUES**

Explanation: The database is damaged. Root Anchor Points (RAPs) and Physical Twin Forward (PTF) pointers do not point correctly to the RBA values of the *segname* root segment occurrences. The root segments, RAPs, PTF pointers, or all of them are damaged.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1960E THE NUMBER OF SEGMENTS: *segname* IS NOT EQUAL TO THE NUMBER OF THE PTF POINTERS

Explanation: The database is damaged. The number of the *segname* segment occurrences should be equal to the number of Physical Twin Forward (PTF) pointers that point to the segment occurrences. The *segname* segment occurrences, PTF pointers, or both are damaged.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271.

FABP1961E MISMATCH BETWEEN PTF POINTERS IN SEGMENT: *segname* AND TARGET RBA VALUES

Explanation: The database is damaged. Physical Twin Forward (PTF) pointers do not point correctly to the Relative Byte Addresses (RBAs) of the *segname* segment occurrences. The *segname* segment occurrences, the PTF pointers, or both are damaged.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271.

FABP1962E THE NUMBER OF SEGMENTS: *segname* IS NOT EQUAL TO THE NUMBER OF THE PTB POINTERS

Explanation: The database is damaged. The number of Physical Twin Backward (PTB) pointers should be equal to the number of the *segname* segment occurrences. The *segname* segment occurrences, the PTB pointers, or both are damaged.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271.

FABP1963E MISMATCH BETWEEN PTB POINTERS IN SEGMENT: *segname* AND TARGET RBA VALUES

Explanation: The database is damaged. Physical Twin Backward (PTB) pointers does not point correctly to the Relative Byte Addresses (RBAs) of the *segname* segment occurrences. The *segname* segment occurrences, the PTB pointers, or both are damaged.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271.

FABP1964E THE NUMBER OF LTF POINTERS IS NOT EQUAL TO THE NUMBER OF LTB POINTERS IN SEGMENT: *segname*

Explanation: The database is damaged. The number of Logical Twin Forward (LTF) pointers in the *segname* segment occurrences should be equal to the number of Logical Twin Backward (LTB) pointers in the *segname* segment occurrences. The *segname* segment occurrences, LTF pointers, LTB pointers, or all of them are damaged.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271.

FABP1965E MISMATCH BETWEEN LTF AND LTB POINTERS IN SEGMENT: *segname*

Explanation: The database is damaged. There is inconsistency between the Logical Twin Forward (LTF) pointers and the Logical Twin Backward (LTB) pointers in the *segname* segment occurrences. The *segname* segment occurrences, LTF pointers, LTB pointers, or all are damaged.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271.

FABP1966E THE NUMBER OF SEGMENTS: *segname* IS NOT EQUAL TO THE NUMBER OF ITS PTF POINTERS & PCF POINTERS IN THE PARENT: *parent*

Explanation: The database is damaged. The number of the *segname* segment occurrences should be equal to the number of physical forward pointers that point to the *segname* segment occurrences. Some or all of the following segment occurrences or pointers are damaged:

- The *segname* segment occurrences
- The *parent* segment occurrences, which are the physical parents of the *segname* segments
- The Physical Twin Forward (PTF) pointers in the *segname* segment occurrences

- The Physical Child First (PCF) pointers in the *parent* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1967E MISMATCH BETWEEN PTF POINTERS IN SEGMENT: *segname* **& PCF POINTERS IN PARENT:** *parent* **AND TARGET RBA VALUES**

Explanation: The database is damaged. Physical forward pointers do not point correctly to the *segname* segment occurrences. Some or all of the following segments or pointers are damaged:

- The *segname* segment occurrences
- The *parent* segment occurrences, which are the physical parents of the *segname* segments
- The Physical Twin Forward (PTF) pointers in the *segname* segment occurrences
- Physical Child First (PCF) pointers in the *parent* segment occurrences, which are parents of the *segname* segments

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1968E THE NUMBER OF SEGMENTS: *segname* **IS NOT EQUAL TO THE NUMBER OF ITS PTB POINTERS & PCL POINTERS IN THE PARENT:** *parent*

Explanation: The database is damaged. The number of the *segname* segment occurrences should be equal to the number of physical backward pointers. Some or all of the following segments or pointers are damaged:

- The *segname* segment occurrences
- The *parent* segment occurrences, which are the physical parents of the *segname* segments
- The Physical Twin Backward (PTB) pointers in the *segname* segment occurrences
- Physical Child Last (PCL) pointers in the *parent* segment occurrences, which are parents of the *segname* segments

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1969E MISMATCH BETWEEN PTB POINTERS IN SEGMENT: *segname* **& PCL POINTERS IN PARENT:** *parent* **AND TARGET RBA VALUES**

Explanation: The database is damaged. Physical backward pointers do not point correctly to the Relative Bytes Addresses (RBAs) of the *segname* segment occurrences. Some or all of the following segments or pointers are damaged:

- The *segname* segment occurrences
- The *parent* segment occurrences, which are the physical parents of the *segname* segments
- The Physical Twin Backward (PTB) pointers in the *segname* segment occurrences
- The Physical Child Last (PCL) pointers in the *parent* segment occurrences, which are parents of the *segname* segments

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1970E MISMATCH BETWEEN PP POINTERS IN THE LAST PHYSICAL CHILD: *segname* **AND RBA VALUES OF PARENT:** *parent* **WITH NON-ZERO PCF**

Explanation: The database is damaged. Physical Parent (PP) pointers in the *segname* segment occurrences do not point correctly to the Relative Bytes Addresses (RBAs) of the *parent* segment occurrences. Some or all of the following segments or pointers are damaged:

- The *segname* segment occurrences
- The *parent* segment occurrences, which are the physical parents of the *segname* segments
- The Physical Parent (PP) pointers in the *segname* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1971E MISMATCH BETWEEN PP POINTERS IN THE FIRST PHYSICAL CHILD: *segname* **AND RBA VALUES OF PARENT:** *parent* **WITH NON-ZERO PCF**

Explanation: The database is damaged. The Physical Parent (PP) pointers in the *segname* segment occurrences do not point correctly to the Relative Bytes Addresses (RBAs) of the *parent* segment occurrences.

Some or all of the following segments or pointers are damaged:

- The *segname* segment occurrences
- The *parent* segment occurrences, which are the physical parents of the *segname* segments
- The Physical Parent (PP) pointers in the *segname* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1972E THE NUMBER OF PREFIX PORTIONS IS NOT EQUAL TO THE NUMBER OF DATA PORTIONS IN SPLIT VARIABLE-LENGTH SEGMENTS: *segname*

Explanation: The database is damaged. Some of *segname* segment occurrences are split into the prefix and the data portion. The number of prefix portions should be equal to the number of data portions; however, they are different. The *segname* segment occurrences, prefix portions, data portions, or all of them are damaged.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1973E MISMATCH BETWEEN POINTERS TO THE DATA PORTIONS AND RBA VALUES OF THE DATA PORTIONS IN THE SPLIT VL SEGMENT: *segname*

Explanation: The database is damaged. Some of the *segname* segment occurrences are split into the prefix and the data portion. The pointers in the prefix portions do not point correctly to the Relative Byte Addresses (RBAs) of the data portions. The *segname* segment occurrences, prefix portions, data portions, or all of them are damaged.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1974E THE NO. OF THE LAST SEGMENTS IN TWIN CHAINS: *segname* **IS NOT EQUAL TO THE NUMBER OF THE PCL POINTERS IN PARENT:** *parent*

Explanation: The database is damaged. The number of Physical Child Last (PCL) pointers in the *parent* segment occurrences should be equal to the number of the last occurrences of the *segname* segment; however, they are different. Some or all of the following segments or pointers are damaged:

- The *segname* segment occurrences
- The *parent* segment occurrences, which are the physical parents of the *segname* segments
- The Physical Child Last (PCL) pointers in the *parent* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1975E MISMATCH BETWEEN PCL POINTERS IN PARENT: *parent* **AND RBA VALUES OF THE LAST SEGMENT IN TWIN CHAINS:** *segname*

Explanation: The database is damaged. The Physical Child Last (PCL) pointers in the *parent* segment occurrences do not point correctly to the *segname* segment occurrences. Some or all of the following segments or pointers are damaged:

- The *segname* segment occurrences
- The *parent* segment occurrences, which are the physical parents of the *segname* segment occurrences
- The Physical Child Last (PCL) pointers in the *parent* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1976E THE NUMBER OF LC: *segname* **IS NOT EQUAL TO THE NUMBER OF ITS LTF POINTERS & LCF POINTERS IN THE LP:** *parent* **DB:** *database*

Explanation: The database is damaged. The number of the *segname* segment occurrences should be equal to the logical forward pointers that point to the *segname* segment occurrences; however, they are different. Some or all of the following segments or pointers are damaged:

- The *segname* segment occurrences

- The *parent* segment (the logical parent of the *segname* segment) occurrences in the *database* database
- The Logical Twin Forward (LTF) pointers in the *segname* segment occurrences
- The Logical Child First (LCF) pointers in the *parent* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1977E MISMATCH BETWEEN LTF POINTERS IN LC: ***segname*** **& LCF POINTERS IN LP:** ***parent*** **DB:** ***database*** **AND TARGET RBA VALUES**

Explanation: The database is damaged. Logical forward pointers do not point correctly to the Relative Bytes Addresses (RBAs) of the *segname* segment occurrences. Some or all of the following segments or pointers are damaged:

- The *segname* segment occurrences
- The *parent* segment (the logical parent of the *segname* segment) occurrences in the *database* database
- The Logical Twin Forward (LTF) pointers in the *segname* segment occurrences
- The Logical Child First (LCF) pointers in the *parent* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1978E THE NUMBER OF LC: ***segname*** **IS NOT EQUAL TO THE NUMBER OF ITS LTB POINTERS & LCL POINTERS IN THE LP:** ***parent*** **DB:** ***database***

Explanation: The database is damaged. The number of the *segname* segment occurrences should be equal to the logical backward pointers that point to the *segname* segment occurrences; however, they are different. Some or all of the following segments or pointers are damaged:

- The *segname* segment occurrences
- The *parent* segment (the logical parent of the *segname* segment) occurrences in the *database* database
- The Logical Twin Backward (LTB) pointers in the *segname* segment occurrences
- The Logical Child Last (LCL) pointers in the *parent* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1979E MISMATCH BETWEEN LTB POINTERS IN LC: ***segname*** **& LCL POINTERS IN LP:** ***parent*** **DB:** ***database*** **AND TARGET RBA VALUES**

Explanation: The database is damaged. Logical backward pointers do not point correctly to the Relative Bytes Addresses (RBAs) of the logical child segment (the *segname* segment) occurrences. Some or all of the following segments or pointers are damaged:

- The *segname* segment occurrences
- The *parent* segment (the logical parent of the *segname* segment) occurrences in the *database* database
- The Logical Twin Backward (LTB) pointers in the *segname* segment occurrences
- The Logical Child Last (LCL) pointers in the *parent* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1980E MISMATCH BETWEEN LP POINTERS IN THE LAST LC: ***segname*** **AND RBA VALUES OF LP:** ***parent*** **DB:** ***database*** **WITH NON-ZERO LCF**

Explanation: The database is damaged. HD Pointer Checker detected pointer errors during the check for the *segname* segment occurrences, which are the first twin of the logical twin chains. The Logical Parent (LP) pointers in the *segname* segment occurrences do not point correctly to the Relative Bytes Addresses (RBAs) of the logical parent segment occurrences. Some or all of the following segments or pointers are damaged:

- The *segname* segment occurrences
- LP pointers in the *segname* segments
- The *parent* segment (logical parent of the *segname* segment) occurrences in the *database* database
- Logical Child First (LCF) pointers in the *parent* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1981E MISMATCH BETWEEN LP POINTERS IN THE FIRST LC: ***segname*** **AND RBA VALUES OF LP:** ***parent*** **DB:** ***database*** **WITH NON-ZERO LCF**

Explanation: The database is damaged. HD Pointer Checker detected pointer errors during the check for the *segname* segment occurrences, which are the first twin in the logical twin chains. The Logical Parent (LP) pointers in the *segname* segment occurrences do not point correctly to the Relative Bytes Addresses (RBAs) of the logical parent segment occurrences. Some or all of the following segments or pointers are damaged:

- The *segname* segment occurrences
- The LP pointers in the *segname* segments
- The *parent* segment (logical parent of the *segname* segment) occurrences in the *database* database
- The Logical Child First (LCF) pointers in the *parent* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1982E MISMATCH BETWEEN PP POINTERS IN THE LAST PHYSICAL CHILD: ***segname*** **AND RBA VALUES OF PARENT:** ***parent*** **WITH NON-ZERO PCL**

Explanation: The database is damaged. HD Pointer Checker detected pointer errors during the check for the *segname* segment occurrences, which are the last twin in the physical twin chains. The Physical Parent (PP) pointers in the *segname* segment occurrences do not point correctly to the Relative Bytes Addresses (RBAs) of the physical parent segment occurrences. Some or all of the following segments or pointers are damaged:

- The *segname* segment occurrences
- The PP pointers in the *segname* segment occurrences
- The Physical Twin Forward (PTF) in the *segname* segment occurrences
- The *parent* segment (physical parent of the *segname* segment) occurrences in the *database* database
- The Physical Child Last (PCL) pointers in the *parent* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1983E MISMATCH BETWEEN PP POINTERS IN THE FIRST PHYSICAL CHILD: ***segname*** **AND RBA VALUES OF PARENT:** ***parent*** **WITH NON-ZERO PCL**

Explanation: The database is damaged. HD Pointer Checker detected pointer errors during the check for the *segname* segment occurrences, which are the first twin in the physical twin chains. The Physical Parent (PP) pointers in the *segname* segment occurrences do not point correctly to the Relative Bytes Addresses (RBAs) of the physical parent segment occurrences. Some or all of the following segments or pointers are damaged:

- The *segname* segment occurrences
- The PP pointers in the *segname* segment occurrences
- The Physical Twin Backward (PTB) pointers in the *segname* segment occurrences
- The *parent* segment (physical parent of the *segname* segment) occurrences in the *database* database
- The Physical Child Last (PCL) pointers in the *parent* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1984E MISMATCH BETWEEN LP POINTERS IN THE LAST LC: ***segname*** **AND RBA VALUES OF LP:** ***parent*** **DB:** ***database*** **WITH NON-ZERO LCL**

Explanation: The database is damaged. HD Pointer Checker detected pointer errors during the check for the *segname* segment occurrences, which are the last twin in the logical twin chains. The Logical Parent (LP) pointers in the *segname* segment occurrences do not point correctly to the Relative Bytes Addresses (RBAs) of the logical parent segment occurrences. Some or all of the following segments or pointers are damaged:

- The *segname* segment occurrences
- The LP pointers in the *segname* segment occurrences
- The Logical Twin Forward (LTF) pointers in the *segname* segment occurrences
- The *parent* segment (logical parent of the *segname* segment) occurrences in the *database* database
- The Logical Child Last (LCL) pointers in the *parent* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1985E MISMATCH BETWEEN LP POINTERS IN THE FIRST LC: ***segname*** **AND RBA VALUES OF LP:** ***parent*** **DB:** ***database*** **WITH NON-ZERO LCL**

Explanation: The database is damaged. HD Pointer Checker detected pointer errors during the check for the *segname* segment occurrences, which are the first twin in the logical twin chains. The Logical Parent (LP) pointers in the *segname* segment occurrences do not point correctly to the Relative Bytes Addresses (RBAs) of the logical parent segment occurrences. Some or all of the following segments or pointers are damaged:

- The *segname* segment occurrences
- The LP pointers in the *segname* segment occurrences
- The Logical Twin Backward (LTB) in the *segname* segment occurrences
- The *parent* segment (physical parent of the *segname* segment) occurrences in the *database* database
- The Logical Child Last (LCL) pointers in the *parent* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1986E THE NUMBER OF SEGMENT OCCURRENCES IN HIERARCHICAL POINTER CHAINS IS NOT EQUAL TO THE NUMBER OF HF POINTERS (SEGMENT: ***segname*** **)**

Explanation: The database is damaged. The number of segment occurrences, which belong to the hierarchical structure with the *segname* segment at the top, should be equal to the number of Hierarchical Forward (HF) pointers, which link the hierarchical path; however they are different. Some or all of the following segment occurrences or pointers are damaged:

- The *segname* segment occurrences
- The dependent segment occurrences that belong to the *segname* segment on the hierarchical path
- HF pointers, which link the segment occurrences under the hierarchical structure

In addition, the following pointers might be damaged:

- Hierarchical Backward (HB) pointers
- Physical Twin Forward (PTF) pointers and Physical Twin Backward (PTB) pointers, which are in the *segname* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1987E MISMATCH BETWEEN HF POINTERS AND RBA VALUES OF SEGMENTS IN HIERARCHICAL POINTER CHAINS (SEGMENT: ***segname*** **)**

Explanation: The database is damaged. Hierarchical Forward (HF) pointers do not point correctly to the Relative Byte Addresses (RBAs) of the segment occurrences that belong to the hierarchical structure with the *segname* segments at the top. Some or all of the following segment occurrences or pointers are damaged:

- The *segname* segment occurrences
- The dependent segment occurrences that belong to the *segname* segment on the hierarchical path
- HF pointers, which link the segment occurrences under the hierarchical structure

In addition, the following pointers might be damaged:

- Hierarchical Backward (HB) pointers
- Physical Twin Forward (PTF) pointers and Physical Twin Backward (PTB) pointers, which are in the *segname* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1988E THE NUMBER OF SEGMENTS IN HIERARCHICAL POINTER CHAINS IS NOT EQUAL TO THE NUMBER OF HB POINTERS (SEGMENT: ***segname*** **)**

Explanation: The database is damaged. The number of segment occurrences, which belong to the hierarchical structure with the *segname* segment at the top, should be equal to the number of Hierarchical Backward (HB) pointers, which link the hierarchical path; however they are different. Some or all of the following segment occurrences or pointers are damaged:

- The *segname* segment occurrences
- The dependent segment occurrences that follow the *segname* segment on the hierarchical path
- HB pointers, which link the segment occurrences under the hierarchical structure

In addition, the following pointers might be damaged:

- Hierarchical Forward (HF) pointers
- Physical Twin Forward (PTF) pointers and Physical Twin Backward (PTB) pointers, which are in the *segname* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1989E MISMATCH BETWEEN HB POINTERS AND RBA VALUES OF SEGMENTS IN HIERARCHICAL POINTER CHAINS (SEGMENT: *segname* **)**

Explanation: The database is damaged. Hierarchical Backward (HB) pointers do not point correctly to the Relative Byte Addresses (RBAs) of the segment occurrences that belong to the hierarchical structure with the *segname* segments at the top. Some or all of the following segment occurrences and pointers are damaged:

- The *segname* segment occurrences
- The dependent segment occurrences that follow the *segname* segment on the hierarchical path
- HB pointers, which link the segment occurrences under the hierarchical structure

In addition, the following pointers might be damaged:

- Hierarchical Forward (HF) pointers
- Physical Twin Forward (PTF) pointers and Physical Twin Backward (PTB) pointers, which are in the *segname* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1990E THE NO. OF SEGMS IN HIERARCHICAL PTR CHAINS IS NOT EQUAL TO THE NO. OF HF & PCF IN THE PARENT: *parent* **(SEGM:** *segname* **)**

Explanation: The database is damaged. The number of segment occurrences, which belong to the hierarchical structure with the *segname* segment at the top, should be equal to the number of hierarchical forward (HF) pointers and Physical Child First (PCF) pointers. Some or all of the following segment occurrences and pointers are damaged:

- The *parent* segment occurrences that are the physical parent segments of the *segname* segments
- The *segname* segment occurrences
- The dependent segment occurrences that follow the *segname* segment on the hierarchical path
- HF pointers, which link the segment occurrences under the hierarchical structure
- PCF pointers in the *parent* segments, which point to the *segname* segments

In addition, the following pointers might be damaged:

- Hierarchical Backward (HB) pointers
- Physical Twin Forward (PTF) pointers, which are in the *segname* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1991E MISMATCH BETWEEN HF IN HIERARCHICAL PTR CHAINS (SEGMENT: *segname* **) & PCF IN THE PARENT:** *parent* **AND TARGET RBA VALUES**

Explanation: The database is damaged. Hierarchical Forward (HF) pointers and Physical Child First (PCF) pointers do not point correctly to the Relative Byte Addresses (RBAs) of the segment occurrences that belong to the hierarchical structure with the *segname* segments at the top. Some or all of the following segment occurrences and pointers are damaged:

- The *parent* segment occurrences that are physical parent segments of the *segname* segments
- The *segname* segment occurrences
- The dependent segment occurrences that follow the *segname* segment on the hierarchical path
- HF pointers, which link the segment occurrences under the hierarchical structure
- PCF pointers in the *parent* segments, which point to the *segname* segments

In addition, the following pointers might be damaged:

- Hierarchical Backward (HB) pointers
- Physical Twin Forward (PTF) pointers, which are in the *segname* segment occurrences

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1992W THE NUMBER OF DIRECT-ADDRESS POINTERS < THE NUMBER OF LOGICAL RECORDS IN THE OVERFLOW DATA SET

Explanation: The number of direct-address pointers is less than the number of logical records in the HISAM overflow database. It could be that no direct-address pointer points to logical record in the overflow data set.

If an application program deleted segment of the HISAM database, the delete flag is not set by DL/I and the space of the deleted segment remains in the database. Therefore, this message is issued for the normal database, and the message can be ignored. However, it suggested to reorganize the database if a lot of the deleted segments exits. To know the number of deleted segments, run the HD Pointer Checker FABPMAIN program with HASH=NO.

If no application program deletes a segment in the HISAM database, the HISAM database is damaged.

System action: Processing continues.

Programmer response: If it is damaged, repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1993E THE NUMBER OF DIRECT-ADDRESS POINTERS > THE NUMBER OF LOGICAL RECORDS IN THE OVERFLOW DATA SET

Explanation: The HISAM database is damaged. The number of direct-address pointers in the HISAM database is greater than the number of logical records in the overflow data set.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1994E THE SUM OF DIRECT-ADDRESS POINTER VALUES IS DIFFERENT FROM THE SUM OF LOGICAL RECORD RBAS IN THE OVERFLOW DATA SET

Explanation: The HISAM database is damaged. The direct-address pointer values are not equal to the RBAs of the logical record in the overflow data set.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1995E MISMATCH BETWEEN THE NUMBER OF SEGMENTS IN SEG (DB: *dbname1* **SEGM:** *segname1***) AND SEG: (DB:** *dbname2* **SEGM:** *segname2***)**

Explanation: The database is damaged. The number of occurrences of logical child segment is not equal to the occurrences of paired logical child segment in physically paired bidirectional logical relationship. This message is issued for every physically paired bidirectional logical relationship.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1996E COUNTER VALUE IN LP (SEGMENT: *segment***) IS NOT EQUAL TO THE NUMBER OF LOGICAL CHILDREN**

Explanation: The database is damaged. The actual number of logical child segments is not equal to the counter in the logical parent.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1997E MISMATCH BETWEEN LP POINTERS AND LOGICAL PARENT RBA VALUES (DB: *dbdname* **SEGMENT:** *segname***)**

Explanation: The database is damaged. The sum of direct LP pointers is not consistent with the counter value and RBA value.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1998E THE NUMBER OF INDEX SOURCE SEGMENTS (DB: *dbname* **SEGMENT:** *segment***) IS NOT EQUAL TO THE NUMBER OF INDEX POINTER SEGMENTS**

Explanation: The database is damaged. The number of index source segments is not equal to the number of index pointer segments. NUMBER OF INDEX SOURCE means the number of index source segments excluding the number of segments that meet the conditions of suppressing index pointer segments.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP1999E MISMATCH BETWEEN INDEX POINTERS AND TARGET RBA VALUES (DB: *dbname* **SEGMENT:** *segment***)**

Explanation: The database is damaged. The pointer values in the index pointer segment are not equal to the target segment RBAs.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271.

FABP2001I EVAL OF DB: *dbdname* **DB#:** *nnn* **PART: NNNNN DSG#:** *xx* **COMPLETED ERRORS:** *t* **TOTAL,** *s* **SEV,** *p* **PHY,** *l* **LOG**

Explanation: This message is printed for each data set group *xx* of database *dbdname* (database number *nnn*).

t = Total number of errors.

s = Number of errors in RAP, physical child/twin pointer, and hierarchical pointer chain.

p = Number of other errors in physical child/twin pointer, and hierarchical pointer chain.

l = Number of errors in logical relationship pointers and secondary index pointers.

System action: Processing continues.

Programmer response: Check the messages that follow.

Problem determination: None.

FABP2002I RUN COMPLETED ERRORS: *t* **TOTAL**

Explanation: This message is printed at the end of HASH process. *t* represents the total number of errors.

System action: Processing continues.

Programmer response: Check following messages.

Problem determination: None.

FABP2003I NO ERRORS DETECTED

Explanation: All formulas that were evaluated are matched.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2004E ERRORS WERE DETECTED IN LOGICAL RELATIONSHIP POINTERS

Explanation: Error in logical relationship pointers or error in physical path pointers except physical child/twin pointer chain. This may be acceptable if not all logically related databases were included in SCAN process.

System action: Processing continues.

Programmer response: Run the HD Pointer Checker job to check the error in detail.

Problem determination: None.

FABP2005E ERRORS WERE DETECTED ON PHYSICAL CHILD/TWIN POINTER OR HIERARCHICAL POINTER CHAIN

Explanation: Error on physical child pointers, physical twin pointers, or hierarchical pointers.

System action: Processing continues.

Programmer response: Run the HD Pointer Checker job with HASH=NO to check the error in detail.

Problem determination: See the message, issued prior to this message, which contains the details of the error.

FABP2006W EVAL OF DB: *dbdname* **DB#:** *nnn* **PID:** *ppppp* **DSG#:** *d* **COMPLETED ERRORS: NO SEGMENT FOUND IN SCAN PROCESS**

Explanation: No HASH total records are found in the data set group *xx* of input database *dbdname* (database number: *nnn*) data set.

System action: Processing continues.

Programmer response: Rerun the HD Pointer Checker job with HASH=NO option.

Problem determination: None.

FABP2007E ERRORS WERE DETECTED IN SECONDARY INDEX RELATIONSHIP

Explanation: Errors were detected in the HASH check of secondary index relationship.

System action: Processing continues.

Programmer response: Run the HD Pointer Checker job to check the error in detail.

Problem determination: None.

FABP2008E ERRORS WERE DETECTED ON RAP, PHYSICAL CHILD/TWIN POINTER, OR HIERARCHICAL POINTER CHAIN

Explanation: Error on Root Anchor Points (RAP), physical child pointers, physical twin pointers, or hierarchical pointers.

System action: Processing continues.

Programmer response: Run the HD Pointer Checker job with HASH=NO to check the error in detail.

Problem determination: See the message, issued prior to this message, which contains the details of the error.

FABP2010I DATABASE *dbdname* IS PROCESSED

Explanation: HD Pointer Checker processes the database that is identified by *dbdname*. This message is issued for the database that is processed by the DBALL=YES keyword of the DATABASE statement.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2011W DBDS OF ABOVE DATABASE STATEMENT IS TREATED AS ″DBALL=NO″

Explanation: HD Pointer Checker treats the database data sets as DBALL=NO because they are already specified with the previous DBALL=YES specification.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2012E SECONDARY INDEX DATABASE DB: *dbdname* CANNOT BE SCANNED WITH "HASH=YES"

Explanation: The indicated secondary index database *dbdname* and the indexed database are not processed with HASH=YES process option.

System action: Processing stops.

Programmer response: Specify HASH= NO process option, and rerun the HD Pointer Checker job.

Problem determination: None.

FABP2013W DB: *dbdname* DB#: *nnn* DD: *ddname* WAS NOT SCANNED FOR MULTIPLE DATA SET GROUP

Explanation: Multiple data set groups were scanned in the SCAN process, but the indicated data set groups of database *dbdname* was not scanned.

System action: Processing continues.

Programmer response: Make sure you process all multiple data set groups. If you omit any of the multiple data set groups, errors might be left undetected.

Problem determination: None.

FABP2014W DB: *dbdname* DB#: *nnn* WAS NOT SCANNED FOR LOGICALLY RELATED DB DBLG#: *xxx*

Explanation: Multiple logically related data set groups were scanned in the SCAN process, but the indicated data set group *xx* of database *dbdname* (database number *nnn*) was not scanned.

System action: Processing continues.

Programmer response: Make sure you process all logically related databases in the CHECK process. If you omit any of the multiple data set groups, errors might be left undetected.

Problem determination: None.

FABP2015E NO HD DATABASE STATEMENT EXISTS IN DBLG#: *xxx*, CANNOT BE SCANNED WITH "HASH=YES"

Explanation: At least one of the logically related databases (DBLG NO is *xxx*) must be an HD database to run HD Pointer Checker with HASH=YES process option.

System action: Processing stops.

Programmer response: Specify HASH=NO process option, and rerun the HD Pointer Checker job.

Problem determination: None.

FABP2016W PARTITION DD:*ddname* OF DB:*dbdname* DB#:*nnn* WAS NOT SCANNED FOR LOGICALLY RELATED DB DBLG#:*xxx*

Explanation: Multiple partitions that has direct logical parent pointers were scanned in the SCAN process, but the indicated partition was not scanned.

System action: Processing continues.

Programmer response: Make sure you process all partitions of a database that has direct logical parent pointers. If you omit any partitions, errors might be left undetected.

Problem determination: None.

FABP2017W SOME PARTITION OF DB: *dbdname* DB#: *nnn* WAS NOT SCANNED FOR LOGICALLY RELATED DB DBLG#: *xxx*

Explanation: Multiple partitions that has direct logical parent pointers were scanned in the SCAN process, but some partitions of a HALDB was not scanned.

System action: Processing continues.

Programmer response: Make sure you process all partitions of a HALDB that have direct logical pointers. If you omit any of the multiple data set groups, errors might be left undetected.

Problem determination: None.

FABP2018I ″DBALL=YES″ FORCES TO RUN THE FOLLOWING PROCESSES:

Explanation: One or more databases are implicitly processed by the DATABASE DBALL=YES statement. The FABP2010I messages with the database names follow this message.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2020E MISSING INDEX POINTER SEGMENT

Explanation: The segment is missing index pointer segment. This message is issued when index target segment type is the same as index source segment type.

System action: Processing continues.

Programmer response: The database is probably damaged. Repair the database.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP2021E INVALID INDEX KEY

Explanation: The index pointer segment has an incorrect index key. This message is issued when index target segment type is the same as index source segment type.

System action: Processing continues.

Programmer response: The database is probably damaged. Repair the database.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP2022E INVALID INDEX POINTER

Explanation: The index pointer does not point to any segment. This message is issued when index target segment type is the same as index source segment type.

System action: Processing continues.

Programmer response: The database is probably damaged. Repair the database.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP2023E NUMBER OF INDEX < NUMBER OF SOURCE FOR SAME INDEX KEY

Explanation: The number of index source segments is greater than one of index pointer segments which contain same key value. There is a missing index pointer or a bad index key in the database. This message is issued when a index target segment type is different from a index source segment type.

System action: Processing continues.

Programmer response: The database is probably damaged. Repair the database.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP2024E NUMBER OF INDEX > NUMBER OF SOURCE FOR SAME INDEX KEY

Explanation: The number of index source segments is less than one of index pointer segments which contain same key value. There is a missing index pointer or a bad index key in the database. This message is issued when a index target segment type is with a different index source segment type.

System action: Processing continues. Repair the database.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP2025E INDEX KEY LENGTH OF REAL DATABASE IS DIFFERENT FROM DBD

Explanation: The length of the index key from DBD (SRCH field and SUBSEQ field) is different from the length of index key in T6, T7 or TA record.

System action: Skip index key checking for this index database and processing continues.

Programmer response: Use correct DBD or sort records, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271 if necessary.

FABP2026E NUMBER OF INDEX DATABASE POINT TO TARGET EXCEED EXPECTED VALUE

Explanation: The detected number of index database point to target segment is greater than the expected number from DBD. This message is issued when index target segment type is the same as index source segment type.

System action: Processing continues.

Programmer response: Use correct DBD or sort records, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines," on page 271.

FABP2027E TOTAL NUMBER OF RECORDS SKIPPED BY INDEX KEY CHECKING

Explanation: The length error of the index key is detected with FABP2025E. HD Pointer Checker ends the index key checking for this database data set group and reports the total number of records to be skipped.

System action: Processing continues.

Programmer response: Use correct DBD or sort records, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271.

FABP2028E DUPLICATE POINTERS AND SOURCE SEGMENTS FOUND

Explanation: Duplicate index pointers (T6 or T7) and source segments (TA) were detected for the same RBA in case of ITS=ISS.

System action: Processing continues.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Probable JCL error. Check the data sets containing the sort records.

FABP2030E MEMBER: *member-name* NOT FOUND IN *ddname* DATA SET

Explanation: A FIND macro was issued for the indicated *member-name*, and no module with that name was in the indicated *ddname* library.

System action: HD Pointer Checker ends the job with RC=8.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Make sure that the indicated *member-name* is in the correct library.

FABP2031E NO OR INCORRECT RECORDS FROM SCAN PROCESS IN DD: SORTEX01 OR SORTEX

Explanation: During the TYPE=CHECK process, HD Pointer Checker found that the SORTEX01 or SORTEX data set has no record or incorrect records. These records are expected to be created by the preceding TYPE=SCAN process.

System action: HD Pointer Checker ends with return code 08.

Programmer response: Specify the correct SORTEX01 or SORTEX data set and rerun the HD Pointer Checker job.

Problem determination: Check whether the SORTEX01 or SORTEX data set was created by the TYPE=SCAN process, or check whether the data set is specified in the JCL of the TYPE=CHECK process.

FABP2032W INDEX KEY CHECK NOT ALLOWED WITHOUT INDEX DATABASE SPECIFIED

Explanation: The index key checking option is not effective when no primary or secondary index database is specified.

