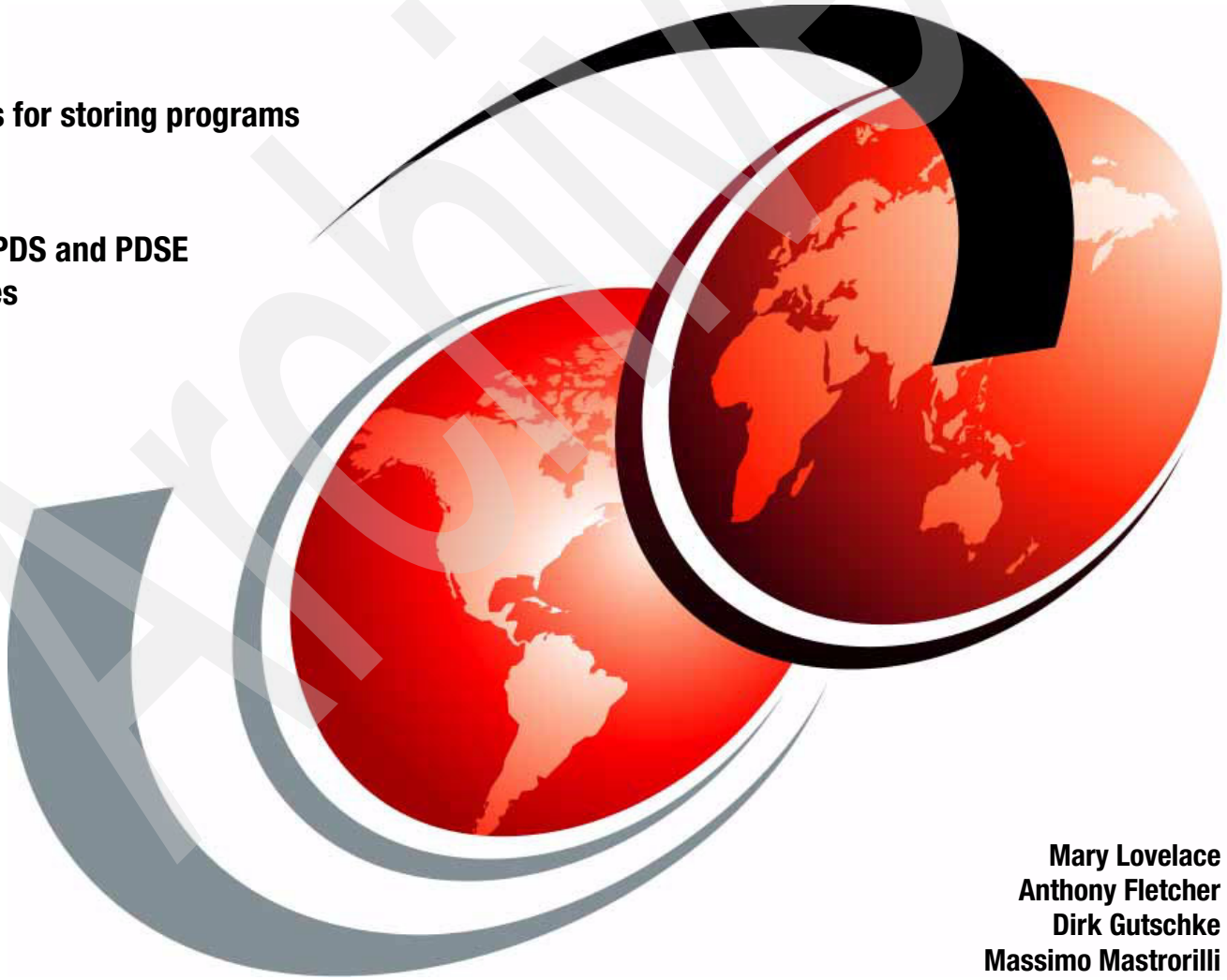
IBM

Partitioned Data Set Extended Usage Guide

- Learn the latest PDSE functions
- Use PDSEs for storing programs
- Compare PDS and PDSE capabilities



Mary Lovelace
Anthony Fletcher
Dirk Gutschke
Massimo Mastrorilli

Redbooks

ibm.com/redbooks

International Technical Support Organization

Partitioned Data Set Extended Usage Guide

May 2005

SG24-6106-01

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

Second Edition (May 2005)

This edition applies to Version1 Release 6 of z/OS (product number 5694-A01).

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Advanced Function Printing™
AFP™
AIX®
DB2®
DFSMS/MVS®
DFSMSdfp™
DFSMSdss™
DFSMShsm™
DFSMSrmm™
DFSORT™
Hiperspace™
IBM®
ibm.com®
IMS™
MVS™
MVS/DFP™
OS/390®
OS/400®
Redbooks™
Redbooks (logo) ™
RACF®
RMF™
S/390®
System/390®
Tivoli®
WebSphere®
z/Architecture™
z/OS®

The following terms are trademarks of other companies:

Microsoft, Windows, 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.

Other company, product, and service names may be trademarks or service marks of others.

Preface

This IBM® Redbook will help you install, manage, and use PDSEs. It is a substantial rewrite of *Partitioned Data Set Extended Usage Guide,* SG24-6106, which was published in March 2001. It brings the originally published material up to date and includes enhancements made to PDSEs since the original book was published.

The book describes how to define and use PDSEs, and shows how PDSEs offer improvements over ordinary partitioned data sets, such as sharing between users and systems with integrity, an automatically extending directory, and reuse of space.

Information for storage administrators and systems programmers includes system-level information about sharing in a sysplex, the use and definition of non-SMS PDSEs, and how to diagnose problems with PDSEs. The advantages of using PDSEs for storing executable code are explained.

Figure 1 shows the team that wrote this redbook.

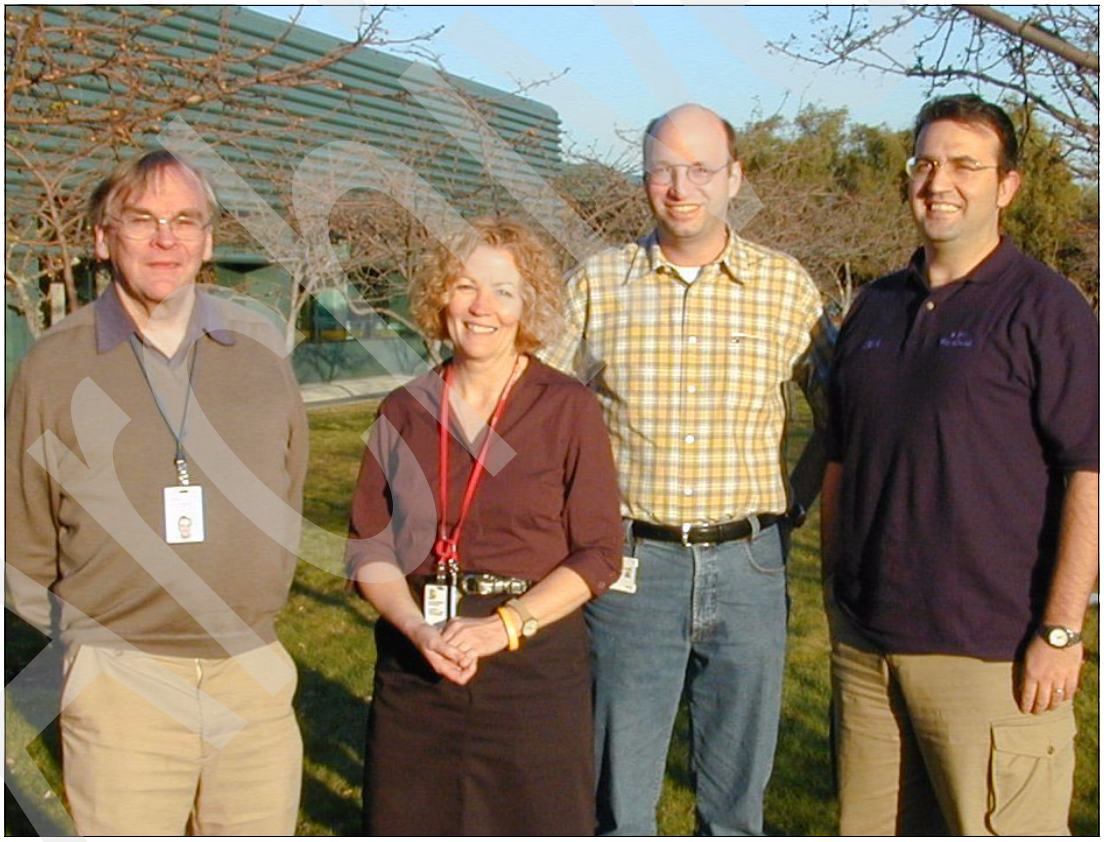


Figure 1 Anthony, Mary, Dirk, and Massimo

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

Mary Lovelace is a Consulting IT specialist at the International Technical Support Organization. She has more than 20 years of experience with IBM in large systems, storage

and Storage Networking product education, system engineering and consultancy, and systems support.

Anthony Fletcher is an IT Specialist working with the Global Services Delivery (GSD) z/OS® software platform of IBM Global Services On Demand Infrastructure Services, based in New Zealand and working for New Zealand and Australia. He has 35 years of experience in z/OS, OS/390®, and their predecessors and related components, both as a customer of IBM and with IBM Global Services. He holds a degree in Electrical Engineering from the University of Salford, Lancashire, UK. His main areas of expertise include DFSMS, DFSMSrmm™, and DFSMShsm™, and he has working knowledge of RACF®. He also has experience installing non-IBM products for the GSD platform.

Dirk Gutschke is an IT specialist working in the IBM Global Services ITS Software Support Center in Mainz, Germany, for z/OS DFSMS. He provides Backoffice (level 2) technical support to customers and IBM staff in EMEA. Dirk has worked at IBM for 15 years. His areas of expertise include DFSMS/MVS® with particular focus on PDSE and HFS.

Massimo Mastrorilli is an Advisory IT Storage Specialist in Switzerland. He joined IBM Italy in 1989, and six years ago he moved to IBM Switzerland, based in Lugano. He has 15 years of experience in implementing, designing, and supporting DFSMS storage solutions in S/390® for IBM Enterprise customers. His areas of expertise include IBM Tivoli® Storage Manager, SAN Storage Area Network, and storage solutions for open systems. He is an IBM Certified Specialist for ADSM/TSM, Storage Sales, and Open System Storage Solutions.

Thanks to the following people for their contributions to this project:

Bob Haimowitz
International Technical Support Organization, Raleigh Center

Sangam Racherla
International Technical Support Organization, San Jose Center

Ron Bretschneider
Jerry Dearing
Stuart Edwards
Elmer Latorre
Tom Mahon
Lyle Merithew
IBM San Jose

Barry Hocken
EMEA DFSMS Back Office, United Kingdom

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners, and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us because we want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. QXXE Building 80-E2
  650 Harry Road
  San Jose, California 95120-6099

1

Release summary

This chapter provides details about the recent enhancements for Partition Data Set Extended (PDSE) data sets. The historical background is discussed, as well as function delivered in z/OS V1R3 DFSMS through z/OS V1R6 DFSMS.

1.1 z/OS V1R3 DFSMS changes

In this section, enhancements to PDSE in z/OS V1R3 DFSMS are discussed. None of these enhancements were delivered in the base code, but a major enhancement was delivered through APAR OW53245.

1.1.1 PDSE enhancements delivered with base code

There were no enhancements delivered for PDSE processing in the base code for z/OS V1R3 DFSMS. At this level there were still two system address spaces, SMXC and SYSBMAS, that did the majority of processing for PDSE in addition to the processing within the user address space.

A problem experienced at this level was that large amounts of CSA and ECSA storage were used for PDSEs with large numbers of members. When the common storage resource is exhausted, an IPL is necessary to recover the system.

In addition to this, system hangs arising out of failure and cancel situations required IPLs for recovery. Also, there were no tools for storage administrators for display or diagnosis of problem situations with PDSEs.

Figure 1-1 shows the PDSE address space situation before APAR OW53245.

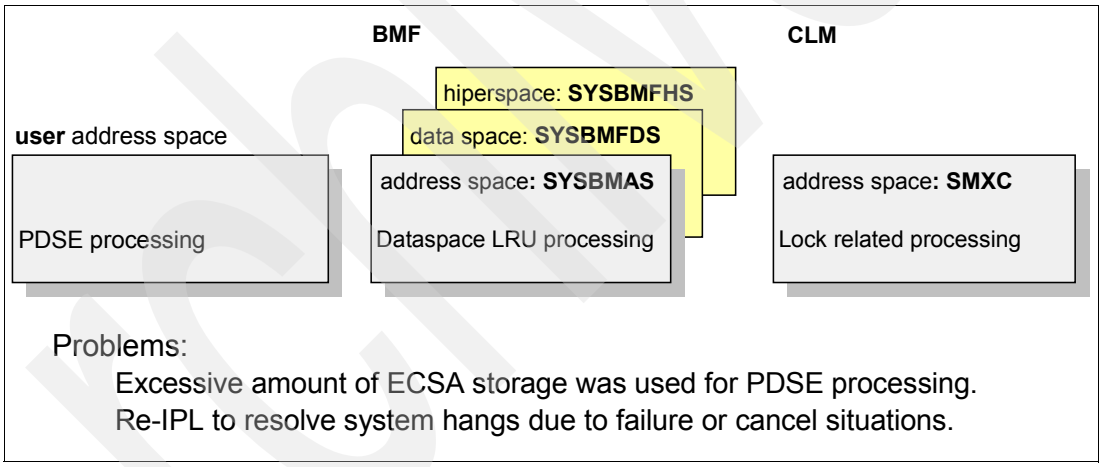


Figure 1-1 PDSE implementation before APAR OW53245

1.1.2 PDSE enhancement delivered with APAR OW53245

After z/OS V1R3 DFSMS was generally available, the PDSE enhancement to eliminate the common storage constraint problem, discussed above, was delivered in July 2002 through APAR OW53245. This APAR improved overall PDSE usability and reliability. A new non-restartable address space, named *SMSPDSE,* has been created to alleviate problems in the amount of ECSA used to support the PDSE data set format. In summary, these are the advantages you get after APAR OW53245:

- Reducing excessive ECSA usage (by moving control blocks into the SMSPDSE address space)
- Reducing re-IPLs due to system hangs (in failure or CANCEL situations)
- Providing storage administrator's tools for monitoring and diagnosis (for example, determining which systems are using a particular PDSE)

Figure 1-2 shows the PDSE address space after OW53245 installation.