System action: Processing continues as IXKEYCHK=NO is specified for the specified databases.

Programmer response: If you did not specify the DATABASE statement for an index database, correct the error and rerun the HD Pointer Checker job.

Problem determination: Index databases and associating primary databases must be specified on the DATABASE statement when running the HD Pointer Checker with IXKEYCHK=YES option.

FABP2035E TARGET OF SYMBOLIC INDEX POINTER NOT FOUND

Explanation: A symbolic pointer in secondary index was found but the target segment having the corresponding key was not found.

System action: Processing continues.

Programmer response: The database may be corrupted. Repair the database.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271.

FABP2036E TARGET OF SYMBOLIC LP POINTER NOT FOUND

Explanation: A symbolic LP pointer was found, but the target segment having the corresponding key was not found.

System action: Processing continues.

Programmer response: The database may be corrupted. Repair the database.

Problem determination: See Chapter 10, "Database repair guidelines," on page 271.

FABP2040W IMAGE COPY DATE OLDER THAN LATEST DBDS RECORD DATE FOR DB: *dbdname* PID: *xxxxx* DSG#: *xx*

Explanation: You attempted to process the indicated data set group *xx* of partition ID *xxxxx* of database *dbdname*, which is an image copy. But the creation date of the image copy data set is older than the latest database data set record date for the database data set in the HISTORY data set.

System action: HD Pointer Checker skips processing HISTORY data set for the database data set, and continues processing.

Programmer response: None.

Problem determination: None.

FABP2041E SPECIFIED SPMNMBR DD IS INCORRECT

Explanation: The SPMNMBR DD statement is not supported in HD Pointer Checker.

System action: HD Pointer Checker ends the job with RC=8.

Programmer response: None.

Problem determination: Specify a valid DD name, and rerun the utility.

FABP2042E (HDPC) STATEMENT NOT SPECIFIED

Explanation: There is no (HDPC) statement in the HPSRETCD data set. You must specify one (HDPC) statement.

System action: Processing stops.

Programmer response: Specify (HDPC) statement, and rerun the HD Pointer Checker job.

Problem determination: Check the HPSRETCD data set.

FABP2043I FOR SOME DBDS, HDPC RUNS WITH HISTORY OPTION MORE THAN *nn* TIMES A DAY

Explanation: Some database data set entries are not recorded, because the number of database data set entries of the day exceeds the maximum number 25.

System action: Processing continues. However, the update after the 25th entry will not be overridden or not be recorded.

Programmer response: None.

Problem determination: See message FABP2044W in the PROCCTL Statement report. The message shows that the data set reaches the maximum number of entries.

FABP2044W THE NUMBER OF DBDS ENTRIES FOR DB: *dbdname* DD: *ddname* IN THE HISTORY DATA SET REACHES *nn*. THE ENTRY IS NOT STORED

Explanation: The number of database data set entries for *dbdname* and *ddname* for today reached the maximum number per day.

System action: Processing continues. The current history information of the database data set is not recorded.

Programmer response: None.

Problem determination: None.

FABP2045W PARTITION DD: *ddname* OF DB: *dbdname* DB#: *nnn* IS NOT SPECIFIED IN DATABASE STATEMENT FOR LOGICALLY RELATED DB DBLG#: *xxx*

Explanation: The partition named is not specified with a DATABASE statement.

System action: Processing continues.

Programmer response: Always process every partition of the database that has direct logical parent pointers. If you omit any partitions, errors might be left undetected.

Problem determination: None.

FABP2046W SOME PARTITION OF DB: *dbdname* DB#: *nnn* IS NOT SPECIFIED IN DATABASE STATEMENT FOR LOGICALLY RELATED DB DBLG#: *xxx*

Explanation: Some partition of a HALDB is not specified with a DATABASE statement.

System action: Processing continues.

Programmer response: Always process every partition of the HALDB that has direct logical pointers. If you omit any partitions, errors might be left undetected.

Problem determination: None.

FABP2047W PHIDAM PRIMARY INDEX OF DB: *dbdname* PART: *partname* IS NOT SCANNED BECAUSE OF AN IMAGE COPY

Explanation: DATASET=IMAGECOPY is specified in the DATABASE statement of a primary index of PHIDAM database. The primary index is not checked.

System action: Processing continues.

Programmer response: The image copy of the PHIDAM primary index cannot be taken by image copy products. Therefore the specification DATASET=IMAGECOPY is ignored. Specify DATASET=REAL to check the primary index of PHIDAM database.

Problem determination: None.

FABP2048I *ddname1* DD STATEMENT IS IGNORED. *ddname2* DD STATEMENT IS USED

Explanation: Two work data sets were specified: one that is compatible with IMS HP Pointer Checker Version 1 (*ddname1*) and one for Version 2 (*ddname2*). The one for *ddname1* will be ignored.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2049I DBD NAME SPECIFIED IN EXEC PARAMETER IS IGNORED

Explanation: In an IMS ULU region, HD Pointer Checker ignores the DBD name specified on the PARM parameter on the EXEC statement and uses the DBDs specified on the DATABASE statements of the PROCCTL data set.

System action: Processing continues.

Programmer response: None.

Problem determination: When running in an IMS ULU region, HD Pointer Checker does not require the name of a DBD on the PARM parameter of the EXEC statement. To avoid getting this message, remove the DBD name from the PARM parameter.

FABP2050E "PROC TYPE= ALL" IS SPECIFIED "BLOCKDUMP" PARAMETER IS SPECIFIED TO SOME DSG'S BUT NOT ALL FOR DB: ***dbdname***

Explanation: An incorrect control statement is detected by the control statement syntax checking. Exclusive option is specified for each data set group of the indicated database *dbdname* when TYPE= ALL is specified.

System action: Processing stops. HD Pointer Checker ends the job to avoid the incomplete pointer checking.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: If you specify the BLOCKDUMP option, pointer checking is not done for the one data set group of the multiple data set group specified in the DATABASE statement. Because you request the pointer checking for the other data set group without the BLOCKDUMP option at the same run (TYPE= ALL).

FABP2051W DB: ***dbdname*** **DB#:** ***nnn*** **DD:** ***ddname*** **IS NOT SPECIFIED IN DATABASE STATEMENT FOR MULTIPLE DATA SET GROUP.**

Explanation: Multiple data set group was specified in the DATABASE statement, but the indicated multiple data set group of database *dbdname* was not specified in the DATABASE statement.

System action: Processing continues.

Programmer response: Always be sure to process every multiple data set group. If you omit a data set group, you are taking a risk that errors will go undetected.

Problem determination: None.

FABP2052W DB: ***dbdname*** **DB#:** ***nnn*** **IS NOT SPECIFIED IN DATABASE STATEMENT FOR LOGICALLY RELATED DB DBLG#:** ***xxx***

Explanation: Multiple databases of the logically related database group *xx* was specified on the DATABASE statement, the indicated database *dbdname* (database number: *nnn*) was not specified on a DATABASE statement. This message is also issued when the database type specified on DBORG parameter prevents the indicated database *dbdname* (database number: *nnn*) from being processed, even if the database was specified on a DATABASE statement.

System action: Processing continues.

Programmer response: Always be sure to process all logically related databases. If you omit a database, you are taking a risk that errors will go undetected.

Problem determination: None.

FABP2053E "PROC TYPE= ALL" IS SPECIFIED DB: ***dbdname*** **IS SPECIFIED AS TARGET OF INDEX DB BUT DATABASE STATEMENT FOR TARGET NOT FOUND**

Explanation: The indicated *dbdname* was specified on PRIMEDB parameter of the index DB DATABASE statement. However, the DATABASE statement of the indicated *dbdname* is not found.

System action: Processing stops.

Programmer response: Specify the DATABASE statement of the indicated *dbdname*, and rerun the HD Pointer Checker job.

Problem determination: None.

FABP2054E "PROC TYPE= ALL" IS SPECIFIED COMBINATION OF DB TYPES SPECIFIED IN "DBORG" PARAMETER IS INVALID FOR "PROC TYPE= ALL"

Explanation: The indicated database type specified in DBORG parameter must be run with logically related database type when TYPE= ALL is specified.

System action: Processing stops.

Programmer response: Specify the logically related database type in DBORG parameter, and rerun the HD Pointer Checker job. Or change TYPE parameter to SCAN, and rerun the HD Pointer Checker job.

Problem determination: None.

FABP2055E ***parm-value*** **IS INVALID FOR "*parm-name*" PARAMETER**

Explanation: An incorrect control statement was detected by the control statement syntax checking. The indicated *parm-value* is incorrect for the indicated *parm-name* parameter.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: None.

FABP2056E "PTRCHK=NO" CAN BE SPECIFIED ONLY IF "HASH=NO" AND "IXKEYCHK=NO"

Explanation: An incorrect control statement was detected in the control statement syntax checking. "HASH=YES or FORCE" or "IXKEYCHK= YES" PROC statement parameter and "PTRCHK=NO" OPTION statement parameter cannot be specified at the same run.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Check the combination error of the control statement.

FABP2057W LOGICALLY RELATED DATABASE GROUP DBLG#: ***xxx*** **ARE SCANNED WITH "HASH=NO"**

Explanation: The HASH=FORCE option was specified on the PROC statement, but all data sets of logically related database group *xx* were scanned with the HASH=NO option, because, the secondary index database was specified on the DATABASE statement.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2058E LOGICAL DATABASE CANNOT BE PROCESSED

Explanation: An incorrect control statement was detected by the control statement syntax checking. The *dbdname* in the DATABASE statement is the name of a logical DBD.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: The PSB may contain PCBs that reference logical DBDs. Since HD Pointer Checker processes only physical databases, your control statement must refer to the physical DBD.

FABP2059E SPECIFIED DATABASE WAS ALREADY FOUND IN ANOTHER DATABASE STATEMENT

Explanation: An incorrect control statement was detected by the control statement syntax checking. Duplicate DATABASE statements are found in the PROCCTL data set.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Only one DATABASE statement is specified for each database data set group in the PROCCTL data set.

FABP2060E DDNAME SPECIFIED IN "DD" PARAMETER NOT FOUND IN DMB

FABP2060E DDNAME SPECIFIED IN "OVERFLOW" PARAMETER NOT FOUND IN DMB

Explanation: The ddname in the DD parameter or OVERFLOW parameter on the control statement is not present in the IMS DMB control block.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: You may have entered the ddname incorrectly on your control statement. It is also possible that you used the wrong PSB name on the EXEC statement PARM. (You can also get this error by running HD Pointer Checker under a different release of IMS from the one used to install IMS HP Pointer Checker.)

FABP2061E "*parm-name*" PARAMETER IS INVALID FOR SPECIFIED DATABASE

Explanation: An incorrect control statement was detected by the control statement syntax checking. The indicated *parm-name* parameter on the DATABASE statement is incorrect.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Check the DATABASE statement for spelling errors or the combination error of parameters.

FABP2062E DBD NAME SPECIFIED IN "DB" PARAMETER NOT FOUND IN DMB

FABP2062E DBD NAME SPECIFIED IN "PRIMEDB" PARAMETER NOT FOUND IN DMB

Explanation: The *dbdname* in the DB parameter or PRIMEDB parameter on the control statement is not present in the IMS DMB control block.

With dynamic PSB generation, this error message also appears if you specify a database whose DBD member does not exist in the IMS data set.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: You may have entered the *dbdname* incorrectly on your control statement. It is also possible that you used the wrong PSB name on the EXEC statement PARM. (You can also get this error by running HD Pointer Checker under a different release of IMS than the one used to install IMS HP Pointer Checker.)

If this error resulted when an unrelated database was encountered on a control statement, remove the DATABASE statements that are not logically related to the first DATABASE statement. Put these statements, if related to each other, in a separate job and run it separately.

FABP2063E DD NOT FOUND FOR DDNAME SPECIFIED IN "DD" PARAMETER

FABP2063E DD NOT FOUND FOR DB: ***dbdname*** **DD:** ***ddname***

Explanation: Because there is no DD statement in your JCL stream that corresponds to the *ddname* that is specified in the control statement, dynamic allocation of the data set is not available. When TYPE=ALL or SCAN is specified in the PROCCTL statement, this message means that the DD name is not defined in the JCL DD statement, in the DFSMDA member, or in the RECON data set.

System action: Processing stops.

Programmer response: Correct your JCL by adding the missing DD statement, or prepare the environment for dynamic allocation, and rerun the HD Pointer Checker job.

Problem determination: If you specified a DD statement, it is probably a JCL error. Check for spelling mistakes. If you did not specify a DD statement, check DFSMDA or RECON.

FABP2064E "BLOCKDUMP" PARAMETER IS INVALID FOR "PROC TYPE=CHECK"

FABP2064E "BLOCKDUMP" PARAMETER IS INVALID FOR "PROC TYPE=BLKMAP"

Explanation: An incorrect control statement was detected by the control statement syntax checking. The BLOCKDUMP parameter on the DATABASE statement cannot be specified when TYPE= CHECK or BLKMAP.

System action: Processing stops.

Programmer response: Correct all errors and rerun the job.

Problem determination: The BLOCKDUMP parameter can be specified when TYPE= ALL or SCAN is specified.

FABP2065I IBUFF|VSAMBF PARAMETER: ***nnnnn*** **OVERRIDDEN BY BUFND|BUFNO PARAMETER:** ***mmmmm*** **SPECIFIED IN JCL FOR PRIM DD:** ***dddddddd***

FABP2065I IBUFF|VSAMBF PARAMETER: ***nnnnn*** **OVERRIDDEN BY BUFND|BUFNO PARAMETER:** ***mmmmm*** **SPECIFIED IN JCL FOR OVER DD:** ***dddddddd***

Explanation: The VSAMBF or IBUFF parameter is overridden by the AMP=('BUFND=') or DCB=BUFNO= parameter specified in the JCL for the primary DD statement or the overflow DD statement. Here, *nnnnn* is the number of buffers specified by the VSAMBF parameter or the buffer size specified by the IBUFF parameter, and *mmmmm* is the number of buffers specified by the BUFND or BUFNO parameter.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2066E DUMMY OR NULLFILE SPECIFIED FOR DDNAME: ***ddname***

Explanation: DSN=NULLFILE or DUMMY parameter specified for the DD.

System action: Processing stops.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Make sure that a DD statement is present for the ddname indicated and that the correct data set is identified.

FABP2067E INVALID NUMBER OF OPERANDS SPECIFIED FOR "*parm-name*" PARAMETER

Explanation: An incorrect control statement was detected when the syntax of the control statement was checked. The number of operands in the indicated *parm-name* parameter on the control statement is incorrect.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Check the control statement for the combination error in the parameter.

FABP2068E "HASH=YES OR FORCE" AND *parm-name*=YES CANNOT BE SPECIFIED AT THE SAME TIME

Explanation: An incorrect control statement was detected by the control statement syntax checking. HASH=YES or FORCE and *parm-name*= YES cannot be specified in the PROC statement at the same time.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Check the combination error of the PROC statement.

FABP2069E MORE THAN ONE "*statement-name*" STATEMENT SPECIFIED

Explanation: An incorrect control statement was detected by the control statement syntax checking. You have specified more than one *statement-name* statement in the PROCCTL or HPSRETCD data set.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: There must be only one *statement-name* statement.

FABP2070E "PROC" STATEMENT NOT SPECIFIED

Explanation: An incorrect control statement was detected by the control statement syntax checking. No PROC statement is specified in the PROCCTL data set.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: There must be only one PROC statement, and it must be the first control statement in the PRCCTL data set.

FABP2071E "*parm-name*" PARAMETER IS REQUIRED FOR SPECIFIED DATABASE

Explanation: An incorrect control statement was detected by the control statement syntax checking. No indicated *parm-name* parameter is specified on the DATABASE statement.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Specify the ddname (as coded in DBD) of the overflowed data set for HISAM or index database in OVERFLOW parameter. Specify the *dbdname* of the primary database indexed by the HIDAM index or secondary index database in PRIMEDB parameter.

FABP2072W INDEX KEY CHECK NOT ALLOWED FOR DBNAME: *dbdname* REASON: /CK FIELD IS SPECIFIED

FABP2072W INDEX KEY CHECK NOT ALLOWED FOR DBNAME: *dbdname* REASON: MISSING PRIME DB STATEMENT

FABP2072W INDEX KEY CHECK NOT ALLOWED FOR DBNAME: *dbdname* REASON: "SPIXCHK=NO" IS SPECIFIED

FABP2072W INDEX KEY CHECK NOT ALLOWED FOR DBNAME: *dbdname* REASON: "EPSCHK=NO" IS SPECIFIED

Explanation: The index key checking option is not effective for the indicated database *dbdname* for the reason given.

System action: Processing continues as if IXKEYCHK=NO is specified for the indicated database *dbdname*.

Programmer response: If you did not specify the DATABASE statement for the primary database, correct the error and rerun the HD Pointer Checker job.

Problem determination: Index databases and associated primary databases must be specified by the succeeding DATABASE statements.

FABP2073E MISSING "DATABASE" STATEMENT

Explanation: An incorrect control statement was detected by the control statement syntax checking. The data set associated with the PROCCTL DD statement had no DATABASE statement, or there was no DATABASE statement that preceded an END statement.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: You must specify the DATABASE statement, unless TYPE= CHECK or BLKMAP is specified on the PROC statement.

FABP2074E MISSING CONTROL STATEMENT

Explanation: An incorrect control statement was detected by the control statement syntax checking. The data set associated with the PROCCTL DD statement was empty or a DUMMY data set.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: None.

FABP2075E MORE THAN ONE "OPTION" STATEMENT SPECIFIED AFTER "PROC" OR "DATABASE" STATEMENT

FABP2075E MORE THAN ONE "REPORT" STATEMENT SPECIFIED AFTER "PROC" OR "DATABASE" STATEMENT

Explanation: An incorrect control statement was detected by the control statement syntax checking. More than one OPTION/REPORT statement was specified for a DATABASE statement or a PROC statement.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: None.

FABP2076E SPECIFIED PRIMARY DATABASE IS NOT HIDAM OR HDAM

Explanation: Specified *dbdname* on PRIMEDB parameter of DATABASE statement is not the name of a valid HIDAM or HDAM DBD.

System action: Processing stops.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: You probably specified the wrong *dbdname* on the PRIMEDB parameter. It is also possible that you specified the wrong DBD library.

FABP2077W "*parm-name*" PARAMETER IS INVALID FOR SPECIFIED "TYPE" PARAMETER

Explanation: The indicated option parameter *parm-name* is not effective for specified TYPE parameter.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2078E SPECIFIED OPTION OF "*parm-name*" PARAMETER IS INVALID FOR "*HISTORY=YES*"

Explanation: An incorrect control statement is detected by the control statement syntax checking. The indicated option parameters "*parm-name*" cannot be specified with "HISTORY=YES" at the same run. This message is issued when:

- "HASH=YES or FORCE" option parameter of the PROC statement is effective at run time.
- "BLOCKDUMP" option parameter of the DATABASE statement is specified.
- "HOMECHK=NO", "INCORE=NO", or "NOCHKP" option parameters of the OPTION statement is specified.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Check the combination error of the control statement.

FABP2079E DD STATEMENT MISSING FOR DDNAME:*ddname*

Explanation: Required DD statement of work data set is not specified for the DD.

System action: Processing stops.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Make sure that a DD statement is present for the ddname indicated and that the correct data set is identified.

FABP2081E MEMBER "*xxxxxxxx*" SPECIFIED IN "DBDEFCTL=" IS INVALID

Explanation: The member name specified by the DBDEFCTL= keyword on the PROC statement is incorrect.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

FABP2082E "*xxxxxxxx*" CANNOT BE USED AS DBDEFCTL MEMBER NAME

Explanation: The control information member name specified by the DBDEFCTL= keyword is the same as the database name.

System action: Processing stops.

Programmer response: Specify another member name for the control information member, and rerun the HD Pointer Checker job.

FABP2083W INDEX KEY CHECK CANNOT VALIDATE FOR INDEX DB#: *nn* **ISS SC:** *xx*

Explanation: The index key check function cannot validate the index pointers and the key data of the indicated primary/secondary index database (database number: *nn*), because at least one index source segment (segment code of index source segment: *xx*) is split to the prefix part and data part. The RBA of index pointer points to the prefix part and differs from the RBA of the data part, which has the key of an index source segment.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2084I I/O BUFFER OF DD: *ddname***(CI/BLOCK SIZE:** *xxxxx***) IS** *yyyy* **KBYTE.**

Explanation: *xxxxx* is the CI or block size of the input data set. *yyyy* is the actual size of I/O buffer.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2085E "*parm-name***" PARAMETER IS INVALID FOR SPECIFIED "TYPE" PARAMETER**

Explanation: The option parameter *parm-name* is not effective for the TYPE parameter specified.

System action: Processing stops.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Check the combination error of the control statement.

FABP2086I SHARED INDEX DATABASE DB: *dbdname* **DD:** *ddname* **IS SCANNED IN SCANGROUP:** *n*

Explanation: More than two shared index databases share one data set. Different scan group numbers, however, were assigned to the same data set. The parameter of SCANGROUP= is ignored, and the second or later *dbdname ddname* of the shared index will be used in the first scan group *n*.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2087W PARAMETER "*old-statement***" IS OBSOLETE AND IGNORED**

Explanation: The specification is ignored, and *old-statement* is not used.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2088W PARAMETER "*old-statement***" IS OBSOLETE AND TREATED AS "***new-statement***"**

Explanation: The parameter *old-statement* will not be used. It is treated as *new-statement*.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2089E PARAMETER 'KEYSIN=YES' MUST BE SCANNED IN THE SAME SCAN TASK

Explanation: More than two data sets have root segments with KEYSIN=YES specified, however, different scan group numbers are assigned to them.

System action: Processing stops.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: All data sets having root segments with KEYSIN=YES specified must be assigned to the same group.

FABP2090E 'EPSCHK' PARAMETER MUST BE SAME BETWEEN SCAN PROCESSES

Explanation: EPSCHK= parameter is different for different SCAN processes.

System action: Processing stops

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: EPSCHK parameter must be the same in every SCAN process.

FABP2091E PARTITION NAME SPECIFIED IN 'PART' PARAMETER NOT FOUND IN PTE

Explanation: The partition name in the PART parameter of the control statement is not present in the IMS PTE control block.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: You may have entered the partition name incorrectly on your control statement.

FABP2092E PARTITION NUMBER SPECIFIED IN 'NUM' PARAMETER IS OVER MAX VALUE

Explanation: The partition number in the NUM parameter on the control statement exceeds the maximum value.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Check the partition number specified in the NUM parameter of DATABASE statement.

FABP2093E DDNAME SPECIFIED IN 'DD' PARAMETER IS INVALID

Explanation: The ddname in the DD parameter on the control statement is incorrect.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Check the length specified in the DD parameter of the DATABASE statement. The ddname must be a 1-digit value, and the data set group ID, A,B,C,...J,L,X, must be specified.

FABP2095E ONLINE REORG IS ACTIVE FOR DB: ***dbname*** **PART:** ***partname***

Explanation: The partition (*partname*) of HALDB (*dbname*) cannot be processed because an online reorganization has not completed.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: None.

FABP2096W SOME SYMBOLIC LP POINTERS IN DB: ***dbdname*** **ARE NOT CHECKED**

Explanation: The symbolic LP pointers that point to dependent segments are not checked. Only symbolic LP pointers that point to the root segments can be checked by specifying SYMLPCHK=YES.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2097W SYMBOLIC INDEX POINTER IN DB: ***dbdname*** **IS NOT CHECKED**

Explanation: The symbolic pointers in the secondary index database are not checked because they point to dependent segments. Only symbolic pointers that point to root segments can be checked by SYMIXCHK=YES.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2098E DBRC IS REQUIRED TO PROCESS HALDB DB: ***dbdname***

Explanation: When the processing database is a HALDB, DBRC should be activated. However, DBRC was not active.

System action: Processing stops.

Programmer response: Specify Y for the 14th parameter on the EXEC statement of the HD Pointer job step. For example,

```
//HDPCPRO EXEC PGM=DFSRRC00,
// PARM='ULU,FABPMAIN,,,,,,,,,,,,Y,N'
```

Problem determination: None.

FABP2099I RETURN CODE ***nn*** **WILL BE RETURNED IF** ***parm-name*** **IS DETECTED**

Explanation: The return code is changed when the *parm-name* event is detected. *parm-name*: T2ERROR, DBERROR, or PROCERROR. This message is shown in the HPSRETCD statement report.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2100E ILK IN EPS NOT FOUND IN ILDS

Explanation: Indirect List Entry key (ILK) in Extended Pointer Set (EPS) was not found in the Indirect List Data set (ILDS).

System action: Processing continues.

Programmer response: Repair the databases, and rerun the HD Pointer Checker job.

Problem determination: ILK in EPS must be in ILDS as the key of ILE in ILDS.

FABP2101E ILK IN EPS IS DIFFERENT FROM ILK OF TARGET SEGMENT

Explanation: Indirect List Entry key (ILK) in the Extended Pointer Set (EPS) was different from ILK of the TARGET segment.

System action: Processing continues.

Programmer response: Repair the databases, and rerun the HD Pointer Checker job.

Problem determination: ILK in EPS must be the same as the ILK of the TARGET segment.

FABP2103W EPS CANNOT BE CHECKED WITHOUT SCANNING DSG 'A' OF TARGET

Explanation: The EPS cannot be checked, because the data set group A of the target partition for the pointers, which reside in the EPS, is not scanned. The reorganization number in the bitmap block is required for EPS check process. The data set group must be included in the processed data sets with 'EPSCHK=YES' is specified.

System action: Processing continues.

Programmer response: If you want to run HD Pointer Checker with 'EPSCHK=YES', specify data set group A of the target partition on DATABASE statement.

Problem determination: None.

FABP2104E THE SEQUENCE OF DATABASE STATEMENTS AMONG MULTIPLE SCAN STEP JOBS CONFLICT WITH EACH OTHER

Explanation: Databases with neither logical nor index relation were processed while the scan steps (TYPE=SCAN steps), which are composed of multiple jobs, were running in a ULU region. The order of DBLG (Data Base Logical Group)—that is, databases with either logical or index relations—among the scan step jobs is not correct.

System action: Processing stops.

Programmer response: Do one of the following three actions:

- Run with TYPE=ALL.
- Run the scan step in a single job.
- Specify databases with either logical or index relations (DBLG) in the same order within multiple scan step jobs.

  Having multiple independent databases means there are multiple DBLGs. When specifying the multiple DBLGs in multiple scan-step jobs, the DBLGs must be specified in the same order for the DATABASE statements in the PROCCTL statement.

  Example:

  There are two DBLGs X and Y: X contains DB1, DB2 and Y contains DBA, DBB. If you want to process the DBLGs in two scan-step jobs, you must specify in the order DBLG X,Y, such as DB1,DBA for one job, and DB2,DBB for another. In this way, for both jobs, the DBLG is specified in the same order—that is for both job, the first DBLG specified is in X and the next one is in Y. But if, for example, you specify DB1,DB2 for one job and DBA,DBB for the other, this will cause an error because the order of the DBLG in the first job begins with X, but in the second job it begins with Y.

Problem determination: None.

FABP2105E ″*XXXXXXXX*″ **PARAMETER IS INCORRECT**

Explanation: An operand is missing in the USER= keyword parameter.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Check the control statement.

FABP2106E DUPLICATE OPERANDS FOR ″*xxxxxxxx*″ **PARAMETER**

Explanation: Two or more same user IDs are specified for USER=.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Check the control statement.

FABP2107E **″*NO″ CANNOT BE SPECIFIED WITH USERID**

Explanation: Both ″*NO″ and USERID are specified.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Check the control statement.

FABP2108I T2 OR POINTER ERRORS NOTIFICATIONS WERE SENT TO TSO USERS: *xxxxxxx*,

Explanation: Notification messages of T2 (unknown data) or pointer errors have been sent to the TSO user IDs.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2109I *mm/dd/yyyy hh:mm:ss xxxxxxx* **ERROR DB:** *dbname* **DD:** *ddname* **JOBNAME:** *jobname*

Explanation: HD Pointer Checker job (*jobname*) detected pointer errors or unknown data and sent this message to TSO users.

System action: Processing continues.

Programmer response: Repair the database.

Problem determination: Check the errors in the HD Pointer Checker reports.

FABP2110E LIU IS NOT INSTALLED OR NOT IN REQUIRED LEVEL

Explanation: IMS Library Integrity Utilities is not found in any library concatenated to the STEPLIB DD, or the maintenance level of IMS Library Integrity Utilities is not supporting the requested IMS Library Integrity Utilities service. For details, see “Software requirements” on page 6.

System action: Processing stops.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: If you want to use the requested IMS Library Integrity Utilities service, install IMS Library Integrity Utilities and apply the required maintenance. Otherwise remove the control statement that requests the IMS Library Integrity Utilities service.

FABP2111E [DECODE DBD | MAP DBD] PROCESSING FAILED WITH RC=*cc*

Explanation: IMS Library Integrity Utilities returned a nonzero return code.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: If RC=8, check the Messages report of DBSRCPRT or DBMAPPRT data set. If RC=16, check message FABL*nnnn*E, issued to the console.

FABP2112E SPMN PROCESS IS BYPASSED. BECAUSE HPIC MAINTENANCE LEVEL IS LOW.

Explanation: Space Monitor cannot be run, because IMS HP Image Copy is not in the required maintenance level.

System action: Image Copy process by IMS HP Image Copy ends with an error.

Programmer response: None.

Problem determination: Apply the required maintenance to IMS HP Image Copy.

FABP2113E INCORRECT DATA SET: *dsname* **IS SPECIFIED IN DD:** *ddname*

Explanation: A real database data set is allocated in the DD *ddname* when DATASET=IMAGECOPY is specified in the PROCCTL data set.

System action: Processing stops.

Programmer response: Specify a correct image copy data set name to HD Pointer Checker JCL. Or, if you want to use the latest image copy data set, the *ddname* DD is not required. Remove the *ddname* DD and specify the RECON data sets to the HD Pointer Checker JCL, and rerun the HD Pointer Checker job.

Problem determination: Check the *ddname* DD statement in the HD Pointer Checker JCL and the DATABASE statement in the PROCCTL data set.

FABP2114I *mm***/***dd***/***yyyy hh***:***mm***:***ss* **THE REST OF THE MESSAGES ARE SUPPRESSED JOBNAME:** *jobname*

Explanation: The number of exception notification messages (FABP2109I) reached the limit of 50. The rest of the messages are not sent to TSO users.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2115W THE ITKBLOAD PARAMETER WAS IGNORED BECAUSE NO SERVER NAME WAS SPECIFIED

Explanation: An IMS Tools KB load module library was specified on the ITKBLOAD parameter, but the specification was ignored because no server name was specified on the ITKBSRVR parameter.

System action: Processing continues, but HD Pointer Checker does not store any reports to the IMS Tools KB Output repository.

Programmer response: If you want to store the reports to the IMS Tools KB Output repository, specify the IMS Tools KB server name on the ITKBSRVR parameter.

Problem determination: None.

FABP2117W THE ITKBLOAD PARAMETER IS NOT USED UNTIL A SERVER NAME IS SPECIFIED

Explanation: In the Site Default Generation utility, a data set name of IMS Tools KB load module was specified on the ITKBLOAD parameter, but server name was not specified. HDPC will not use this value until server name is specified.

System action: Processing continues.

Programmer response: If you want to store the reports to the IMS Tools KB Output repository, specify the server name on the ITKBSRVR parameter by using the Site Default Generation utility or at run-time.

Problem determination: None.

FABP2119E THE LATEST IMAGE COPY RECORD FOR DD: *ddname* **WAS USER IC RECORD**

Explanation: HD Pointer Checker attempted to get the image copy data set name for the *ddname* from the latest image copy record in RECON data set, but the image copy record was a user IC record. HD Pointer Checker supports only the standard IC records.

System action: Processing stops.

Programmer response: Specify the DD statement for the image copy data set and rerun the HD Pointer Checker job.

Problem determination: None.

FABP2120W RETURN CODE 04 IS RETURNED FROM VSAM OPEN DB: *dbdname* **DD:** *ddname*

Explanation: When opening the database data set specified by *dbdname* and *ddname*, the VSAM data set was opened successfully, but an attention message was issued from the OPEN macro.

System action: Processing continues.

Programmer response: To avoid getting this message, resolve the attention of VSAM OPEN. If the attention is not a problem to your system, you can ignore it.

Problem determination: The reason code and data set name is shown in IEC161I. For the details of the reason code, see *DFSMS Macro Instructions for Data Sets* (SC26-7408).

FABP2121I T2 OR POINTER ERRORS NOTIFICATIONS WERE CANCELED, BECAUSE USERS WERE NOT LOGGED ON OR TERMINAL DISCONNECTED

Explanation: HD Pointer Checker detected pointer errors or T2 errors (unknown data), and attempted to send the notification message to TSO user IDs, but failed to send the message. The TSO users were not logged on or were disconnected. The notification messages were discarded.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2122I MARKED AS RECOVERY NEEDED FOR DB: *dbdname* **PART:** *partname*

Explanation: The database *dbdname* or the partition *partname* of the database *dbdname* is marked as recovery needed in the RECON data set.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP2140I SITE DEFAULT TABLE SOURCE CODE IS GENERATED

Explanation: The site default table source code was generated successfully.

System action: This message reports that the Site Default Generation utility job ended normally with RC=00.

Programmer response: None.

Problem determination: None.

FABP2141E PROCCTL CONTROL STATEMENT ERROR

Explanation: The site default table source code was not generated, because there are errors in the PROCCTL data set.

System action: The Site Default Generation utility ended the job with RC=8.

Programmer response: Correct the control statements and rerun the job.

Problem determination: Check the errors for the PROCCTL Statements report.

FABP2142I THE FOLLOWING STATEMENTS ARE IGNORED

Explanation: The Site Default Generation utility ignored the control statements that follow this message when it set the site defaults.

System action: The Site Default Generation utility sets the site default values from the statements preceding this message, and ignores the statements that follow this message.

Programmer response: None.

Problem determination: None.

FABP2143I KEYWORD: ***keyword*** **IS IGNORED**

Explanation: A site default value cannot be specified for keyword *xxxxxxx*. It is ignored.

System action: The Site Default Generation utility skips the keyword, and it sets the site default values for other keywords.

Programmer response: None.

Problem determination: None.

FABP2144E NO KEYWORD TO GENERATE THE SITE DEFAULT TABLE

Explanation: The site default table source code was not generated, because there is no valid specification in the PROCCTL data set.

System action: The Site Default Generation utility ends the job with RC=08.

Programmer response: Specify the keywords that are valid to set the site default values in the PROCCTL data set. For available keywords for site defaults, see "FABPTGEN PROCCTL data set" on page 246.

Problem determination: None.

FABP2145E TYPE=CHECK OR BLKMAP CANNOT BE SPECIFIED IN THE SITE DEFAULT TABLE

Explanation: The site default table source code was not generated, because TYPE=CHECK or TYPE=BLKMAP is specified in the PROC control statement in the PROCCTL data set. They cannot be specified to Site Default Generation utility.

System action: Site Default Generation utility ends with RC=08.

Programmer response: Specify TYPE=ALL or TYPE=SCAN in the PROC control statement in the PROCCTL data set.

Problem determination: Check the error for the PROCCTL data set.

FABP2146E *parm-values* **IS INCORRECT FOR PARM PARAMETER OF EXEC STATEMENT**

Explanation: An incorrect value was specified for the EXEC parameter in the Site Default Generation utility JCL.

System action: The Site Default Generation utility ends with RC=08.

Programmer response: Specify PARM='GEN' or PARM='REPORT' for the EXEC statement.

Problem determination: None.

FABP2150I SITE DEFAULT TABLE FABPCTL0 IS USED

Explanation: HD Pointer Checker used the site default table module (FABPCTL0).

System action: HD Pointer Checker loads the site default table module (FABPCTL0) and sets the site default values that are specified in it.

Programmer response: None.

Problem determination: None.

FABP2151E SITE DEFAULT TABLE FABPCTL0 IS NOT FOUND

Explanation: The site default values were not reported, because the site default table module (FABPCTL0) is not in the STEPLIB data sets.

System action: The Site Default Generation utility ends with RC=08.

Programmer response: Specify the data set that includes the site default table module (FABPCTL0) member to the STEPLIB statement.

Problem determination: None.

FABP2152E SITE DEFAULT TABLE FABPCTL0 IS CORRUPTED

Explanation: The site default table module (FABPCTL0) is corrupted. If this message appears in the Site Default Generation utility job, the Site Default Values report is not generated. If this message appears in the HD Pointer Checker job, site default values are not used.

System action: The Site Default Generation utility or HD Pointer Checker ends with RC=08.

Programmer response: Specify the correct FABPCTL0 module. If it is damaged, recreate another site default table module and store it in the STEPLIB data set.

Problem determination: Check the FABPCTL0 member in the STEPLIB data set.

FABP2601E LOAD FAILED FOR DDNAME: ***ddname*** **MODULE:** ***module-name***

Explanation: After issuing the LOAD macro to load module *module-name* from the *ddname* data set, register 15 contained a nonzero return code.

System action: The Site Default Generation utility ends with RC=08.

Programmer response: Correct the error and rerun the job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Make sure that the DD statement is specifying the correct data set.

FABP2602W THE ACCESS TO OUTPUT REPOSITORY WAS CANCELED REASON: *reason*

Explanation: HD Pointer Checker canceled its access to the IMS Tools KB Output repository because the initialization process failed. The reason is one of the following:

- Dynamic allocation for the library in the ITKBLOAD parameter failed
- Failed to open the library specified in the ITKBLOAD parameter
- Failed to load the HKTXXLI module

System action: Processing continues, but HD Pointer Checker does not store any reports to the IMS Tools KB Output repository.

Programmer response: If you want to store the reports to the IMS Tools KB Output repository, specify the correct load module library of the IMS Tools KB product.

Problem determination: Check the following to see if the IMS Tools KB product load module library name is correct:

- The specification of the ITKBLOAD parameter on the PROC statement in the PROCCTL data set
- STEPLIB, JOBLIB, or LINKLIST concatenations

FABP2603E GETMAIN FAILED. SIZE: *nn* **K ITEM-ID:** *xx*

Explanation: After issuing a GETMAIN macro to get the size *nn* K bytes of storage, register 15 contained a nonzero return code. The ITEM-ID *xx* represents the internal location where the GETMAIN macro was issued.

System action: Site Default Generation utility ends with RC=08.

Programmer response: Increase the region size parameter in the JCL and rerun the job.

Problem determination: None.

FABP2604E FREEMAIN FAILED. ITEM-ID: *xx*

Explanation: An internal error occurred during the FREEMAIN process for internal storage.

System action: The Site Default Generation utility ends with RC=08.

Programmer response: Save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any runtime errors.

FABP2605E OPEN FAILED FOR DDNAME: *ddname*

Explanation: The OPEN macro failed when opening the data set that is associated with *ddname* DD.

System action: The Site Default Generation utility ends with RC=08.

Programmer response: Correct the error and rerun the job.

Problem determination: Check that the *ddname* indicated in the DD statement is specifying the correct data set.

FABP2606E CLOSE FAILED FOR DDNAME: *ddname*

Explanation: An internal error occurred. The CLOSE macro failed when closing the data set that is associated with *ddname* DD.

System action: The Site Default Generation utility ends with RC=08.

Programmer response: Correct the error and rerun the job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any runtime errors.

FABP2607E DATA SET: *dsname* **IS NOT FOUND**

Explanation: The *dsname* data set was not found. The data set name is specified in RETCDDSN= in the PROCCTL data set or the site defined as the site default.

System action: Processing stops.

Programmer response: If the data set name is stored in the site default table module (FABPCTL0), correct the data set name and run the Site Default Generation utility. If the data set name is specified in the PROCCTL data set in the HD Pointer Checker JCL, specify the correct data set name. Then, rerun the HD Pointer Checker job.

Problem determination: Check that the data set name is correct. If it is correct, check that the data set name is cataloged correctly, because the data set name must be cataloged.

FABP2608W ERROR OCCURRED IN ACCESSING OUTPUT REPOSITORY FUNC: *function* **RC:** *rc* **RSN:** *rsn*

Explanation: An error occurred while getting access to the IMS Tools KB Output repository.

System action: Processing continues. If the return code is equal to or greater than 08, HD Pointer Checker does not store its reports to the IMS Tools KB Output repository.

Programmer response: If you want to store the reports to the IMS Tools KB Output repository, correct the error.

Problem determination: If any of the messages, FABP2611W, FABP2612W, FABP2613W, or FABP2614W, which describes the cause of the error, are issued following this message, see the explanation for those messages. If the above messages are not issued, check the return code and the reason code specified in this message. The codes are in hexadecimal. For the description of the return code and reason code, see *IMS Tools Knowledge Base for z/OS User's Guide* (SC18-9963).

FABP2609W DB: *dbdname* **PART:** *partname* **DD:** *ddname* **REPORT:** *report name*

Explanation: This message follows the FABP2608W message.

System action: Processing continues. If the return code in the FABP2608W message is equal to or greater than 08, HD Pointer Checker does not store the *report name* report for the database name, partition name, and DD name that are shown in this message.

Programmer response: See the description for message FABP2608W.

Problem determination: See the description for message FABP2608W.

FABP2611W RECON ENTRY WAS NOT FOUND IN ITKB

Explanation: The RECON entry was not defined in your IMS Tools Knowledge Base information management environment.

System action: Processing continues, but HD Pointer Checker does not store any reports to the IMS Tools KB Output repository.

Programmer response: If you want to store the reports to the IMS Tools KB Output repository, add a RECON environment. For more information about adding the RECON environment, see *IMS Tools Knowledge Base for z/OS User's Guide* (SC18-9963).

Problem determination: Check the RECON information on the IMS Tools Knowledge Base panel of the ISPF dialogue. For more details of the IMS Tools Knowledge Base panel, see *IMS Tools Knowledge Base for z/OS User's Guide* (SC18-9963).

FABP2612W ITKB SERVER NAME WAS INCORRECT

Explanation: The connection to the IMS Tools KB server failed, because the server name specified by the ITKBSRVR parameter in the PROC control statement was incorrect.

System action: Processing continues, but HD Pointer Checker does not store any reports to the IMS Tools KB Output repository.

Programmer response: If you want to store the reports to the IMS Tools KB Output repository, specify the correct IMS Tools KB server name.

Problem determination: Check the IMS Tools KB server name on the ITKBSRVR parameter.

FABP2613W HPPC WAS NOT DEFINED IN ITKB

Explanation: IMS HP Pointer Checker was not defined in the IMS Tools Knowledge Base information management environment as a product that can store reports to the IMS Tools KB Output repository.

System action: Processing continues, but HD Pointer Checker does not store any reports to the IMS Tools KB Output repository.

Programmer response: If you want to store the reports to the IMS Tools KB Output repository, register the IMS HP Pointer Checker product by using the IMS Tools KB product administration utility (HKTAPRA0).

Problem determination: Check the listing of registered products by using the LIST command of the IMS Tools KB HKTAPRA0 utility. For more details of HKTAPRA0, see *IMS Tools Knowledge Base for z/OS User's Guide* (SC18-9963).

FABP2614W REPORT WAS NOT DEFINED IN ITKB

Explanation: The report was not defined in the IMS Tools Knowledge Base information management environment.

System action: Processing continues, but HD Pointer Checker does not store the report to the IMS Tools KB Output repository.

Programmer response: If you want to store the report to the IMS Tools KB Output repository, register the report by using the IMS Tools KB product administration utility (HKTAPRA0).

Problem determination: Check the listing of registered products and reports by using the LIST command of the IMS Tools KB HKTAPRA0 utility. For more details of HKTAPRA0, see *IMS Tools Knowledge Base for z/OS User's Guide* (SC18-9963).

FABP3504E MESSAGE TEXT NOT FOUND

Explanation: HD Pointer Checker attempted to print an error message that could not be found in the message table in module FABUMSGS.

System action: HD Pointer Checker issues a USER 3504 abend.

Programmer response: Verify that IMS HP Pointer Checker, including all current maintenance, was installed correctly. Reinstall IMS HP Pointer Checker, including all maintenance, if necessary. Then rerun the job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Either IMS HP Pointer Checker or its maintenance may have been installed incorrectly.

FABP3505E INVALID MESSAGE FLAG

Explanation: HD Pointer Checker attempted to print an error message, and an incorrect flag was supplied to module FABUMSGS.

System action: HD Pointer Checker issues a USER 3505 abend.

Programmer response: Verify that IMS HP Pointer Checker, including all current maintenance, was installed correctly. Reinstall IMS HP Pointer Checker, including all maintenance, if necessary. Then rerun the job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Either IMS HP Pointer Checker or its maintenance may have been installed incorrectly.

FABP3509E NUMBER OF ENTRIES FOR BLOCK MAP EXCEED SIZE OF GETMAIN AREA

Explanation: A database block (or control interval) contains more segments and free space elements than module’s internal tables allow.

System action: HD Pointer Checker issues a USER 3509 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Verify if there are too many segments in the block. Otherwise, check for run-time errors.

FABP3510E MORE THAN *nnnn* DATABASES ARE REFERRED IN PSB: *psbname*

Explanation: The number of databases entered in the PSB *psbname* exceeds the limit *nnnn*. The limit is 2500.

Programmer response: Redefine the PSB not to exceed the limit number of referred database data set, and rerun the HD Pointer Checker job.

Problem determination: None.

FABP3513E INVALID USER STATISTICS INTERVAL SPECIFIED

Explanation: An internal error occurred.

System action: HD Pointer Checker issues a USER 3513 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors.

FABP3565E UNSUCCESSFUL GENCB FOR VSAM EXLST

Explanation: After issuing a GENCB macro to create an EXLST control block, register 15 contained a nonzero return code.

System action: HD Pointer Checker issues a USER 3565 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Use standard VSAM debugging techniques.

FABP3566E UNSUCCESSFUL GENCB FOR VSAM ACB

Explanation: After issuing a GENCB macro to create an ACB control block, register 15 contained a nonzero return code.

System action: HD Pointer Checker issues a USER 3566 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Use standard VSAM debugging techniques.

FABP3567E UNSUCCESSFUL GENCB FOR VSAM RPL

Explanation: After issuing a GENCB macro to create an RPL control block, register 15 contained a nonzero return code.

System action: HD Pointer Checker issues a USER 3567 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Use standard VSAM debugging techniques.

FABP3568E DATABASE NAME SPECIFIED IN CONTROL STATEMENT NOT FOUND IN DMB

Explanation: The *dbdname* in column 1 on your HD Pointer Checker control statement is not present in the IMS DMB control block.

System action: HD Pointer Checker issues a USER 3568 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: It is possible that you entered the DBD name incorrectly on your control statement. It is also possible that you used the wrong PSB name on the EXEC statement PARM. (You can also get this error by running HD Pointer Checker under a release of IMS different from the one used to install IMS HP Pointer Checker.)

FABP3569E DDNAME IN CTL STMT NOT FOUND IN DMB

Explanation: The ddname in column 11 on your HD Pointer Checker control statement is not present in the IMS DMB control block.

System action: HD Pointer Checker issues a USER 3569 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: You may have entered the ddname incorrectly on your control statement. It is also possible to get this error by running HD Pointer Checker under a release of IMS different from the one used to install IMS HP Pointer Checker.

FABP3570E FIND FAILED FOR DBD IN DBDLIB

Explanation: A FIND macro was issued for the *dbdname*, and no module with that name was in the library on your IMS DD statement.

System action: HD Pointer Checker issues a USER 3570 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Probable JCL error. Make sure your DBD library is part of the IMS DD statement. Check your control statement for spelling errors.

FABP3571E SEGMENT CODE NOT FOUND IN DBD SEGMENT TABLE

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3571 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3572E EXPECTED PHYSICAL PARENT NOT IN SEGMENT INFORMATION TABLE

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3572 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3573E MORE THAN 1 LOGICAL PARENT

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3573 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker and contact IBM Software Support.

Problem determination: Check for obvious run-time errors. Make sure the DBD and PSB are valid.

FABP3574E PHYSICAL PARENT NOT FOUND IN SEGMENT INFORMATION TABLE

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3574 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3575E DATABASE NAME SPECIFIED IN CONTROL STATEMENT NOT FOUND IN IMAGE COPY HEADER

Explanation: The *dbdname* on your control statement is not contained in the first record of your image copy data set.

System action: HD Pointer Checker issues a USER 3575 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: You probably entered the wrong *dbdname* on the control statement. You could also have used the wrong image copy data set, or your image copy data set could have been destroyed.

FABP3576E ERROR IN DETERMINING END OF BLOCK IN BUFFER OR BAD FSE

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3576 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3577E INVALID POINTER TYPE OBTAINED FROM POINTER PROFILE

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3577 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3578E POINTER TO LOGICAL CHILD TABLE = 0

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3578 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3579E ERROR IN CALCULATING NUMBER OF LC POINTERS OR SEGMENT CODES DO NOT MATCH

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3579 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3580E LOGICAL PARENT NOT IN LOGICAL CHILD TABLE

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3580 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3581E PHYSICAL PARENT NOT IN PHYSICAL CHILD TABLE

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3581 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3582E ERROR IN CALCULATING NUMBER OF PC POINTERS, UNABLE TO MATCH PC TO PP IN PC TABLE

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3582 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3583E INVALID POINTER TYPE OBTAINED FROM SEGMENT INFORMATION TABLE

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3583 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3586E OPEN FAILED FOR PRINT FILE

Explanation: After issuing an OPEN macro for a printer file, DCBOFLGS is not zero.

System action: HD Pointer Checker issues a USER 3586 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Check for any run-time errors.

FABP3587E INPUT RECORDS OUT OF SEQUENCE

Explanation: The data set defined by the CHECKREC DD statement is not in the proper sequence.

System action: HD Pointer Checker issues a USER 3587 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Check for JCL errors. Make sure that MERGE process was run properly.

FABP3588E INPUT RECORD TYPE NOT HEADER, 0, 1, 2, 3, 6, 7, 8, OR 9

Explanation: The data set defined by the CHECKREC DD statement contains incorrect records.

System action: HD Pointer Checker issues a USER 3588 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Check for JCL errors. Make sure that MERGE process was run properly.

FAB3589E OPEN FAILED FOR CONTROL FILE

Explanation: After issuing an OPEN macro for a control statement file, DCBOFLGS is not zero.

System action: HD Pointer Checker issues a USER 3589 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Check for any run-time errors.

FABP3590E INVALID POINTER OR SEGMENT CODE IN SEGMENT STATISTICS TABLE

Explanation: An internal error occurred in the HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3590 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3591E DATA SET TYPE MISMATCH FOR DB: *dbdname* DD: *ddname*

Explanation: The data set type of the database data set does not match the DATASET=REAL|IMAGECOPY parameter of the DATABASE statement in the PROCCTL data set.

System action: HD Pointer Checker issues a USER 3591 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Check for any obvious runtime errors. This error can also be caused by running HD Pointer Checker with an empty OSAM data set which has been allocated by specifying DCB parameters. Such data sets have record format 'U'. HD Pointer Checker recognizes the data sets as image copies created with IMS Image Copy 2 Utility.

FABP3600E NO DMB FOUND IN DMB DIRECTORY

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3600 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3601E LOAD FAILED FOR DDNAME: *ddname* **MODULE:** *module-name*

Explanation: After issuing a LOAD macro to load the indicated module *module-name* from the library specified by the indicated *ddname* on the DD statement, register 15 contained a nonzero return code.

System action: HD Pointer Checker issues a USER 3601 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Make sure that the DD statement is specifying the proper data set.

FABP3602E DELETE FAILED FOR MODULE: *module-name*

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3602 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors.

FABP3603E GETMAIN FAILED. SIZE: *nn* **K ITEM-ID:** *xx*

Explanation: After issuing a GETMAIN macro to acquire the indicated size *nn* K bytes of storage, register 15 contained a nonzero return code. The ITEM-ID *xx* represents the internal location of issuing a GETMAIN macro.

System action: HD Pointer Checker issues a USER 3603 abend.

Programmer response: Increase the region size parameter on the JOB statement and rerun the HD Pointer Checker job.

Problem determination: None.

FABP3604E FREEMAIN FAILED. ITEM-ID: *xx*

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3604 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors.

FABP3605E OPEN FAILED FOR DDNAME: *ddname*

Explanation: In issuing an OPEN macro to open the data set associated with the specified ddname, the ABEND exit routine was called.

The following causes are possible for a tape data set:

- Tape BLKSIZE is over 32760. There is a restriction that you have to make the block size of the image copy data set to be equal to or less than 32760.
- DCB is not specified on the DD statement of a non-label (NL) tape. DCB is required for an NL tape.

System action: The HD Pointer Checker issues a USER 3605 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Make sure that the ddname indicated in the DD statement is specifying the proper data set.

FABP3606E CLOSE FAILED FOR DDNAME: *ddname*

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3606 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors.

FABP3607E DEVTYPE FAILED FOR DDNAME: *ddname* **(RC =** *xx***)**

Explanation: After issuing a DEVTYPE macro to get information about the device associated with the ddname, the return code indicated that the attempt to do so was unsuccessful.

System action: The HD Pointer Checker issues a USER 3607 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Make sure that the ddname in the DD statement is specifying the correct data set.

FABP3609E SORT FOR VALIDATION/EVALUATION FAILED

FABP3609E SORT FOR INDEX KEY CHECK FAILED

FABP3609E SORT FOR BLKMAP PROCESS FAILED

FABP3609E SORT FOR EPS HEALING FAILED

FABP3609E SORT FOR SYMBOLIC POINTER CHECK FAILED

Explanation: DFSORT returned non-zero return code. An internal sort error occurred in the process indicated in the message.

System action: HD Pointer Checker issues a USER 3609 abend.

Programmer response: Check the message in the SYSOUT*nn* data set generated by DFSORT, take necessary actions, and rerun HD Pointer Checker.

Problem determination: Check the SYSOUT*nn* for any run-time errors.

FABP3610E DUPLICATE HEADER RECORD FOUND FOR DB: *dbdname* **PID:** *xxxxx* **DSG#:** *xx*

Explanation: Two or more header records were found for the specified data set group *xx* of partition ID *xxxxx* of database *dbdname*.

System action: HD Pointer Checker issues a USER 3610 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Probably a JCL error. Check the data set containing the header records.

FABP3611E SOME HEADER RECORD(S) NOT FOUND FOR DB: *dbdname* **PID:** *xxxxx* **DSG#:** *xx*

Explanation: Some of the five types of header records (H1, H2, h3, H4, and H5) were not found in the CHECKREC data set for the specified data set group *xx* of partition ID *xxxxx* of database *dbdname*.

System action: HD Pointer Checker issues a USER 3611 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Probably a JCL error. Check the data set containing the header records.

FABP3612E INVALID TYPE OF HEADER RECORD FOUND FOR DB: *dbdname* **PID:** *xxxxx* **DSG#:** *xx*

Explanation: A type of the header record read from the CHECKREC data set for the specified data set group *xx* of partition ID *xxxxx* of database *dbdname* is not correct.

System action: HD Pointer Checker issues a USER 3612 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the error persists, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors.

FABP3613E DATA OF HEADER RECORD FOR DB#: *nnn* **PID:** *xxxxx* **DSG#:** *xx* **IS DIFFERENT FROM DMB**

Explanation: Data of the header records read by the CHECKREC data set is different from that of the DMB read by the IMS for the data set group *xx* of partition ID *xxxxx* (database number: *nnn*) when checking one of the following:

- DB name
- DD name
- DB logical group number (DBLG#)
- block size (BLKSIZE)
- logical record length (LRECL)

System action: HD Pointer Checker issues a USER 3613 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: You may have used a wrong PSB name on the EXEC statement PARM. It is also possible that you specified a wrong PSB/DBD library.

FABP3614E HEADER RECORD FOR DB#: ***nn*** **PID:** ***xxxxx*** **DSG#:** ***xx*** **WAS READ BUT CORRESPONDING DMB NOT FOUND**

Explanation: A header record for the specified database data set group *xx* of partition ID *xxxxx* of database (database number: *nn*) was read from the CHECKREC data set, but a DMB for the data set group was not found.

System action: HD Pointer Checker issues a USER 3614 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: You may have used a wrong PSB name on the EXEC statement PARM. It is also possible that you specified a wrong PSB/DBD library.

FABP3615E MEMBER: ***member-name*** **NOT FOUND IN** ***ddname*** **DATA SET**

Explanation: A FIND macro was issued for the indicated *member-name*, and no module with that name was in the specified *ddname* library.

System action: HD Pointer Checker issues a USER 3615 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Make sure that the indicated *member-name* is in the correct library.

FABP3617E DIFFERENT HASH OPTIONS WERE SPECIFIED TO DATA SET GROUPS FOR DB: ***dbdname*** **DB#:** ***nnn***

Explanation: For database *dbdname* (database number: *nnn*), some data set groups were scanned by the HASH Check function and some by the regular check function.

System action: HD Pointer Checker issues a USER 3617 abend.

Programmer response: Rerun the HD Pointer Checker run with the same HASH option (YES or NO) for all data set groups of the database.

Problem determination: You probably ran HD Pointer Checker by different job steps with the TYPE=SCAN option specified on the PROC statement, with HASH=YES option for some data set groups, and with HASH=NO option for other data set groups.

FABP3618E HD POINTER CHECKER FAILED IN A SCAN PROCESS (SCANGROUP: ***xx*****)**

Explanation: The scan task ended abnormally.

System action: HD Pointer Checker issues a USER 3618 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: None.

FABP3630E HASH RECORD NOT FOUND IN HASH CHECK PROCESS.

Explanation: No HASH record was passed to the evaluation process, while some data sets were scanned by the HASH Check function.

System action: HD Pointer Checker issues a USER 3630 abend.

Programmer response: Save the entire run listing (including the dump, JCL, and reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: None.

FABP3631E TOTAL NUMBER OF HASH RECORDS NOT EQUAL TO HEADER INFORMATION

Explanation: Number of HASH records is different from SCAN process and CHECK process.

System action: HD Pointer Checker issues a USER 3631 abend.

Programmer response: Correct the error, and rerun the HD Pointer Checker job with "PROC TYPE= ALL."

Problem determination: It is possible that you have a JCL error. It could be either an old CHECKREC data set or a corrupted CHECKREC data set.

FABP3633E DBD: ***dbdname*** **DB#:** ***nnn*** **CANNOT BE CHECKED WITH HASH OPTION**

Explanation: Database *dbdname* (database number: *nnn*), which is a secondary index database, cannot be checked with the HASH option.

System action: HD Pointer Checker issues a USER 3633 abend.

Programmer response: Correct the error, and rerun the HD Pointer Checker job with "PROC TYPE=ALL."

Problem determination: DBD can be changed between the SCAN process and the CHECK process.

FABP3640E HISTORY DATA SET CONTROL RECORD NOT FOUND

Explanation: There is no HISTORY data set control record in the HISTORY data set.

System action: HD Pointer Checker issues a USER 3640 abend.

Programmer response: Correct the error, and rerun the HD Pointer Checker job. If you used newly allocated HISTORY data set and got the error, run the DB

Historical Data Analyzer and initialize the HISTORY data set first; then rerun the HD Pointer Checker job.

Problem determination: You may have specified the incorrect data set on your HISTORY DD statement. It is also possible that you specified a data set on your HISTORY DD statement that was not initialized by the DB Historical Data Analyzer yet.

FABP3641E HISTORY DATA SET NO MORE SPACE FOR RECORD: *record type* **KEY:** *key-value*

Explanation: The HISTORY data set record specified by *record type* and *key-value* has no more space to add information gathered by HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3641 abend.

Programmer response: Delete the old and unnecessary database data set records on the HISTORY data set (as many as possible) and reorganize the data set by using the DB Historical Data Analyzer; then rerun the HD Pointer Checker job.

Problem determination: None.

FABP3642E DATA OF DBDS TABLE FOR DB#: *nnn* **PART: NNNNN DSG#:** *xx* **IS DIFFERENT FROM DMB**

Explanation: Data of the DBDS table record read by the HISTORY data set is different from that of the DMB read by the IMS data set for the specified data set group *xx* of database (database number: *nnn*) when checking one of the following:

- Primary DD name
- Overflow DD name
- Database organization
- Database access method

System action: HD Pointer Checker issues a USER 3642 abend.

Programmer response: Correct the error, and rerun the HD Pointer Checker job. If you regenerated the DBD for the database data set, first delete all DBCS records for the database data set from the HISTORY data set by using DB Historical Data Analyzer; then rerun the HD Pointer Checker job.

Problem determination: You may have specified the incorrect data set on your HISTORY DD statement. It is also possible that you regenerated the DBD for the database data set with the attribute that is different from the previous DBD.

| **FABP3643E MORE THAN 15 IMSIDS CANNOT BE CREATED TO HISTORY DATA SET**

| **Explanation:** More than 15 IMSIDs cannot be created in the HISTORY data set that is Multiple-IMSID option enabled.

| **System action:** HD Pointer Checker issues a USER 3643 abend.

| **Programmer response:** Do not create more than 15 IMSIDs.

| **Problem determination:** None.

FABP3650E INVALID CALL TO U1212AV

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3650 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3651E INVALID CALL TO U1214AV

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3651 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3660E INVALID RECORD TYPE DETECTED IN IXKEY FILE

Explanation: Other than T1, T6, T7 or TA type of record was detected in the IXKEY data set.

System action: HD Pointer Checker issues a USER 3660 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors.

FABP3661E DB#: *nnn* PID: *xxxxx* DSG#: *xx* OF *rec-type* RECORD IS DIFFERENT FROM DBD

Explanation: The information on the indicated database (database number: *nnn* or partition ID: *xxxxx* or data set group number: *xx*) which is contained in the indicated *rec-type* (such as T1, T6, T7, and TA) record of the IXKEY file and the information in DBD is different.

System action: HD Pointer Checker issues a USER 3661 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: It is also possible that you regenerated the DBD for the database data set with an attribute different from the previous DBD.

FABP3663E ERROR IN CALCULATING ADDRESS OF POINTER

Explanation: An internal error occurred in calculating the actual direct access address of the record that contains the pointer.

System action: HD Pointer Checker issues a USER 3663 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Make sure that the input data set specified in DD statement is a real database data set. Do not provide a DD statement for any image copy data sets.

FABP3664E INCOMPATIBLE BLOCK NUMBER DB: *dbname* PID: *xxxxx* DSG: *xx* SCANNED: *mmmmmmmm* BITMAP: *nnnnnnn*

Explanation: The database or the image copy data set may be damaged. The number of physical blocks in the database data set conflicts with the information obtained from the bit map block. The indicated number of physical blocks scanned (*mmmmmmmm*) is smaller than the indicated number of insufficient free space blocks (*nnnnnnnn*) obtained from the bit map block of the indicated database data set. The number of physical blocks scanned must be equal to the sum of insufficient free space blocks and sufficient free space blocks obtained from the bit map blocks.

The following reasons can be considered as the cause of the bit map error:

- The bit map block is damaged.
- Some physical blocks are lost from the database data set or the image copy data set.
- Incorrect EOF is found in the database data set or the image copy data set.

System action: Processing continues.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP3665E DATABASE CREATION DATE '*xxxxxx*' IS INVALID FOR DB: *dbdname* PID: *xxxxx* DSG#: *xx*

Explanation: The database is damaged. HD Pointer Checker cannot accept the database creation date obtained from the database (DB: *dbdname*, PID: *xxxxx*, DSG#: *xx*) because of the incorrect data format.

System action: HD Pointer Checker issues a USER 3665 abend.

Programmer response: Repair the database, and rerun the HD Pointer Checker job.

Problem determination: See Chapter 10, “Database repair guidelines,” on page 271.

FABP3666E FAILED IN PARTITION SELECTION; REQUEST=*xxxxxx*, RC=*yy*, RSN=*zzzz*

Explanation: During the Partition Selection Process by the IMS DFSPSEL macro, HD Pointer Checker received the error return code. *xxxxxx* is the request function for DFSPSEL macro, *yy* is RC from DFSPSEL, and *zzzz* is RSN from DFSPSEL (If RC=16, *zzzz* is ABEND code from DFSPSEL).

System action: HD Pointer Checker issues a USER 3666 abend.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: You may have specified an incorrect partition selection exit routine or a broken database. Make sure that the STEPLIB, LOADLIB, or LINKLIST library contains the correct HALDB Partition Selection exit routine.

FABP3671E OFFSET OF KEY IN VLS ROOT SEGMENT CANNOT BE ZERO

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3671 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3672E LOGICAL CHILD NOT FOUND IN SEGMENT INFORMATION TABLE

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3672 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3673E LOGICAL PARENT NOT FOUND IN SEGMENT INFORMATION TABLE

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3673 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3674E SEGMENT CODE OF INDEX TARGET NOT FOUND IN DMB DIRECTORY

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3674 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3676E ERROR LOCATING PHYSICAL PARENT OF SOURCE SEGMENT

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3676 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3677E PHYSICAL PARENT OF THE SOURCE SEGMENT CANNOT BE LOCATED

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3677 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3678E INVALID POINTER TYPE WAS DETECTED

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3678 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Check for any run-time errors. Make sure the DBD and PSB are valid.

FABP3679E INVALID POINTER TYPE PASSED TO CHECK PROCESS

Explanation: The work file passed to CHECK processor contains a record with an incorrect pointer type.

System action: HD Pointer Checker issues a USER 3679 abend.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: It is possible that you have a JCL error. You could be using either an old CHECKREC data set or a corrupted CHECKREC data set.

FABP3680E NO SEQUENCE FIELD IS SPECIFIED IN *dbdname* WITH KEYSIN=YES OPTION

Explanation: The KEYSIN=YES option is not effective for the indicated database “dbdname” because no sequence field is defined in the DBD.

System action: The HD Pointer Checker issues a USER 3680 abend.

Programmer response: Check the error, and rerun the HD Pointer Checker job with “KEYSIN=NO” option.

Problem determination: None.

FABP3690E ATTACH FAILED FOR MODULE: *module-name* (RC=*xx*)

Explanation: A nonzero return code was returned from an ATTACH macro for the module *module-name*. Here *xx* is the return code returned from the ATTACH macro in a decimal format.

System action: The HD Pointer Checker issues a USER 3690 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: For each return code, see *Assembler Service Reference* for the operating system.

FABP3751E UNSUCCESSFUL VSAM MODCB (FOR ACB)

Explanation: After issuing a MODCB macro to modify an ACB control block, register 15 contained a nonzero return code.

System action: HD Pointer Checker issues a USER 3751 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Use standard VSAM debugging techniques. The code returned by VSAM is at label ERROR in the HD Pointer Checker module that issued the MODCB macro.

FABP3752E UNSUCCESSFUL VSAM OPEN

Explanation: After issuing an OPEN macro for an ACB control block, register 15 contained a nonzero return code.

System action: HD Pointer Checker issues a USER 3752 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Use standard VSAM debugging techniques. The code returned by VSAM is at label ERROR in the HD Pointer Checker module that issued the OPEN macro.

FABP3753E UNSUCCESSFUL VSAM SHOWCB (FOR ACB)

Explanation: After issuing a SHOWCB macro for an ACB control block, register 15 contained a nonzero return code.

System action: HD Pointer Checker issues a USER 3753 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Use standard VSAM debugging techniques.

FABP3754E UNSUCCESSFUL VSAM GET

Explanation: After issuing a GET macro for an ACB control block, register 15 contained a nonzero return code.