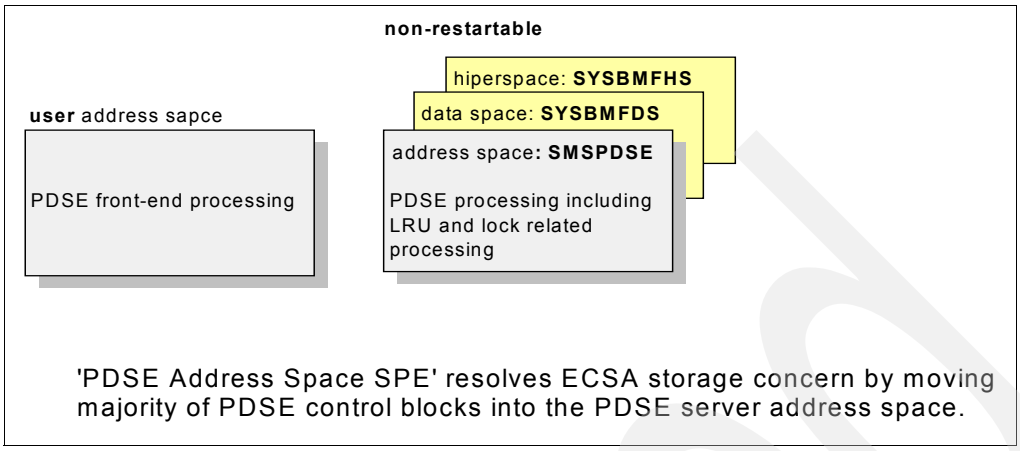


Figure 1-2 PDSE implementation after OW53245

Technical benefits

Approximately 1000 to 2000 bytes of CSA and ECSA per member in a PDSE are saved. For PDSEs that have thousands of members that can be used or “connected” concurrently, this is substantial relief. IEBCOPY and other programs that do a BLDL for all members (such as IEBPTPCH) fall into this category. The storage is not released until a STOW DISCONNECT is requested for each member, or the PDSE is closed. In either case, if circumstances dictate that all members be connected concurrently, the relief is significant.

All processing that took place in the SMXC and SYSBMAS system address spaces has been moved to the new system address space SMSPDSE, which also replaces those address spaces. This enables the removal of the majority of the control blocks from CSA/ECSA subpools to extended private storage within the SMSPDSE address space, though some still remain. Much of the processing that was previously done in the user address space has been moved to the SMSPDSE address space.

Implementation

To implement the new function provided by OW53245, apply the applicable PTFs:

- UW99418 for z/OS V1R3 DFSMS (FMID HDZ11G0), plus any other required maintenance as documented in informational APARs II13336.
- For z/OS V1R4 DFSMS or later, this support is included in the base code.

Note: An IPL must be performed to implement the change, as there will be a new system address space initialized.

Operational changes

As just noted, the SMSPDSE address space replaces the SMXC and SYSBMAS address spaces. In SYSPLEXes, new messages are seen indicating successful connection to XCF.

Specifically, there are changes to message `IGW040I` and two new messages: `IGW041A` and `IGW042A`. Both `IGW040I` and `IGW041A` relate to XCF signalling, and `IGW042A` indicates if there was a problem with end-of-memory processing for the address space. These messages are also documented in APAR OW53245 as shown in Figure 1-3 on page 4.

```
IGW04OI PDSE Connecting to XCF for IPLID
IGW04OI PDSE Connected to XCF for IPLID
IGW04OI PDSE IGWLGEDC Connected
IGW04OI PDSE Connecting to XCF for Signaling
IGW04OI PDSE Connected to XCF for Signaling
IGW04OI PDSE Posting initialization
IGW043I PDSE MONITOR IS ACTIVE 248
++ INVOCATION INTERVAL:60 SECONDS
++ SAMPLE DURATION:15 SECONDS
IGW061I SMSPDSE INITIALIZATION COMPLETE.
```

Figure 1-3 SMSPDSE initialization messages at IPL time

Diagnostic information

There are two commands:

1. **VARY SMS,PDSE,ANALYSIS**

   This gives the storage administrator diagnostic information to use in error situations to determine PDSE lock and latch usage. This should help you respond with a more precise corrective action in error situations. Figure 1-4 shows partial output from the ANALYSIS command. The latch address is shown along with the holder's ASID and TCB address.

```
IGW031I PDSE ANALYZE Start of Report
    ---------data set name---------------------- ---vsgt----------
    SYSPLEX.TEMP.PDSE                           01-XP0201-000104
    ++ Message to SYSTEM2 pending for 32 seconds
    ++ Unable to latch Dib:074FC000
       Holder(003A:007D2858)
    PDSE ANALYZE End of Report
```

Figure 1-4 Output of V SMS,PDSE,ANALYSIS command

2. **VARY SMS,PDSE,FREELATCH(latchaddr,asid,tcbaddr)[,RETRIES(n|1500)]**

   The latch address, ASID, and TCB address will be displayed in the output of an ANALYSIS command if any contention exists; hence, it is always recommended to precede the FREELATCH command with the ANALYSIS command. For additional information, see Chapter 10, "PDSE problem determination" on page 209.

1.2 z/OS V1R5 DFSMS changes

In this section we confirm the current status of enhancements for PDSE in z/OS V1R5 DFSMS.

1.2.1 PDSE enhancements delivered with base code

There were no enhancements delivered for PDSE processing in the base code for z/OS V1R5 DFSMS.

Even though OW53245 reduced the number of system hangs and the need for IPLs, some have still been known to occur. After cause analysis, design changes have been planned to reduce these problems further and make PDSE processing more reliable and recoverable. This leads to the new function included in z/OS V1R6 DFSMS.

1.3 z/OS V1R6 DFSMS changes

In this section we cover the enhancements for PDSE in z/OS V1R6 DFSMS, delivered as base code.

1.3.1 PDSE RAS restartable address space enhancement

With the introduction of the SMSPDSE address space, the majority of PDSE processing was moved into its own address space. With a continued focus on high availability and outage elimination, the PDSE component of z/OS V1R6 DFSMS delivers a new restartable PDSE address space named SMSPDSE1.

In a z/OS V1R6 or later system you have either the single SMSPDSE address space, or both SMSPDSE and SMSPDSE1 address spaces.The non-restartable SMSPDSE will be the *only* PDSE address space in the following circumstances:

- If PDSESHARING(NORMAL) is defaulted, or is specified in the IGDSMSxx initialization parameters.
- In a sysplex coupled systems environment, if PDSESHARING(EXTENDED) is specified, as is PDSE_RESTARTABLE_AS(NO) in the IGDSMSxx initialization parameters.

The new SMSPDSE1 address space will be restartable. It will provide connections to, and process requests for, those PDSE data sets that are not part of the global connections associated with SMSPDSE. Global connections are used for PDSEs in the linklist DCBs. In a sysplex coupled systems environment, when PDSESHARING(EXTENDED) and PDSE_RESTARTABLE_AS(YES) is specified in the IGDSMSxx PARMLIB parameters, SMSPDSE1 will be created during IPL NIP processing.

Figure 1-5 shows the PDSE address space at the z/OS V1R6 DFSMS level.

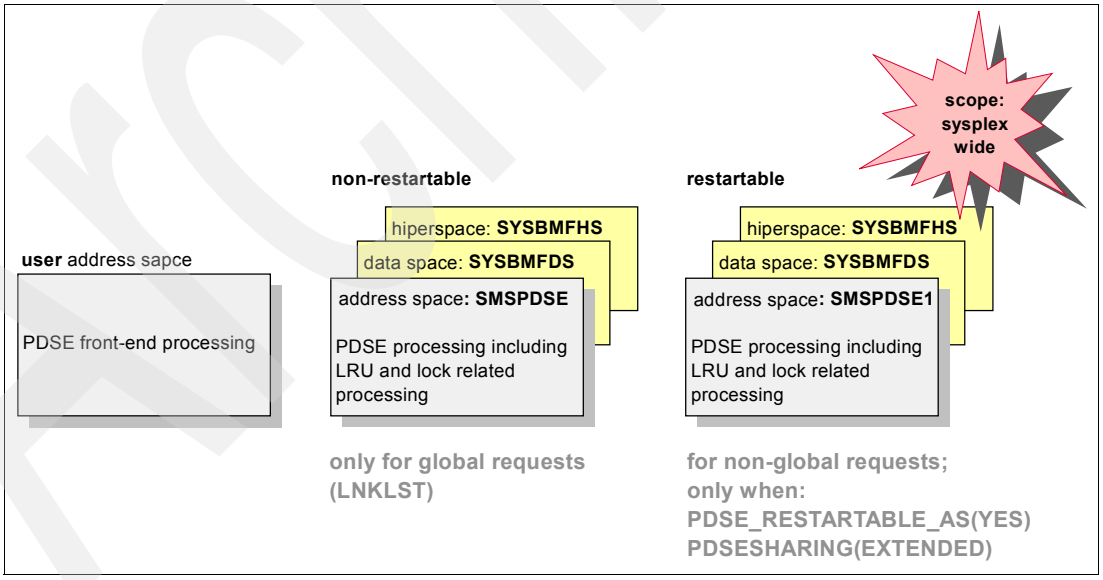


Figure 1-5 PDSE RAS restartable address space support (HDZ11J0)

With the new restartable SMSPDSE1 address space, an IPL will no longer be required to restart PDSE processing after most critical errors. This enhancement includes new commands to affect termination, activation, and restart. The previous ANALYSIS and FREELATCH commands remain and they can be used against the new address space when active.

Technical benefits

Through the implementation of the optional, second, restartable address space SMSPDSE1, the need to IPL to recover from a PDSE critical error is greatly reduced.

Implementation

To install and use the new enhancements provided by the design enhancement, you must:

- Apply SMPE coexistence and toleration PTFs.
- Review and add new parameters in the IGDSMSxx PARMLIB member.
- Review new options available for the **D SMS** and **VARY SMS** commands.

For further details, see Chapter 3, “Managing the PDSE environment” on page 33.

2

PDSE overview and concepts

This chapter presents a brief history of PDSEs. It reviews the partitioned data set (PDS), its uses, its advantages, and some improvements that have been widely requested. It also examines the structure of Partitioned Data Set Extended (PDSE). It explains how PDSEs meet the requirements for improvements to PDSs and describes the logical views of PDSEs for different users.

This introduction does not replace the chapters about processing PDSs and PDSEs found in *z/OS V1R6.0 DFSMS/MVS Using Data Sets,* SC26-7410. Rather, it complements those chapters and provides some additional background material.

2.1 History of PDSEs

PDSEs were first introduced in MVS/DFP™ 3.2 with corresponding support in DFDSS 2.5 and DFHSM 2.5. At that time, PDSEs had to be system-managed.

Since then, several significant enhancements have been provided, as outlined below and in Figure 2-1.

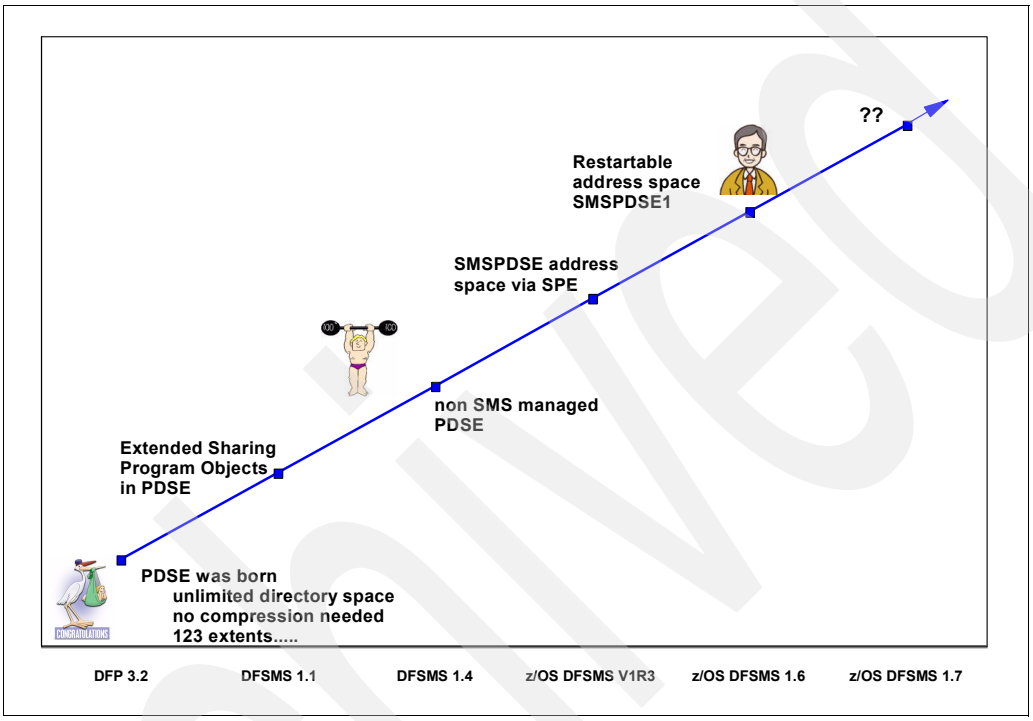


Figure 2-1 History of PDSE

2.1.1 Extended sharing

Extended sharing was introduced in DFSMS/MVS 1.1. It provides the ability for users on different MVS systems in a sysplex to create, read, or update members of a PDSE concurrently.

Any number of users or systems can concurrently share a PDSE or members in it for input (read) or output (write). Multiple members can be updated by different users on the same system, but only one user can update a member in place at a given time. In addition, only one system can access the PDSE when update in place is being done.

Sharing is described in detail in Chapter 8, “PDSE sharing and serialization” on page 153.

2.1.2 Storing executable code in PDSEs

DFSMS/MVS 1.1 introduced the program management binder and loader. The binder was designed as a replacement for the old linkage editor and batch loader, and the loader as a replacement for program fetch, in order to lift a number of restrictions inherent in the linkage editor and the old load module format.

The binder has the option of writing executable code in two distinct forms:

- Load modules in PDSs
- Program objects in PDSEs

Further enhancements for program objects were made in DFSMS/MVS 1.3 and 1.4.

The format of program objects and the advantages of using them are discussed in Chapter 7, “Program management and PDSEs” on page 145.

2.1.3 Non-SMS PDSEs

Originally, PDSEs had to be managed by SMS. SMS management has these advantages:

- Using the storage class to define performance objectives for the system with the provision of guaranteed synchronous write.
- Using the management class to give great flexibility in how PDSEs should be managed at a data set level.
- System-managed PDSEs must be cataloged and must be on system-managed volumes. Uncataloged non-SMS PDSEs are not backed up by DFSMShsm. Cataloged, SMS-managed data sets can be backed up by DFSMShsm.

Because system-managed PDSEs must be cataloged and must be on system-managed volumes, it is difficult to use PDSEs for system software. With non-SMS managed PDSEs, it is easy to place PDSEs on the IPL volume because they need not be cataloged, making cloning simpler. You can also use indirect cataloging (using a volume serial number of ******).

Support for non-SMS PDSEs is included in the base code of DFSMS 2.10 or later.

Important: We recommend that user PDSEs *not* be allocated as non-SMS data sets. The use of SMS still provides benefits for storage management, enabling management class controls to be used and allocation to be made to storage groups rather than specific volumes.

The latest information about non-SMS PDSEs can be found in information APAR II12221.

2.2 Partitioned data set review

If you do not need to review how PDSs work, you can begin with 2.2.4, “PDS customer requirements and limitations” on page 11.

2.2.1 Partitioned data set organization

The simplest data structure in an MVS system is a *sequential* data set. It consists of one or more records that are processed in sequence. Examples are an output data set for a line printer or a deck of punch cards. Sequential data sets are defined in Job Control Language (JCL) with a data set organization of PS (DSORG=PS), which stands for *physical sequential*. In other words, the records in the data set are physically arranged one after another.

A *partitioned* data set adds a layer of organization to this simple structure. A PDS consists of a directory and zero or more members. Each member is like a sequential data set and has a simple name, which can be up to eight characters long. The directory is a series of keyed 256-byte blocks, which contain entries for the members in the PDS. Each entry consists of a

member name and a pointer, plus some optional user data. The eight-byte hardware key contains the name of the last member in the directory block.

This structure is defined in JCL with a data set organization of PO (DSORG=PO), which stands for *partitioned organization*.

2.2.2 Where to use partitioned organization

The PDS structure was designed to provide efficient access to libraries of related members, whether load modules, program source modules, JCL, or many other types of content.

Table 2-1 summarizes the uses of the three main types of MVS data set organization.

Table 2-1 Typical uses of different data set organizations

Sequential	Partitioned	VSAM
Listings, tape files, print files, or any simple file not related to any other file.	Load module libraries, object libraries, macro libraries, program source libraries, text files, or any related set of sequential files.	Databases, files requiring anything other than sequential access.

Note: You cannot store load modules in a PDSE. See 5.9, "PDSE restrictions" on page 114, for more information. However, the program management binder, the replacement for the linkage editor introduced in DFSMS/MVS 1.1, writes executable code to PDSEs as program objects.

Many system data sets are also kept in PDSs, especially when they consist of many small, related files. For example, the definitions for ISPF panels are kept in PDSs. (ISPF is the IBM System/390® dialog manager.)

Programmers use ISPF to create and manipulate most PDSs. These data sets typically consist of source code for programs, text for manuals or help screens, or JCL to allocate data sets and run programs. The text for this guide was originally kept in a partitioned data set (a PDSE actually) with each chapter in a separate member.

2.2.3 Advantages of partitioned organization

Partitioned data sets offer a simple and efficient way to organize related groups of sequential files. This section describes the advantages enjoyed by users of PDSs. One of the main design objectives of PDSEs was to keep these advantages, so the following list applies to *both PDSs and PDSEs*:

- Grouping of related data sets under a single name makes MVS data management easier. Files stored as members of a PDS either can be processed individually or all the members can be processed as a unit.
- The space allocated for MVS data sets always starts at a track boundary on disk, so using a PDS is a way to store more than one small data set on a track. This saves disk space if you have many data sets that are much smaller than a track. A track is 56,664 bytes for 3390.
- Members of a PDS can be used as sequential data sets, and they can be concatenated to sequential data sets.
- PDSs can be concatenated to form large libraries.
- PDSs are easy to create with JCL or ISPF; they are easy to manipulate with ISPF utilities or TSO commands.

2.2.4 PDS customer requirements and limitations

PDSs are simple, flexible, and widely used. However, some aspects of the PDS design affect both performance and the efficient use of disk storage.

The requirements for partitioned data sets highlighted in this section are those expressed by IBM customers and user groups.

More efficient use of disk space

When a member in a PDS is replaced, the new data area is written to a new section within the storage allocated to the PDS. When a member is deleted, its pointer is deleted too, so there is no mechanism to reuse its space. This wasted space is often called *gas* and must be removed periodically by reorganizing the PDS (for example, by using the utility IEBCOPY to compress it).

A PDS that automatically reuses space without needing to be compressed would save both disk space and storage administration time. The SHARE study highlighted this as the most important requirement for new partitioned data sets.

Unlimited number of tracks

The number of tracks for PDSE on one volume is unlimited, which means more than 65535. This is the limit for a PDS.

Expandable directory size

The size of a PDS directory is set at allocation time. Figure 2-2 shows an example.

```
//DDCARD DD DSN=PDS.EXAMPLE,...,SPACE=(80,(5,5,20)),AVGREC=K,..
```

Figure 2-2 JCL sample to allocate PDS

This specifies a PDS with 400 KB (`80 x 5 x 1024`) as the primary space allocation and 20 directory blocks (each block 256 bytes long for all PDSs). As the data set grows, it can acquire more space in units of the amount you specified as its secondary space. In the example, this unit is also 400 KB. These extra units are called *secondary extents*.

However, you can store only a fixed number of member entries in the PDS directory because its size is fixed when the data set is allocated. If you need to store more entries than there is space for, you have to allocate a new PDS with more directory blocks and copy the members from the old data set into it. This means that when you allocate a PDS, you must calculate the amount of directory space you need.

A directory that grows dynamically as the data set expands would save time and effort for storage administrators and application programmers. They would not have to waste time calculating the space a directory needs or reallocating and copying PDSs with full directories.

Improved directory and member integrity

There is no mechanism to stop the PDS directory from being overwritten if a program mistakenly opens it for sequential output. If this happens, the directory is destroyed, and all the members are lost. An improved PDS would protect the directory from accidental changes.

Similarly, the data control block (DCB) attributes of a PDS can easily be changed by mistake. If you add a member whose DCB characteristics differ from those of the other members, you will change the DCB attributes of the entire PDS, and all the old members become unusable.

Also, with a PDS you can create only one member at a time. If a job tries to write two different members into a PDS at the same time, the members will overlay each other.

Better directory search time

As previously described, an entry in a PDS directory consists of a name and a pointer to the starting location of the member. Entries are stored in alphabetical order by member names. Inserting an entry near the front of a large directory can cause a large amount of I/O activity, as all of the entries behind the new one are moved along to make room for it. This is sometimes called a *ripple stow*.

Entries are also searched sequentially in alphabetical order. The search is done using a hardware search by key. While this is happening, the entire path to the device is in use on older hardware. (Newer hardware has a microcode performance feature known as PDS search assist.) If the directory is very large and the members small, it may take longer to search the directory than to retrieve the member after its starting location is found.

Figure 2-3 shows PDS structure.

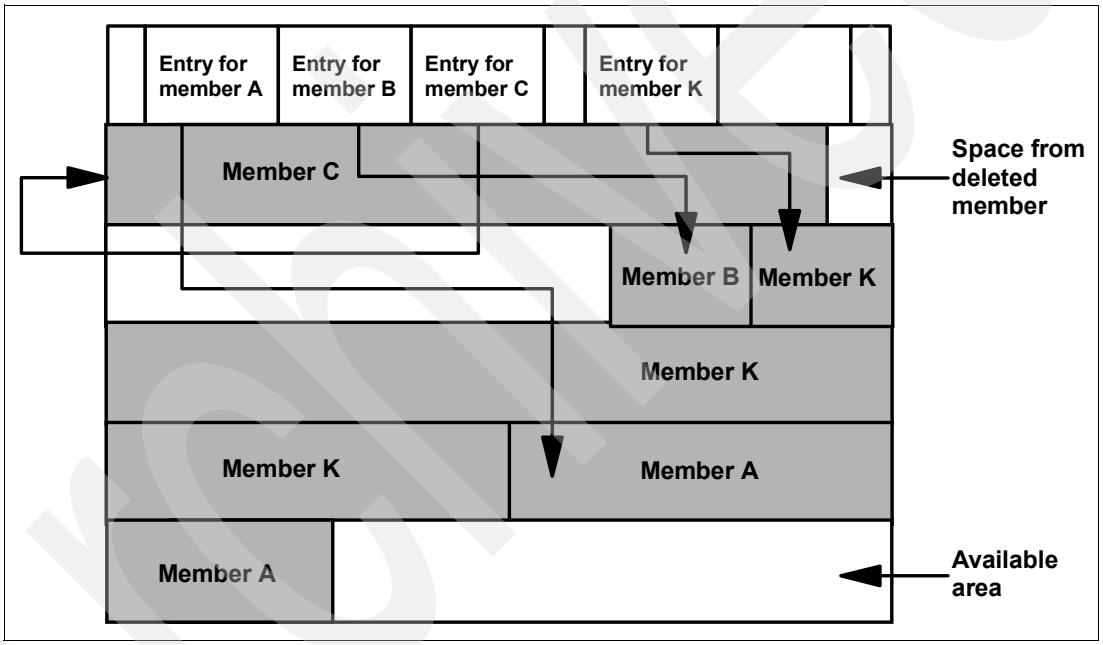


Figure 2-3 A partitioned data set PDS structure

The method of storing member entries and searching the directory must be improved to enable very large partitioned data sets to have good performance.

Improved device independence

The PDS addressing structure is closely tied to the physical structure of the disk storage device. If programs use this fact and perform calculations based on real addresses, a migration to a disk with a different track size and cylinder geometry would be difficult.

An addressing scheme not tied to any single type of physical device would make it easier to move data from one device to another.

Improved sharing facilities

To change a member of a PDS, you need exclusive access to the entire data set. This means that no other jobs can have access to the data set during an update. If the data set is always

accessed for input by a system task that runs 24 hours a day, there is a problem: If you want to update the data set, you must either shut down the system task or turn off the exclusive locking control for this data set. Neither option is acceptable to many installations.

A facility to control sharing of partitioned data sets at the member level is needed, so that if you access one member to update it, you do not lock all the other members in the data set.

Other requirements

Other requirements include member-level security access control and the ability to delete separate partitioned data set members with JCL.

The final requirement is to make all of these improvements to PDSs without losing compatibility with existing access methods and macro-level interfaces.

2.3 PDSE structure

This section first gives a brief overview of PDSEs, then summarizes how PDSEs meet the requirements for improvements to PDSs. It does not replace the PDSE sections in the manuals *z/OS V1R6.0 DFSMS/MVS Using Data Sets*, SC26-7410 and in *z/OS V1R1.0-V1R6.0 DFSMS/MVS Macro Instructions for Data Sets,* SC26-7408. These manuals contain full details about processing PDSEs.

2.3.1 What is a PDSE?

A PDSE is a *partitioned data set extended*. It consists of a directory and zero or more members, just like a PDS. It can be created with JCL, TSO/E, and ISPF, just like a PDS, and can be processed with the same access methods.

The directory can expand automatically as needed, up to the addressing limit of 522,236 members. It also has an index, which provides a fast search for member names.

Space from deleted or moved members is automatically reused for new members, so you do not have to compress a PDSE to remove wasted space. Each member of a PDSE can have up to 15,728,639 records.

A PDSE can have a maximum of 123 extents, but it cannot extend beyond one volume. PDSEs were first available with DFP Version 3.2 with DFDSS 2.5 and DFHSM 2.5.

When a directory of a PDSE is in use, it is kept in processor storage in a z/OS data space (SYSBMFDS). If a PDSE member needs fast access, it can be kept in processor storage in a z/OS Hiperspace™ (SYSBMFHS). See 9.5, “PDSE use of processor storage” on page 188, for further details.

PDSEs can be used in place of nearly all PDSs that are used to store data. But PDSE format is not intended as a PDS replacement. A PDSE cannot be used to store load modules, but it can contain program objects. You can convert load modules into program objects using IEBCOPY. For details refer to the section “Using Utilities for Program Management,” in *z/OS: MVS Program Management: User's Guide and Reference*, SA22-7643.

2.3.2 Improvements provided by PDSE

This section shows how PDSE meets the requirements listed in 2.2.4, “PDS customer requirements and limitations” on page 11.