System action: HD Pointer Checker issues a USER 3754 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Use standard VSAM debugging techniques. The feedback field into which VSAM puts a return code is at label FDBK in the HD Pointer Checker module that issued the GET macro.

FABP3755E UNSUCCESSFUL VSAM SHOWCB (FOR RPL)

Explanation: After issuing a SHOWCB macro for an RPL control block, register 15 contained a nonzero return code.

System action: HD Pointer Checker issues a USER 3755 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Use standard VSAM debugging techniques. The feedback field into which VSAM puts a return code is at label FDBK in the HD Pointer Checker module that issued the GET macro.

FABP3756E UNSUCCESSFUL VSAM MODCB (FOR EXLST)

Explanation: After issuing a MODCB for an EXLST control block, register 15 contains a nonzero return code.

System action: The HD Pointer Checker issues a USER 3756 abend.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Use standard VSAM debugging techniques.

FABP3757E UNSUCCESSFUL VSAM MODCB (FOR RPL)

Explanation: After issuing a MODCB for an RPL control block, register 15 contains a nonzero return code.

System action: The HD Pointer Checker issues a USER 3757 abend.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Use standard VSAM debugging techniques.

FABP3758E UNSUCCESSFUL VSAM CLOSE DDNAME: *ddname*

Explanation: After issuing an CLOSE macro for ACB control block, register 15 contained a nonzero return code.

System action: HD Pointer Checker issues a USER 3758 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Use standard VSAM debugging techniques.

FABP3759E UNSUCCESSFUL VSAM *macro-name***. VSAM ERROR DATA: RETURN CODE:** *xx* **RPL "FDBK":** *aaa (bb)*

Explanation: After issuing the indicated *macro-name* (GET, PUT, MODCB or DELETE macro) for ACB control block, register 15 contained an *xx* (nonzero) return code.

System action: HD Pointer Checker issues a USER 3759 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Use standard VSAM debugging techniques. VSAM return code and FDBK code are shown in a decimal (*xx, aaa*) and hexadecimal (*bb*) format.

FABP3760E VSAM KSDS WITH EXTENDED ADDRESSABILITY ATTRIBUTE IS NOT SUPPORTED

Explanation: It was determined that the KSDS is a SMS data set with the extended addressability attribute, which IMS does not support.

System action: HD Pointer Checker issues a USER 3760 abend.

Programmer response: Allocate the data set with a data class that does not specify extended addressability, and rerun the job.

Problem determination: None.

FABP3770E BLANK DDNAME ON CONTROL CARD

Explanation: Your FABPCHR0 control statement contains blanks in the ddname field (column 1).

System action: Processing continues.

Programmer response: Correct or remove the bad control statement. Rerun the HD Pointer Checker job, if necessary.

Problem determination: Control statement error.

FABP3775E NO DD STATEMENT = *ddname*

Explanation: There is no DD statement in your JCL that corresponds to the *ddname*.

System action: Processing continues.

Programmer response: Correct your JCL by adding the missing DD statement, and rerun the HD Pointer Checker Checker job.

Problem determination: Probable JCL error. Check for mistakes in spelling.

FABP3780E FABPDADR RDJFCB RC = *xxx*

Explanation: A return code of *xxx* resulted from a RDJFCB macro issued by module FABPDADR.

System action: Processing continues.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Use standard debugging techniques.

FABP3785E FABPDADR LOCATE RC = *xxx*

Explanation: A return code of *xxx* resulted from a LOCATE macro issued by module FABPDADR.

System action: Processing continues.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Use standard debugging techniques.

FABP3790E NOT DIRECT ACCESS DEVICE

Explanation: Module FABPDADR attempted to process a data set that was not on a direct access device.

System action: Processing continues.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Use standard debugging techniques.

FABP3795E FABPDADR OBTAIN RC = *xxx*

Explanation: A return code of *xxx* resulted from an OBTAIN macro issued by FABPDADR.

System action: Processing continues.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Use standard debugging techniques.

FABP3800E NUMBER OF EXTENTS = 0

Explanation: Module FABPDADR attempted to process a data set that has no extents.

System action: Processing continues.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Use standard debugging techniques.

FABP3805E BLKSIZE = 0

Explanation: Module FABPDADR attempted to process a data set for which the block size (DS1BLKL) in the DSCB is zero.

System action: Processing continues.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Use standard debugging techniques.

FABP3810E RBA LESS THAN BLKSIZE

Explanation: The deleted RBA is less than BLKSIZE. If you are running FABPCHRO, the control statement contains an RBA (in column 11) that is smaller than your data set block size.

System action: Processing continues.

Programmer response: Correct or remove the bad control statement. Rerun the HD Pointer Checker job, if necessary.

Problem determination: Control statement error. This message can also occur if you code blanks in column 31 for a VSAM data set.

FABP3815E RBA BEYOND EXTENTS

Explanation: Your FABPCHRO control statement contains an RBA that points beyond the extents of your data set.

System action: Processing continues.

Programmer response: Correct or remove the bad control statement. Rerun the HD Pointer Checker job, if necessary.

Problem determination: Control statement error.

FABP3820E VSAM CATALOG I/O ERROR

Explanation: Module FABPDADR attempted to LINK to IDCAMS, and the resulting return code was greater than 4.

System action: Processing continues.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Use standard debugging techniques.

FABP3840E UNSUPPORTED LEVEL OF IMS IS BEING USED: *mmm* **IMS LEVEL OF THIS RUN**

Explanation: You are running an IMS batch region controller (DFSRRC00) that is not supported by HD Pointer Checker (*mmm* represents the IMS level).

System action: HD Pointer Checker issues a USER 3840 abend.

Programmer response: You must run HD Pointer Checker with the correct version of IMS.

Problem determination: None.

FABP3850W TIMESTAMP MISMATCH. FROM exitname (*mm/dd/yy hh:mm***) HDR TIME(***mm/dd/yy hh:mm***)**

Explanation: Assemble timestamp in Image Copy compression exit is different from timestamp in image copy header.

System action: Processing continues.

Problem determination: None.

Programmer response: None.

FABP3851E LENGTH FROM EXIT RTN DOES NOT MATCH ORIGINAL. AT DB <RBAIRBN>=X'*xxxxxxxx***'**

Explanation: Record length returned by Image Copy compression exit in decompress process is different from original record length.

System action: The HD Pointer Checker issues a USER 3851 abend.

Problem determination: None.

Programmer response: Correct the user exit routine and rerun the job.

FABP3852E TERMINATION REQUESTED BY *exitname* IN INITIALIZATION CALL RC=X'*xx*'

FABP3852E TERMINATION REQUESTED BY *exitname* IN BLOCK READ CALL RC=X'*xx*'

FABP3852E TERMINATION REQUESTED BY *exitname* IN TERMINATION CALL RC=X'*xx*'

Explanation: SCAN process termination is requested by Image Copy compression exit.

System action: The HD Pointer Checker issues a USER 3852 abend.

Problem determination: None.

Programmer response: If user exit routine has an error, correct the error and rerun the job.

FABP3860E RDJFCB MACRO FAILED FOR DDNAME: *ddname*

Explanation: Register 15 contained a non-zero return code, after issuing a RDJFCB macro for *ddname*.

System action: HD Pointer Checker issues a USER 3860 abend.

Programmer response: Correct the error and rerun the HD Pointer job.

Problem determination: Use standard debugging methods.

FABP3861E OBTAIN MACRO FAILED FOR DSNAME: *dsname*

Explanation: Register 15 contained a non-zero return code, after issuing a OBTAIN macro for *dsname*.

System action: HD Pointer Checker issues a USER 3861 abend.

Programmer response: Correct the error and rerun the HD Pointer job.

Problem determination: Use standard debugging methods.

FABP3862E IDCAMS LISTCAT FAILED FOR DSNAME: *dsname*

Explanation: After internally linking IDCAMS LISTCAT routine for *dsname*, register 15 contained a non-zero return code.

System action: HD Pointer Checker issues a USER 3862 abend.

Programmer response: Correct the error and rerun the HD Pointer job.

Problem determination: Use standard debugging methods.

FABP3863E OBTAIN MACRO FAILED FOR VOLUME: *volume*

Explanation: Register 15 contained a nonzero return code, after an OBTAIN macro for the volume was issued.

System action: HD Pointer Checker issues a USER 3863 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Use standard debugging methods.

FABP3864E IDENTIFY MACRO FAILED FOR ENTRY NAME: *module-name* RC=X'*xx*'

Explanation: An internal error occurred in HD Pointer Checker.

System action: HD Pointer Checker issues a USER 3864 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: The HD Pointer Checker product library might be broken or specified incorrectly to STEPLIB, JOBLIB or LINKLIST. Make sure the module name is in the library.

FABP3901E UNKNOWN FUNCTION REQUESTED

Explanation: The module FABPIC20 was called with an unknown function code.

System action: The HD Pointer Checker issues a USER 3901 abend.

Programmer response: Correct all errors and rerun the job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

FABP3902E "GET" FAILED FOR DDNAME *ddname*

Explanation: The HD Pointer Checker tried to issue a GET macro for the file associated with the DD statement. The SYNAD exit routine was called during the GET processing.

System action: HD Pointer Checker ends with the abend code 3902.

Programmer response: Make sure that a DD statement is specifying the correct data set. Correct the error and rerun the job.

FABP3903E "GETMAIN" FAILURE OCCURRED (RC = *xx*) FOR IMAGE COPY 2 BUFFER

Explanation: The HD Pointer Checker issued a GETMAIN macro to get storage for the buffer. The return code indicates that the attempt to do so was unsuccessful.

System action: The HD Pointer Checker issues a USER 3903 abend.

Programmer response: Increase the region size parameter value in the EXEC statement and rerun the job.

FABP3904E "FREEMAIN" FAILURE OCCURRED (RC = *xx*) FOR IMAGE COPY 2 BUFFER

Explanation: The HD Pointer Checker issued a FREEMAIN macro. The return code indicates that the attempt to do so was unsuccessful.

System action: HD Pointer Checker ends with the abend code 3904.

Programmer response: Correct all errors and rerun the job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

FABP3911E FIRST RECORD IS NOT A TAPE HEADER RECORD

Explanation: The first record of the image copy taken with the Database Image Copy 2 utility is not a tape header record. The HD Pointer Checker expected the first record of the image copy taken with the Database Image Copy 2 utility to be a tape header record, but it was not.

System action: The HD Pointer Checker issues a USER 3911 abend.

Programmer response: You may have specified an incorrect or a broken data set. Make sure that the DD statement is identifying the correct data set. Correct the errors and rerun the job.

FABP3912E INITIALIZATION FOR IMAGE COPY 2 READING INCOMPLETE - DATA SET HEADER RECORD NOT FOUND

FABP3912E INITIALIZATION FOR IMAGE COPY 2 READING INCOMPLETE - VOLUME DEFINITION RECORD NOT FOUND

Explanation: The HD Pointer Checker could not find the record indicated. The HD Pointer Checker collects the information which is necessary to analyze the data set while initialization processing, but the processing was incomplete.

System action: The HD Pointer Checker issues a USER 3912 abend.

Programmer response: You may have specified an incorrect or a broken data set. Make sure that the DD statement identifies the correct data set. Correct the errors and rerun the job.

FABP3913E UNEXPECTED LOGICAL END OF FILE ON IMAGE COPY 2 DATA SET

Explanation: An unexpected trailer record was encountered while the HD Pointer Checker was reconstructing the logical record image from the image copy taken with the Database Image Copy 2 utility. The image copy taken with the Database Image Copy 2 utility has two trailer records and the HD Pointer Checker regards them as the end-of-data.

System action: The HD Pointer Checker issues a USER 3913 abend.

Programmer response: You may have specified an incorrect or a broken data set. Make sure that the DD statement is specifying the correct data set. Correct the error and rerun the job.

FABP3914E INTERNAL RECORD LENGTH ERROR ON IMAGE COPY 2 DATA SET

Explanation: The HD Pointer Checker checked the logical record length of the dumped data set in the image copy and found an incorrect length.

System action: The HD Pointer Checker issues a USER 3914 abend.

Programmer response: You may have specified an incorrect or a broken data set. Make sure that the DD statement is specifying the correct data set. Correct the error and rerun the job.

FABP3915E SPECIFIED IMAGE COPY 2 DATA SET IS EMPTY

Explanation: The HD Pointer Checker attempted to get a record from the specified image copy taken with the Database Image Copy 2 utility, but the data set was empty.

System action: The HD Pointer Checker issues a USER 3915 abend.

Programmer response: Make sure that the DD statement is specifying the correct data set. Correct the error and rerun the job.

FABP3916E UNEXPECTED PHYSICAL END OF FILE ON IMAGE COPY 2 DATA SET

Explanation: The HD Pointer Checker attempted to get a record from the specified image copy taken with the Database Image Copy 2 utility, but an unexpected physical end-of-file was encountered.

Note: The image copy taken with the Database Image Copy 2 utility has two trailer records. Usually HD Pointer Checker ends image copy reading on the first trailer record instead of the physical end-of-file.

System action: The HD Pointer Checker issues a USER 3916 abend.

Programmer response: You may have specified an incorrect or a broken data set. Make sure that the DD statement is specifying the correct data set. Correct the error and rerun the job.

FABP3921E UNSUPPORTED DUMP FORMAT. LOGICAL DUMP REQUIRED

Explanation: The image copy taken with the Database Image Copy 2 utility must be a DFSMSdss™ logical dump format, but it was not.

System action: The HD Pointer Checker issues a USER 3921 abend.

Programmer response: You may have specified an incorrect or a broken data set. Make sure that the DD statement is specifying the correct data set. Correct the error and rerun the job.

FABP3922E MORE THAN ONE DATA SET IS DUMPED IN THE IMAGE COPY

Explanation: Input data set contained more than one database data set. It will be an error because the image copy taken with the Database Image Copy 2 utility contains only one database data set.

System action: The HD Pointer Checker issues a USER 3922 abend.

Programmer response: You may have specified an incorrect or a broken data set. Make sure that the DD statement is specifying the correct data set. Correct the error and rerun the job.

FABP3923E DATA SET ORGANIZATION CONFLICT - EXPECTED *: xxxx*

FABP3923E DATA SET ORGANIZATION CONFLICT - DUMPED DATA SET *: yyyy*

Explanation: The dumped data set organization conflicted with the parameter which was passed by the caller. (*xxxx*/*yyyy* : VSAM/OSAM/ESDS/KSDS)

System action: The HD Pointer Checker issues a USER 3923 abend.

Programmer response: You may have specified an incorrect or a broken data set. Make sure that the DD statement is specifying the correct data set. Correct the error and rerun the job.

FABP3924E VSAM KSDS DATA SET WAS DUMPED WITHOUT VALIDATION

Explanation: VSAM KSDS must be dumped with validation by IMS Database Image Copy 2 utility, but the input data set was not.

System action: The HD Pointer Checker issues a USER 3924 abend.

Programmer response: You may have specified an incorrect or a broken data set. Make sure that the DD statement is specifying the correct data set. Correct the error and rerun the job.

FABP3925E DUMPED DATA SET IS UNSUPPORTED FORMAT
- FOR OSAM DATA SET, BLOCKED RECORD

FABP3925E DUMPED DATA SET IS UNSUPPORTED FORMAT
- FOR VSAM DATA SET, NOT ESDS/KSDS

FABP3925E DUMPED DATA SET IS UNSUPPORTED FORMAT
- FOR VSAM DATA SET, SPANNED RECORD

FABP3925E DUMPED DATA SET IS UNSUPPORTED FORMAT
- FOR DEDB/HDAM/HIDAM DATA, BLOCKED RECORD

Explanation: The dumped database data set organization is not supported. Details are shown by sub-messages.

System action: The HD Pointer Checker issues a USER 3925 abend.

Programmer response: You may have specified an incorrect or a broken data set. Make sure that the DD statement is specifying the correct data set. Correct the error and rerun the job.

FABP3988E UNIT: *unitname* **VOL-COUNT:** *volcount* **DISP: (***disp1***,***disp2***,***disp3***) DSN:** *dsname*

Explanation: When the dynamic allocations of the work data set fails, this message follows message FABP3998E. This message shows the parameters to the dynamic allocation macro. Because some of the resources shown in the parameters are insufficient, the dynamic allocation might be failed.

System action: HD Pointer Checker issues a USER 3998 abend.

Programmer response: Specify the DD statement explicitly in the HD Pointer Checker JCL, with the appropriate unit name and volume count of your system.

Problem determination: None.

FABP3989E SYSIN, SYSOUT, OR SUBSYSTEM (SUBSYS=) IS UNSUPPORTED DATA SET TYPE. DD: *ddname*

Explanation: Data sets with the specified SYSIN, SYSOUT, or SUBSYS DD are not supported.

System action: The HD Pointer Checker issues a USER 3989 abend.

Programmer response: Correct the error and rerun the HD Pointer Checker job.

Problem determination: Make sure that the DD statement of ddname shown in the message is specified correctly.

FABP3990E INVALID IMS RELEASE LEVEL IN RECON DATA SET: *ddname*

Explanation: The data set used for the DD name *dddddddd* was not the correct IMS release level of the RECON data set.

System action: HD Pointer Checker issues a USER 3990 abend.

Programmer response: Specify the correct IMS release level of the RECON data set, and rerun the job.

Problem determination: None.

FABP3991E HEADER RECORD NOT FOUND IN RECON DATA SET: *ddname*

Explanation: The data set used for the DD name was not the RECON data set.

System action: HD Pointer Checker issues a USER 3991 abend.

Programmer response: Specify the correct RECON data set, and rerun the job.

Problem determination: None.

FABP3992E RECON HEADER EXTENSION RECORD NOT FOUND IN RECON DATA SET: *ddname*

Explanation: The data set used for the DD name was not the RECON data set or the correct IMS release level of the RECON data set.

System action: HD Pointer Checker issues a USER 3992 abend.

Programmer response: Specify the correct IMS release level of the RECON data set, and rerun the job.

Problem determination: None.

FABP3993E TWO VALID RECON DATA SETS NOT PROVIDED

Explanation: The two IMS release levels of RECON data sets are not correct.

System action: HD Pointer Checker issues a USER 3993 abend.

Programmer response: Specify two correct IMS release levels of the RECON data sets, and rerun the job.

Problem determination: None.

FABP3994E *xxxxxxxx* **DEFINED INCORRECTLY IN RECON DATA SET:** *ddname*

Explanation: The RECON data set had incorrect information. *xxxxxxxx* is "COEX" or "MINVERS".

System action: HD Pointer Checker issues a USER 3993 abend.

Programmer response: Specify the correct COEX/MINVERS for the RECON data set when initializing RECON, and rerun the job.

Problem determination: None.

FABP3995E DSN: *dsname* **VOL:** *vol-ser*

Explanation: If the dynamic allocation of the input database data set or the image copy data set fails, this message is issued along with message FABP3998E. Here, *dsname* and *vol-ser* are the name and volume serial of the data set that caused the error. If the data set is a cataloged data set, `'******'` is shown for *vol-ser*.

System action: HD Pointer Checker issues a USER 3998 abend.

Programmer response: Check message FABP3998E.

Problem determination: None.

FABP3996E UNABLE TO BUILD CONTROL BLOCKS FOR DBD: *dbdname*

Explanation: HD Pointer Checker requested construction of control blocks for Full Function Database. The request was not successfully completed.

System action: HD Pointer Checker issues a USER 3996 abend.

Programmer response: Make sure that a valid DBD or RECON data set exists for the named database and rerun the job.

Problem determination: None.

FABP3997E DATA SET FOR DB: *dbdname* **DD:** *ddname* **NOT ALLOCATED**

Explanation: Either a DD statement of the required data set is not specified, or the data set cannot be allocated dynamically.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Make sure that a valid DBD or data set name is defined in DFSMDA or RECON.

FABP3998E UNSUCCESSFUL DYNALLOC REQUEST (SVC 99: RC= *nn***, CC =** *xxxx***)**

Explanation: After issuing the DYNALLOC request for database data set, image copy data set, or work data sets, SVC 99 routines return a nonzero (*nn*) return code in register 15 and hexadecimal (*xxxx*) reason code.

System action: The HD Pointer Checker issues a USER 3998 abend.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: For the return code and reason code of SVC 99, refer to *z/OS MVS Programming: Authorized Assembler Services Guide*. If you allocate an uncataloged image copy data set on tapes dynamically, make sure you specify the correct unit name in the ICUNIT parameter in the PROCCTL statement. If you use image copy data sets that are stacked in a tape, you cannot specify both dynamic and explicit allocation.

FABP3999E UNKNOWN ERROR OCCURRED IN *module-name* **MODULE (CC =** *nnn***)**

Explanation: An internal errors occurred in the indicated module HD Pointer Checker *module-name*.

System action: HD Pointer Checker issues a USER 3999 abend.

Programmer response: Save the entire run listing (including the dump, JCL, and reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: None.

FABP4001E DEVTYPE MACRO FAILURE OCCURRED FOR *ddname* **DATA SET**

Explanation: No data set is specified on the DD statement of ddname. This data set is required as output.

System action: HD Pointer Checker issues a USER 4001 abend.

Programmer response: Specify the correct ddname data set. Rerun the job.

Problem determination: None.

FABP4002E I/O FAILURE OCCURRED DURING *xxxxx* **FOR DBDEFCTL DATA SET** *(member)*

Explanation: An attempt to receive input from or send output to the DBDEFCTL data set failed. *xxxxx* is one of I/O, BLDL, WRITE, or STOW. R15 shows a return code from a macro.

System action: A user 4002 abend is issued.

Programmer response: Correct the error and rerun the job.

Problem determination: Check system messages in the JOBLOG.

FABP4003W DATABASE STRUCTURE MAY HAVE BEEN CHANGED: *dbdname*

Explanation: The creation date of the DBDEFCTL data set is older than the assemble date of the specified DBD. If the database structure has been changed, the result of the HASH check is unpredictable.

System action: Processing continues.

Programmer response: Make sure that the database structure has not been changed. If it has been changed, rerun the DBD Analysis Program and then rerun the job.

Problem determination: None.

FABP4004W HISAM DB: *dbdname* **DB#:** *nnn* **DD:** *ddname* **WAS NOT SCANNED**

Explanation: The HISAM DB that has overflow DD was not scanned in the same step as the primary and overflow data sets.

System action: Processing continues.

Programmer response: If the indicated DD was already scanned in another step, no action is required. If it has not yet been scanned, run the IMS Image Copy Extensions Image Copy with HASH Check function job for the HISAM database data set.

Problem determination: None.

FABP4005W DB: ***dbdname*** **DB#:** ***nnn*** **DD:** ***ddname*** **WAS NOT SCANNED FOR MULTIPLE DATA SET GROUP**

Explanation: Multiple data set groups were scanned in the IMS Image Copy Extensions Image Copy utility job, but the indicated data set group was not scanned.

System action: Processing continues.

Programmer response: Make sure that all database data sets in the multiple data set group are processed in your HASH evaluation process. If you omit any of database data sets, errors might not be detected.

Problem determination: None.

FABP4006W DB: ***dbdname*** **DB#:** ***nnn*** **WAS NOT SCANNED FOR LOGICALLY RELATED DB DBLG#:** ***mmm***

Explanation: Multiple databases have logical relationships with the database, but the indicated database was not scanned.

System action: Processing continues.

Programmer response: Make sure that all databases in the logical relationship are processed in your HASH evaluation process. If you omit any of databases, errors might not be detected.

Problem determination: None.

FABP4007E INVALID PROCESS OPTION SPECIFIED FOR "TYPE=" PARAMETER

Explanation: An incorrect TYPE= operand for HASH Check function was specified. The operand that can be used for the TYPE= parameter for the HASH Check function is SCAN or ALL.

System action: Processing stops.

Programmer response: Specify TYPE=SCAN or TYPE=ALL. Rerun the job.

Problem determination: None.

FABP4008E DB: ***XXXXXXXX*** **CANNOT BE SCANNED WHEN HD POINTER CHECKER IS INVOKED FROM OTHER PRODUCTS**

Explanation: DB *XXXXXXXX* is not supported when HD Pointer Checker is called from IMS Database Recovery Facility.

System action: Processing continues.

Programmer response: Remove the database data set from the control statement of the IMS Database Recovery Facility, and rerun the job. If the problem remains, save the entire run listing (including the dump, JCL, and all reports from IMS Database Recovery Facility and HD Pointer Checker), and contact IBM Software Support.

Problem determination: Check the control statement of IMS Database Recovery Facility.

FABP4009E "HASH=YES/FORCE" WAS NOT SPECIFIED IN PROC STATEMENT

Explanation: The parameter HASH=YES or HASH=FORCE is required for the DBD Analysis Program. The DBDEFCTL members must not be created.

System action: Processing stops.

Programmer response: Specify HASH=YES or HASH=FORCE. Rerun the job.

Problem determination: None.

FABP4010E "HASH=FORCE" WAS SPECIFIED IN PROC STATEMENT, BUT THERE IS NO DATABASE DATA SET FOR APPLYING HASH CHECK OPTION

Explanation: Self-explanatory. The DBDEFCTL members are not be created.

System action: Processing stops.

Programmer response: Make sure that you specified the correct PSB name and DATABASE statements.

Problem determination: None.

FABP4011I HASH RECORDS WRITTEN TO ***dsname***

Explanation: This is just an information message. The HASH total records were written into the indicated data set.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP4012W NO DATABASE DATA SET PROCESSED BY HASH CHECKING

Explanation: The DBDEFCTL DD statement specified on JCL for the IMS Image Copy Extensions Image Copy with HASH Check function job, but no database data set was processed by the HASH Check function

System action: Processing continues.

Programmer response: Check if the DBDEFCTL members you want to perform hash checking on exist. If not, run DBD Analysis Program with the database specified.

Problem determination: None.

FABP4013W DATA SET INFORMATION NOT FOUND IN DBDEFCTL

Explanation: The DB information was found in the DBDEFCTL data set, but the data set information belonging to the DB was not found in the DBDEFCTL data set.

System action: Processing continues.

Programmer response: Check the DATABASE statement in the preceding DBD Analysis Program job. If it is incorrect, rerun the job with the correct database specification.

Problem determination: None.

FABP4014I THE ABOVE DATABASE DATA SET IS PROCESSED WITH HASH CHECK FUNCTION

Explanation: This is an informational message. The HASH Check function is invoked for this database. HASH total records are written for the database.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP4015E RESERVED NAME WAS USED FOR DBD

Explanation: The reserved name PROCTL01 is specified as an input database name.

System action: Processing stops.

Programmer response: Specify the correct process control information member name on DBDEFCTL DD card. If the both name of control information member and DBD is "PROCTL01", rerun DBD analysis program and recreate the control information member by another name.

FABP4016W "HASH=NO" IS APPLIED TO THE ABOVE DATABASE DATA SET

Explanation: The HASH Check function is not applied to the database data set. The HASH checking is processed for other databases and not for this database.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP4017W REPORT CONTAINS STATISTICS FOR ONLY DDNAME: ***ddname*** **PART OF HISAM DB:** ***dbdname***

Explanation: The HISAM Statistics report contains partial DB information only (either primary DB information or overflow DB information), because of a restriction of image copy hash. For complete DB statistics, refer to another HISAM Statistics report in a pair of primary and overflow DDs.

System action: Processing continues.

Programmer response: For complete DB statistics, refer to another HISAM statistics report in a pair, too.

Problem determination: None.

FABP4018W PARTITION DD: ***ddname dbdname*** **DB#:** ***nnn*** **WAS NOT SCANNED FOR LOGICALLY RELATED DB DBLG#:** ***nnn***

Explanation: Multiple partitions that have direct logical parent pointer were scanned in the Enhanced Image Copy utility job, but the indicated partition was not scanned.

System action: Processing continues.

Programmer response: Make sure that all partitions of a database that has direct logical parent pointers are processed in your HASH evaluation. If you omit any partitions, errors might not be detected.

Problem determination: None.

FABP4019E MEMBER "*xxxxxxxx*" FROM DDNAME DBDEFCTL NOT FOUND

Explanation: The member *xxxxxxxx* does not exist in the data set defined by the DBDFECTL DD statement.

System action: HD Pointer Checker issues a user 4019 abend.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

FABP4020E MEMBER "*xxxxxxxx*" FROM DDNAME DBDEFCTL IS NOT A CONTROL BLOCK

Explanation: The data format of the member *xxxxxxxx* in the data set specified by DBDEFCTL is an incorrect control information.

System action: HD Pointer Checker issues a user 4020 abend.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

FABP4021W PHIDAM PRIMARY INDEX DATABASE DB: ***dbdname*** **CANNOT BE SCANNED WITH "HASH=YES"**

Explanation: The PHIDAM primary index database *dbdname* was not processed with the HASH=YES process option.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABP4022E DLI BATCH REGION IS NEEDED FOR THIS EXECUTION

Explanation: The program FABPHCTL was invoked as an MVS batch program, and High Availability Large Database was included to be processed. However, the DLI region is needed to process High Availability Large Database.

System action: HD Pointer Checker issues a USER 4022 abend.

Programmer response: Modify JCL to use the DLI region, and rerun the job.

Problem determination: None.

FABP4024W PROCCTL DD STATEMENT IS IGNORED

Explanation: HD Pointer Checker does not accept the PROCCTL statements during a single-step HASH checking with IMS Image Copy Extensions or IMS HP Image Copy.

System action: HD Pointer Checker ignores the specifications in the PROCCTL statements and continues processing.

Programmer response: Specify the processing options to the ICEIN statements and, if necessary, remove the PROCCTL statement.

Problem determination: Check the PROCCTL DD in the control statement.

FABP4025W SECONDARY INDEX DB: ***dbdname*** **CANNOT BE CHECKED WITH HASH=YES**

Explanation: The pointer of the secondary index database cannot be checked for one of the following reasons:

- A secondary index database maintenance exit routine is defined in the source segment, but no load module is in the IMS2 DD data set or in the STEPLIB, LOADLIB, or LINKLIST library.
- A segment edit/compression exit routine and a sparse index are defined in the source segment.
- The source segment is a variable length and a segment edit/compression exit routine is defined in it.
- Some of the source segments are suppressed, split to prefix and data portions, and physically deleted.

The secondary index database is scanned, but the index pointer is not checked. Only statistics data is written in a report.

System action: Processing continues.

Programmer response: If you do not need to check the secondary index nor the statistics data, this message can be ignored.

Problem determination: None.

FABP4026E PARTITION DB CANNOT BE PROCESSED

Explanation: HD Pointer Checker does not support IMS Partition DB (5697-A06, 5697-D85) or any other products with an equivalent function.

System action: Pointer validation is taken for the Partition DB. Return Code 08 is returned.

Programmer response: Change the PROCCTL or the ICEIN statement not to invoke the HASH check option.

- For the single-step HASH check option, specify HDPC=N in the Partition DB in the ICEIN control statement.

or

- For the multiple-step HASH check option, remove the DATABASE statement of the Partition DB from the PROCCTL data set.

Problem determination: None.

FABP4027W INDEX DB: ***dbdname*** **CANNOT BE CHECKED WITH IXKEYCHK=YES**

Explanation: Index key in index database cannot be checked for one of the following reasons:

- Some of the source segments are split to prefix and data portions and some are physically deleted.
- The source segment is compressed, but the segment/edit compression routine is not in the IMS2 DD data set or in the STEPLIB, LOADLIB, or the LINKLIST library.

System action: Processing continues.

Programmer response: If you do not need to check the index key, this message can be ignored.

Problem determination: None.

FABP4030E DSPSERV CREATE FAILED. DSPNAME: ***dspname*** **RC=*nn* RSN=*nnnnnnnn***

Explanation: Failed in creating a data space. RC is the return code set in register 15 and RSN is the reason code set in register 0 by the DSPSERV macro.

System action: HD Pointer Checker issues a USER 4030 abend.

Programmer response: Correct the error, and rerun the job.

Problem determination: None.

FABP4031E ALESERV ADD FAILED. DSPNAME: *dspname* **RC=***nn*

Explanation: Failed in getting ALET. RC is the return code set in register 15 by the ALESERV macro.

System action: HD Pointer Checker issues a USER 4031 abend.

Programmer response: Correct the error, and rerun the job.

Problem determination: None.

FABP4032E NAME/TOKEN *service* **FAILED. NAME:** *nametoken* **RC=***nn*

Explanation: Failed in the NAME/TOKEN service. The service is in one of 'IEANTRC', 'IEANTRT', or 'IEANTDL'. RC is the return code of the NAME/TOKEN service.

System action: HD Pointer Checker issues a USER 4032 abend.

Programmer response: Correct the error, and rerun the job.

Problem determination: None.

FABP4033E ASCRE FAILED. PROCNAME: *procname* **RC=***rtncd* **RSN=***rsncd*

Explanation: ASCRE macro failed. RC and RSN are the return and reason code that are set in register 15 and 0 by the ASCRE macro.

System action: HD Pointer Checker issues a USER 4033 abend.

Programmer response: Correct the error, and rerun the job.

Problem determination: For the return code and reason code of the ASCRE macro, see *z/OS MVS Programming Authorized Assembler Services Reference Volume 1* (ALESERV-DYNALLOC).

FABP4034E FABAPTH0 ENDED ABNORMALLY. PROCNAME: *procname*

Explanation: HD Pointer Checker created a FABPATH0 address space by using the PROCNAME procedure, but the address space ended abnormally.

System action: HD Pointer Checker issues a USER 4034 abend.

Programmer response: Check the FABPATH0 job log, correct the error and rerun the job.

Problem determination: None.

FABP4035E ASEXT FAILED. RC=*rtncd* **RSN=***rsncd*

Explanation: The ASEXT macro failed. RC and RSN are the return and reason code that are set in register 15 and 0 by the ASEXT macro.

System action: HD Pointer Checker issues a USER 4035 abend.

Programmer response: Correct the error, and rerun the job.

Problem determination: For the return code and reason code of the ASEXT macro, see *z/OS MVS Programming Authorized Assembler Services Reference Volume 1* (ALESERV-DYNALLOC).

FABP4036E INCORRECT LIBRARY IS USED IN FABPATH0

Explanation: The level of the STEPLIB library in the FABPATH0 procedure is different from the master address space.

System action: HD Pointer Checker issues a USER 4036 abend.

Programmer response: Specify the same IMS HP Pointer Checker load module library as the MASTER JCL in the FABPATH0 STEPLIB.

Problem determination: Check the STEPLIB data set in the FABPATH0 procedure.

FABP4037E ATTACH FAILED FOR DFSRRC00 IN FABPATH0

Explanation: A nonzero return code was returned from the ATTACH macro when DFSRRC00 was attached in the FABPATH0 address space, which is an IMS HP Pointer Checker subordinate address space. See message in the FABPATH0 job.

System action: HD Pointer Checker issues a USER 4037 abend.

Programmer response: Correct the error, and rerun the job.

Problem determination: For the reason the ATTACH macro failed, see the return code shown in message FABP3690E, which is issued in the FABPATH0 address space.

FABP4095E RECON ACCESS FAILED. DBRC LIST COMMAND IS NOT COMPLETED. RC=*xxxxxxxx*

FABP4095E RECON ACCESS FAILED. SYSPRINT DD FOR DBRC LIST COMMAND IS SPECIFIED AS DUMMY

FABP4095E RECON ACCESS FAILED. INTERNAL ERROR OCCURED

FABP4095E RECON ACCESS FAILED. FUNC=*ffffffff* RETURN CODE=*xxxxxxxx* REASON CODE=*xxxxxxxx* KEYS: DBD=*dbdname* DDN=*ddname* KEYTYPE=*xx xxxxxxxxxx*

Explanation: An error was detected in the RECON access processing.

System action: HD Pointer Checker issues a USER 4095 abend.

Programmer response: Correct the error, and rerun the job.

Problem determination: None.

FABP8001E STATEMENT FORMAT ERROR

Explanation: An incorrect control statement was detected by the control statement syntax checking.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Check the control statement.

FABP8002E *parm-name* IS INVALID AS STATEMENT PARAMETER

Explanation: An incorrect control statement was detected by the control statement syntax checking.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Check the control statement.

FABP8003E *parm-name* PARAMETER IS INVALID FOR *ctl-stmt-name* STATEMENT

Explanation: An incorrect control statement was detected by the control statement syntax checking.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Check the control statement.

FABP8004E NUMBER OF *parm-name* PARAMETERS EXCEEDED THE LIMIT, MAX IS *nn*

Explanation: An incorrect control statement was detected by the control statement syntax checking.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Check the control statement.

FABP8005E *parm-name* PARAMETER IS REQUIRED FOR *ctl-stmt-name* STATEMENT

Explanation: An incorrect control statement was detected by the control statement syntax checking.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Check the control statement.

FABP8006E NUMBER OF OPERAND FOR *parm-name* PARAMETER EXCEED THE LIMIT, MAX IS *nn*

Explanation: An incorrect control statement was detected by the control statement syntax checking.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Check the control statement.

FABP8007E LENGTH ERROR IN *n-th* OPERAND OF PARAMETER: *parm-name*

Explanation: An incorrect control statement was detected by the control statement syntax checking.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Check the control statement.

FABP8008E *n-th* OPERAND IS REQUIRED FOR PARAMETER: *parm-name*

Explanation: An incorrect control statement was detected by the control statement syntax checking.

System action: Processing stops.

Programmer response: Correct the error, and rerun the HD Pointer Checker job.

Problem determination: Check the control statement.

FABP8009I (HPIC) AND SUBSEQUENT PARAMETERS ARE IGNORED

Explanation: Only ″(HDPC)″ statement is valid for HD Pointer Checker. ″(HPIC)″ and subsequent parameters are ignored because they are for IMS HP Image Copy.

System action: Processing continues. HD Pointer Checker does not check the syntax of the ″(HPIC)″ and subsequent parameters.

Programmer response: None.

Problem determination: None.

HD Tuning Aid

This section describes the abend codes, return codes, and messages issued by HD Tuning Aid.

Abend codes

Every 3*nnn* abend code is accompanied by a FABT3*nnn*E message. (3*nnn* is a four-digit identification number of the abend code and message.) See the associating FABT3*nnn*E message description for the *3nnn* abend code.

Return codes

These are the return codes set by HD Tuning Aid:

Code	Meaning
0	HD Tuning Aid has been successfully executed. No severe errors have been detected.
8	HD Tuning Aid has experienced severe errors and the processing stops. Check the JES log and reports for messages to correct the error and rerun the job.

Messages

This section explains the messages that are issued by HD Tuning Aid.

Message format

All the messages have the following format:

```
FABTnnnnx text
```

Where:

nnnn Is a four-digit message identification number.

x Indicates the severity of the message as follows:

I Information message

W Warning message

E Error message.

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 may represent such things as:

- The name of a data set
- The number of data records
- A name or value provided by the user
- The name of a partition
- The name of a DBD

Messages

FABT3504E MESSAGE TEXT NOT FOUND

Explanation: HD Tuning Aid attempted to print an error message that could not be found in the message table in module FABTMSGS.

System action: HD Tuning Aid ended with the USER 3504 abend.

Programmer response: Verify that IMS HP Pointer Checker, including all current maintenance, was installed correctly. Reinstall IMS HP Pointer Checker, including all maintenance, if necessary. Then rerun the job. If the problem persists, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Either IMS HP Pointer Checker or its maintenance may have been installed incorrectly.

FABT3505E INVALID MESSAGE FLAG

Explanation: HD Tuning Aid attempted to print an error message, and an incorrect flag was supplied to module FABTMSGS.

System action: HD Tuning Aid ended with the USER 3505 abend.

Programmer response: Verify that IMS HP Pointer Checker, including all current maintenance, was installed correctly. Reinstall IMS HP Pointer Checker, including all maintenance, if necessary. Then rerun the job. If the problem persists, save the entire run listing (including the dump, JCL, and all reports from IMS HP Pointer Checker), and contact IBM Software Support.

Problem determination: Either IMS HP Pointer Checker or its maintenance may have been installed incorrectly.

FABT3521E OPEN FAILED FOR KEYS*xxx* FILE

Explanation: The indicated data set was not opened successfully. Where KEYS*xxx* is:

KEYSIN It contains root segment keys (created by the HD Pointer Checker).

KEYSOUT It contains block/RAP assignments (created by FABTROOT).

System action: Processing stops.

Programmer response: Correct the error and rerun the job.

Problem determination: Check for JCL errors. Otherwise, rerun the HD Pointer Checker job to re-create the indicated data set.

FABT3522E EMPTY KEYSIN DATA SET

FABT3522I EMPTY KEYSIN DATA SET

FABT3522W EMPTY KEYSIN DATA SET

Explanation: The data set containing the root segment keys (created by the HD Pointer Checker) is empty. The change of severity can be requested by the %OPTION statement in the CTL data set.

System action: Processing stops.

Programmer response: Correct the error and rerun the job.

Problem determination: Check for JCL errors.

FABT3523E NO TYPE R RECORD CREATED FOR ALL OF CONTROL STATEMENTS

Explanation: FABTROOT processed for all input control statements. But, None of DB name specified in the control statements could be found in the KEYSIN data set.

System action: Processing stops.

Programmer response: Correct the error and return the job.

Problem determination: None.

FABT3525I PROCESSING COMPLETED FOR THIS DATABASE

Explanation: FABTROOT completed processing of all KEYSIN records for the database.

System action: If there are more databases to be processed, then processing continues. Otherwise, processing stops.

Programmer response: None.

Problem determination: None.

FABT3530E ERROR FOUND IN COLUMN *nn* OF %OPTION STATEMENT

Explanation: Incorrect value was found in column *nn* in the %OPTION statement.

System action: Processing stops.

Programmer response: Correct the error and rerun the job.

Problem determination: None.

FABT3531E MORE THAN ONE %OPTION STATEMENT SPECIFIED

Explanation: More than one %OPTION statements were specified. Only one %OPTION statement can be specified in the CTL data set.

System action: Processing stops.

Programmer response: Correct the error and rerun the job.

Problem determination: None.

FABT3532E %OPTION STATEMENT SPECIFIED AFTER DB OR PART STATEMENT

Explanation: The %OPTION statement was specified after one or more DB or PART statements. It should be the first statement in the CTL data set.

System action: Processing stops.

Programmer response: Correct the error and rerun the job.

Problem determination: None.

FABT3535I END-OF-FILE ON CONTROL FILE OR STOP REQUEST

Explanation: Either FABTROOT processed all input control statements, or it processed a control statement with a **1** in column 10.

System action: Processing stops.

Programmer response: None.

Problem determination: None.

FABT3537E MAXIMUM NUMBER OF CONTROL STATEMENTS (DB STATEMENTS) EXCEEDED KEYSIN FILE

Explanation: More than 100 DB statements are specified.

System action: HD Tuning Aid ended with USER 3537 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: The maximum number of DB statements is 100.

FABT3538E DUPLICATE HEADER RECORD FOUND IN KEYSIN DATA SET FOR DB: *dbdname*

Explanation: HD Tuning Aid found the duplicate header record in the data set containing the root segment keys (created by the HD Pointer Checker) for the database *dbdname* requested on the control statement.

System action: HD Tuning Aid issues a USER 3538 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Check for JCL errors. Rerun the HD Pointer Checker job to recreate the KEYSIN data set.

FABT3545I ACTUAL ROOTS/BLOCK MAP WILL NOT BE REPRINTED

Explanation: FABTROOT is being run with control statement input.

System action: The Actual Roots per Block report is not printed. It is assumed that this report was printed by an earlier HD Tuning Aid run that did not use control statement input.

Programmer response: None.

Problem determination: None.

FABT3568E NO DMB FOUND FOR THE DB NAME IN CONTROL STATEMENT

Explanation: FABTROOT could not find the name specified in column 1 of your control statement in the DMB.

System action: FABTROOT issues a USER 3568 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Check the FABTROOT control statement for spelling errors. Make sure the correct PSB name is used in the EXEC statement. The PSB must contain a PCB for that database.

FABT3570E FIND FAILED FOR DBD IN DBDLIB

Explanation: A FIND macro was issued for the dbdname. But no module with that name could be found in the library on the IMS DD statement.

System action: FABTROOT issues a USER 3570 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Check the FABTROOT control statement for spelling errors. Make sure the DBD library is part of the IMS DD statement.

FABT3590E HEADER RECORD IS MISSING IN KEYS*xxx* FILE

Explanation: The first record in the indicated data set containing the root segment keys (created by the HD Pointer Checker) is not in the correct format. Its second half word is not XX'0000'. Where KEYS*xxx* is:

KEYSIN	It contains root segment keys (created by the HD Pointer Checker).
KEYSOUT	It contains block/RAP assignments (created by FABTROOT).

System action: HD Tuning Aid issues a USER 3590 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Check for JCL errors. If necessary, rerun the HD Pointer Checker job to recreate the indicated data set.

FABT3601I *nnnnnnnnnn* TYPE K RECORDS PROCESSED

Explanation: *nnnnnnnnnn* data records from the data set containing the root segment keys (created by the HD Pointer Checker) were processed for this data set. (*nnnnnnnnnn* indicates the number of data records.)

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABT3602I *nnnnnnnnnn* TYPE R RECORDS CREATED

FABT3602I *nnnnnnnnnn* TYPE R RECORDS PROCESSED

Explanation: *nnnnnnnnnn* data records from the work data set containing the block/RAP assignments were created by FABTROOT or processed by FABTRAPS. (*nnnnnnnnnn* indicates the number of data records.)

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABT3603I *dbdname* DBD NOT EITHER HDAM OR PHDAM, SKIPPING FOR NEXT FILE

Explanation: The indicated HIDAM or PHIDAM database dbdname is processed with no control statement. No reports describing HDAM randomization are produced.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABT3610E RBA OF KEY BLOCK LESS THAN CURRENT BEGINNING BLOCK

Explanation: The data records from the data set containing the root segment keys (created by the HD Pointer Checker) are not in the correct sequence.

System action: HD Tuning Aid issues a USER 3610 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Rerun the HD Pointer Checker to recreate the bad data set.

FABT3611E KEYSOUT BLOCK NUMBERS ARE OUT OF SEQUENCE

Explanation: The data records from the work data set containing the block/RAP assignments (created by FABTROOT and sorted by DFSORT) are not in the correct sequence.

System action: HD Tuning Aid issues a USER 3611 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Check the DFSORT step (between the FABTROOT and FABTRAPS steps) for errors.

FABT3615E DBD SUPPLIED NOT HDAM TYPE, AND NO HDAM OR PHDAM OVERRIDE SPECIFIED IN CONTROL STATEMENT

Explanation: The input DBD is not type HDAM, but you did not override the control statement by specifying HDAM in columns 58-61 or PHDAM in columns 58-62.

System action: HD Tuning Aid ends with a USER 3615 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Check errors in the control statement.

FABT3616E DATABASE ORGANIZATION NOT EITHER HDAM OR PHDAM

Explanation: The input DBD is not either HDAM or PHDAM, but you did not override this with the control statement by specifying "HDAM" or "PHDAM" in columns 58-61.

System action: HD Tuning Aid issues a USER 3616 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Check for control statement errors.

FABT3620E RAP VALUE GREATER THAN NUMBER OF RAP'S PER BLOCK

Explanation: One of the data records from the work data set containing the block/RAP assignments (created by FABTROOT) contains a RAP assignment that is larger than allowed by the DBD.

System action: HD Tuning Aid issues a USER 3620 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Check for JCL errors. An old version of the work data set might have been used.

FABT3625E KEYSOUT FILE DD STATEMENT MISSING

Explanation: The DD statement for the work data set containing the block/RAP assignments (created by FABTROOT) is missing.

System action: HD Tuning Aid issues a USER 3625 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Check for JCL errors.

FABT3627E DUMMY SPECIFIED FOR KEYS*xxx* FILE

Explanation: The DD statement for the indicated data set was coded as DUMMY. Where KEYS*xxx* is:

KEYSIN
It contains root segment keys (created by the HD Pointer Checker).

KEYSOUT
It contains block/RAP assignments (created by FABTROOT).

System action: The HD Tuning Aid issues a USER 3627 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Check for JCL errors.

FABT3630E ROOT SEGMENT NOT FOUND IN DBD SEGTAB

Explanation: The DBD control block in memory was damaged.

System action: HD Tuning Aid issues a USER 3630 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Check that the DBD load module is valid.

FABT3640E DATABASE ORGANIZATION IS NOT EITHER HDAM OR PHDAM AND SOME CTL STATEMENT FIELDS ARE ZERO/BLANK

Explanation: HD Tuning Aid control statement was coded incorrectly. A required field was blank.

System action: HD Tuning Aid ends with a USER 3640 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: If the DBD is not either HDAM or PHDAM, all DBD override fields on the control statement must not be blanks.

FABT3650E UNSUPPORTED IMS LEVEL OF IMS IS BEING USED: *vv* IMS LEVEL OF THIS RUN

Explanation: Self-explanatory.

System action: HD Tuning Aid ends with a USER 3560 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Check IMS level coded on JCL.

FABT3660E KEY START POSITION OR KEY LENGTH ON CONTROL STATEMENT IS NOT WITHIN KEY FIELD

Explanation: A key start position or key length was not specified within the root segment key field.

System action: HD Tuning Aid issued a USER 3660 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Check the DBD for location and length of the root segment key field.

FABT3665W CALCULATED KEY LENGTH=*nnn*

Explanation: A key start position other than the beginning of the root segment key field had been specified, but HD Tuning Aid adjusted the key length to be *nnn* which is the new start and end of the key field.

System action: Processing continues.

Programmer response: None

Problem determination: None

FABT3670E FIND FAILED FOR HDAM RANDOMIZING MODULE IN RESLIB

Explanation: A FIND macro was issued for the HDAM randomizing routine that was specified either on the control statement or in the DBD load module. No module with that name could be found in the library on the IMS2 DD statement.

System action: HD Tuning Aid issues a USER 3670 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Check the control statement for spelling errors in the *mod* (column 11) field. Make sure the correct partitioned data set is specified on the IMS2 DD statement.

FABT3680W KEY FILE SEARCH FOR DBD= *dbdname* **FAILED THAT WAS SPECIFIED BY CONTROL STATEMENT**

Explanation: The data set containing the root segment keys (created by the HD Pointer Checker) does not contain any records for the database *dbdname* requested on the control statement.

System action: Processing continues.

Programmer response: Correct the error and rerun the job.

Problem determination: Check the control statement for spelling errors in the *dbdname* (column 1) field. Make sure the correct input KEYSIN data set is used.

FABT3685I KEY FILE FOR *dbdname* **DBD SKIPPED AS REQUESTED**

Explanation: The records on the data set containing the root segment keys (created by the HD Pointer Checker) that refer to the database *dbdname* requested on the control statement were not processed by the HD Tuning Aid.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABT3697W SPECIFIED ALGORITHM RANDOMIZED; RC4= *nnnnnnnn* **RC8=** *mmmmmmmm*

Explanation: HD Tuning Aid received the indicated number of return code 4/8 from the randomizer routine for the root segments.

System action: Processing continues.

Programmer response: An application program will receive the FM status code and/or abend U812 for the root segment keys by using the specified randomizer routine.

Problem determination: None.

FABT3699E ’BLKSZIN’ OR ’NBLKSIN’ (ON CONTROL STATEMENT) IS TOO LARGE

Explanation: An incorrect value or combination of values were found in the rbn (column 20) or blksz (column 34) field of control statement.

System action: HD Tuning Aid issues a USER 3699 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Verify that the fields in question are numeric (including leading zeros), and the combination of the 'BLKSZ' multiplied by the 'RBN' does not exceed 4 gigabytes for a VSAM data set, or an even number and 8 gigabytes for an OSAM data set.

FABT3700I NO MORE RECORDS FOR THIS FILE

Explanation: No more data records containing the block/RAP assignments (created by FABTROOT and sorted by DFSORT) remains for this data set.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABT3710E INVALID PARTITION NUMBER ’*xxxxxxxx***’ RETURNED FROM RANDOMIZER: DBF***yyyyy*

Explanation: Fast Path randomizer DBF*yyyyy* returned an incorrect partition number.

System action: HD Tuning Aid ended with the USER 3710 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Make sure the correct randomizer is used for the run.

FABT3711E INVALID RAP NUMBER ’*xxxxxxxx***’ FOR PART#:** *nnn* **RETURNED FROM RANDOMIZER: DBF***yyyyy*

Explanation: Fast Path randomizer DBF*yyyyy* returned an incorrect relative RAP number for the indicated partition (*nnn*).

System action: HD Tuning Aid ended with the USER 3711 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Make sure the correct randomizer is used for the run.

FABT3801E MAXIMUM NUMBER OF CONTROL STATEMENTS (PART STATEMENTS) EXCEEDED FOR DBD *dbdname*

Explanation: More than 32 PART statements are specified for a database.

System action: HD Tuning Aid ended with the USER 3801 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: The maximum number of the PART statements is 32.

FABT3802E ’PARTINI’ ON DB STATEMENT REQUIRES PART STATEMENT(S)

Explanation: No PART statement was specified when the PARTINI keyword was specified on the DB statement.

System action: HD Tuning Aid ended with the USER 3802 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: PARTINI requires a complete set of PART statements for a PHDAM database.

FABT3803E ’PARTDEL’ ON DB STATEMENT DOES NOT ALLOW PART STATEMENT(S)

Explanation: The PART statement was specified when the PARTDEL keyword was specified on the DB statement.

System action: HD Tuning Aid ended with the USER 3803 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: PARTDEL allows no PART statement to be specified.

FABT3804E PART STATEMENT(S) FOR NON-PARTITIONED DB ’PARTINI’ ON DB STATEMENT REQUIRED

Explanation: Self-explanatory.

System action: HD Tuning Aid ended with the USER 3804 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: For migration to a PHDAM database from the other database, 'PARTINI' and a complete set of PART statements are required.

FABT3807E ’PARTDEL’ ON DB STATEMENT IS SPECIFIED FOR HALDB

Explanation: Self-explanatory.

System action: HD Tuning Aid ends with a USER 3807 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: The PARTDEL parameter is used only for migration from HALDB to a HDAM database.

FABT3810E INVALID STATEMENT SPECIFIED ON CONTROL STATEMENT

Explanation: The statement is not recognized as a DB statement, a PART statement, or a comment statement.

System action: Processing continues. The incorrect statement is skipped.

Programmer response: None.

Problem determination: See the FABTROOT CTL data set format.

FABT3811E DDNAME: *ddname* IS NOT FOUND IN DBD

Explanation: DDNAME *ddname* specified on the PART statement is not found in the DBD of the partitioned database. *ddname* must match the ddname on the DBD when overriding parameters without changing the number of partitions.

System action: HD Tuning Aid ends with the USER 3811 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: None.

FABT3812E ’SIZE1’ ON PART STATEMENT IS NOT AN EVEN NUMBER

Explanation: The *size1* value specified on the PART statement is an odd number. It muse be an even number.

System action: HD Tuning Aid ends with the USER 3812 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: None.

FABT3813E MIGRATION FROM HIDAM TO PARTITIONED HDAM. ’PARTINI’ ON DB STATEMENT REQUIRED

Explanation: Self-explanatory. This type of migration requires ’PARTINIT’ on the DB statement.

System action: HD Tuning Aid ends with the USER 3813 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: ’PARTINI’ is required in this case.

FABT3814E PART RECORD FOR PART#: ***nn*** **IS OUT OF SEQUENCE IN KEYSIN FILE**

Explanation: The PART record in the KEYSIN data set is out of partition number sequence. *nn* indicates the partition number of the PART record in error.

System action: HD Tuning Aid ends with the USER 3814 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Make sure that the KEYSIN files are concatenated in ascending order of the partition number.

FABT3850E FIND FAILED FOR PARTITION SELECTION MODULE ***modname*** **IN IMS2**

Explanation: A FIND macro was issued for the partition selection exit specified in the control statement, but no module with that name could be found in the library on the IMS2 DD statement.

System action: HD Tuning Aid ends with a USER 3850 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Check the control statement for spelling errors in the xxx (column xx) field. Make sure the correct partitioned data set is specified on the IMS2 DD statement.

FABT3851E ’AUTOINI’ ON DB STATEMENT DOES NOT ALLOW PART STATEMENT(S)

Explanation: The PART statement was specified when the AUTOINI keyword was specified on the DB statement.

System action: HD Tuning Aid ends with a USER 3851 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: AUTOINI allows only partition high key or partition selection strings after the DB statement.

FABT3852E ’PARTINI’ REQUIRES PARTITION SELECTION EXIT OR PARTITION HIGH KEY

Explanation: The PARTINI keyword was specified in the DB statement, but neither S nor H was specified in column 72.

System action: HD Tuning Aid ends with a USER 3852 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: PARTINI requires a complete set of PART statements for PHDAM.

FABT3853E ’AUTOINI’ REQUIRES PARTITION SELECTION EXIT OR PARTITION HIGH KEY

Explanation: The PARTINI keyword was specified on the DB statement, but neither S nor H was specified in column 72.

System action: HD Tuning Aid ends with a USER 3853 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: AUTOINI requires complete information about the partition selection.

FABT3854E PARTITION SELECTION EXIT NAME IS REQUIRED ON DB STATEMENT

Explanation: 'S' was specified in column 72 on the DB statement, but the exit module name of the partition selection is not specified.

System action: HD Tuning Aid ends with a USER 3854 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: The partition selection exit name is required in column 2-9 on the continuous line of the DB statement.

FABT3855E PARTITION SELECTION STRING OR HIGH KEY WAS NOT SPECIFIED ON PART STATEMENT

Explanation: '+' was specified in column 72 on the PART statement line 1, but the partition selection string or high key is not specified correctly.

System action: HD Tuning Aid ends with a USER 3855 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: The partition selection string or high key is required on the continuous line of the PART statement.

FABT3856E ’AUTOINI’ ON DB STATEMENT REQUIRES PARTITION HIGH KEY OR PARTITION SELECTION STRING

Explanation: The AUTOINI keyword was specified on the DB statement, but neither the partition high key nor the partition selection string was specified on the continuous line of the PART statement.

System action: HD Tuning Aid ends with a USER 3856 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: AUTOINI requires complete information about the partition selection.

FABT3857E GETMAIN FAILED. RETURN CODE: ***xx*** **SIZE:** ***nnnn*** **K**

Explanation: A GETMAIN macro was issued, but failed.

System action: HD Tuning Aid ends with a USER 3857 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Analyze the return code. If the reason is that there is not enough storage available to GETMAIN, increase the region size of the job or decrease the number of databases specified by CTL statement.

FABT3858E LOAD FAILED FOR DDNAME ***ddname*** **MODULE** ***modname***

Explanation: A LOAD macro was issued for the specified module, but no module could be loaded.

System action: HD Tuning Aid ends with a USER 3858 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Make sure that the member specified exists in the data set specified for the specified ddname.

FABT3859E PARTITION SELECTION FAILED FOR HALDB. FUNCTION: ***xxxxx*** **RETURN CODE:** ***nn*** **REASON CODE:** ***mmmmm***

Explanation: HD Tuning Aid received the specified return code from partition selection. The return code and reason code are specified in hexadecimal.

System action: HD Tuning Aid ends with a USER 3859 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Verify that the partition selection exit name and the partition selection strings, or the partition high keys, are correct.

FABT3860E MAXIMUM NUMBER OF PARTITIONS EXCEEDED FOR DBD ***dbdname***

Explanation: The specifications in the CTL statement have caused the total number of partitions for the specified DBD to exceed 1001.

System action: HD Tuning Aid ends with a USER 3860 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: The maximum number of partitions for HALDB is 1001. Verify the total number of partitions defined in DBD and specified by ADD/DEL parameters in the PART statements.

FABT3861E SPECIFIED PARTITION NAME ***partname*** **DOES NOT EXIST IN DBD** ***dbdname*****.**

Explanation: The partition name specified to be deleted by the DEL parameter in PART statement was not defined in the DBD.

System action: HD Tuning Aid ends with a USER 3861 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Verify the partition name.

FABT3862I NO TYPE R RECORD CREATED FOR DBD ***dbdname*** **PARTITION** ***partname***

Explanation: HD Tuning Aid processed all KEY records of the DBD specified, but no data record containing the block/RAP assignments was created for the specified partition.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABT3864I PARTITION ***partname*** **WAS DELETED AS REQUEST**

Explanation: The partition specified was deleted as requested by the DEL parameter on the PART statement.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABT3865E PARTITION NAME REQUIRES FOR PART STATEMENT FOR DBD: *dbdname*

Explanation: The HD Tuning Aid control statement was coded incorrectly. The ’partname’ parameter is required for the PART statement.

System action: HD Tuning Aid ends with a USER 3865 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Verify columns 2 through 8 in the PART statement.

FABT3866E ’ADD’ SPECIFIED BUT SOME CTL STATEMENT FIELDS ARE ZERO/BLANK

Explanation: The HD Tuning Aid control statement was coded incorrectly. The ADD parameter was specified in the PART statement, but some required fields were not specified.

System action: HD Tuning Aid ends with a USER 3866 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: For each additional partition, a complete set of parameters is required in the DB statement or the PART statement.

FABT3867E RANDOMIZER NAME IS NOT SPECIFIED FOR MIGRATION TO PHDAM

Explanation: The HD Tuning Aid control statement was coded incorrectly. No randomizer name was specified on the DB statement or the PART statement.

System action: HD Tuning Aid ends with a USER 3867 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: For migration from non-PHDAM to PHDAM, the randomizer name is required in the DB statement or the PART statement.

FABT3868E RAA BLOCK NUMBER IS NOT SPECIFIED FOR MIGRATION TO PHDAM

Explanation: The HD Tuning Aid control statement was coded incorrectly. No RAA block number was specified in the DB statement or the PART statement.

System action: HD Tuning Aid ends with a USER 3868 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: For migration from non-PHDAM to PHDAM, the RAA block number parameter is required in the DB statement or the PART statement.

FABT3869E RAP NUMBER PER BLOCK IS NOT SPECIFIED FOR MIGRATION TO PHDAM

Explanation: The HD Tuning Aid control statement was coded incorrectly. The RAP number per block was not specified in the DB statement or the PART statement.

System action: HD Tuning Aid ends with a USER 3869 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: For migration from non-PHDAM to PHDAM, the RAP number per block parameter is required in the DB statement or the PART statement.

FABT3870E ’AUTOINI’ BUT SOME DB STATEMENT FIELDS ARE ZERO/BLANK

Explanation: The HD Tuning Aid control statement was coded incorrectly. The AUTOINI parameter is specified, but some required fields are not specified in the DB statement.

System action: HD Tuning Aid ends with a USER 3870 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: The AUTOINI parameter requires a complete set in the DB statement.

FABT3871E EITHER ’PARTINI’ OR ’AUTOINI’ IS NOT SPECIFIED FOR MIGRATION TO PHDAM

Explanation: The HD Tuning Aid control statement was coded incorrectly. The PHDAM parameter is specified without either PARTINI or AUTOINI parameter in the DB statement.

System action: HD Tuning Aid ends with a USER 3871 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: For migration from non-PHDAM database to a PHDAM database, the

’PARTINI’ or the ’AUTOINI’ parameter is required in the DB statement.

FABT3872E PARTDEL SPECIFIED FOR MIGRATION FROM PHIDAM TO PHDAM

Explanation: The HD Tuning Aid control statement was coded incorrectly. The PHDAM parameter is specified with PARTDEL in the DB statement for migration from a PHIDAM database to a PHDAM database.

System action: HD Tuning Aid ends with a USER 3872 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: For migration from a PHIDAM database to a PHDAM database, PARTDEL is not allowed in the DB statement.

FABT3873E ’PARTDEL’ IS NOT SPECIFIED FOR MIGRATION FROM HALDB TO NON-HALDB

Explanation: The HD Tuning Aid control statement was coded incorrectly. The HDAM parameter is specified without PARTDEL in the DB statement for migration from a HALDB to a HDAM database.

System action: HD Tuning Aid ends with a USER 3873 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: For migration from a HALDB database to a HDAM database, 'PARDEL' is required in the DB statement.

FABT3874E COLUMN 72 ON DB STATEMENT IS NOT BLANK FOR MIGRATION FROM HALDB TO NON-HALDB

Explanation: The HD Tuning Aid control statement was coded incorrectly. The HDAM parameter is specified with a character in column 72 on the DB statement for migration from a HALDB to a HDAM database.

System action: HD Tuning Aid ends with a USER 3874 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: For migration from a HALDB database to a HDAM database, only a blank is allowed in column 72 on the DB statement.

FABT3875E PART STATEMENT EXISTS FOR MIGRATION FROM HALDB TO NON-HALDB

Explanation: The HD Tuning Aid control statement was coded incorrectly. HDAM parameter is specified with a PART statement for migration from a HALDB to a HDAM database.

System action: HD Tuning Aid ends with a USER 3875 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: For migration from a HALDB database to a HDAM database, a PART statement is not allowed.

FABT3876E SOME DB STATEMENT FIELDS ARE ZERO/BLANK FOR MIGRATION FROM HALDB TO NON-HALDB

Explanation: The HD Tuning Aid control statement was coded incorrectly. The HDAM parameter is specified, but some required fields are not specified in the DB statement for migration from a HALDB to a HDAM database.

System action: HD Tuning Aid ends with a USER 3876 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: For migration from a HALDB database to a HDAM database, a complete set is required in the DB statement.

FABT3877E PARTITION *partname* ADD SPECIFIED AGAINST ALREADY EXIST PARTITION FOR DBD *dbdname*

Explanation: The HD Tuning Aid control statement was coded incorrectly. The ADD parameter is specified, but the partition name is already defined for the specified database.

System action: HD Tuning Aid ends with a USER 3876 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Verify the partition name in the PART statement. Duplicate partition names are not allowed in a HALDB.

FABT3878E PARTITION ID EXCEEDED 32767 FOR HALDB. DBD: *dbdname*

Explanation: An attempt to add partitions to the database failed, because the maximum partition ID for HALDB is 32767.

System action: HD Tuning Aid ends with a USER 3878 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Reorganize the whole partitions and reassign the partition IDs by the ISPF/PDF partition definition utility.

FABT3879E PARTITION HIGH KEY OR SELECTION STRING EXCEEDED 1001 FOR HALDB. DBD: *dbdname*

Explanation: The HD Tuning Aid control statements were coded incorrectly. The maximum number of partition high keys or partition selection strings is 1001 for HALDB.

System action: HD Tuning Aid ends with a USER 3879 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Verify the specified number of partition high keys or partition selection strings.

FABT3880E INVALID IMS RELEASE LEVEL OR NON IMS ENVIRONMENT FOR HALDB KEYSIN

Explanation: The input KEYSIN file contained some key data for HALDB, but HD Tuning Aid was run under the MVS batch environment.

System action: HD Tuning Aid ends with a USER 3880 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: For HALDB key data processing, HD Tuning Aid must be run under the IMS batch environment.

FABT3881E KEYSIN HEADER RECORD INDICATE INVALID DB ORGANIZATION

Explanation: The input KEYSIN file contained incorrect database organization information in a header record.

System action: HD Tuning Aid ends with a USER 3880 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Use the KEYSIN file that is created by HD Pointer Checker Version 3 Release 1.

FABT3882E PART RECORD FOR PARTITION ID: *nn* **IS OUT OF SEQUENCE IN KEYSIN FILE**

Explanation: The PART record in the KEYSIN data set is out of partition ID sequence. *nn* indicates the partition ID of the PART record in error.

System action: HD Tuning Aid ends with a USER 3882 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Make sure that the KEYSIN files are concatenated in the ascending order of the partition IDs.

FABT3883E KEYSIN HEADER RECORD INDICATE *xxxxxx* **BUT DBD WAS** *yyyyyy*

Explanation: The database organization defined in DBD is not the same as that of the KEYSIN file.