Desirable features retained from PDS

One of the main requirements of a PDSE was to retain the desirable features of PDSs. The following features, as stated in 2.2.3, “Advantages of partitioned organization” on page 10, are retained:

- You can handle a set of related files as a unit.
- PDSE lets you store more than one file on a track. You can store up to 12 members on one 3390 track, as each member is allocated to one or more fixed blocks (pages) of 4 KB each. However, you may be able to store more members per track in a PDS, depending on the block size.
- You can access the members as sequential files and concatenate them to sequential data sets.
- You can create a new member easily with no overhead for allocation, OPEN, or CLOSE overhead.

Space reclaimed without compressing

A PDSE automatically reuses space, without needing an IEBCOPY compress. This saves both disk space and storage administration time. A list of available space is kept in the directory, and a space reuse algorithm chooses where to store a new member. The algorithm aims to strike a balance between keeping each member contiguous and reusing space as efficiently as possible, so the member might not be stored in contiguous pages.

A PDSE can become fragmented depending on the access pattern. This does not normally occur when the addition and deletion of members is balanced, but it might occur when members are deleted and new members are not added to reclaim that space. To reclaim the space and reorganize the PDSE, copy it into a new PDSE using IEBCOPY and DFSMSdss™ COPY. For further details, see 5.1.5, “Frequency of data set compression” on page 81.

Directory improvements

The directory can grow dynamically as the data set expands. When the pages at the front of the data set are used up, an extra directory page is put in the first available slot in the allocated space. If no space is left in the current extents, a new secondary extent is added.

The PDSE directory is an indexed structure, which improves search performance and the time taken to add a member entry. The PDSE directory cannot be overwritten by opening it for sequential output. See 2.5.3, “Directory structure” on page 22 for more information regarding the PDSE directory.

Member integrity improvements

If you try to add a member to a PDSE with DCB characteristics that differ from the rest of the members, you will receive an error. This prevents the DCB for the entire partitioned data set from becoming overlaid and all members from becoming unusable. However, because all PDSEs are assumed to be reblockable, different applications may open a PDSE with different block sizes. For fixed-length records, the block size can be any multiple of the record length up to the maximum block size, 32760 bytes. For variable-length records, the block size can be any value greater than the record size by at least 4 bytes.

The same job step, or separate jobs, may create more than one PDSE member at the same time.

Improved device independence

The PDSE addressing scheme is not tied to any single type of physical device. The members and records are located with simulated 3-byte track-and-record (TTR) addresses.

Improved sharing facilities

You can open a PDSE member for output or update without locking out other users from the entire data set. The sharing control is at the member level, not the data set level. See Chapter 8, “PDSE sharing and serialization” on page 153 for details.

Caching control

DFSMSdfp™ provides automated, system-level member caching to improve PDSE performance under the control of storage class attributes. Some facilities are available to improve PDSE performance by using data in memory techniques; however, they require that I/O statistics be analyzed and that VLF and LLA be set up and maintained. These facilities are described in 9.4.2, “Caching with LLA and VLF” on page 185.

2.4 Logical views of PDSEs

A central design objective for PDSEs was to make PDSE access as transparent as possible for existing PDS users. Although PDSEs are implemented in a very different way from PDSs, their external interfaces are similar.

This section reviews what different types of users will see on the screen, in the DFSMS configuration, or in their own programs when using PDSEs instead of PDSs. Chapter 5, “Using PDSEs” on page 79 offers details about using these functions.

2.4.1 For the user

If you are a typical TSO/E user, the only visible differences between a PDSE and a PDS are some fields on the ISPF enhanced allocation and data set information panels. If you use JCL you may use the data set name type keyword (`DSNTYPE`) on data set allocation statements. DSNTYPE must be set to `LIBRARY` for a PDSE; `PDS` or blank for a PDS. If you use the ISMF data set application, notice that there is a PDSE Indicator column and that some of the space information fields are blank for PDSEs.

PDSEs are simpler to use than PDSs. Because PDSEs reuse deleted space, you do not need to compress them to get rid of gas. See 5.1.4, “PDS gas and PDSE unused space” on page 80 for more information. Because the directory can expand as needed, you should not get directory-out-of-space errors when you add new members.

The significance of the block size keyword (`BLKSIZE`) has changed slightly. All PDSEs are stored on disk as fixed 4 KB blocks. These 4 KB physical blocks are also known as pages. The PDSE is logically reblocked to the block size you specify when you open the data set. The block size does not affect how efficiently a PDSE is stored on disk, but it can influence how efficiently the system reads from it and writes to it because it affects the size of the storage buffers allocated for a PDSE. You should allow the DFSMSdfp System-Determined Blocksize function to calculate the best block size for your data set, by not specifying any value for it.

Note: Applications that depend on block boundaries will not work with a PDSE.

2.4.2 For the storage administrator

As the storage administrator, you will see PDSEs in three places.

Data class

A field called DATA SET NAME TYPE in the data class definition corresponds to the JCL keyword DSNTYPE. You may want to provide some data classes that specify `DATA SET NAME TYPE ===> LIBRARY` and some that specify `DATA SET NAME TYPE ===> PDS or blank`.

Storage class

You will also see changes in the storage class definitions that support PDSEs. First, a field called *Guaranteed Synchronous Write (GSW)* has been included. Storage classes with this attribute can be used on an exception basis for PDSEs with high data integrity needs. GSW ensures that updates to a PDSE are written to disk immediately. This only applies if the PDSE is open for update.

Moreover, the use of performance objective specifications in the storage class has been extended. These performance objectives represent the millisecond response (MSR) that you would prefer for data sets assigned to a particular storage class.

The MSR value is used to determine whether a data set is a candidate for cache storage in a storage controller. This is known as *Dynamic Cache Management*. The MSR value also controls which PDSEs will use the PDSE Hiperspace. Therefore, you should set up storage classes with low MSR values for PDSEs that require very good performance.

You should update your ACS routines to select the new classes and to control which data sets are allocated as system-managed PDSEs. The SMS address space must be active for you to allocate managed PDSEs. It need not be active for you to use PDSEs or to allocate non-managed PDSEs.

For a complete description of SMS environment, SMS classes, and ACS routines, refer to 3.8, “SMS considerations” on page 49.

2.4.3 For the system programmer

For the system programmer and others writing assembler programs that process PDSs, the logical view of PDSE is the same as PDS, if you write “well-behaved” code. Unlike PDS, PDSE is not dependent on the physical characteristics of the disk on which it is stored. Although all of the common access method macros that you use with PDSs are mapped to PDSEs and work as they do with PDSs, the functions that depend on the physical characteristics of a device work differently.

DS1LSTAR

DS1LSTAR is a field in the Format 1 DSCB for a data set. It contains the last-used track and block for a PDS. It is always zero for a PDSE.

Note: Instead of using the DS1LSTAR field, PDSE services use an internal field called the *high formatted relative frame number* (HFRFN) to track the highest formatted page within a PDSE. For more information see “High formatted relative frame number” on page 21.

Assembler macros for PDSEs

The only data access assembler macros supported for PDSEs are those provided by the three standard PDS access methods:

- Basic Partitioned Access Method (BPAM)
- Basic Sequential Access Method (BSAM)
- Queued Sequential Access Method (QSAM)

You cannot process PDSEs directly with EXCP, EXCPVR, or XDAP.

For more information, refer to 5.4, “Accessing PDSEs” on page 99. For full details about using assembler macros with PDSEs, refer to *z/OS V1R3.0-V1R6.0 DFSMS Macro Instructions for Data Sets*, SC26-7408.

Connections

In PDSE terms, we are concerned more often about connections to a PDSE directory, and about connections to the individual members within a PDSE data set, than about OPENs. A connection is represented by internal data control structures maintained in virtual storage. The number of connections may be important because of the space needed for these control structures.

A connection to a PDSE *directory* is established when a program opens a PDSE data set. The connection is dropped when the program closes the data set.

A connection to a PDSE *member* provides a temporary version of that member. The connection enables the member to remain available to applications that were accessing the member.

Connections to a member within a PDSE are established by using these ways of accessing a member:

- JCL using DSNAME=libname(memname). This connection occurs at OPEN.
- OPEN for output without a member name, where a connection is made to a member that does not yet have a name.
- BLDL
- DESERV FUNC=GET
- DESERV FUNC=GET_ALL
- FIND by name
- FIND by TTR
- POINT

The connections are released if the PDSE is closed. The STOW or DESERV FUNC=RELEASE macro can also be used to disconnect members of a PDSE.

See 5.4, “Accessing PDSEs” on page 99, for more information about the system macros and their usage, and 9.5.3, “Pending delete considerations” on page 193 regarding deferred delete processing of PDSE members.

Deleted members and aliases

With a PDSE, when you delete a primary member name, all of that name’s aliases are deleted automatically. This is good news because now you do not have to remember to delete these aliases yourself. However, be aware that you can no longer use old aliases to retrieve members that have been deleted accidentally, which was possible with a PDS.

Programs that have a member open when another program deletes or replaces that member will keep a *connection* to the old version of the member until they close it. See 9.5.3, “Pending delete considerations” on page 193 for more information regarding deferred delete processing. When all connections to an old version of a member have been dropped, the space is marked for reuse and the old version of the member cannot be accessed at all.

Note: This does not apply when a member is open for update. If you open a PDSE member for update, you lock out all other users, so if you delete it, there can be no connections to the old data.

Member and record addressing

The PDSE directory is indexed, permitting more direct searches for members. Hardware-defined keys are not used to search for members. Instead, the name and the relative track address of a member are used as keys to search for members. The TTRs in the directory can change if you move the PDSE, because for PDSE members the TTRs are not relative track and record numbers but rather generated aliases for the PDSE members. These TTRs sometimes may be referred to as Member Locator Tokens (MLTs). A PDSE directory is limited to 522,236 members. The PDSE directory is expandable; you can keep adding entries up to the maximum number of members or until the data set runs out of space. The system uses the space it needs for the directory entries from storage available to the data set.

TTR token X'000001' points to the directory. TTR tokens in the range X'000002' to X'07FFFF' are used to point to the first record of each member, which is why there is a limit of 522,236 members. TTR tokens from X'100001' to X'FFFFFF' point to records within each member. This is why there is a limit of 15,728,639 records in each member.

Some points to note about these TTR addresses:

- The addresses from X'080000' to X'100000' are reserved.
- The records within each member are numbered from X'100001' upward, so to point to a record correctly, a program must point first to the member, then to the record.
- You cannot perform track calculations on these addresses. You can use the TRKCALC macro, but it will give you unusable results.

Updating the directory

The STOW macro updates a PDSE directory by adding, changing, replacing, or deleting an entry in the directory. For BSAM or QSAM access, after PDSE members are written or updated, the CLOSE macro synchronizes member data to disk by performing an automatic STOW.

Notes:

- An OPEN for OUTPUT creates a “new” member (a new file sequence number, as defined on page 22) within the in-storage control structures. This “new” member does not have any connection to a member name and therefore is not visible to any other task until a STOW is performed.
- The STOW updates the directory (the attribute directory), establishes the relation between the file sequence number and the member name (the name directory), and synchronizes the directory changes to disk.

2.5 PDSE physical components

This section introduces aspects of the physical implementation of PDSEs that are visible to the end user.

2.5.1 PDSE address spaces

The two address spaces below are used for PDSE server processing:

- *SMSPDSE* is a non-restartable address space that is intended to handle global requests such as loading from LNKLST.
- *SMSPDSE1* is restartable through operator commands and is intended to handle non-global processing.

The PDSE subcomponents are modified as follows:

- The majority of PDSE processing is now performed in the PDSE address spaces.
- The majority of PDSE control blocks is moved from ECSA into the PDSE address spaces.
- Recovery is improved, particularly EOM processing, to ensure that resources are released and structures are left in valid states.

2.5.2 Space usage on disk

All PDSE data sets are stored on disk volumes in fixed 4 KB blocks (4,096 bytes). These 4 KB physical blocks are also known as *pages* or *frames*. You cannot change the physical block size. However, the block size specified when you allocate a PDSE is the logical block. The logical block size affects the size of buffers used within application programs.

A page containing PDSE member data contains data from only one member. PDSE members do not share pages.

See 5.1, “PDSE space usage” on page 80 for more information about the space usage within PDSE data sets. Table 5-3 on page 87 summarizes the maximum capacity for a PDSE data set for track or cylinder allocations for both 3380 and 3390 format volumes.

Structure of a new PDSE

A PDSE always occupies at least five pages of storage. These five directory pages are created at the same time as the data set.

Figure 2-4 on page 20 illustrates the structure of a newly allocated PDSE data set on a 3390 volume.