System action: HD Tuning Aid ends with a USER 3883 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Make sure that the KEYSIN files and the DBD are correct. These two definitions must match.

FABT3884E PHDAM ’SPECIFIED’ BUT SOME CTL STATEMENT *xxxxxxxx* **FIELDS ARE ZERO/BLANK**

Explanation: The HD Tuning Aid control statement was coded incorrectly. PHDAM parameter is specified, but the indicated required field is not specified on a DB statement or a PART statement.

System action: HD Tuning Aid ends with a USER 3884 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: For migration to a PHDAM, a complete DB statement or a PART statement is required.

FABT3885E ’HDAM’ SPECIFIED AGAINST ’AUTOINI’ ON DB STATEMENT

Explanation: The HD Tuning Aid control statement was coded incorrectly. The HDAM parameter is specified with AUTOINI on the DB statement.

System action: HD Tuning Aid ends with a USER 3885 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: For migration to an HDAM, AUTOINI is not allowed in the DB statement.

FABT3886E PHDAM IS NOT SPECIFIED FOR MIGRATION FROM NON-HALDB TO HALDB BY 'AUTOINI'

Explanation: The HD Tuning Aid control statement was coded incorrectly. The input database organization is non-HALDB and AUTOINI parameter is specified, but PHDAM is not specified in the DB statement.

System action: HD Tuning Aid ends with a USER 3886 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: For migration from non-HALDB to a PHDAM, PHDAM is required in the DB statement.

FABT3887E NON ZERO RETURN CODE FROM CSVQUERY FOR PARTITION SELECTION EXIT *modname*. RC=*nn*

Explanation: CSVQUERY macro failed. The return code was returned from CSVQUERY.

System action: HD Tuning Aid ends with a USER 3887 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Check the HD Tuning Aid control statement for spelling errors. Make sure that the IMS2 data set contains the module specified by *modname*.

FABT3888E PARTITION SELECTION EXIT *modname* IS NOT REENTRANT

Explanation: The partition selection exit module specified was not reentrant.

System action: HD Tuning Aid ends with a USER 3888 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: The partition selection exit for HALDB must be a reentrant module.

FABT3890E LINK FAILED FOR IDCAMS

Explanation: HD Tuning Aid issued a LINK macro for an IDCAMS program, but failed.

System action: HD Tuning Aid issues a USER 3890 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Check that the input database data set actually exists, or make sure the IDCAMS module is in the system library correctly.

FABT3891E PR10 DD DATA SET OPEN FAILED

Explanation: PR10 DD data set was not opened successfully. OPEN macro failed for PR10 data set.

System action: HD Tuning Aid issues a USER 3891 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Check the PR10 data set.

FABT3893E 'PARTINI' IS SPECIFIED FOR MIGRATION FROM HALDB TO NON-HALDB

Explanation: The HD Tuning Aid control statement was coded incorrectly. The HDAM parameter is specified with PARTINI in the DB statement for migration from a HALDB to a HDAM database.

System action: HD Tuning Aid ends with a USER 3893 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: For migration from a HALDB database to a HDAM database, PARTINI is not allowed in the DB statement.

FABT3895E DUPLICATE VALUE SPECIFIED FOR PARTITION HIGH KEY

Explanation: The same value is specified for the partition high key of two partitions.

System action: HD Tuning Aid issues a USER 3587 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Check the PART statement (in the CTL DD statement) and the actual HALDB definitions for the partition high key.

FABT3896E INVALID LENGTH OF PARTITION SELECTION STRING OR HIGH KEY

Explanation: The partition string length or high key length specified on the PART statement is incorrect.

System action: HD Tuning Aid issues a USER 3896 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: The partition string or high key must contain no more than 255 characters. The partition high key length must be equal to the root key length.

FABT3897E INVALID VALUE SPECIFIED FOR PARTITION SELECTION STRING OR HIGH KEY

Explanation: The partition string or high key characters specified on the PART statement is incorrect.

System action: HD Tuning Aid issues a USER 3897 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Verify the PART statement line 2 and after.

FABT3899E DUPLICATE PARTITION NAME *xxxxxxx* ON PART STATEMENT

Explanation: The same partition name is specified in more than two PART statements.

System action: HD Tuning Aid issues a USER 3899 abend.

Programmer response: Correct the error and rerun the job.

Problem determination: Verify PART statements.

DB Historical Data Analyzer

This section describes the abend codes, return codes, and messages issued by DB Historical Data Analyzer.

Abend codes

- **MVS Batch Environment**

  Every 3*nnn* abend code is accompanies by a FABG3*nnn*E messages. (3*nnn* is a four-digit identification number of the abend code and message.) See the associated FABG3*nnn*E message description for the 3*nnn* abend code.

  The abend code 3999 is accompanied by the FABG3999E or FABG8999E message.
- **TSO/ISPF Environment**

  The abend code 3999 is accompanied by the FABG399E message.

Return codes

Return codes are provided only when DB Historical Data Analyzer is run in MVS batch environment. The return codes are described in the following sections.

FABGHIST

The following list shows the return codes set by FABGHIST.

Code **Meaning**

0 The requested operation has completed successfully.

4 Warning message is issued, but the requested operation has completed.

8 Severe errors occurred. The job was ended.

FABPXEXP

The following list shows the return codes set by FABPXEXP.

Code **Meaning**

0 The requested operation has completed successfully.

2 In the FABGRECI syntax checking process, Export Utility found that some field's length is not enough for storing the field. Warning message is issued, but the syntax checking is completed.

4 Warning message is issued, but the requested operation is completed. The reason of the warning is one or more than one of the following:

- There is a minor syntax error in the HISTIN control statement or the FABGRECI statement. Export Utility ignores the error.
- Overflow occurred in one or more fields.
- Some of the flat records are not created because the specified database, IMS ID, and date entry was not found in the HISTORY data set.

8 Severe errors occurred and the job ended. The reason of the error is one of the following:

- There is a syntax error in the HISTIN control statement or the FABGRECI statement.
- No flat record was created.

Messages

This section lists the messages of DB Historical Data Analyzer as follows:

- FABG001E—FABG399E are messages in TSO/ISPF environment.
- FABG1000I—FABG8999E are messages in MVS Batch environment.

Messages of the TSO/ISPF Environment

FABG001E Invalid group number entered.

Explanation: The group number entered was not correct.

System action: The system ignores the input and waits for the next input.

Programmer response: Enter the correct input.

FABG002E Member name is required.

Explanation: The group number 1 was selected, but the member name has not been entered yet.

System action: The system waits for the next input.

Programmer response: Enter the correct member name.

FABG004E Date is required.

Explanation: The group number 3 was selected, but date has not been entered yet.

System action: The system waits for the next input.

Programmer response: Enter the correct date.

FABG005E Specified member is not found in SPMNMBR data set.

Explanation: The member entered on the “Group Selection Menu” panel was not found in the SPMNMBR data set.

System action: The system ignores the input and waits for the next input.

Programmer response: Terminate the dialog and make sure that you allocate the SPMNMBR data set correctly in your TSO command list FABGCMD0. Correct the error and restart the dialog.

FABG006E Invalid date entered.

Explanation: The date entered was not correct.

System action: The system ignores the input and waits for the next input.

Programmer response: Enter the correct input.

FABG007I No DBDS record found on History data set.

Explanation: The system searched the HISTORY data set based on the group information but no database data set record was found for the group.

System action: The system waits for the next attempt.

Programmer response: Try a next attempt.

FABG008E Only S command is valid to select a data set.

Explanation: An input on the CMD field was not S (select) command.

System action: The system waits for the next input.

Programmer response: Enter the S command.

FABG009E More than one data set were selected.

Explanation: More than one data set was selected at a time.

System action: The system waits for the next input.

Programmer response: Select only one data set.

FABG010E Invalid item number entered.

Explanation: The item number entered was not correct.

System action: The system ignores the input and waits for the next input.

Programmer response: Enter the correct input.

FABG011E Invalid item group number entered.

Explanation: The item group number entered was not correct.

System action: The system ignores the input and waits for the next input.

Programmer response: Enter the correct input.

FABG012E Member has no control statement.

Explanation: The user-specified member in SPMNMBR data set contains no effective control statement.

System action: The system ignores the input and waits for the next input.

Programmer response: Ensure that the SPMNMBR data set is specified correctly in the FABGCMD0 CLIST.

Also ensure that you specified the correct member name.

FABG013E DD name is entered but DB name is not entered.

Explanation: The user selected group 2 on the “Group Selection Menu” panel, and specified only the DD name field without specifying the DB name.

System action: The system ignores the input and waits for the next input.

Programmer response: Enter the DB name.

FABG014E One database data set must be selected.

Explanation: A database data set was not selected on one of the “Data Set Selection Menu” panels.

System action: The system waits for the next input.

Programmer response: Select one database data set to be processed.

FABG015E One or more items must be selected.

Explanation: A group that has two or more items to be processed was selected, but no items to be processed were specified.

System action: The system waits for the next input.

Programmer response: Enter S in front of items to be processed (at least one item must be specified).

FABG016E Invalid ISPF command is entered.

Explanation: The command entered was not correct.

System action: The system ignores the input and waits for the next input.

Programmer response: Enter a correct ISPF command.

FABG017E The option that was entered was not valid.

Explanation: The option entered was not correct.

System action: The system ignores the input and waits for the next input.

Programmer response: Enter a correct option.

FABG020E Selected item not supported for the database.

Explanation: The item selected is not supported for the specified database. Refer to Table 59 on page 453 for the database analysis items and supported database types.

System action: The system ignores the input and waits for the next input.

Programmer response: Enter the correct input.

FABG021E Selected item not supported for the data set group.

Explanation: The item selected is not supported for the data set group requested. Refer to Table 59 on page 453 for the database analysis items and supported database types.

System action: The system ignores the input and waits for the next input.

Programmer response: Enter a correct input.

FABG022E No space allocation information available for the data set.

Explanation: The system found that there was no space allocation information in the SPMNSPDT data set for the database data set specified.

System action: The system ignores the input and waits for the next input on the same panel.

Programmer response: Return to the previous panel (Group Selection Menu) and continue the dialog.

FABG023E More than 999 entries are selected between specified dates.

Explanation: More than 999 key date entries were selected from the range between the date specified on "From Date" field and the date specified on "To Date" field. DB Historical Data Analyzer passes a maximum of 999 key date entries and control to GDDM-ICU.

System action: The system ignores the input and waits for the next input on the same panel.

Programmer response: Reduce the range between the date specified on "From Date" field and the date specified on "To Date" field by changing the specified date.

FABG024E Specified date on "To Date" is older than "From Date".

Explanation: The specified date on "To Date" field is older than the date specified on "From Date" field.

System action: The system ignores the input and waits for the next input on the same panel.

Programmer response: Enter the correct date in which "To Date" is not older than "From Date".

FABG025E Invalid date entered on the date field.

Explanation: The date entered on "To Date" field and/or "From Date" field was not correct.

System action: The system ignores the input and waits for the next input on the same panel.

Programmer response: Enter the correct date.

FABG026E No DBDS record found between specified dates.

Explanation: The system searched the history date set based on the range between the specified dates but no key date entry of DBDS record was found.

System action: The system ignores the input and waits for the next input on the same panel.

Programmer response: Enter the appropriate date according to the dates on "1st Entry" and "Last Entry" fields on the same panel.

FABG027E No bucket found between specified dates.

Explanation: The system searched the SPMNSPDT data set based on the range between the specified dates but no key date entry of bucket was found.

System action: The system ignores the input and waits for the next input on the same panel.

Programmer response: Enter the appropriate date according to the dates on "1st Entry" and "Last Entry” fields on the same panel.

FABG100E HISTORY data set OPEN failed (RC = *xx*).

Explanation: OPEN processing failed for the HISTORY data set. Refer to *z/OS DFSMS Macro Instructions for Data Sets* for the explanation of the return code *xx*.

System action: The system ignores the input and waits for the next input on the same panel.

Programmer response: Terminate the dialog and make sure that you allocate the HISTORY data set correctly. Correct the error and restart the dialog.

FABG102E SPMNMBR data set OPEN error (RC = *xx*).

Explanation: OPEN processing failed for the SPMNMBR data set. Refer to *z/OS DFSMS Macro Instructions for Data Sets* for the explanation of the return code *xx*.

System action: The system ignores the input and waits for the next input on the same panel.

Programmer response: Terminate the dialog and make sure that you allocate the SPMNMBR data set correctly. Correct the error and restart the dialog.

FABG103E SPMNSPDT data set OPEN error (RC = *xx*).

Explanation: OPEN processing failed for the SPMNSPDT data set. Refer to *z/OS DFSMS Macro Instructions for Data Sets* for the explanation of the return code *xx.*

System action: The system ignores the input and waits for the next input on the same panel.

Programmer response: Terminate the dialog and make sure that you allocate the SPMNSPDT data set correctly. Correct the error and restart the dialog.

FABG104E POINT macro failed for HISTORY data set, RC=*xx* RPL FDBK=*aaa* (*bb*).

Explanation: An error was encountered with the VSAM POINT macro while attempting to access a record on the HISTORY data set.

The return code is shown in hexadecimal (*xx*), and the RPL FDBK code value is shown in both decimal (*aaa*) and hexadecimal (*bb*) format.

Refer to *z/OS DFSMS Macro Instructions for Data Sets* for the further explanation of the error.

System action: The system ignores the input and waits for the next input on the same panel.

Programmer response: Terminate the dialog, determine the cause of the error indicated by the VSAM status code, and correct it. Perform a VSAM VERIFY on the KSDS, and rerun the job. If this situation persists, report it to the systems operations personnel.

FABG105E GET macro failed for HISTORY data set, RC=*xx* RPL FDBK=*aaa* (*bb*).

Explanation: An error was encountered with the VSAM GET macro while attempting to access a record on the HISTORY data set.

The return code is shown in hexadecimal (*xx*), and the RPL FDBK code value is shown in both decimal (*aaa*) and hexadecimal (*bb*) format.

Refer to *z/OS DFSMS Macro Instructions for Data Sets* for the further explanation of the error.

System action: The system ignores the input and waits for the next input on the same panel.

Programmer response: Terminate the dialog, determine the cause of the error indicated by the VSAM status code, and correct it. Perform a VSAM VERIFY on the KSDS, and rerun the job. If this situation persists, report it to the systems operations personnel.

FABG106E DBDS SEQ=1 record not found for DB: *dbname* **DD:** *ddname***.**

Explanation: A history record for the database data set (*dbname*, *ddname*) that has a sequence number 1 was not found in the HISTORY data set.

System action: The system stops processing the database data set and returns to the last menu panel.

Programmer response: Terminate the dialog, reorganize the HISTORY data set by running a DB Historical Data Analyzer batch job, and rerun the dialog. If this situation persists, contact IBM Software Support.

FABG107E DBDS SEQ=2 record not found for DB: *dbname* **DD:** *ddname***.**

Explanation: A history record for the database data set (*dbname*, *ddname*) that has a sequence number 2 was not found in the HISTORY data set.

System action: The system stops processing the database data set and returns to the last menu panel.

Programmer response: Terminate the dialog, reorganize the HISTORY data set by running a DB Historical Data Analyzer batch job, and rerun the dialog. If this situation persists, contact IBM Software Support.

FABG108E Invalid history data set entry found for DB: *dbname***.**

Explanation: The system found an incorrect HISTORY data set entry while processing HISTORY data set records for the database (*dbname*). An incorrect HISTORY data set entry may be created when HD Pointer Checker run fails during processing a database data set group associated with the database.

System action: The system stops processing the database data set and returns to the last menu panel.

Programmer response: Terminate the dialog. Run DB Historical Data Analyzer, and generate the HISTORY Data Set by Key Date report and the HISTORY Data Set by DB-DS report with the PROC TYPE=LIST option.

Collect database data set and key date information of the incorrect HISTORY data set entries, prepare necessary DB Historical Data Analyzer control statements, then delete these incorrect entries by running DB Historical Data Analyzer with the PROC TYPE=DELETE option. Rerun the dialog.

FABG109E IMSID CONTROL STATEMENT IS INCORRECT

Explanation: The IMSID control statement was specified while Multiple-IMSID is disabled, or was not specified while Multiple-IMSID is enabled.

System action: The system waits for the next input.

Programmer response: Correct the IMSID control statement.

Problem determination: None.

FABG399E Unknown error occurred in *modulename* **module (RC =** *nn***).**

Explanation: The system found that an unknown error occurred in the internal module (*modulename*). This kind of error should not occur. The return code (*nn*) is the internal reason code.

System action: The dialog ends with an abend code of 3999.

Programmer response: Contact IBM Software Support.

Messages of the MVS/ Batch Environment

FABG1000I FABGHIST ENDED NORMALLY

Explanation: The job is completed successfully.

System action: The system completes the job normally with a return code of 0.

Programmer response: None.

FABG1001W FABGHIST ENDED WITH WARNINGS

Explanation: Minor error conditions were detected by the system.

System action: The system ends with a return code of 4.

Programmer response: Refer to the other message generated by the system to determine the nature and causes of the errors detected. Correct the problem and rerun the job.

FABG1002E FABGHIST ENDED WITH ERRORS

Explanation: Major error conditions were detected by the system.

System action: Program FABGHIST ends with a return code of 8.

Programmer response: Refer to the other message generated by the system to determine the nature and causes of the errors detected. Correct the problem and rerun the job.

FABG1010I REQUESTED PROCESS ENDED NORMALLY (TYPE = ***type*****)**

Explanation: The requested process (*type*) ended normally.

System action: The system sets a return code of 0 and continues processing.

Programmer response: None.

FABG1011W REQUESTED PROCESS ENDED WITH WARNINGS (TYPE = ***type*****)**

Explanation: Minor error conditions were detected during the requested process (*type*).

System action: The system sets a return code of 4 and continues processing.

Programmer response: Refer to the other message generated by the system to determine the nature and causes of the errors detected. Correct the problem and rerun the job.

FABG1012E REQUESTED PROCESS ENDED WITH ERROR (TYPE = ***type*****)**

Explanation: Major error conditions were detected during the requested process (*type*).

System action: The system sets a return code of 8 and continues processing.

Programmer response: Refer to the other message generated by the system to determine the nature and causes of the errors detected. Correct the problem and rerun the job.

FABG1101E VALID STATEMENT NOT FOUND

Explanation: No valid statement (that is, statement that is not a comment) was found in the HISTIN DD data set or the FABGRECI member.

System action: The system ends with a return code of 8.

Programmer response: Specify the necessary statements and rerun the job.

FABG1102W "DATABASE" STATEMENT IS IGNORED

Explanation: A DATABASE statement is coded following the PROC statement with TYPE=LIST, REORG, CREATE, or UPDATE. A DATABASE statement is not required when the associated PROC statement has the TYPE=LIST, REORG, CREATE, or UPDATE option.

System action: The system sets a return code of 4, ignores this DATABASE statement, and continues processing.

Programmer response: Correct the sequence of the control statements and rerun the job, if necessary.

FABG1104W "PROC TYPE=CREATE" STATEMENT IGNORED

Explanation: After processing one or more PROC statement (except TYPE=CREATE), a PROC TYPE=CREATE statement was read. A PROC TYPE=CREATE statement must be specified at first.

System action: The system sets a return code of 4, ignores this PROC statement, and continues processing.

Programmer response: Correct the sequence of the control statements and rerun the job, if necessary.

FABG1105E NEITHER "DB" PARAMETER NOR "MEMBER" PARAMETER IS SPECIFIED ON "DATABASE" STATEMENT

Explanation: On the DATABASE statement, neither a DB parameter nor MEMBER parameter was specified. A

DB parameter or MEMBER parameter must be specified on a DATABASE statement.

System action: The system ends with a return code of 8.

Programmer response: Correct the DATABASE statement and rerun the job.

FABG1106E "DD" PARAMETER IS NOT SPECIFIED ON "DATABASE" STATEMENT

Explanation: On the DATABASE statement, the DD keyword parameter was not specified.

System action: The system ends with a return code of 8.

Programmer response: Correct the DATABASE statement and rerun the job.

FABG1107E BOTH "DB" AND "MEMBER" PARAMETERS SPECIFIED ON "DATABASE" STATEMENT

Explanation: On the DATABASE statement, both the DB and MEMBER keyword parameters were specified. These two keyword parameters are mutually exclusive.

System action: The system ends with a return code of 8.

Programmer response: Correct the DATABASE statement and rerun the job.

FABG1108E INVALID DATE IS SPECIFIED FOR "FROM"/"TO" PARAMETER ON "DATABASE" STATEMENT

Explanation: On the DATABASE statement, the date specified by the FROM or TO keyword parameter was not correct.

System action: The system ends with a return code of 8.

Programmer response: Correct the DATABASE statement and rerun the job.

FABG1109E "DATABASE" STATEMENT NOT SPECIFIED FOR "PROC TYPE=DELETE" STATEMENT

Explanation: The last statement is a PROC TYPE=DELETE statement, but no DATABASE statement follows it. At least one DATABASE statement must follow a PROC statement with TYPE=DELETE option.

System action: The system ends with a return code of 8.

Programmer response: Add necessary DATABASE statements and rerun the job.

FABG1110E “DB” PARAMETER IS NOT SPECIFIED ON "DATABASE" STATEMENT

Explanation: On the DATABASE statement, DB keyword parameter was not specified.

System action: The system ends with a return code of 8.

Programmer response: Correct the DATABASE statement and rerun the job.

FABG1111W DATABASE DATA SET WAS ALREADY SPECIFIED BY PREVIOUS "DATABASE" STATEMENT

Explanation: The database data set specified on the DATABASE statement was already specified by the previous DATABASE statement.

System action: The system ignores the statement and continues processing. The return code will be 4.

Programmer response: Correct the DATABASE statement and rerun the job, if necessary.

FABG1112W DATABASE DATA SET WAS ALREADY SPECIFIED BY PREVIOUS CONTROL STATEMENT

Explanation: The database data set specified on the Space Monitor control statement in the control member data set was already specified by the previous Space Monitor control statement.

System action: The system ignores the statement and continues processing. The return code will be 4.

Programmer response: Correct the DATABASE statement and rerun the job, if necessary.

FABG1113I *80-byte Space Monitor control statement image*

Explanation: The message text shows the 80-byte Space Monitor control statement image read from the member of the SPMNMBR data set.

System action: None.

Programmer response: None.

FABG1114E "ENDPROC" STATEMENT NOT FOUND

Explanation: The system encountered EOF status of HISTIN data set, or read another PROC control statement before the ENDPROC control statement. Every PROC control statement must have a matching ENDPROC control statement.

System action: The system ends with a return code of 8.

Programmer response: Specify ENDPROC control statement where missing, and rerun the job.

FABG1115E INVALID TYPE IS SPECIFIED FOR "TYPE" PARAMETER ON "PROC" STATEMENT

Explanation: On the PROC statement, the type specified by the TYPE keyword parameter is not correct.

System action: The system ends with a return code of 8.

Programmer response: Correct the PROC statement and rerun the job.

FABG1116E DBNAME FIELD (COLUMN 1-8) IS INVALID

Explanation: The system found that the DB name specified in columns 1-8 of the control statement was not correct.

System action: The system ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABG1117E DDNAME FIELD (COLUMN 10-17) IS INVALID

Explanation: The system found that the DD name specified in columns 10-17 of the control statement was not correct.

System action: The system ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABG1118E MEMBER HAS NO VALID CONTROL STATEMENT WHICH SPECIFIES DBNAME AND DDNAME

Explanation: The member specified by DATABASE statement has no valid control statement that specifies a database name and ddname.

System action: The system ends with a return code of 8.

Programmer response: Specify the correct member name of the control member data set (SPMNMBR) that contains valid control statements, and rerun the job.

FABG1119E INTERNAL WORK SPACE FOR CONTROL STATEMENTS FULL

Explanation: The system found that the internal work space for stacking DATABASE control statements and the Space Monitor control statements in the member of SPMNMBR data set was full. Total number of valid DATABASE control statements and valid Space Monitor control statements that the system can process per one PROC control statement is 1000.

System action: The system ends with a return code of 8.

Programmer response: Do not exceed a total of 1000 DATABASE control statements and/or Space Monitor control statements (specified by the DATABASE control statement with MEMBER= parameter). Rerun the job.

FABG1120E INVALID COMBINATION IS SPECIFIED FOR "FROM=" AND "TO=" PARAMETERS

Explanation: The date specified on the FROM= parameter must be older than the date specified on the TO= parameters of the DATABASE control statement.

System action: The system ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABG1121W DBDS RECORD OF DB: ***dbname*** **DD:** ***ddname*** **NOT FOUND FOR SPECIFIED FROM=*mmddyy*, TO=*mmddyy***

Explanation: The DBDS record of indicated database data set was not found in the range between the specified dates on FROM= and TO= parameters of the DATABASE control statement.

System action: The system sets a return code of 4 and continues processing.

Programmer response: Correct the control statement and rerun the job, if necessary.

FABG1122W ″*statement*″ **STATEMENT IS IGNORED**

Explanation: The specified control statement ″*statement*″ cannot be specified with the preceding PROC control statement.

System action: The system sets a return code of 4, ignores this statement, and continues processing.

Programmer response: Correct the sequence of the control statements and, if necessary, rerun the job.

FABG1123E ″*statement*″ **STATEMENT NOT SPECIFIED FOR ″PROC TYPE=*function*″ STATEMENT**

Explanation: ″*statement*″ control statement is required, because ″PROC TYPE=*function*″ is specified previously. But, it is not specified.

System action: The system ends with a return code of 8.

Programmer response: Add the necessary DATABASE statements and rerun the job.

FABG1124E *″parameter″* **PARAMETER IS INCORRECT**

Explanation: Incorrect operand is specified for the *″parameter″* parameter, or the specified *″parameter″* parameter is incorrect for the record type that is specified in the TYPE= parameter.

System action: The system ends with a return code of 8.

Programmer response: Specify the correct parameter or operand, and rerun the job

FABG1125E *″statement″* **STATEMENT DUPLICATED**

Explanation: More than one *″statement″* control statements are specified. Only one *″statement″* is allowed.

System action: The system ends with a return code of 8.

Programmer response: Correct the statement and rerun the job.

FABG1126E *″statement″* **STATEMENT IS INCORRECT**

Explanation: The *″statement″* statement is missing, specified incorrectly, or a required parameter is not specified for the *″statement″* control statement.

System action: The system ends with a return code of 8.

Programmer response: Correct the statement, parameters, or both, and rerun the job.

FABG1127E THE NUMBER OF STATEMENTS SPECIFIED IS NOT CORRECT

Explanation: The number of the statements specified in a FABGRECI member is either 0 or is over 255.

System action: The system ends with a return code of 8.

Programmer response: Correct the statements.

Problem determination: Check the flat record definitions in the FABGRECI member.

FABG1128W DBDS RECORD OF DB: *dbname* **DD:** *ddname* **NOT FOUND FOR VALUE SPECIFIED BY KEEP=***nnn*

Explanation: The DBDS record of the specified database data set was not found of which retention period exceeds value specified by the KEEP= parameter of the DATABASE control statement.

System action: The system sets a return code of 4 and continues processing.

Programmer response: Correct the control statement and, if necessary, rerun the job.

FABG1129W *″XXXXXXXXXXXX″* **PARAMETER IS IGNORED**

Explanation: Parameter shown in the message cannot be specified.

System action: System ignores the specified parameter.

Programmer response: None.

Problem determination: None.

FABG1130E DBDS RECORD NOT FOUND ON HISTORY DATA SET FOR DBDNAME: *dbname* **(DDNAME:** *ddname***)**

Explanation: On the HISTORY data set, there is no database data set record for the database *dbname*, or the database data set specified by the *dbname* and *ddname*.

System action: The system ignores processing the database or the database data set and continues processing. The return code will be 8.

Programmer response: Run HD Pointer Checker for the database data set at least once, specifying HISTORY=YES on the OPTION statement, and rerun the job.

FABG1131E DATA SET NOT EMPTY FOR DDNAME: HISTORY

Explanation: An attempt was made to format a HISTORY data set which was not empty.

System action: The system ends with a return code of 8.

Programmer response: Use the appropriate VSAM options to scratch, delete, allocate, and define the HISTORY data set, and rerun the job.

FABG1132E NO DBDS RECORD ON HISTORY DATA SET

Explanation: There is no database data set record in the HISTORY data set.

System action: The system ignores the processing for the current request type and continues processing.

Programmer response: Ensure that the HISTORY DD statement specifies the correct data set. Correct any errors and rerun the job, if necessary.

FABG1133E HISTORY DATA SET CONTROL RECORD NOT FOUND

Explanation: There is no control record in the HISTORY data set.

System action: The system ignores the current request type and continues processing. The return code will be 8.

Programmer response: Make sure that the HISTORY DD statement specifies the correct data set. Correct any errors and, if necessary, and rerun the job.

FABG1134E DBDS RECORD NOT FOUND ON HISTORY DATA SET FOR IMSID: ***imsid***

Explanation: There is no record in the HISTORY data set that matches the specified IMSID.

System action: Processing stops.

Programmer response: Specify any of the IMSIDs of records that are in the HISTORY data set.

Problem determination: None.

FABG1135E DUPLICATE OPERANDS FOR ***″XXXXXXXX″*** **PARAMETER**

Explanation: You cannot specify duplicate operands for the parameter shown in the message.

System action: Processing stops.

Programmer response: Remove the duplicate operands.

Problem determination: None.

FABG1136E IMSID PARAMETER CANNOT BE SPECIFIED WITH HISTORY DATA SET OF MULTIIMSID=NO

Explanation: You cannot use the IMSID= parameter while the Multiple-IMSID option is disabled.

System action: Processing stops.

Programmer response: Enable the Multiple-IMSID option when you use the IMSID= parameter.

Problem determination: None.

FABG1140E MEMBER: ***membername*** **NOT FOUND ON SPMNMBR DATA SET**

Explanation: The member (*membername*) specified by the EXEC parameter was not found in the SPMNMBR data set.

System action: The system ends with a return code of 8.

Programmer response: Ensure that the SPMNMBR DD statement specifies the correct data set. Correct any errors and rerun the job.

FABG1150W NO RECORD IS EXPORTED FOR DB: ***dbdname*** **IMSID:** ***imsid*** **FROM:** ***mmddyyyy*** **TO:** ***mmddyyyy*** **DAYS:** ***nnn***

Explanation: Flat record cannot be generated, because there is no history record entry for the database *dbdname* to meet the conditions specified in the HISTIN control statements and the FABGRECI members.

System action: Completes the process, and ends with a return code of 4.

Programmer response: Correct the control statements or the flat record definition, and, if necessary, rerun the job.

Problem determination: Check the HISTIN control statements and the flat record definitions in the FABGRECI members.

FABG1151E MEMBER: ***member*** **NOT FOUND**

Explanation: *member* specified in the MEMBER= parameter in the HISTIN data set cannot be found in the FABGRECI data set.

System action: The system ends with a return code of 8.

Programmer response: Correct the HISTIN control statement and rerun the job.

Problem determination: Check the HISTIN control statement and the FABGRECI data set.

FABG1152E HISTORY DATA SET IS NOT EXPORTABLE=YES OPTION

Explanation: The attribute of the HISTORY data set is EXPORTABLE=NO.

System action: The system ends with a return code of 8.

Programmer response: Specify a correct data set name for the HISTORY DD statement, or change the EXPORTABLE attribute to YES by the FABGHIST program PROC TYPE=UPDATE and OPTION EXPORTABLE=YES.

Problem determination: You may have specified an incorrect data set on your HISTORY DD statement. It is also possible that you specified a data set on your HISTORY DD statement of which attribute is EXPORTABLE=NO. To check the attribute of the HISTORY data set, run the FABGHIST program with PROC TYPE=ATTRLIST and see the History Attribute report.

FABG1153E ***parm01*** **AND** ***parm02*** **ARE MUTUALLY EXCLUSIVE**

Explanation: You cannot specify *parm01* and *parm02* together.

System action: The system ends with a return code of 8.

Programmer response: Correct the control statements in HISTIN or the flat record definitions in the FABGRECI member, and rerun the job.

Problem determination: Check the control statements in HISTIN or the flat record definitions in the FABGRECI member.

FABG2020E ***parameter*****:** ***operand*** **IS UNAVAILABLE**

Explanation: The *operand* specified for the *parameter* is incorrect.

System action: The system ends with a return code of 8.

Programmer response: Correct the flat record definitions in the FABGRECI member by referring to Table 45 on page 421 through Table 55 on page 438.

Problem determination: Check the flat record definitions in the FABGRECI member.

FABG2021E FIELD: ***fieldname*** **IS UNSUITABLE FOR TYPE:** ***record type***

Explanation: The *field name* cannot be specified for FIELD, when TYPE is *record type*.

System action: The system ends with a return code of 8.

Programmer response: Correct the flat record definitions in the FABGRECI member by referring to Table 45 on page 421 through Table 55 on page 438.

Problem determination: Check the flat record definitions in the FABGRECI member.

FABG2024E THE LENGTH OF FLAT RECORD IS TOO LONG

Explanation: The result of analyzing the flat record definitions in the FABGRECI members shows that the maximum length of flat record is greater than 32752.

System action: The system ends with a return code of 8.

Programmer response: Correct the flat record definitions in the FABGRECI member by referring to Table 45 on page 421 through Table 55 on page 438.

Problem determination: Check the flat record definitions in the FABGRECI member.

FABG2026W LEN=*XXX* SPECIFIED HAS NOT ENOUGH LENGTH FOR FIELD: ***XXXX***

Explanation: LEN=*xxx* has not enough length for field *xxxx*. It is possible to cause an overflow in the field. It is recommended that you increase the length of this field.

System action: Completes processing, and ends with a return code of 2.

Programmer response: Correct the flat record definitions in the FABGRECI member by referring to Table 45 on page 421 through Table 55 on page 438.

Problem determination: Check the flat record definitions in the FABGRECI member.