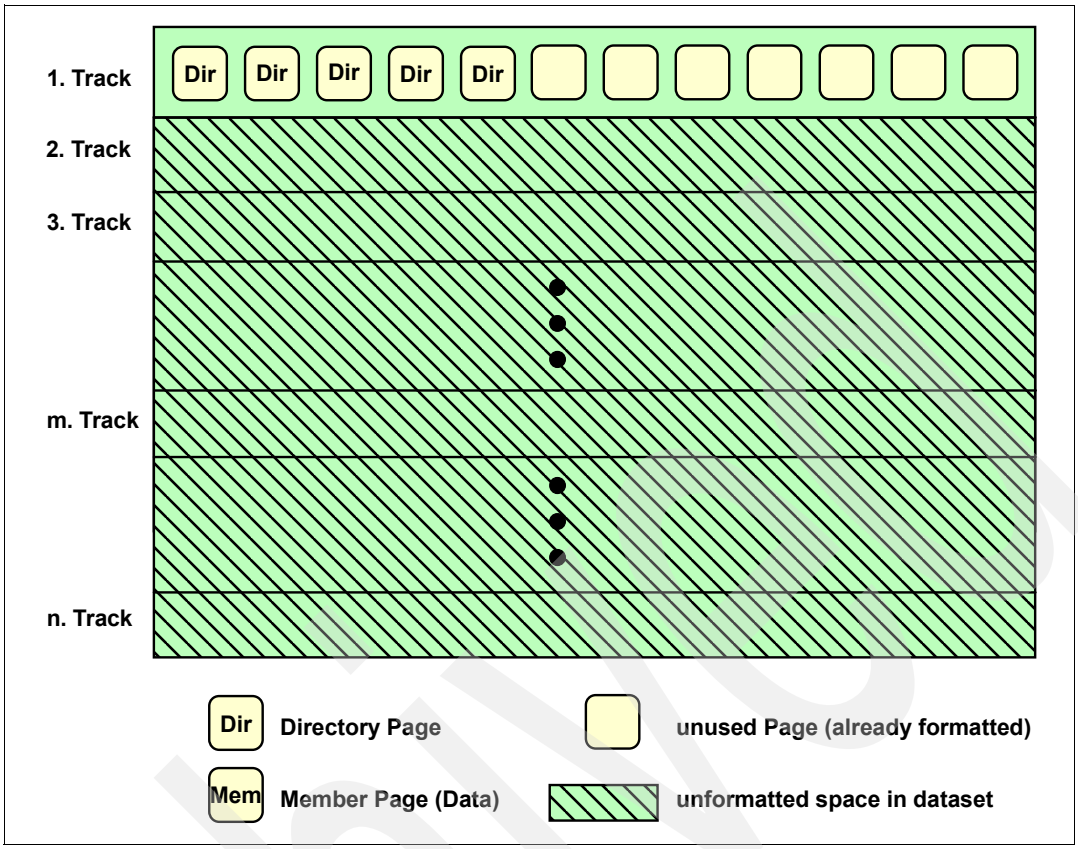


Figure 2-4 Structure of a new PDSE

Reuse of space

When a PDSE member is updated or replaced, it is written in the first available space. This is either at the end of the data set or in a space in the middle of the data set marked for reuse. This space does not have to be contiguous. The space reuse algorithm tries to balance two objectives: not to extend the data set unnecessarily, and not to fragment members unnecessarily.

With the exception of an OPEN for UPDATE, a member is never immediately written back to its original space. The old data in this space is available to programs that had a connection to the member before it was rewritten. The space is marked for reuse only when all connections to the old data are dropped. However, after they are dropped, there are no pointers to the old data, so no program can access it.

Figure 2-5 on page 21 shows the structure of a PDSE after it has had several members added and deleted. We assume that it is stored on a 3390 volume.

- For a 3390 volume, 12 pages (4 KB blocks) fit on one track.
- Directory pages, data pages, and free pages are within the same extent.
- Directory blocks are added dynamically as the number of members increases.
- Unformatted space is formatted automatically into pages when they are needed.
- Unused pages from deleted members and freed directory pages are dynamically reused.

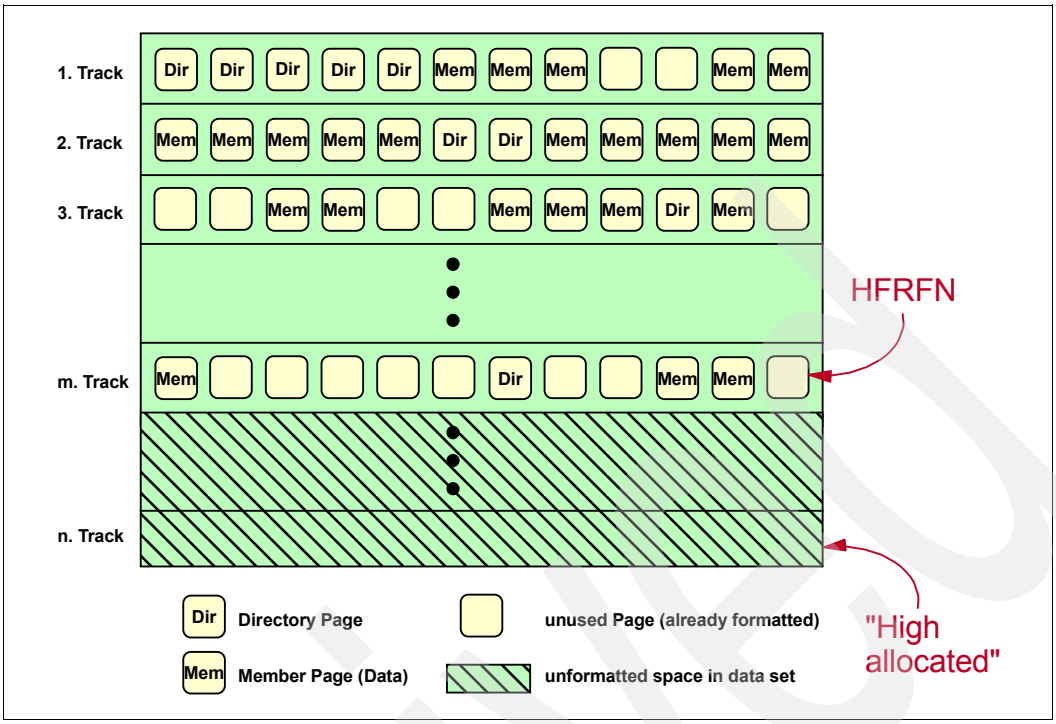


Figure 2-5 Structure of a used PDSE

The space used on a volume is formatted only as the space is needed. This applies to the space for newly allocated PDSE data sets as well as the space for new extents.

You can get an EXTMAP of your PDSE using PHATOOL. Refer to 10.6.5, “PDSE and HFS Analysis Tool (PHATOOL)” on page 306.

High formatted relative frame number

The number of formatted pages is stored inside the PDSE. This value is referred to as the *high formatted relative frame number* (HFRFN). Figure 2-5 also illustrates the HFRFN. The HFRFN is handled as a high-water mark. This means that the HFRFN is not decreased by deleting members, because it represents the highest formatted page number, not the number of used pages.

Important: The HFRFN reflects the high-water mark of formatted pages for a PDSE and not the number of used pages inside a PDSE data set.

It is not decreased if members are deleted.

Backup and restore processing by DFSMSdss and DFSMShsm, and DFSMSdfp space management functions such as partial release operate with the HFRFN as the limit. Backup and restore normally brings back a data set with an allocation sufficient to hold pages up to the HFRFN. Partial release will not release space below the HFRFN.

IEBCOPY can be used to reset the HFRFN by copying the PDSE into a new PDSE.

Refer to Chapter 6, “Backup and recovery of PDSEs” on page 117 for detailed information regarding the storage management provided by DFSMSdss, DFSMShsm, and IEBCOPY, and 5.1.9, “Partial release, free space” on page 82 for more details about partial release.

2.5.3 Directory structure

Logically, a PDSE directory looks the same as a PDS directory: It consists of a series of directory records in a block. Physically, it is a set of *directory pages* at the front of the data set, plus additional directory pages interleaved with member pages. Five directory pages are created initially at the same time as the data set. New directory pages are added, interleaved with the member pages, as new directory entries are required. A PDSE always occupies at least five pages of storage.

Figure 2-6 provides an overview of the internal directory structure of a PDSE.

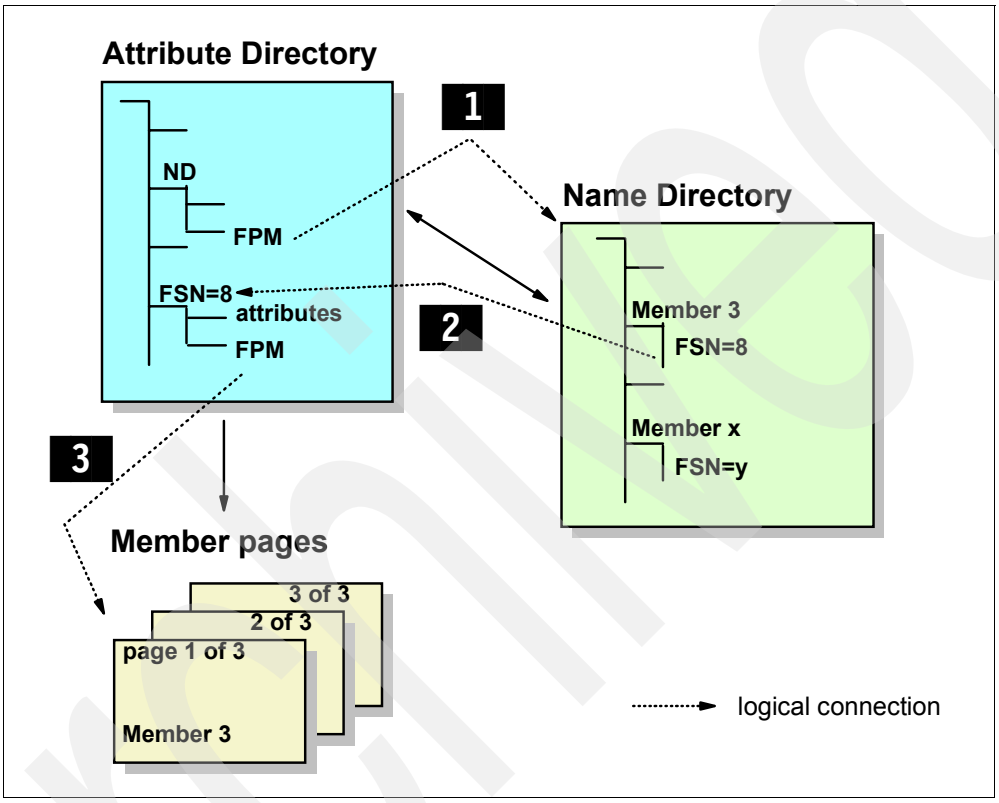


Figure 2-6 PDSE index structure

This structure is maintained internally by allocating pages for different purposes:

- Attribute directory (AD) pages

  The *attribute directory* is an internal hierarchical structure, implemented as a B-tree (a balanced tree, where balancing minimizes the tree depth and speeds up searching), which contains attribute information about individual members and attributes of the PDSE itself.

  PDSE members are tracked internally by a *file sequence number* (FSN) — 2 in Figure 2-6. The name directory (see below) provides the mapping between the member and the FSN.

  The AD also provides the connection to the volume's individual data pages that represent the members. These connections are represented by a map called a *fragment parcel map* (FPM) — see 3 in Figure 2-6.

- Name directory (ND) pages

  The *name directory* pages represent the external members of a PDSE. The ND is also implemented in a B-tree structure. The ND provides the connection between the individual member names and the *file sequence number* (FSN).

  A PDSE data set has only one ND and that is anchored at the AD — see **1** in Figure 2-6 on page 22.

- Data pages

  The data pages contain the data in the individual members in a PDSE. A member may contain data or executable programs (program objects).

The attribute directory and name directory information is also referred to as *metadata*, meaning data about data. Refer to Figure 2-7.

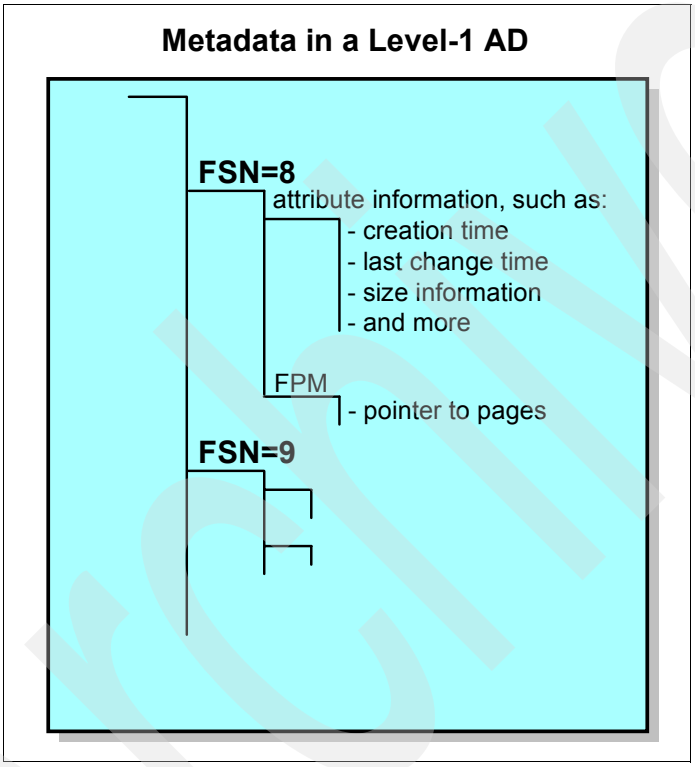


Figure 2-7 Metadata in an AD sequence set page

The attribute directory and the name directory are index structured.

As members are added to a PDSE, the first-level page in an attribute directory or a name directory eventually becomes full. When this happens, another level is added to the directory. The second-level page is now the root page and it contains pointers to the first-level page. In a similar way, if a second-level index becomes full, yet another level is added, and the directory now has three levels.

Only the first-level (lowest-level) pages contain metadata. The second-level and higher pages contain pointers to pages at lower levels. The first-level pages are also known as the *sequence set pages*. The higher-level pages are known as the *index set pages*. The highest-level page is called the *root page of the index*.

For example, as shown in Figure 2-8, for a second-level AD structure, note that the root page contains no metadata, only pointers to the first-level (lowest-level) pages.