FABG2027W DB: ***zzzzzzzz*** **MEMBER:** ***yyyyyyyy*** **FIELD:** ***xxxxxxxxxxxxxxxx*** **CAUSED AN OVERFLOW**

Explanation: Field *xxxx* that was specified caused an overflow. The flat record was created, but some data was truncated.

System action: Completes processing, and ends with a return code of 4.

Programmer response: By referring to Table 45 on page 421 through Table 55 on page 438, specify for LEN= a number with enough digits to store the field.

Problem determination: The LEN= specification is too small for the overflowed field. Check the LEN= in the FIELD statement in the FABGRECI member.

FABG2028E NO RECORD IS EXPORTED

Explanation: No flat record is exported, because there is no history record entry to meet the conditions specified in the HISTIN control statements and the FABGRECI members.

System action: The system ends with a return code of 8.

Programmer response: Correct the control statements or the flat record definitions, and if necessary, rerun the job.

Problem determination: Check the HISTIN control statements and the flat record definitions in the FABGRECI members.

FABG3501E "OPEN" FAILED FOR DDNAME ***ddname*** **(RC =** ***nn*****)**

Explanation: OPEN processing failed for the data set associated with the DD statement (*ddname*). Refer to *z/OS DFSMS Macro Instructions for Data Sets* for the explanation of the return code (*nn*). If the message that the IEC130I DD statement is missing message is issued before this message, the required DD statement is not specified in the JCL.

System action: The system ends with an abend code of 3501.

Programmer response: Ensure that a DD statement is present for the ddname indicated, and that it is correctly specified. Correct any errors and rerun the job.

FABG3502E "CLOSE" FAILED FOR DDNAME *ddname*

Explanation: CLOSE processing failed for the data set associated with the DD statement (*ddname*).

System action: The system ends with an abend code of 3502.

Programmer response: Rerun the job. If the data set indicated is a VSAM data set, perform a VSAM VERIFY on it, and rerun the job. If this situation persists, report it to systems operations personnel.

FABG3503E "FIND" FAILED FOR DDNAME SPMNMBR MEMBER: *membername* **(RC =** *nn***)**

Explanation: The system issued an OS FIND macro to find the member (*membername*) on the data set specified by the SPMNMBR DD statement. The return code (*nn*) indicated that the attempt was unsuccessful. Refer to *z/OS DFSMS Macro Instructions for Data Sets* for the explanation of the return code.

System action: The system ends with an abend code of 3503.

Programmer response: Ensure that the SPMNMBR DD statement specifies the correct data set. Correct any errors and rerun the job. If this situation persists, report it to the systems operations personnel.

FABG3504E VSAM "MODCB"/"SHOWCB" ERROR FOR HISTORY DATA SET - REG 15: *xx* **REG 0:** *yy*

Explanation: The system issued a MODCB or SHOWCB macro to modify or get information on the HISTORY data set. The return code indicates that the attempt was unsuccessful. The values returned in register 15 (*xx*) and 0 (*yy*) are shown in hexadecimal format. Refer to *z/OS DFSMS Macro Instructions for Data Sets* for further explanation.

System action: The system ends with an abend code of 3504.

Programmer response: Ensure that the HISTORY DD statement properly specifies the correct data set. Correct any errors and rerun the job. If this situation persists, report it to the systems operations personnel.

FABG3505E VSAM I/O ERROR ACCESSING HISTORY DATA SET FOR "*yyyyy***" VSAM ERROR DATA: RETURN CODE:** *xx* **RPL "FDBK":** *aaa* **(***bb***)**

Explanation: The system encountered an error while attempting to access a record in the HISTORY data set. *yyyyy* is either POINT, GET, or PUT.

The return code is shown in hexadecimal (*xx*), and the RPL FDBK code value is shown in both decimal (*aaa*) and hexadecimal (*bb*) format.

System action: The system ends with an abend code of 3505.

Programmer response: Refer to *z/OS DFSMS Macro Instructions for Data Sets* for the explanation of the error. Determine the cause of the error indicated by the VSAM status code, and correct it. Perform a VSAM VERIFY on the KSDS, and rerun the job. If this situation persists, report it to the systems operations personnel.

FABG3506E INVALID HISTORY DATA SET ENTRY WAS FOUND FOR DBNAME: *dbname*

Explanation: The system found an incorrect HISTORY data set entry while processing HISTORY data set records for the database (*dbname*). An incorrect HISTORY data set entry may be created when HD Pointer Checker run fails during processing a database data set group associated with the database.

System action: The system ends with an abend code of 3506.

Programmer response: Run DB Historical Data Analyzer with the TYPE=LIST option, and generate the HISTORY Data Set by Key Date report and HISTORY Data Set by DB-DS report.

Collect database data sets and key date information of the incorrect HISTORY data set entries, prepare necessary DB Historical Data Analyzer control statements, then delete these incorrect entries by running DB Historical Data Analyzer with the TYPE=DELETE option. Rerun the job.

FABG3507E DD DUMMY WAS SPECIFIED FOR DDNAME *ddname*

Explanation: The system found that DUMMY was specified for the data set associated with the DD statement (*ddname*).

System action: The system ends with an abend code of 3507.

Programmer response: Specify a correct data set name for the ddname. Rerun the job.

FABG3509E GETMAIN FAILED. SIZE: ***nnnn*** **K ITEM-ID:** ***xxx***

Explanation: GETMAIN failed.

System action: DB Historical Data Analyzer ends with an abend code of 3509.

Programmer response: Specify more region size for your Export Utility job.

Problem determination: The region size might be insufficient. Specify more region size for your Export Utility job.

FABG3510E HISTORY DATA SET CONTROL RECORD NOT FOUND DURING REORGANIZATION

Explanation: No HISTORY data set control record was found in the HISTORY data set during its reorganization.

System action: The system ends with an abend code of 3510.

Programmer response: Ensure that the HISTORY DD statement properly specifies the correct data set. Correct any errors. Rerun the job.

FABG3511E HISTORY DATA SET RUN TIME RECORD NOT FOUND DURING REORGANIZATION

Explanation: During the reorganization of the HISTORY data set, the system detected a HISTORY data set control record that has one or more run date entries, but HISTORY data set run time record was not found in the data set.

System action: The system ends with an abend code of 3511.

Programmer response: Ensure that the HISTORY DD statement properly specifies the correct data set. Correct any errors. Rerun the job.

FABG3512E NUMBER OF DBDS RECORDS (SEQ=1) UNMATCHED WITH CONTROL RECORD DATA

Explanation: The number of database data set records with sequence number 1 does not match the number that the HISTORY data set control record contains.

System action: The system ends with an abend code of 3512.

Programmer response: Ensure that the HISTORY DD statement properly specifies the correct data set. Correct any errors. Rerun the job.

FABG3513E VSAM I/O ERROR ACCESSING DDNAME ddname FOR ***yyyyy*** **VSAM ERROR DATA: RETURN CODE:** ***xx*** **RPL "FDBK":** ***aaa*** **(*****bb*****);** ***zzzzzzzz***

Explanation: The system encountered an error while attempting to access a record in the data set (ddname). *yyyyy* is either POINT, GET, or PUT. *zzzzzzzz* is either CUTLERY, DBDSTBL, DBDSREC, or RUNTIME. The return code is shown in hexadecimal (*xx*), and the RPL FDBK code value is shown in both decimal (*aaa*) and hexadecimal (*bb*) format.

System action: The system ends with an abend code of 3513.

Programmer response: Refer to DFSMS/MVS Macro Instructions for Data Sets for the explanation of the error. Determine the cause of the error indicated by the VSAM status code, and correct it. Perform a VSAM VERIFY on the KSDS, and rerun the job. If this situation persists, report it to the system operations personnel.

FABG3514E DDNAME: ***ddname*** **NOT FOUND**

Explanation: DD name *ddname* is not specified.

System action: DB Historical Data Analyzer ends with an abend code of 3514.

Programmer response: Correct any errors and rerun the job.

Problem determination: Check the DD statements of your JCL.

FABG3515E RDJFCB FAILED FOR DDNAME: ***ddname*** **RC =** ***xxx***

Explanation: RDJFCB was attempted for DD name *ddname*, but failed.

System action: DB Historical Data Analyzer ends with an abend code of 3515.

Programmer response: Correct any errors and rerun the job.

Problem determination: Search the return code of RDJFCB of LOOKAT web site, and use the standard debugging methods. You can access IBM message explanations directly from the LookAt Web site:

http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/

FABG3999E UNKNOWN ERROR OCCURRED IN ***modulename*** **MODULE (RC =** ***nn*****)**

Explanation: An unknown error occurred in the system internal module (*modulename*). This kind of error should not occur. Return code (*nn*) indicates the internal status code.

System action: The system ends with an abend code of 3999.

Programmer response: Contact IBM Software Support.

FABG8001E STATEMENT FORMAT ERROR

Explanation: An error was detected in the specifications of statements.

System action: The system ends with a return code of 8.

Programmer response: Correct the error. Rerun the job.

FABG8002E ***parm-name*** **PARAMETER IS INVALID FOR STATEMENT NAME**

Explanation: An incorrect word *parm-name* was specified for the statement in the HISTIN data set or the FABGRECI member.

System action: The system ends with a return code of 8.

Programmer response: Correct the error. Rerun the job.

FABG8003E ***parm-name*** **PARAMETER IS INVALID FOR** ***stmt-name*** **STATEMENT**

Explanation: An incorrect parameter (*parm-name*) was specified for the statement (*stmt-name*).

System action: The system ends with a return code of 8.

Programmer response: Correct the error. Rerun the job.

FABG8004E NUMBER OF ***parm-name*** **PARAMETERS EXCEEDED THE LIMIT, MAX IS** ***nn***

Explanation: Too many parameters (*parm-name*) were specified. The parameter in error is shown on the message. The maximum number of specifications for the parameter is *nn*.

System action: The system ends with a return code of 8.

Programmer response: Correct the error and rerun the job.

FABG8005E ***parm-name*** **PARAMETER IS REQUIRED FOR** ***stmt-name*** **STATEMENT**

Explanation: A required parameter (*parm-name*) is missing for the control statement (*stmt-name*).

System action: The system ends with a return code of 8.

Programmer response: Specify the required parameter for the control statement, and rerun the job.

FABG8006E NUMBER OF OPERAND FOR ***parm-name*** **PARAMETER EXCEED THE LIMIT, MAX IS** ***nn***

Explanation: Too many parameter values were specified for the parameter (*parm-name*). The maximum number of parameter values is *nn*.

System action: The system ends with a return code of 8.

Programmer response: Correct the error. Rerun the job.

FABG8007E LENGTH ERROR IN ***n-th*** **OPERAND OF PARAMETER:** ***parm-name***

Explanation: The length of the *n-th* parameter value for the parameter (*parm-name*) was not correct.

System action: The system ends with a return code of 8.

Programmer response: Correct the error and rerun the job.

FABG8008E ***n-th*** **OPERAND IS REQUIRED FOR PARAMETER:** ***parm-name***

Explanation: The *n-th* operand was not specified for the parameter (*parm-name*).

System action: The system ends with a return code of 8.

Programmer response: Specified the missing parameter value. Rerun the job.

FABG8999E UNKNOWN ERROR OCCURRED IN ***module-name*** **MODULE (RC =** ***nn*****)**

Explanation: The system found that an unknown error occurred in the internal module (*module-name*). This kind of error should not occur. The return code (*nn*) is the internal status code.

System action: The system ends with an abend code of 3999.

Programmer response: Contact IBM Software Support.

Space Monitor

This section describes the abend codes, return codes, and messages issued by Space Monitor.

Abend codes

Every 3*nnn* abend code is accompanies by a FABK3*nnn*E messages. (3nnn is a four-digit identification number of the abend code and message.) See the associated FABK3*nnn*E message description for the 3*nnn* abend code.

Return codes

This section describes the return codes that are set by each module of Space Monitor.

FABKSPMN

The following list shows the return codes set by FABKSPMN.

Code	Meaning
0	The requested operation has completed successfully.
4	Warning message was written in the SPMNMSG data set, but the requested operation has completed. Or, Warning message was written on the Space Monitor Exception report.
8	Severe errors; job terminated. Warning message was written in the SPMNMSG data set. Or, Error message was written on the Space Analysis by Data Set report.

FABKTGEN

The following list shows the return codes that are set by FABKTGEN.

Code	Meaning
0	Successfully completed. A report or the default table source is generated.
8	Unsuccessfully completed; control statement errors or other errors were detected. For the details, see the error messages.

Messages

This section explains the messages that are issued by Space Monitor.

FABK0001I FABKSPMN ENDED NORMALLY

Explanation: This message is generated when FABKSPMN has been completed successfully.

System action: Program FABKSPMN completes the job normally with a return code of zero.

Programmer response: None.

FABK0002W FABKSPMN ENDED WITH WARNINGS

Explanation: Program FABKSPMN encountered minor error conditions, or, one or more threshold warning messages were generated in the Space Monitor Exception report.

System action: FABKSPMN completes the job with a return code of 4.

Programmer response: Refer to the other message generated by FABKSPMN to determine the nature and causes of the errors detected. Correct the problem and rerun the job, if necessary.

FABK0003E FABKSPMN ENDED WITH ERRORS

Explanation: Program FABKSPMN encountered major error conditions, or, one or more error messages were generated in the Space Analysis by Data Set report.

System action: FABKSPMN completes the job with a return code of 8.

Programmer response: Refer to the other message generated by FABKSPMN to determine the nature and causes of the errors detected. Correct the problem and rerun the job.

FABK0005E NO CONTROL STATEMENT FOUND IN SPMNMBR/SPMNIN DATA SET

Explanation: Program FABKSPMN found that there was no valid control statement in the member of the SPMNMBR data set or in the SPMNIN data set. Valid control statement here means a control statement that is not a comment.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Ensure that the member you specified by the EXEC parameter contains correct control statements, or that you specified the correct data set on the SPMNMBR DD statement, or that correct control statements are specified in the SPMNIN data set. Correct the error and rerun the job.

FABK0006E WARNING THRESHOLD VALUE FOR NUMBER OF EXTENTS (COLUMN 64-65) IS INCORRECT

Explanation: Program FABKSPMN found that the warning threshold value for the number of extents specified in the columns 64-65 of the control statement was not correct.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0007E WARNING THRESHOLD VALUE FOR % FREE SPACE (COLUMN 66-68) IS INCORRECT

Explanation: Program FABKSPMN found that the warning threshold value for the percentage of free space specified in the columns 66-68 of the control statement was not correct.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0008E WARNING THRESHOLD VALUE FOR DAYS WITHOUT HDPC RUN (COLUMN 70-71) IS INCORRECT

Explanation: Program FABKSPMN found that the warning threshold value for the days without HD Pointer Checker run specified in the columns 70-71 of the control statement was not correct.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0009E THE DATA SET *dsname* IS NOT AVAILABLE

Explanation: Program FABKSPMN cannot process the data set (identified by *dsname*).

System action: FABKSPMN ends with a return code of 8.

Programmer response: See the error message written on the Space Analysis by Data Set report, correct the error and rerun the job.

FABK0010W HISTORY RECORD NOT FOUND FOR DBNAME: *dbname* DDNAME: *ddname*

Explanation: In the HISTORY data set specified by the HISTORY DD statement, program FABKSPMN found no history record for the database data set (identified by the *dbname* and *ddname*), specified by the control statement. Or, the HISTORY DD statement was not specified.

System action: FABKSPMN treats the database data set as an OS data set; that is, it does not collect the IMS information for it, and continues processing. The return code will be 4.

Programmer response: None.

FABK0011E DBNAME FIELD (COLUMN 1 - 8) IS INCORRECT

Explanation: Program FABKSPMN found that the database name specified in columns 1-8 of the control statement was not correct.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0012E DDNAME FIELD (COLUMN 10 - 17) IS INCORRECT

Explanation: Program FABKSPMN found that the ddname specified in columns 10-17 of the control statement was not correct.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0013E DSNAME FIELD (COLUMN 19 - 62) IS INCORRECT

Explanation: Program FABKSPMN found that the data set name specified in columns of the control statement was not correct.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0014E DUPLICATE CONTROL STATEMENTS WERE FOUND FOR DBNAME ***dbname*** **DDNAME** ***ddname***

Explanation: Program FABKSPMN found that more than one control statement specified the same database data sets *dbname* and *ddname*.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0015E DUPLICATE CONTROL STATEMENTS WERE FOUND FOR DSNAME ***dsname***

Explanation: Program FABKSPMN found that more than one control statement specified the same data set (*dsname*).

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0016E WARNING THRESHOLD VALUES ARE SPECIFIED WITHOUT DBNAME

Explanation: Program FABKSPMN found that the secondary control statement is specified but no first control statement was specified.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0017E WARNING THRESHOLD VALUE FOR NUMBER OF AVAILABLE EXTENTS (COLUMN 10 - 11) IS INCORRECT

Explanation: Program FABKSPMN found that the warning threshold value for the number of available extents specified in columns 10-11 of the second control statement was not correct.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0018E WARNING THRESHOLD VALUE FOR % CA SPLIT (COLUMN 21 - 23) IS INCORRECT

Explanation: Program FABKSPMN found that the warning threshold value for the percentage of CA split specified in columns 21-23 of the second control statement was not correct.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0019E WARNING THRESHOLD VALUE FOR % CI SPLIT (COLUMN 25 - 27) IS INCORRECT

Explanation: Program FABKSPMN found that the warning threshold value for the percentage of CI split specified in columns 25-27 of the second control statement was not correct.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0020E LAST EXTENT FIELD (COLUMN 13) IS INCORRECT

Explanation: Program FABKSPMN found that the Last extent field specified in column 13 of the second control statement was not correct.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0021E WARNING THRESHOLD VALUE FOR % USED SPACE (COLUMN 15 - 17) IS INCORRECT

Explanation: Program FABKSPMN found that the warning threshold value for the percentage of used space specified in columns 15-17 of the second control statement was not correct.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0022E WARNING THRESHOLD VALUE FOR DAYS SINCE LAST REORGANIZATION (COLUMN 29 - 31) IS INCORRECT

Explanation: Program FABKSPMN found that the warning threshold value for the days since the last reorganization, specified in columns 29-31 of the second control statement, was not correct.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0023E LAST SPACE FIELD (COLUMN 19) IS INCORRECT

Explanation: Program FABKSPMN found that the last space field specified in column 19 of the second control statement was not correct.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0024E WARNING THRESHOLD VALUE FOR % DATASET SIZE (COLUMN 33-35) IS INCORRECT

Explanation: Program FABKSPMN found that the warning threshold value for the percentage of data set size that is specified in columns 33-35 of the second control statement was not correct.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0025E WARNING THRESHOLD VALUE FOR DATASET SIZE (COLUMN 37-40) IS INCORRECT

Explanation: Program FABKSPMN found that the warning threshold value for the data set size that is specified in columns 37-40 of the second control statement was not correct.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0030E TSO USER ID (COLUMN *xx*-*xx*) IS INCORRECT

Explanation: FABKSPMN found that the TSO user ID specified in columns *xx*-*xx* of the USER ID control statement was not correct.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0031E MORE THAN 20 TSO USER IDS ARE SPECIFIED

Explanation: FABKSPMN found that there are more than 20 TSO user IDs specified.

System action: FABKSPMN ends with a return code of 8.

Programmer response: Correct the control statement and rerun the job.

FABK0032E IMSID IS NOT SPECIFIED WITH HISTORY DATA SET OF MULTIIMSID=YES

Explanation: You should specify an IMSID when the Multiple-IMSID option is enabled for the HISTORY data set.

System action: Processing stops.

Programmer response: Specify an IMSID of database data sets when the Multiple-IMSID option is enabled for the HISTORY data set.

Problem determination: None.

FABK0033E IMSID CANNOT BE SPECIFIED WITH HISTORY DATA SET OF MULTIIMSID=NO

Explanation: You cannot specify an IMSID when Multiple-IMSID is disabled for the HISTORY data set.

System action: Processing stops.

Programmer response: Do not specify an IMSID when Multiple-IMSID is disabled for the HISTORY data set.

Problem determination: None.

FABK0034E IMSID (COLUMN *XX*-*XX*) IS INCORRECT

Explanation: The position of the IMSID control statement is incorrect.

System action: Processing stops.

Programmer response: Specify the IMSID control statement in the correct position.

Problem determination: None.

FABK0040I EXCEPTION NOTIFICATIONS WERE SENT TO TSO USERS: *user001, user002, user003, user004, user005,*
user006, user007, user008, user009, user010,
user011, user012, user013, user014, user015,
user016, user017, user018, user019, user020

Explanation: FABKSPMN sent exception notifications to TSO users successfully.

System action: None.

Programmer response: None.

FABK0041I EXCEPTION NOTIFICATIONS WERE CANCELED, BECAUSE USERS WERE NOT LOGGED ON OR TERMINALS DISCONNECTED.

Explanation: FABKSPMN attempted to send exception notification messages to TSO USER IDs specified in the %USER control statement, however, the TSO user IDs were not logged on or the terminal sessions were disconnected. The messages were discarded.

System action: None.

Programmer response: None.

FABK0042I *mm/dd/yyyy hh:mm:ss* **THRESHOLD EXCEPTION DB:** *dbname__* **DD:** *ddname__* **JOBNAME:** *jobname_*

Explanation: FABKSPMN detected one or more threshold exceptions for the database data set (shown by the *dbname* and *ddname*) in the Space Monitor job (*jobname*).

System action: FABKSPMN sent this message to TSO user IDs.

Programmer response: None.

FABK0043I *mm/dd/yyyy hh:mm:ss* **THRESHOLD EXCEPTION DS:** *dsname_________________________* _____________ **JOBNAME:** *jobname_*

Explanation: FABKSPMN detected one or more threshold exceptions for the data set shown by *dsname* on the Space Monitor job *jobname*.

System action: None.

Programmer response: None.

FABK0044I *mm/dd/yyyy hh:mm:ss* **THE REST OF THE MESSAGES ARE SUPPRESSED JOBNAME:** *jobname*

Explanation: The number of exception notification messages (FABK0042I or FABK0043I) reached the limit of 50. The rest of the messages are not sent to TSO users.

System action: Program FABKSPMN sent this message to TSO user IDs.

Programmer response: None.

FABK0051I FABKTGEN ENDED NORMALLY

Explanation: This message is generated when FABKTGEN is completed successfully.

System action: FABKTGEN completes the job normally with a return code of zero.

Programmer response: None.

FABK0053E FABKTGEN ENDED WITH ERRORS

Explanation: FABKTGEN encountered major error conditions.

System action: FABKTGEN completes the job with a return code of 8.

Programmer response: See the other message generated by FABKTGEN to determine the nature and causes of the errors detected. Correct the problem and rerun the job.

FABK0055I SITE DEFAULT TABLE SOURCE CODE IS GENERATED

Explanation: FABKTGEN generates the site default table source code successfully.

System action: FABKTGEN completes the job normally with a return code of zero.

Programmer response: None.

FABK0057I SITE DEFAULT TABLE FABKCTL0 IS PRINTED

Explanation: FABKTGEN prints the site default values that stored in the site default table module FABKCTL0.

System action: FABKTGEN completes the job normally with a return code of zero.

Programmer response: None.

FABK0059I SITE DEFAULT TABLE FABKCTL0 IS USED

Explanation: FABKSPMN uses the site default table FABKCTL0.

System action: FABKSPMN uses the site default values that stored in module FABKCTL0.

Programmer response: None.

FABK0061E SITE DEFAULT TABLE FABKCTL0 IS NOT FOUND

Explanation: The site default table module FABKCTL0 was not found in the data set concatenated to the STEPLIB DD.

System action: FABKTGEN ends with a return code of 8.

Programmer response: Make sure that the Site Default Table FABKCTL0 was in the STEPLIB DD. Correct any errors and rerun the job.

FABK0063E SITE DEFAULT TABLE FABKCTL0 IS CORRUPTED

Explanation: The site default table module FABKCTL0 is corrupted.

System action: FABKTGEN or FABKSPMN ends with a return code of 8.

Programmer response: Make sure that the Site Default Table FABKCTL0 was link-edited correctly in the STEPLIB DD. Correct any errors and rerun the job.

FABK0065E *parm-values* **IS INCORRECT FOR THE PARM PARAMETER OF THE EXEC STATEMENT**

Explanation: FABKTGEN found that the value for the process type specified in the *parm-values* parameter of the EXEC statement was not correct.

System action: FABKTGEN ends with a return code of 8.

Programmer response: Specify PARM='GEN' or PARM='REPORT' to the EXEC statement and rerun the job.

FABK0071E ″OPEN″ FAILED FOR DDNAME *ddname__*

Explanation: The OPEN processing failed for the data set associated with *ddname*.

System action: FABKTGEN ends with a return code of 8.

Programmer response: Make sure that a DD statement is present for the *ddname*, and that it is correctly specified. Correct any errors and rerun the job.

FABK0073E ″LOAD″ FAILED FOR DDNAME *ddname__* **MODULE** *module__*

Explanation: After issuing a LOAD macro to load module *module* from the library specified by *ddname* DD statement, register 15 contained a nonzero return code.

System action: FABKTGEN or FABKSPMN ends with a return code of 8.

Programmer response: Make sure that the module was link-edited correctly in the library. Correct any errors and rerun the job.

FABK3501E "OPEN" FAILED FOR DDNAME *ddname* **(RC=***nn***)**

Explanation: OPEN processing failed for the data set associated with the *ddname*. Refer to *z/OS DFSMS Macro Instructions for Data Sets* for the explanation of the return code (*nn*).

System action: Program FABKSPMN ends with an abend code of 3501.

Programmer response: Ensure that a DD statement is present for the ddname, and that it is correctly specified. Correct any errors. Rerun the job.

FABK3502E "CLOSE" FAILED FOR DDNAME *ddname*

Explanation: CLOSE processing failed for the data set associated with the *ddname*.

System action: Program FABKSPMN ends with an abend code of 3502.

Programmer response: Rerun the job. If the data set indicated is the HISTORY data set, perform an VSAM VERIFY on it, and rerun the job. If this situation persists, report it to systems operations personnel.

FABK3503E "FIND" FAILED FOR DDNAME SPMNMBR MEMBER: *membername* **(RC =** *nn***)**

Explanation: Program FABKSPMN issued an OS FIND macro to find the member (*membername*) on the data set specified in the SPMNMBR DD statement. The return code *nn* indicates that the attempt was unsuccessful. Refer to *z/OS DFSMS Macro Instructions for Data Sets* for the explanation of the return code.

System action: FABKSPMN ends with an abend code of 3503.

Programmer response: Ensure that the SPMNMBR DD statement properly specifies the correct data set, and the specified member exists in the data set. Correct any errors. Rerun the job. If this situation persists, report it to systems operations personnel.

FABK3504E VSAM "*aaaaa*" ERROR FOR HISTORY DATA SET - REG 15: *xx* REG 0: *yy*

Explanation: Program FABKSPMN issued a MODCB or SHOWCB macro to modify or get information of the HISTORY data set. *aaaaa* is either MODCB or SHOWCB. The return code indicates that the attempt was unsuccessful. The values returned in register 15 (*xx*) and 0 (*yy*) are shown (in hexadecimal format).

System action: FABKSPMN ends with an abend code of 3504.

Programmer response: Refer to *z/OS DFSMS Macro Instructions for Data Sets* for a further explanation of the error. Ensure that the data set associated with the HISTORY DD statement is correctly specified. Correct any errors. Rerun the job. If this situation persists, report it to systems operations personnel.

FABK3505E VSAM I/O ERROR ACCESSING HISTORY DATA SET FOR "*yyyyy*" VSAM ERROR DATA: RETURN CODE: *xx* RPL "FDBK": *aaa* (*bb*)

Explanation: Program FABKSPMN encountered an error while attempting to access a record in the HISTORY data set. *yyyyy* is either POINT, GET, or PUT.

The return code is shown in hexadecimal (*xx*), and the RPL FDBK code value is shown in both decimal (*aaa*) and hexadecimal (*bb*) format.

System action: FABKSPMN ends with an abend code of 3505.

Programmer response: Refer to *z/OS DFSMS Macro Instructions for Data Sets* manual for a further explanation of the error. Correct any errors. Perform an VSAM VERIFY on the KSDS, and rerun the job. If this situation persists, report it to systems operations personnel.

FABK3506E INCORRECT HISTORY DATA SET ENTRY WAS FOUND FOR DBNAME: *dbname*

Explanation: Program FABKSPMN found an incorrect HISTORY data set entry while processing the HISTORY data set records for the database (*dbname*). An incorrect HISTORY data set entry may be created when the HD Pointer Checker run fails while processing a database data set group associated with the database (*dbname*), and the rerun is not done for the database data set group on the same day.

System action: FABKSPMN ends with an abend code of 3506.

Programmer response: Run DB Historical Data Analyzer, and generate the HISTORY Data Set by Key Date report and HISTORY Data Set by DB-DS report with the "PROC TYPE=LIST" option. Collect database data set and key date information of the incorrect HISTORY data set entries, prepare necessary DB Historical Data Analyzer control statements, then delete these incorrect entries by running DB Historical Data Analyzer with the "PROC TYPE=DELETE" option. Rerun the job.

FABK3507E DD DUMMY WAS SPECIFIED FOR DDNAME *ddname*

Explanation: Program FABKSPMN found that DUMMY was specified for the data set associated with the DD statement (*ddname*).

System action: FABKSPMN ends with an abend code of 3507.

Programmer response: Specify a correct data set name for the *ddname*. Then rerun the job.

FABK3508I NO DD STATEMENT AVAILABLE FOR SPMN REPORT

Explanation: No effective DD statements are specified for either of the Space Monitor reports. Maybe no DD statement specified, or DUMMY for all specified DD statements.

System action: Processing continues.

Programmer response: If you don't intended to, specify DD statements you want, rerun the job.

FABK3510E MORE THAN 2000 CONTROL STATEMENTS SPECIFIED

Explanation: Program FABKSPMN found that more than 2000 control statements were specified in the member of the data set specified by the SPMNMBR DD statement, or in the data set specified by the SPMNIN DD statement.

System action: FABKSPMN ends with an abend code of 3510.

Programmer response: Divide the specified control statements into two or more groups so that each group contains no more than 2000 control statements. Run a separate job for each control statement group.

FABK3511E SORT FAILED FOR DDNAME SORTIN (RC=*nn*)

Explanation: Program FABKSPMN internally linked DFSORT module to sort a data set specified on the SORTIN DD statement. The return code *nn* indicates that the attempt was unsuccessful. Refer to *DFSORT Application Programming Guide* for the explanation of the return code.

System action: FABKSPMN ends with an abend code of 3511.

Programmer response: Ensure that the DD statements required for DFSORT are properly specified and the sort work data sets have enough space. Correct any errors. Rerun the job. If this situation persists, report it to systems operations personnel.

FABK3520E IDCAMS LISTCAT FAILED FOR DDNAME *ddname* **(RC =** *nn***)**

Explanation: Program FABKSPMN internally linked IDCAMS LISTCAT routine to obtain the data set space information for the data set specified by the ddname (*ddname*). The return code *nn* indicates that the attempt was unsuccessful. Refer to *z/OS V1 R6.0 DFSMS Access Method Services for Catalogs* for the explanation of the return code.

System action: FABKSPMN ends with an abend code of 3520.

Programmer response: Rerun the job. If this situation persists, report it to systems operations personnel.

FABK3530E UNABLE TO OBTAIN DSCB (RC=*nn***) FOR DSNAME:** *dsname*

Explanation: Program FABKSPMN could not obtain a DSCB for the data set (*dsname*). Refer to *z/OS DFSMSdfp Advanced Services* for the explanation of the return code *nn*.

System action: FABKSPMN ends with an abend code of 3530.

Programmer response: Ensure that the volume that contains the data set indicated is mounted. Mount it or catalog the data set and rerun the job. If this situation persists, report it to systems operations personnel.

FABK3531E UNABLE TO OBTAIN EXTENT INFORMATION FROM DSCB FOR DSNAME: *dsname*

Explanation: Program FABKSPMN could not obtain extent information from a DSCB for the data set (*dsname*).

System action: FABKSPMN ends with an abend code of 3531.

Programmer response: Ensure that the volume that contains the data set indicated is mounted. Mount it or catalog the data set and rerun the job. If this situation persists, report it to systems operations personnel.

FABK3533E RDJFCB FAILED FOR DDNAME: *ddname* **(RC=***rc***)**

Explanation: Program FABKSPMN issued a RDJFCB macro and received the non-zero return code *rc*.

System action: FABKSPMN ends with an abend code of 3533.

Programmer response: Contact IBM Software Support.

FABK3540E WRONG FORMAT OF SPMNSPDT DATA SET RECORD FOUND FOR DSNAME: *dsname*

Explanation: Program FABKSPMN found that the SPMNSPDT data set record for the data set (*dsname*) has a wrong format.

System action: FABKSPMN ends with an abend code of 3540.

Programmer response: Ensure that the data set associated with the SPMNSPDT DD statement is correctly specified. You might have copied the correct SPMNSPDT data set to a data set which has a smaller LRECL. Correct any errors and rerun the job.

FABK3541E INCORRECT LRECL OF SPMNSPDT DATA SET

Explanation: Program FABKSPMN found that the SPMNSPDT data set has an incorrect logical record length.

System action: FABKSPMN ends with an abend code of 3541.

Programmer response: Ensure that the data set associated with the SPMNSPDT DD statement is correctly specified. The logical record length of the SPMNSPDT data set should be determined by following guidelines in “Output” on page 508 and “Format” on page 509. Correct any error. Rerun the job.

FABK3542E ″GET″ FAILED FOR DDNAME *ddname***:** *error description*

Explanation: Program FABKSPMN issued a GET macro to get access to the data set associated with the DD statement (ddname). The SYNAD exit routine was called during the GET processing. The error condition is shown in the *error description*.

System action: FABKSPMN ends with an abend code of 3542.