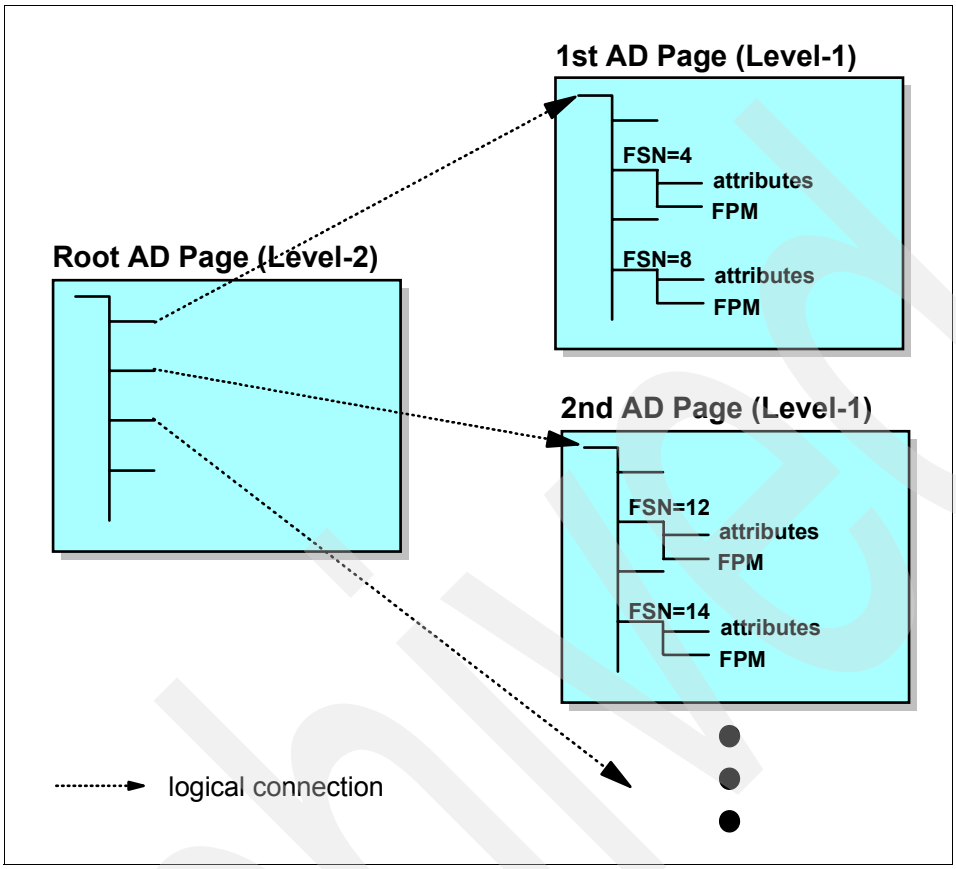


Figure 2-8 Second-level AD index overview

The recently used index pages are stored in the SYSBMFDS data space, which is owned by the SMSPDSE address space. Control blocks are also created in the PDSE address space (common storage) that reflect the index structure in the virtual storage. So, we have two index structures, one in-storage version and the other on disk.

An in-storage index structure exists on any system that is connected to the PDSE, as shown in Figure 2-9. Obviously, these structures must be synchronized among all instances to prevent index damage. DFSMSdfp provides a number of services (such as locking services and notification services) to ensure the integrity of a PDSE index.

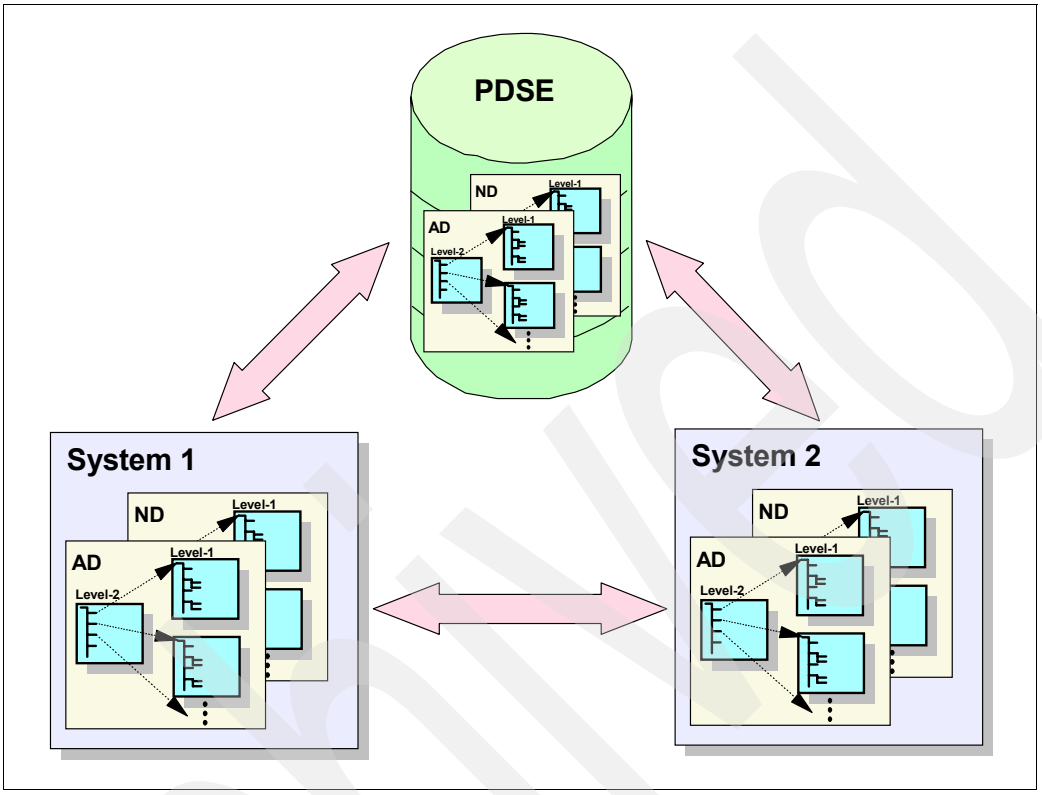


Figure 2-9 Index structures in a multisystem environment

An internal routine performs consistency checks to ensure that a level-2 page refers to the corresponding level-1 page. If there is an inconsistency, an abend is issued with a reason code identifying index damage.

An inconsistency may be caused by sharing across GRSPLEXes (which is incorrectly implemented sharing), as shown in Figure 2-10. An index can be damaged if one system makes an update to the index without notifying the other systems about the changes.

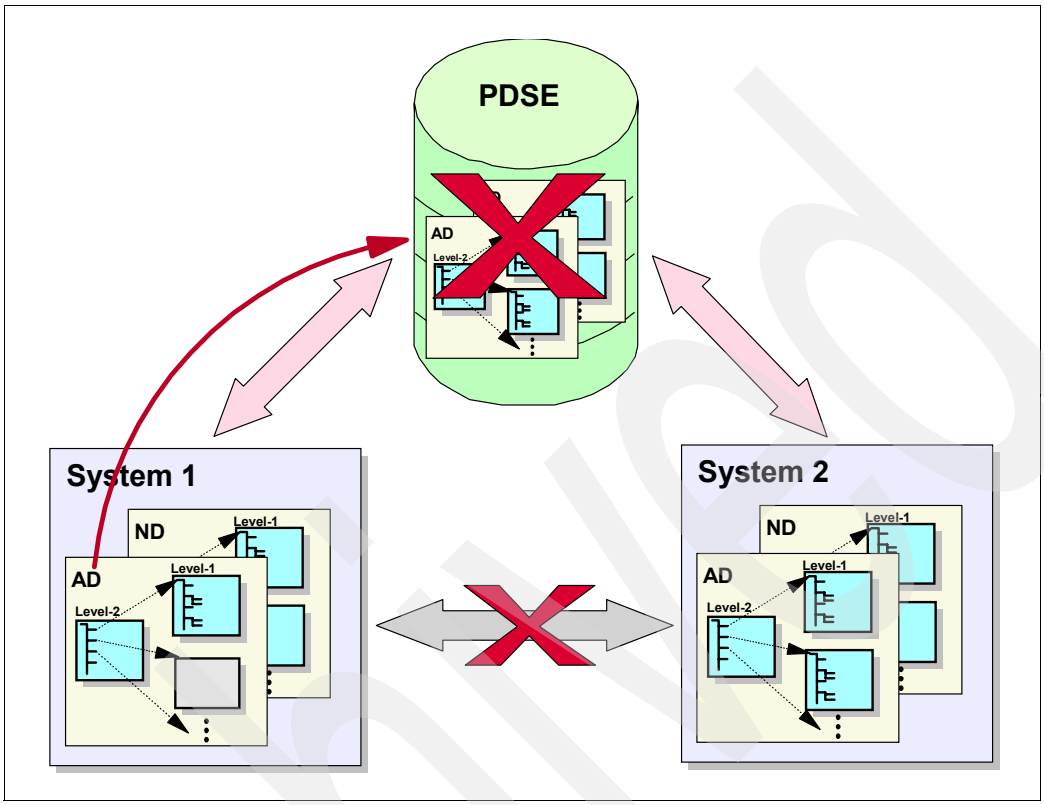


Figure 2-10 Index consistency in a multisystem environment

The space in a PDSE that is used for index information is obtained from the same storage (extents on a disk volume) that is available to the files themselves (data pages). This means that data pages and index pages are combined into the same extents on a disk volume.

Pages from deleted members (data pages) or index pages are available for reuse after the last connection to the member has been dropped.

2.6 Summary of PDSE characteristics and limits

Table 2-2 summarizes the characteristics of PDSEs and PDSs. It notes several architectural limits for PDSEs that cannot be reached because a PDSE must be on a single volume.

Table 2-2 PDSE limits

Attribute	PDSE value	PDS value
Number of tracks on one volume	Unlimited (greater than 65535)	65535
Number of volumes	One: multivolume PDSEs not supported	One
Number of extents	123	16
Maximum architected size	16 Terabytes	Limited by number of tracks
Number of members	522,236	No architectural limit - but limited by 65,535 tracks
Size of members	15,728,639 records	No architectural limit, but limited by 65,535 tracks

2.7 PDSE storage management concepts

To exploit storage management for PDSE data sets, use components such as *LLA, VLF, Hiperspaces,* and *data spaces* (Figure 2-11). This section gives an overview of these components. (Also see Chapter 9, “Performance considerations” on page 179.)

Information about implementing these components for PDSEs is in Chapter 3, “Managing the PDSE environment” on page 33.

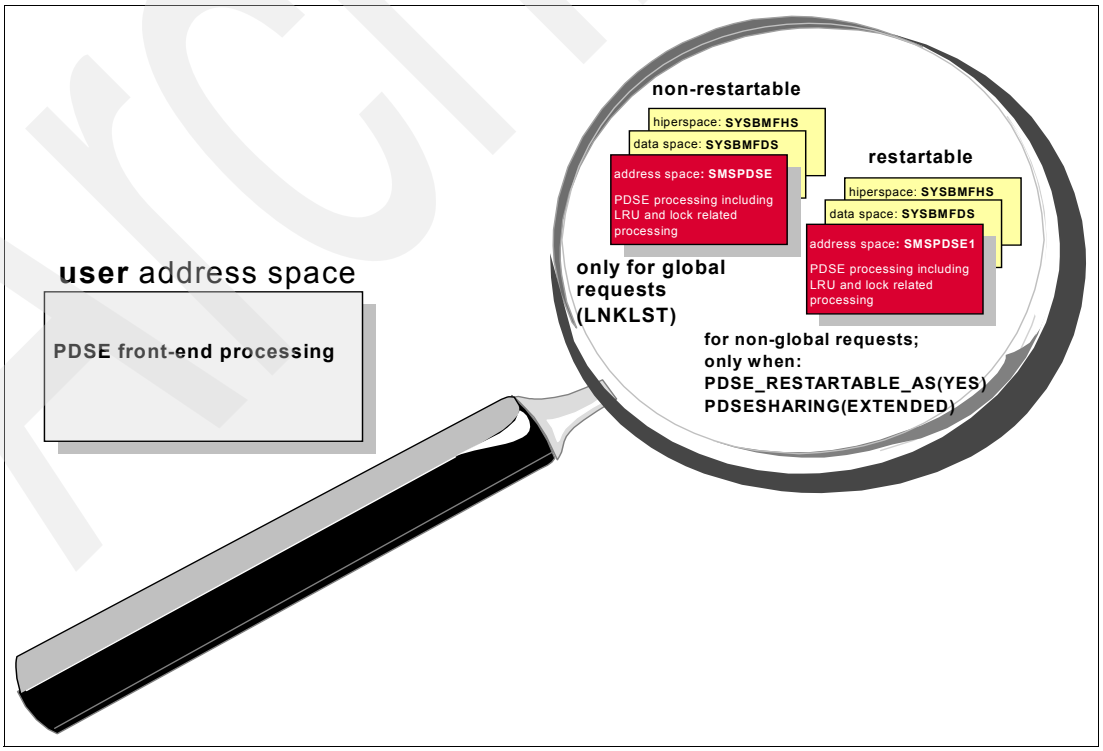


Figure 2-11 Focus on PDSE system address space

2.7.1 Data space and Hiperspace concepts

Data space (SYSBMFDS)

A data space is an area of contiguous storage backed by real, expanded, or auxiliary storage, whichever is necessary as determined by the system. One data space is related to each address space. For PDSE, as mentioned in Figure 2-11 on page 27, there is one data space for SMSPDSE and one for SMSPDSE1.

A data space is a type of virtual storage address space with a range up to 2 GB of contiguous virtual storage. The virtual storage map of a data space is quite different from a normal address space: Except for the first 4K, the entire 2 GB is available for user data. A data space can hold only data; it does not contain programs in execution. Program code does not execute in a data space, although a program can reside in a data space as data. A program can refer to data in a data space at the byte level, as it does in a work file. A program references data in a data space directly, in much the same way that it references data in an address space. It addresses the data by the byte, manipulating, comparing, and performing arithmetic operations. The program uses the same instructions (such as load, compare, add, and move character) that it would use to access data in its own address space. Before accessing data in a data space use special assembler instructions to change its access mode. The data space for PDSE is named SYSBMFDS.

Figure 2-12 shows a data space structure.