Programmer response: Check the error description and make sure that a DD statement is specifying the correct data set. Correct the error and rerun the job.

FABK3550E INSUFFICIENT STORAGE: INCREASE REGION SIZE

Explanation: Program FABKSPMN issued a GETMAIN to acquire storage for internal control blocks. The return code indicated that the attempt was unsuccessful.

System action: FABKSPMN ends with an abend code of 3550.

Programmer response: Increase the region parameter on the EXEC statement for FABKSPMN. Rerun the job.

FABK3560E *mmmmmmmm* **MACRO ERROR (RC =** *nn***) FOR VOLUME** *vvvvvv*

Explanation: Program FABKSPMN issued a UCBSCAN or LSPACE macro to get information about the VOLUME (*vvvvvv*). *mmmmmm* is either UCBSCAN or LSPACE. The return code indicates that the attempt was unsuccessful.

System action: FABKSPMN ends with an abend code of 3560.

Programmer response: Make sure that the volume is mounted, and rerun the job. If this situation persists, report it to systems operations personnel.

FABK3561E **IGWASYS FAILED (RC =** *nn***)**

Explanation: Program FABKSPMN calls the IGWASYS routine internally to obtain the DFSMS™ level.

System action: FABKSPMN ends with an abend code of 3561.

Programmer response: The return code *nn* indicates that the attempt was unsuccessful. For the explanation of the return code, refer to *DFSMS/MVS Advanced Services*.

FABK3562E **DYNAMIC ALLOCATION(/ DEALLOCATION) FAILED FOR** *ddname* **RC=***xx* **CC=***yyyy*

Explanation: An attempt to dynamically allocate or deallocate the *ddname* data set failed. *xx* is the hexadecimal return code, and *yyyy* is the hexadecimal error reason code.

System action: Space Monitor ends with an abend code of 3562.

Programmer response: None.

Problem determination: For the return code and reason code of SVC 99, see *z/OS MVS Programming: Authorized Assembler Services Guide*, or its higher version.

FABK3999E **UNKNOWN ERROR OCCURRED IN** *modulename* **MODULE (RC =** *nn***)**

Explanation: Program FABKSPMN found that an unknown error occurred in the FABKSPMN internal module (*modulename*). This kind of error should not occur. The return code *nn* is the internal reason code.

System action: FABKSPMN ends with an abend code of 3999.

Programmer response: Contact IBM Software Support.

DB Segment Restructure

This section describes the abend codes, return codes, and messages issued by DB Segment Restructure.

Abend codes

Every 3*nnn* abend code is accompanied by a FABR3*nnn*E message. (3*nnn* is a four-digit identification number of the abend code and message.) See the associated FABR3*nnn*E message description for the 3*nnn* abend code.

Return codes

The following list shows the return codes set by FABRUNLD and FABRRELD.

Code	Meaning
0	The DB Segment Restructure utility has been successfully executed with no errors of any kind. This is the normal return code for all DB Segment Restructure runs.
4	Potential errors have been detected, but execution was able to continue. *This return code is abnormal and indicates that some of the processing you wanted has not been done*. Warning messages that explain the errors will be issued. They will be printed in the run listing, either in the JES log or in one of the SYSPRINT data sets.

Messages

This section explains the messages that are issued by DB Segment Restructure.

FABR3501W DB RECORD MUST PRECEDE OTHER REC TYPES

Explanation: The first control statement in the FABRRELD SYSIN data set was not a DB record.

System action: Program FABRRELD ignores the incorrect control statement and continues processing. The return code will be 4.

Programmer response: Correct the FABRRELD SYSIN data set, and rerun the FABRRELD job.

Problem determination: Check the FABRRELD SYSIN data set to ensure that the control statements are in the proper sequence.

FABR3502W DBD NOT REFERENCED BY PSB.

Explanation: The dbdname specified on a FABRUNLD primary request control statement or a FABRRELD DB control statement was not referenced by the PSB whose name was specified on your EXEC statement.

System action: Program FABRUNLD or FABRRELD ignores the input control statement and continues processing. The return code will be 4.

Programmer response: Correct the input SYSIN data set or JCL, and rerun the job.

Problem determination: Check the spelling of the dbdname on the input control statement. If that is correct, then you have used the wrong PSB.

FABR3503W EXCEEDED MAXIMUM OF 1500 LITERAL BYTES PER DB

Explanation: A FABRRELD DB control statement was followed by CHG control statements that contain literals whose total number of bytes is greater than 1500.

System action: Program FABRRELD ignores the incorrect control statement(s) and continues processing. The return code will be 4.

Programmer response: Correct the FABRRELD SYSIN data set, and rerun the FABRRELD job.

Problem determination: Check the FABRRELD SYSIN data set for errors. A single DB control statement can have a total of at most 1500 bytes of literals on its related CHG control statements.

FABR3504E FABRRELD ABENDS DUE TO *xx* STATUS CODE

Explanation: A DL/1 insert (ISRT) call returned a nonblank status code *xx*.

System action: The program ends with a USER 3504 abend. If a SYSUDUMP DD statement is provided, a dump will be produced.

Programmer response: Determine the cause of the bad status code and correct the problem. Then rerun the FABRRELD job.

Problem determination: Use standard IMS application program debugging procedures.

FABR3505E FABRUNLD ABENDS DUE TO *xx* STATUS CODE

Explanation: A DL/1 get next (GN) call returned a status code (*xx*) other than GA, GB, GK, or blank.

System action: The program ends with a USER 3505 abend. If a SYSUDUMP DD statement is provided, a dump will be produced.

Programmer response: Determine the cause of the bad status code and correct the problem. Then rerun the FABRUNLD job.

Problem determination: Use standard IMS application program debugging procedures.

FABR3506W GU STATUS CODE=*xx*

Explanation: The DL/1 GU call, using the SSA provided on a FABRUNLD primary request control statement, resulted in an unacceptable DL/1 status code (*xx*).

System action: Program FABRUNLD ignores the primary request control statement and continues processing. The return code will be 4.

Programmer response: Determine the cause of the bad status code and correct the problem. Then rerun the FABRUNLD job.

Problem determination: Check the SSA on the primary request control statement. If that is correct, then use standard IMS application program debugging procedures.

FABR3507I *xxxxxxxxx* INPUT SEGMENTS *yyyyyyyyy* SEGMENTS CANCELED

Explanation: This is not an error message. The system is reporting the total number of input segments (*xxxxxxxxx*) and the total number of segments (*yyyyyyyyy*) that were canceled by exit routines for that database.

System action: None.

Programmer response: None.

Problem determination: None.

FABR3508W INVALID HEX DIGIT

Explanation: An incorrect hexadecimal digit was encountered in a hexadecimal literal on a FABRRELD CHG control statement.

System action: Program FABRRELD ignores the incorrect control statement and continues processing. The return code will be 4.

Programmer response: Correct the FABRRELD SYSIN data set, and rerun the FABRRELD job.

Problem determination: Check the FABRRELD CHG control statement for errors. A hexadecimal literal must contain only the characters 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, or F.

FABR3509W I/O ERROR IN USEREXIT PDS DIRECTORY SRCH

Explanation: A BLDL macro using the partitioned data set on the USEREXIT DD statement resulted in a return code of 08.

System action: Program FABRUNLD or FABRRELD ignores the control statement and continues processing. The return code will be 4.

Programmer response: Correct the problem, and rerun the job.

Problem determination: Use standard debugging procedures.

FABR3510W I/P POS MUST BE 0 < 5 DIGITS <= '*xxxxx*'

Explanation: The *inpos* field on a FABRRELD CHG control statement contains an incorrect number (*xxxxx*) (see “CHG control statement” on page 561).

System action: Program FABRRELD ignores the incorrect control statement and continues processing. The return code will be 4.

Programmer response: Correct the FABRRELD SYSIN data set control statement, and rerun the FABRRELD job.

Problem determination: Check the format of the FABRRELD CHG control statement. All FABRRELD CHG control statements that do not contain a literal must contain a valid *inpos* field in column 21.

FABR3511W KFB STRING NOT IN QUOTES OR MISSING

Explanation: A key feedback string is expected to be on a FABRUNLD primary request control statement or continuation request record. Program FABRUNLD was unable to find the key feedback string (which must be enclosed in single quotes).

System action: FABRUNLD ignores the primary request control statement (and continuation request control statement, if any) and continues processing. The return code will be 4.

Programmer response: Correct the input SYSIN data set, and rerun the FABRUNLD job.

Problem determination: Check the format of the

FABRUNLD primary request control statement or continuation request control statement. If the quotes are present, the key feedback string may start in the wrong column.

FABR3512W LENGTH MUST BE 0 < 5 DIGITS <= '*xxxxx*'

Explanation: The *length* field on a FABRRELD CHG control statement contains an incorrect number (*xxxxx*).

System action: Program FABRRELD ignores the incorrect control statement and continues processing. The return code will be 4.

Programmer response: Correct the FABRRELD SYSIN data set control statement, and rerun the FABRRELD job.

Problem determination: Check the format of the FABRRELD CHG control statement. All FABRRELD CHG control statements that do not contain a literal must contain a valid *length* field in column 31.

FABR3513W LITERAL MUST BE ENCLOSED IN QUOTES

Explanation: A literal on a FABRRELD CHG control statement did not contain the correct number of single quotes.

System action: Program FABRRELD ignores the incorrect control statement and continues processing. The return code will be 4.

Programmer response: Correct the FABRRELD SYSIN data set, and rerun the FABRRELD job.

Problem determination: Check the FABRRELD CHG control statement for errors. The literal must begin in column 21 of the CHG control statement.

FABR3514E *xxxxxxxx* DATA SET LRECL IS TOO SMALL

Explanation: The maximum LRECL of the sequential data set (*xxxxxxxx*) that will contain the unloaded database is too small to contain the last retrieved segment.

System action: Program FABRUNLD issues a User 3514 abend.

Programmer response: Increase the LRECL of the *xxxxxxxx* data set, and rerun the FABRUNLD job.

Problem determination: Check the JCL of the *xxxxxxxx* DD statement.

FABR3515W MEMBER NOT FOUND IN 'USEREXIT' DATA SET

Explanation: The *userexit* field on a FABRRELD SEG control statement contains a name that is not a member name in the USEREXIT partitioned data set. A BLDL macro using this data set resulted in a return code of 04.

System action: Program FABRUNLD or FABRRELD ignores the control statement and continues processing. The return code will be 4.

Programmer response: Correct the FABRRELD SYSIN data set, and rerun the FABRRELD job.

Problem determination: Check the FABRRELD SYSIN data set for errors.

FABR3516W MORE THAN 150 'CHG' CARDS PER DB

Explanation: A FABRRELD DB control statement was followed by more than 150 CHG records.

System action: Program FABRRELD ignores the incorrect control statement(s) and continues processing. The return code will be 4.

Programmer response: Correct the FABRRELD SYSIN data set, and rerun the FABRRELD job.

Problem determination: Check the FABRRELD SYSIN data set for errors. A single DB control statement can have at most 150 related CHG control statements.

FABR3517W MORE THAN 50 ’SEG’ CARDS PER DB

Explanation: A FABRRELD DB control statement was followed by more than 50 SEG control statements.

System action: Program FABRRELD ignores the incorrect control statement(s) and continues processing. The return code will be 4.

Programmer response: Correct the FABRRELD SYSIN data set, and rerun the FABRRELD job.

Problem determination: Check the FABRRELD SYSIN data set for errors. A single DB control statement can have at most 50 related SEG control statements.

FABR3518W NO SYSIN CONTROL STATEMENTS SUPPLIED

Explanation: An empty FABRRELD SYSIN data set was used.

System action: Program FABRRELD ends without reloading any databases. The return code will be 4.

Programmer response: Correct the FABRRELD SYSIN data set or JCL, and rerun the FABRRELD job.

Problem determination: FABRRELD requires a nonempty SYSIN data set.

FABR3519W NULL LITERAL INVALID

Explanation: A literal of length 0 was encountered on a FABRRELD CHG control statement.

System action: Program FABRRELD ignores the incorrect control statement and continues processing. The return code will be 4.

Programmer response: Correct the FABRRELD SYSIN data set, and rerun the FABRRELD job.

Problem determination: Check the FABRRELD CHG control statement for errors. The literal must begin in column 21 of the CHG control statement.

FABR3520W O/P POS MUST BE 0 < 5 DIGITS <= '*xxxxx*'

Explanation: The *outpos* field on a FABRRELD CHG control statement contains an incorrect number (*xxxxx*).

System action: Program FABRRELD ignores the incorrect control statement and continues processing. The return code will be 4.

Programmer response: Correct the FABRRELD SYSIN data set control statement, and rerun the FABRRELD job.

Problem determination: Check the format of the FABRRELD CHG control statement. It must contain a valid *outpos* field in column 11.

FABR3521W OPEN FAILED FOR *xxxxxxx* DD

Explanation: The sequential data set whose ddname (*xxxxxxx*) was specified on a FABRUNLD primary request control statement (*OUTPUT*), or a FABRRELD DB control statement (*INPUT*) could not be opened successfully.

System action: Program FABRUNLD or FABRRELD ignores the input control statement and continues processing. The return code will be 4.

Programmer response: Correct the input SYSIN data set or JCL, and rerun the job.

Problem determination: Check the spelling of the ddname on the input control statement. If that is correct, then use standard IMS application program debugging procedures.

FABR3522W OPEN FAILED FOR 'USEREXIT' DD

Explanation: The partitioned data set on the USEREXIT DD statement could not be opened successfully.

System action: Program FABRUNLD or FABRRELD ignores the control statement and continues processing. The return code will be 4.

Programmer response: Correct the problem, and rerun the job.

Problem determination: Use standard debugging procedures.

FABR3523I *xxxxxxxxx* SEGMENTS RELOADED

Explanation: This is not an error message. It is just a message reporting the total number of database segments (*xxxxxxxxx*) that were reloaded for that database.

System action: None.

Programmer response: None.

Problem determination: None.

FABR3524I *xxxxxxxx* SEGMENTS UNLOADED

Explanation: This is not an error message. It is just a message reporting the total number of database segments (*xxxxxxx*) that were unloaded.

System action: None.

Programmer response: None.

Problem determination: None.

FABR3525W TYPE MUST BE ’DB’, ’SEG’, OR ’CHG’

Explanation: A control statement in the FABRRELD SYSIN data set contained an incorrect character string in its first three columns.

System action: Program FABRRELD ignores the incorrect control statement and continues processing. The return code will be 4.

Programmer response: Correct the FABRRELD SYSIN data set control statement, and rerun the FABRRELD job.

Problem determination: All FABRRELD input control statements must contain 'DB', 'SEG', or 'CHG' in column 1.

FABR3526W USER EXIT MARKED 'NOT EXECUTABLE'

Explanation: The *userexit* field on a FABRRELD SEG control statement contains a name that is a member name in the USEREXIT partitioned data set. A BLDL macro using this data set found that member to be marked not executable.

System action: Program FABRUNLD or FABRRELD ignores the control statement and continues processing. The return code will be 4.

Programmer response: Correct the problem, and rerun the job.

Problem determination: Use standard debugging procedures.

FABR3527W STATUS CODE FM RECEIVED, nnnnnnnnnn SEGMENTS NOT RELOADED

Explanation: A FABRRELD received a status code FM for DL/I insert call. A FABRRELD did not reload the root segment and all of its dependent segments. Indicated number of segments have been discarded.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABR3528W ISRT STATUS CODE=FM

FABR3528W ASRT STATUS CODE=FM

Explanation: A DL/I insert (ISRT) or assert (ASRT) call returns status code FM. A FABRRELD did not reload the root segment and all of its dependent segments.

System action: Processing continues.

Programmer response: None.

Problem determination: None.

FABR3625E *xxxxxxxx* DD STATEMENT IS MISSING

Explanation: A required DD statement, *xxxxxxxx*, is not present in the FABRUNLD JCL.

System action: The program ends with a USER 3625 abend. If a SYSUDUMP DD statement is provided, a dump will be produced.

Programmer response: Correct the problem, and rerun the job.

Problem determination: JCL error or control statement error. Either the DD statement has been omitted, or the ddname has been spelled incorrectly on either your primary request control statement or your DD statement.

FABR3626E FAILURE IN DEVTYPE MACRO FOR THE *xxxxxxxx* FILE

Explanation: Program FABRUNLD issued a DEVTYPE macro for the *xxxxxxxx* ddname. The resulting return code was 8.

System action: The program ends with a USER 3626 abend. If a SYSUDUMP DD statement is provided, a dump will be produced.

Programmer response: Correct the problem, and rerun the job.

Problem determination: Use standard debugging procedures.

FABR3627E DUMMY SPECIFIED FOR *xxxxxxxx* FILE

Explanation: A required DD statement, *xxxxxxxx*, was coded as DUMMY in the FABRUNLD JCL.

System action: The program ends with a USER 3627 abend. If a SYSUDUMP DD statement is provided, a dump will be produced.

Programmer response: Correct the problem, and rerun the job.

Problem determination: JCL error.

FABR3628W BLKSIZE OR LRECL OF *xxxxxxxx* FILE IS '0'; MAX ALLOWABLE SIZE FOR DEVICE IS USED

Explanation: BLKSIZE or LRECL is not coded on the *xxxxxxxx* DD statement.

System action: FABRUNLD internally determines the BLKSIZE (block size) for the device assigned to the *xxxxxxxx* DD statement. Refer to the description of “dataout DD” on page 555 for the default block size. It uses BLKSIZE-4 as the LRECL. The return code will be 4.

Programmer response: None, unless the program ends due to an IMS or MVS error. In that case, code both LRECL and BLKSIZE on your DD statement.

When using DISP=MOD (as in Figure 223 on page 569), you may get a system 013 abend if you code BLKSIZE on the DD statement without coding LRECL.

Problem determination: For some DASD devices (such as 3375 or 3380), the default block size and record length result in wasted space. When the maximum record size for the device is considerably smaller than the maximum track length, it is desirable to code BLKSIZE and LRECL on your DD statement. Refer to the description of “dataout DD” on page 555 for the default block size.

FABR3629W THE VALUE IN COLUMN 19 IS INVALID

Explanation: A control statement in the FABRUNLD SYSIN data set contained an incorrect character string in column 19.

System action: Program FABRUNLD ignores the incorrect control statement and continues processing. The return code will be 4.

Programmer response: Correct the FABRUNLD SYSIN data set control statement, and rerun the FABRUNLD job.

Problem determination: All FABRUNLD input control statements must contain 'H' or ' ' in column 19.

FABR3630E LOGICAL DBD *xxxxxxxx* CANNOT BE SPECIFIED WITH 'H' FORMAT OPTION

Explanation: The HD unload record format option is not effective for the indicated database xxxxxxxx.

System action: The program ends with a USER 3630 abend.

Programmer response: Specify the physical DBD with the HD unload record format option, and rerun the job.

Problem determination: None.

FABR3640E UNSUPPORTED LEVEL OF IMS IS BEING USED: *mmm* IMS LEVEL OF THIS RUN

Explanation: Program FABRUNLD or FABRRELD was executed with the IMS version/release which was not supported by FABRUNLD or FABRRELD. *mmm* is the version/release of IMS that was executed.

System action: The program ends with a USER 3640 abend.

Problem determination: None.

FABR3641E UNSUPPORTED RELEASE OF DFP

Explanation: The FABRUNLD or FABRRELD job is running under an unsupported release of MVS/DFP™. The DFSMSdfp of DFSMS/MVS Version 1 Release 1 or later is necessary to run an FABRUNLD or FABRRELD job.

System action: The program terminates with a USER 3641 abend.

Programmer response: None.

Appendix B. Diagnostics Aid

If you have a problem that you think is not a user error, you should run the Diagnostics Aid (FABPDIAG), 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:

- "How to run Diagnostics Aid with JCL"
- "Load Module/Macro APAR Status report" on page 718
- "Messages and codes" on page 720

How to run Diagnostics Aid with JCL

To run the Diagnostics Aid (FABPDIAG), supply an EXEC statement and a DD statement that defines the output data set.

EXEC

This statement must be in the following form:

```
//stepname  EXEC PGM=FABPDIAG
```

SHPSLMD DD

This statement defines the library containing the load modules (usually HPS.SHPSLMD0) for which you have a problem.

The Load Module APAR Status report is not generated if this DD statement is not provided or if DD DUMMY is specified.

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.

The Macro APAR Status report is not generated if this DD statement is not provided or if DD DUMMY is specified.

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
```

Load Module/Macro APAR Status report

The Diagnostics Aid generates the following two reports for the maintenance by IBM:

- Load Module APAR Status report
- Macro APAR Status report

Load Module APAR Status report

This report contains the following information:

MODULE LIBRARY
This 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 is the name of the load module member or the alias.

ALIAS-OF
This is 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 is the name of the included CSECT in the module. The CSECT names are reported in the included order in the module.

APAR NUMBER
This is 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 is 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 HP Pointer Checker 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

This report contains the following information:

MACRO LIBRARY
This 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 is the name of the macro member or the alias.

ALIAS-OF
This is 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 is the latest APAR number applied to the macro. If no APAR is applied, NONE is shown.

APAR FIX-DATE
This is 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 HP Pointer Checker 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

This section describes the messages and codes of FABPDIAG.

Return codes

FABPDIAG contains 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 an FABU36*xx* message. Refer to the appropriate message for problem determination.

Messages

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.

Programmer response: None.

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.

Programmer 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.

Programmer 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 DD STATEMENT NOT FOUND

FABU1005W 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.

Programmer 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.

Programmer response: Ensure which libraries have correct module/macro libraries. Correct the error and rerun the job if necessary.

FABU1007W DUMMY SPECIFIED FOR SHPSLMD DD STATEMENT

FABU1007W DUMMY SPECIFIED FOR 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.

Programmer response: If you did not intend to specify the dummy DD statement, correct the error and rerun the job.

FABU1008W NO MODULE MEMBERS FOUND IN DDNAME SHPSLMD

FABU1008W NO MACRO MEMBERS FOUND IN DDNAME 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.

Programmer 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.

Programmer 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.

Programmer 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.

Programmer 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.

Programmer 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.

Programmer 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.

Programmer 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.

Programmer 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.

Programmer 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.

Programmer 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.

Programmer 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.

Programmer 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.

Programmer 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.

Programmer response: Increase the region size and rerun the job.

FABU3612E TOO MANY MODULE MEMBERS DETECTED IN DDNAME SFABMOD

FABU3612E TOO MANY MACRO MEMBERS DETECTED IN DDNAME 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.

Programmer response: Specify the correct data set for the indicated DD statement and rerun the job.

Appendix C. Migration considerations

This appendix describes points that must be considered in migrating from a previous version or release of IMS HP Pointer Checker or IMS DBT.

Topics:

- "HD Pointer Checker migration"
- "HD Tuning Aid migration" on page 726

HD Pointer Checker migration

This section explains the migration considerations of HD Pointer Checker.

Migrating from DBT Version 1

This HD Pointer Checker provides new HD Pointer Checker JCL procedures that does not support the control statements of DBT Version 1 HD Pointer Checker. The JCL procedures provided with DBT Version 1 HD Pointer Checker are not supported by this HD Pointer Checker.

Migrating from DBT Version 2 Releases 1 or Version 2 Release 2

Table 70 summarizes the differences between the DD statements in the JCL procedures for this version of HD Pointer Checker and those of DBT Version 2 Release 1 or DBT Version 2 Release 2.

Table 70. Format of DD statements for HD Pointer Checker and DBT Version 2 Releases 1 and 2

DDNAME	DBT V2R1 or V2R2 HD Pointer Checker format	DDNAME in IMS HP Pointer Checker V2	IMS HP Pointer Checker V2 format
HDRECS	LRECL=26	—	Removed
SORTEX	LRECL=26	SORTEX01	Removed in the TYPE=ALL process LRECL=40 in the TYPE=SCAN and the CHECK processes
SORTIN	LRECL=26	—	Removed
MERGIN	LRECL=26	MERGIN01	RECFM=VB
RECOUT	LRECL=26	—	Removed
JRM	LRECL=26	JRM	LRECL=40

For how to change the DD names, see "Changes in the DD names" on page 725.

If the DD names of removed data sets or the previous formats are specified, HD Pointer Checker can bypass them. But, it is recommended to remove such DDs because they waste DASD space.

The names of cataloged procedures and sample control statement member names have been changed in DBT Version 2 Release 2. Thus, if you were using the IBM-supplied cataloged procedure in DBT Version 2 Release 1, you must modify your JCLs so that the new names of cataloged procedures are used in your JCLs.

The format and record length of the KEYSIN data set of DBT Version 2 have been changed. HD Tuning Aid cannot process the KEYSIN data set created by DBT Version 2 HD Pointer Checker. HD Tuning Aid can process only the KEYSIN data set input created by HD Pointer Checker of IMS HP Pointer Checker Version 1 or Version 2, or one created by a user in the same format.

The HISTORY data set of DBT Version 2 Release 1 and Version 2 Release 2 cannot be used in IMS HP Pointer Checker, because the format and record length are different.

Migrating from DBT Version 2 Release 3

Table 71 summarizes the differences between the DD statements in the JCL procedure for DBT Version 2 Release 3 and those in the JCL for this version of HD Pointer Checker.

Table 71. Format of DD statements for Version 2 Release 3 and earlier releases

DDNAME	DBT V2R3 HD Pointer Checker format	DDNAME in IMS HP Pointer Checker V2	IMS HP Pointer Checker V2 format
HDRECS	LRECL=28	—	Removed
SORTEX	LRECL=28	SORTEX01	Removed in the TYPE=ALL process LRECL=40 in the TYPE=SCAN and the CHECK processes
SORTIN	LRECL=28	—	Removed
MERGIN	LRECL=28	MERGIN01	RECFM=VB
RECOUT	LRECL=28	—	Removed
CHECKREC	LRECL=28	CHECKREC	LRECL=40
JRM	LRECL=28	JRM	LRECL=40
IXKEY	LRECL=291	IXKEY	RECFM=VB

For how to change the DD names, see “Changes in the DD names” on page 725.

If the DD names of removed data sets or the previous formats are specified, HD Pointer Checker can bypass them. But, it is recommended to remove such DDs because they waste DASD space.

The names of the cataloged procedures and the sample control statement member have not been changed, but SORT field of FABPSRT2 have been changed. If you are using the IBM-supplied cataloged procedure in DBT Version 2, you must use the procedure in this release.

The format and record length of the KEYSIN data set of DBT Version 2 have been changed. HD Tuning Aid cannot process the KEYSIN data set created by the DBT Version 2 HD Pointer Checker. HD Tuning Aid can process only the KEYSIN data set input created by HD Pointer Checker of IMS HP Pointer Checker Version 1 or one created by a user in the same format.

The format and record length of the HISTORY data set of DBT Version 2 have been changed. You must migrate them by the procedure described in “Migrating from DBT Version 2 Release 3 format” on page 370.

Migrating from IMS HP Pointer Checker Version 1

See also "Compatibility with IMS HP Pointer Checker Version 1" on page 9.

Work data sets

Some work data sets have been removed or the format have been changed as follows:

- HDRECS, SORTIN, SORTEX, and RECOUT are removed from a single-step process (TYPE=ALL).
- HDRECS, SORTIN, and RECOUT are removed from multiple-step processes (TYPE=SCAN and CHECK).
- MERGIN has been changed to RECFM=VB from RECFM=FB, LRECL=40.

If the DD names of removed data sets or the previous formats are specified, HD Pointer Checker can bypass them. But, it is recommended to remove such DDs because they waste DASD space.

Read "Changes in the DD names," because some DD names are changed.

DFSORT work data sets

SORTWK*nn* that is DFSORT work data set DD names have been changed to SORTWK*nn*,SRTXWK*nn*, and SRTEWK*nn*. *nn* is a two-digit number. These work data sets are allocated dynamically by DFSORT. But if you specify SORTWKnn in your JCL for some reason, for example the sizes are too large to be allocated dynamically, you may need to change the DD names.

TYPE=BLKMAP process

If you process TYPE=BLKMAP, you need to specify CHECKREC=YES in TYPE=ALL or TYPE=CHECK that is preprocess of TYPE=BLKMAP.

Changes in the DD names

The DD names changed are shown in Table 72.

Table 72. DD names that are changed

Old name	New name
MERGIN	MERGIN01
SORTEX	SORTEX01
MERGIN2	MERGI201
SORTEX2	SORTE201
SORTIL	SORTIL01

IMS HP Pointer Checker Version 2 replaces old DD names with new ones during its processing. If both old and new names are specified, the new one is used. Therefore, delete the old names from your JCL.

Warning:

New DD names are predefined in the FABPP procedure of IMS HP Pointer Checker Version 2, and old names will be ignored even if you specify them in your JCL. Change, therefore, the old names to new ones in your JCL.

IMS HP Pointer Checker Version 2 provides a new parameter, SCANGROUP=. It enables a parallel scan of data sets and improves performance. If you use more than two scan groups, you need to know new DD names of work data sets MERGIN*xx* and others. For more detailed information about SCANGROUP and DD statements, see “DATABASE statement” on page 80 and “DD statements” on page 38.

HD Tuning Aid migration

This section explains the migration considerations of HD Tuning Aid.

Migrating from DBT Version 2

The cataloged procedures of the HD Tuning Aid utility have been changed. You must use the new ones. If you are using those shipped with IMS System Utilities/Data Base Tools Version 2 HD Tuning Aid, you must modify them as follows:

- In some data sets, the value of LRECL has been changed to 42:
  - RAPSIN of FABTROOT program
  - SORTOUT of SORT program

The SORT parameter that is input to DFSORT has been changed as follows:

```
//SYSIN  DD  *
 SORT  FIELDS=(1,8,BI,A),FILSZ=En
 END
/*
```

Migrating from IMS HP Pointer Checker Version 1

The cataloged procedure FABPPTAM, that is a combination of HD Pointer Checker and the HD Tuning Aid utility, has been changed in HD Pointer Checker part. You must use the new one. For more detail information, see “Compatibility with IMS HP Pointer Checker Version 1” on page 9.

Appendix D. Layout of KEYSIN data set record

The layout for the KEYSIN data set records is a product-sensitive programming interface. See "Programming interface information" on page 732 to understand the restrictions associated with this type of material.

The layouts for the KEYSIN header record and the KEYSIN key record can be referenced by using the FABUKEYS macro to map theses records.

If you specify RECFM=F or FB on the DCB parameter in the KEYSIN DD statement, the RDW field (first four bytes) is not generated.

Topics:

- "Macro intended for customer use"

Macro intended for customer use

This section describes a macro that enables customers to write programs for the HD Pointer Checker. You should use this macro to request or receive the HD Pointer Checker services.

The following product-sensitive macro is provided by HD Pointer Checker:

FABUKEYS

As stated earlier in this appendix, this macro is used to present the layout for the KEYSIN data set records.

Appendix E. Layout of HISTORY data set record

This appendix describes product-sensitive programming interface information. See "Programming interface information" on page 732 to understand the restrictions associated with this type of material.

The layout has the four types of HISTORY data set records. You can use the FABGHIR macro to map these records with the DSECT name HISTREC.

Topics:

HISTORY data set control record

This control record contains the following information:

- HD Pointer Checker run time list which shows all dates of the HD Pointer Checker runs and the number of runs a day
- Number of database data set table records in the HISTORY data set
- Number of database data set records in the HISTORY data set.

Database data set (DBDS) table record

This record contains key date list entries. Each entry contains the following information:

- Key date field of the database data set records for the database data set
- Number of database data set records of the same key date for the database data set
- A flag indicating whether HD Pointer Checker CHECK process was done for the database data set.

Database data set (DBDS) record

This record contains analysis information of a database data set collected by HD Pointer Checker.

There are three types of database data set records as follows:

- **Statistics record 1**

  This is the first record of database data set records with the same key date for the database data set. It contains following information:

  - Database description information
  - HD Pointer Checker run time option information
  - Pointer validation information
  - HISAM, HIDAM index, or secondary index information
  - HD Pointer Checker output records count information.
- **Statistics record 2**

  This is the second record of database data set records with the same key date for the database data set.

- Free space statistics information
  - Tuning statistics information
  - DB records distribution statistics information
  - HD Pointer Checker output records count information.
- **Segment statistics record**

  This is the 3rd or up to 255th record of database data set records with the same key date for the database data set. One record can contain the information of two segments.
  - Segments statistics information

HD Pointer Checker run time record

This record contains the key date and the names of all database data set processed by the last HD Pointer Checker run in one day.

Macro intended for customer use

The macro identified in this section is provided to allow a customer installation to write programs that use the services of DB Historical Data Analyzer. Use only this macro to request or receive the services of DB Historical Data Analyzer.

DB Historical Data Analyzer provides only the following product-sensitive macro:

FABGHIR

This macro is used to map the HISTORY data set records with the DSECT name HISTREC.

Appendix F. Notices

This guide was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This guide 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 guide to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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 Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA
U.S.A.
95141-1003

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 guide 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.

Programming interface information

This publication primarily documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IMS HP Pointer Checker. This publication also documents information that is NOT intended to be used as Programming Interfaces of IMS HP Pointer Checker. This information is identified where it occurs by an introductory statement to a topic or section.

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 may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-Sensitive Programming Interface and Associated Guidance Information is identified where it occurs, either as an introductory statement to a chapter or section or by the following marking:

Product-Sensitive Programming Interface

Description of Product-Sensitive Programming Interface and Associated Guidance Information...

End of Product-Sensitive Programming Interface

This guide also documents Diagnosis, Modification, and Tuning information, which is provided to help the customer to administer IMS databases.

Warning: Do not use this Diagnosis, Modification, and Tuning Information as a programming interface.

Diagnosis, Modification, and Tuning Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

Diagnosis, Modification, and Tuning Information

Description of Diagnosis, Modification, and Tuning Information...

End of Diagnosis, Modification, and Tuning Information

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A complete and current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml.

Microsoft, Windows, Windows NT®, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Index

Program Number: 5655-K53

Printed in USA

SC18-9974-01