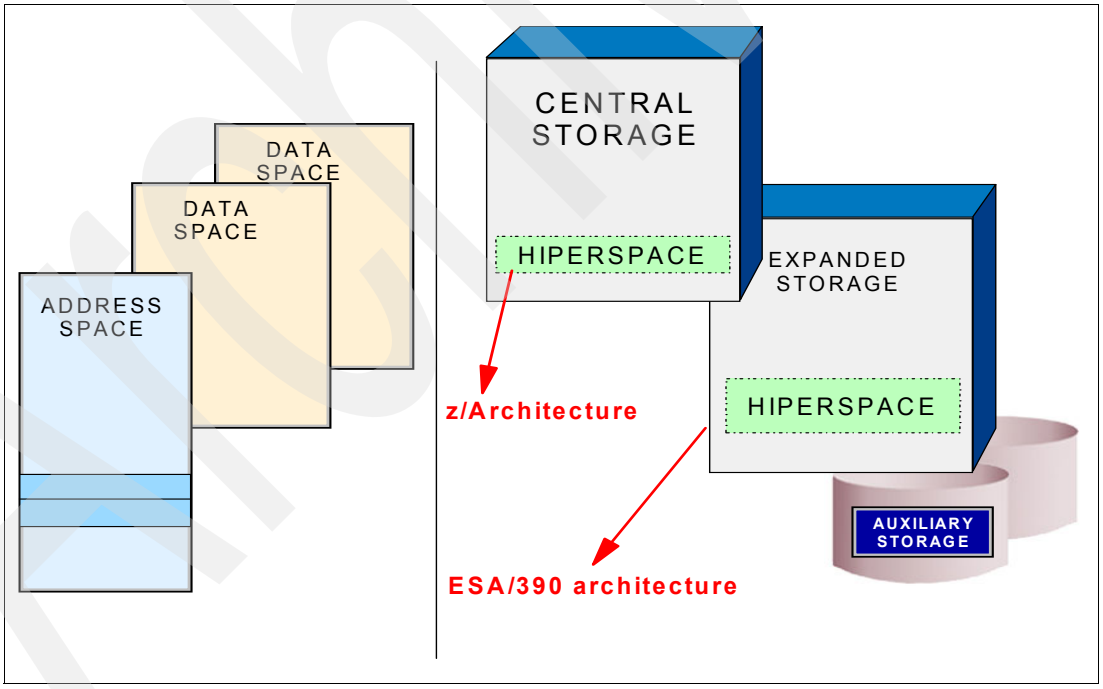


Figure 2-12 Data space management

Hiperspace (SYSBMFHS)

A Hiperspace is a high-performance virtual storage space of up to 2 GB. A high-performance data access Hiperspace is a kind of data space created with the same RSM services used to create a data space. Unlike an address space, a Hiperspace contains only user data and does not contain system control blocks or common areas. The data in a Hiperspace cannot be referenced directly but must be moved to and from real storage.

Hiperspaces are used by VSAM and by PDSE. *SYSBMFHS* is the name of the Hiperspace dedicated to PDSE. Hiperspace provides an opportunity to the applications to use expanded storage as a substitute to I/O operations. With z/OS architecture a Hiperspace is in real storage.

Programs can use data spaces and Hiperspaces to:

- Obtain more virtual storage than a single address space gives a user.
- Isolate data from other tasks in the address space. Data in an address space is accessible to all programs executing in that address space. You might want to move some data to data spaces or Hiperspaces for security or integrity reasons. You can restrict access to data in those spaces to one or more units of work.
- Share data among programs that are executing in the same address space or different address spaces. Instead of keeping the shared data in common areas, create a data space or Hiperspace for the data you want your programs to share. Use this space as a way to separate your data logically by its own particular use.
- Provide an area in which to map a data-in-virtual object.

2.7.2 LLA and VLF concepts

The library lookaside (LLA) facility uses a virtual lookaside facility (VLF) data space to hold the most active modules of linklist and user-specified program libraries. When an address space requests an LLA-managed program that is in the data space, the load module is retrieved from VLF instead of from the program library on DASD. LLA and VLF functions apply to PDS and PDSE objects.

LLA

LLA provides services that improve system performance by reducing the amount of I/O needed to locate and fetch modules from DASD storage.

LLA services are in the LLA address space. They improve module fetch performance as follows: LLA maintains copies of the library directories in the LLA address space. To fetch a module, the system has to first search the directory for the module location. The system can quickly search the LLA copy of a directory in virtual storage instead of using costly I/O to search the directories on DASD. LLA places copies of selected modules in a virtual lookaside facility (VLF) data space (when the LLA class is defined to VLF). The system can quickly fetch modules from virtual storage rather than use slower I/O to fetch the modules from DASD. LLA determines which modules, if staged, would provide the most benefit to module fetch performance. LLA evaluates modules as candidates for staging based on statistics LLA collects about the members of the libraries it manages, such as module size, fetch count, and the time required to fetch a particular module.

You obtain the most benefit from LLA when both LLA and VLF are functioning, so you should plan to use both. Figure 2-13 on page 30 shows LLA structure.

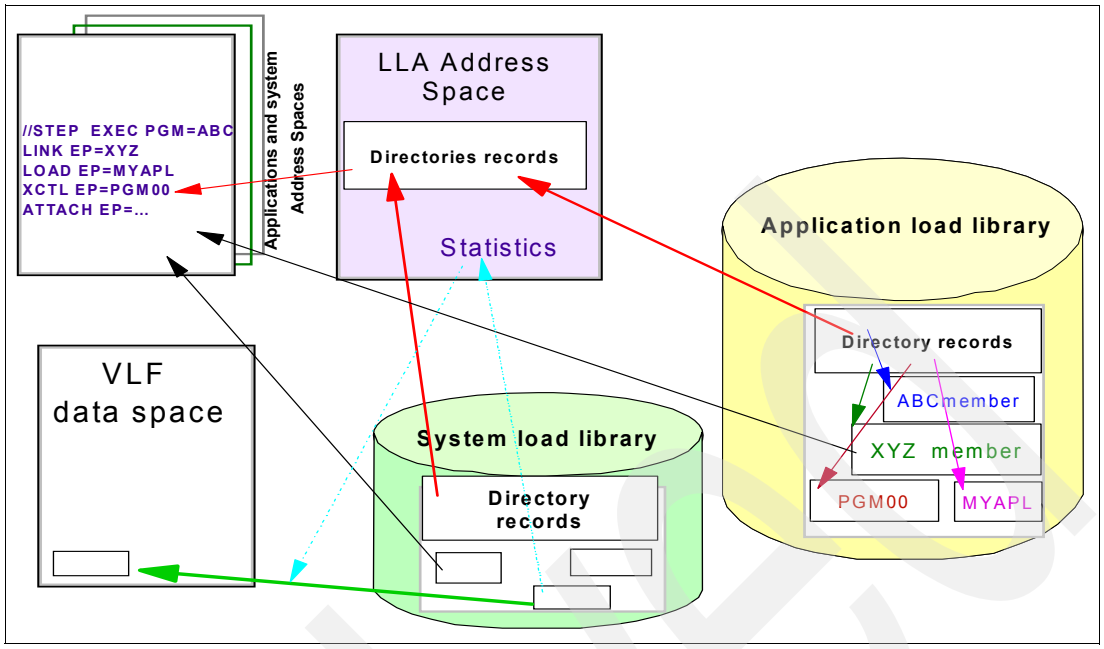


Figure 2-13 Library lookaside

VLF: Virtual lookaside facility (VLF) is a set of services that can improve the response time of applications that must retrieve a set of data for many users. VLF creates and manages a data space to store an application's most frequently used data. When the application makes a request for data, VLF checks its data space for the data. If the data is present, VLF can rapidly retrieve it without requesting I/O to DASD. To take advantage of VLF, an application must identify the data it needs. The data is known as a data object. Data objects should be small to moderate in size, named according to the VLF naming convention, and associated with an installation-defined class of data objects.

Certain IBM products or components such as LLA, TSO/E, CAS, and RACF use VLF as an alternate way to access data. Because VLF uses virtual storage for its data spaces, each installation must weigh performance considerations when planning for the resources required by VLF.

Note: VLF is intended for use with major applications. Because VLF runs as a started task that the operator can stop or cancel, it cannot take the place of any existing means of accessing data on DASD. Any application that uses VLF must also be able to run without it.

When you define the LLA class to VLF, and start VLF, the most active modules from LLA-managed libraries are staged into the DCSVLLA data space managed by VLF. You obtain the most benefit from LLA when both LLA and VLF are functioning, so you should plan to use both. Figure 2-14 on page 31 shows the VLF structure.

.

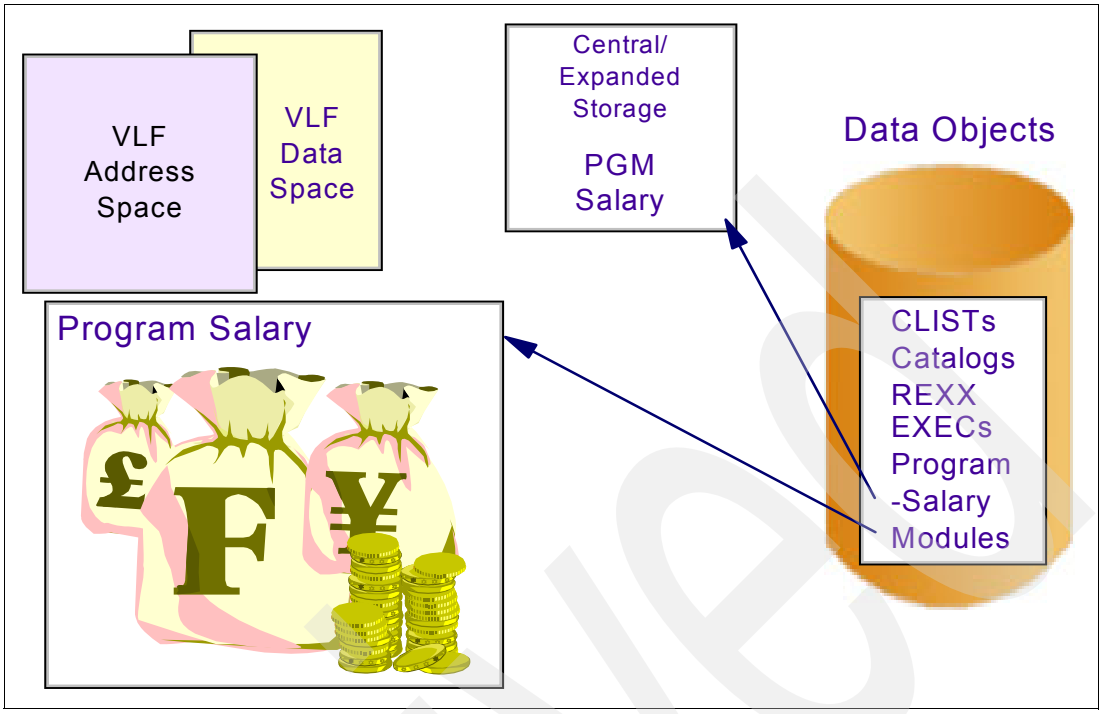


Figure 2-14 Virtual lookaside facility

3

Managing the PDSE environment

This chapter contains information that operators and system programmers need to know to manage the PDSE environment. The introduction of the restartable address space (SMSPDSE1) requires changes to initialization, but the end user does not have to take any specific actions. The information in this chapter is not required for the end user to use PDSEs. In the event of a suspected problem with a PDSE, end users should not attempt any recovery themselves.

Topics included in this chapter:

- PARMLIB requirements
- Restartable address space considerations
- Link Pack Area (LPA) considerations
- Library lookaside (LLA) considerations
- Virtual lookaside facility (VLF) considerations
- Hiperspace considerations
- SMS considerations
- JES2 PROCLIB considerations

For more information see “Initializing SMS through the IGDSMSxx member” in the *z/OS DFSMSdfp Storage Administration Reference,* SC26-7402 and *z/OS MVS Initialization and Tuning Reference,* SA22-7592.

3.1 PARMLIB requirements

The PDSE function is a component of DFSMS, so its control parameters are specified along with the other DFSMS parameters in member IGDSMSxx.

For easier distinction between the parameters intended for management of the PDSE environments, the parameters for the restartable address space have a prefix of PDSE1. Some existing parameters that apply to either address space have been given synonyms using the prefix PDSE for the non-restartable address space. The old form for these parameters is still accepted.

3.1.1 IGDSMSxx PARMLIB global setting affecting data set allocation

The default specification for allocating a PDS, PDSE, or HFS data set is PDS, which retains compatibility with previous operating system levels. It is possible to change the systemwide default from PDS.

DSNTYPE(PDS | PDSE | HFS)

Specifies the installation default for data sets allocated with directory space but without a data set type specified. The options are PDS, PDSE, or HFS.

Default: PDS

We recommend that this parameter not be specified so that the default remains PDS. Use the SMS routines, the DSNTYPE keyword, or both to explicitly allocate the type of data set required.

3.1.2 PARMLIB required IGDSMSxx parameters for SMSPDSE1 support

PARMLIB changes are required to activate the second system address space, SMSPDSE1. The IGDSMSxx member active at IPL must have both PDSESHARING(EXTENDED) and PDSE_RESTARTABLE_AS(YES) specified. The defaults for these options are NORMAL and NO, respectively. If either of these default values is specified or implied, then only the non-restartable address space, SMSPDSE, will be initiated and activated.

PDSESHARING(EXTENDED|NORMAL)

Specify PDSESHARING(EXTENDED)

Default: NORMAL

PDSE_RESTARTABLE_AS(YES|NO)

Specify PDSE_RESTARTABLE_AS(YES)

Default: NO

Note: You can update the IGDSMSxx member with PDSE_RESTARTABLE_AS(YES) then issue the SET SMS=xx command to reinitialize SMS. If the v sms,pdse1,activate command is issued, this may start the SMSPDSE1 address space. Starting SMSPDSE1 in this way is not supported and if the address space is started this way, without an IPL, PDSE problems may be encountered.

3.1.3 PARMLIB optional IGDSMSxx changes for SMSPDSE support

The existing IGDSMSxx parameters are enhanced with synonym parameters for the global PDSE address space (original SMSPDSE address space). Although the old form of the parameters is still accepted, we recommend that the new forms be adopted when a system has been converted to z/OS V1R6. The D SMS,OPTIONS command in z/OS V1R6 displays only the new form.

PDSE_LRUCYCLES(nnn | 240)

This specifies the maximum number of times the BMF LRU routine allows unused buffers to remain unavailable for reuse. The allowable range is from 5 to 240. This sets the maximum, and BMF dynamically adjusts the actual number of LRU cycles before inactive buffers are reused.

PDSE_LRUCYCLES is related to PDSE_LRUTIME. A change to PDSE_LRUCYCLES will take effect the next time the LRU routine runs. Most installations should use the default value of 240. This value can be tuned in the case of very high data rates. SMF type 42 subtype 1 records should be monitored to see the caching activity in the BMF dataspace.

Default: 240

This option was LRUCYCLES prior to z/OS V1R6. LRUCYCLES is still accepted but only applies to the global SMSPDSE address space.

Note: APAR OA10091 should be applied to all z/OS V1R6 systems to fix a PDSE SMS recording error.

The SETSMS command can be used to change the value of PDSE_LRUCYCLES dynamically. There is no specific response to the SETSMS PDSE_LRUCYCLES(nnn) command. The D SMS,OPTIONS command can be used to verify that the change was made.

Figure 3-1 shows the SETSMS PDSE_LRUCYCLES(239) command and part of the output from the D SMS,OPTIONS command showing that the value has been set to 239.

Refer to 3.7, “Hiperspace considerations” on page 47 for information about setting the PDSE_LRUCYCLES parameter.

```
SETSMS PDSE_LRUCYCLES(239)
D SMS,OPTIONS
IGD002I 17:47:59 DISPLAY SMS 193
ACDS     = SYS1.SMS.ACDS
COMMDS   = SYS1.SMS.COMMDS
INTERVAL = 15            DINTERVAL = 150
SMF_TIME = YES           CACHETIME = 3600
CF_TIME = 1800           PDSE_RESTARTABLE_AS = YES
PDSE_BMFTIME = 3600      PDSE1_BMFTIME = 3600
PDSE_LRUTIME = 15        PDSE1_LRUTIME = 15
PDSE_LRUCYCLES = 239     PDSE1_LRUCYCLES = 200
```

Figure 3-1 Example of the SETSMS PDSE_LRUCYCLES(nnn) command and D SMS,OPTIONS

PDSE_LRUTIME(nnn|15)

This specifies the number of seconds between each run of the BMF LRU routine. The allowable range is from 5 to 60 seconds.

Most installations should use the default value of 15. This value can be tuned in the case of very high data rates or if the LRU routines are using excessive CPU

cycles. SMF type 42 subtype 1 records should be monitored to see the caching activity in the BMF data space.

Default: 15

This option was LRUTIME prior to z/OS V1R6. LRUTIME is still accepted but only applies to the global SMSPDSE address space.

See *z/OS MVS System Management Facilities*, SA22-7630, for more information about the SMF type 42 record and the statistics it records.

Refer to 3.7, “Hiperspace considerations” on page 47 for setting PDSE_LRUTIME.

The SETSMS command can be used to change the value of PDSE_LRUTIME dynamically.

There is no specific response to the SETSMS PDSE_LRUTIME(nn) command. The D SMS,OPTIONS can be used to verify that the change was made. Figure 3-2 shows the command SETSMS PDSE_LRUTIME(16), and part of the output from the D SMS,OPTIONS command showing that the value has been set to 16.

```
SETSMS PDSE_LRUTIME(16)
D SMS,OPTIONS
IGD002I 17:54:09 DISPLAY SMS 199
ACDS     = SYS1.SMS.ACDS
COMMDS   = SYS1.SMS.COMMDS
INTERVAL = 15           DINTERVAL = 150
SMF_TIME = YES          CACHETIME = 3600
CF_TIME = 1800          PDSE_RESTARTABLE_AS = YES
PDSE_BMFTIME = 3600     PDSE1_BMFTIME = 3600
PDSE_LRUTIME = 16       PDSE1_LRUTIME = 15
```

Figure 3-2 Example of the SETSMS PDSE_LRUTIME(nn) command and D SMS,OPTIONS

PDSE_HSP_SIZE(nnn | 256)

On z/Series systems where there is no longer expanded storage, the default is 256 MB or one-fourth of the available real storage, whichever is smaller. If the amount of central storage is 64 MB or less, the amount of storage used is limited to one-eighth of that available.

The PDSE_HSP_SIZE parameter can be used to request up to 2047 MB for the PDSE Hiperspace. You can indicate that the Hiperspace is not to be created by setting PDSE_HSP_SIZE to 0.

Note: APAR OA07807 must be applied to increase the upper limit from 512 MB to 2047 MB. See “OA07807” on page 332.

If a valid value is specified for PDSE_HSP_SIZE, the system uses it to create the PDSE Hiperspace at IPL. PDSE_HSP_SIZE cannot be altered after IPL.

If there is not enough available storage resource to satisfy the specified values for both PDSE_HSP_SIZE and PDSE1_HSP_SIZE, then the system will use some portion of what is available. This portion will be distributed among the PDSE and PDSE1 Hiperspaces depending on the amount of caching activity. Member caching for PDSE, PDSE1, or both will cease if the storage resources required for the Hiperspaces become fully utilized elsewhere in the system.

The value for this parameter should be chosen with the utmost care. Too large a value can potentially have a serious impact on system-wide performance.

Default: 256

This option was HSP_SIZE prior to z/OS V1R6. HSP_SIZE is still accepted and only applies to the global SMSPDSE address space.

Attention: Inappropriate use of PDSE_HSP_SIZE could have a serious impact on your system's performance. Care is required in the specification of the PDSE_HSP_SIZE parameter to avoid overloading the real and auxiliary storage managers.

The DISPLAY SMS,OPTIONS command can be used to see the current value in use for all SMS options, in particular PDSE_HSP_SIZE, and its effectiveness may be evaluated by examining SMF type 42 subtype 1 records.

Refer to 3.7, "Hiperspace considerations" on page 47 to set PDSE_HSP_SIZE.

PDSE_BMFTIME(nnn | 3600)

This specifies the interval in seconds between which statistics are recorded and SMF type 42 records are written for Buffer Management Facility (BMF) cache use for the SMSPDSE address space. You can specify a value from 1 to 86399 (24 hours minus 1 second).

According to the DFSMS documentation, if the SMF_TIME keyword is set to YES (the default), it overrides this keyword and SMF records are created at the expiration of SMF intervals. We found that this was not the case. Specify a value that corresponds to the SMF INTVAL. Note that SMF INTVAL is expressed in minutes, but PDSE_BMFTIME is expressed in seconds.

Default: 3600 (1 hour).

The SETSMS command can be used to change the value of PDSE_BMFTIME dynamically. The new value takes effect after the previous value expires.

This option was BMFTIME prior to z/OS V1R6. BMFTIME is still accepted but only applies to the global SMSPDSE address space.

PDSE_MONITOR(NO | YES [, interval | 60 [, duration | 15]])

This specifies whether the SMSPDSE1 address space will be monitored. Optionally, interval and duration values can be specified in seconds. The interval and duration operands can only be specified with the YES option.

Interval is the number of seconds between successive scans of the monitor.

The duration is the number of seconds an exception must exist before it is treated like an error.

Default: Yes, 60, 15

Recommendation: The PDSE monitor is an important function that gives warnings of potential problems with PDSE data sets being managed by the SMSPDSE address space. If problems may be occurring, messages will be issued recommending that the PDSE analysis commands be issued. The timing of the initial messages could be significant in assisting with problem determination. We recommend that you accept the default YES and that PDSE_MONITOR not be set to NO.

There is no change to the PDSE_MONITOR option for the SMSPDSE address space in z/OS 1.6.

3.1.4 PARMLIB IGDSMSxx parameters for SMSPDSE1 support

The following IGDSMSxx PARMLIB parameters are in support of the SMSPDSE1 address space:

PDSE_RESTARTABLE_AS(YES | NO)

As stated in 3.1.2, “PARMLIB required IGDSMSxx parameters for SMSPDSE1 support” on page 34, you must specify YES in conjunction with PDSESHARING(EXTENDED) before IPL so that the restartable SMSPDSE1 address space will be initialized.

Note: Implementing a change in this parameter option requires a system IPL.

Default: NO

PDSE1_MONITOR(NO | YES [, interval | 60 [, duration | 15]])

This specifies whether the SMSPDSE1 address space will be monitored. Optionally, interval and duration values can be specified in seconds. The interval and duration operands can be specified only with the **YES** option.

Interval is the number of seconds between successive scans by the monitor.

The duration is the number of seconds an exception must exist before it is treated like an error.

Default: YES, 60, 15

Recommendation: The PDSE1 monitor is an important function that gives warnings of possible problems with PDSE data sets being managed by the SMSPDSE1 address space. If problems may be occurring, messages will advise that the PDSE analysis commands be issued. The timing of the initial messages could be significant in assisting with problem determination. We recommend that you accept the default YES; that PDSE1_MONITOR not be set to NO.

PDSE1_LRUCYCLES (nnn | 240)

This specifies the maximum number of times the BMF LRU routine allows unused buffers to remain unavailable for reuse. The allowable range is from 5 to 240. This sets the maximum; BMF dynamically adjusts the actual number of LRU cycles before inactive buffers are reused.

PDSE1_LRUCYCLES is related to PDSE1_LRUTIME. A change to PDSE1_LRUCYCLES will take effect the next time the LRU routine runs. Most installations should use the default value of 240. This value can be tuned in the case of very high data rates. SMF type 42 subtype 1 records should be monitored to see the caching activity in the BMF dataspace.

Default: 240

The SETSMS command can be used to change the value of PDSE1_LRUCYCLES dynamically.

There is no specific response to the SETSMS PDSE1_LRUCYCLES(nnn)command, but the D SMS,OPTIONS command can be used to verify that the change was made. Figure 3-3 on page 39 shows the SETSMS PDSE1_LRUCYCLES(201) command as issued and part of the output from the D SMS,OPTIONS command showing that the value has been set to 201.

Refer to 3.7, “Hiperspace considerations” on page 47 to set PDSE1_LRUCYCLES.

```
SETSMS PDSE1_LRUCYCLES(201)
D SMS,OPTIONS
IGD002I 18:00:52 DISPLAY SMS 202
ACDS     = SYS1.SMS.ACDS
COMMDS   = SYS1.SMS.COMMDS
INTERVAL = 15           DINTERVAL = 150
SMF_TIME = YES          CACHETIME = 3600
CF_TIME = 1800          PDSE_RESTARTABLE_AS = YES
PDSE_BMFTIME = 3600     PDSE1_BMFTIME = 3600
PDSE_LRUTIME = 16       PDSE1_LRUTIME = 15
PDSE_LRUCYCLES = 239    PDSE1_LRUCYCLES = 201
```

Figure 3-3 Example of the SETSMS PDSE1_LRUCYCLES(nnn) command and D SMS,OPTIONS

PDSE1_LRUTIME (nnn | 15)

This specifies the number of seconds between each run of the BMF LRU routine. The allowable range is from 5 to 60 seconds.

Most installations should use the default value of 15. This value can be tuned in the case of very high data rates and for SMF statistics interpretation. SMF type 42 subtype 1 records should be monitored to see the caching activity in the BMF dataspace.

The SMF42LRU (BMF LRU interval time) value corresponds to the PDSE[1]_LRUTIME. A slightly different value for PDSE1_LRUTIME compared to PDSE_LRUTIME is recommended to interpret SMF type 42 subtype 1 records. The system writes similar SMF type 42 subtype 1 records for the SMSPDSE and SMSPDSE1 address spaces. A slightly different value in PDSE[1]_LRUTIME and therefore in SMF42LRU can be used to distinguish the SMF type 42 subtype 1 records that were written on behalf of either the SMSPDSE or SMSPDSE1 address space.

Default: 15

SETSMS can be used to change the value of PDSE_LRUTIME dynamically.

There is no specific response to the SETSMS PDSE1_LRUTIME(nn) command, but the D SMS,OPTIONS command can be used to verify that the change was made. Figure 3-4 shows the SETSMS PDSE1_LRUTIME(51) command as issued and part of the output from the D SMS,OPTIONS command showing that the value has been set to 51.

Refer to 3.7, “Hiperspace considerations” on page 47 for setting PDSE1_LRUTIME.

```
SETSMS PDSE1_LRUTIME(51)
D SMS,OPTIONS
IGD002I 18:05:54 DISPLAY SMS 213
ACDS     = SYS1.SMS.ACDS
COMMDS   = SYS1.SMS.COMMDS
INTERVAL = 15           DINTERVAL = 150
SMF_TIME = YES          CACHETIME = 3600
CF_TIME = 1800          PDSE_RESTARTABLE_AS = YES
PDSE_BMFTIME = 3600     PDSE1_BMFTIME = 3600
PDSE_LRUTIME = 16       PDSE1_LRUTIME = 51
```

Figure 3-4 Example of the SETSMS PDSE1_LRUTIME(nn) command and D SMS,OPTIONS

PDSE1_HSP_SIZE (nnn | 256)

This specifies the size of the Hiperspace (in megabytes) that is used for the restartable SMSPDSE1 address space member caching. It is applicable only if PDSESHARING(EXTENDED) and RESTARTABLE_PDSE_AS(YES) are specified.

On z/Series systems where there is no longer expanded storage, the default is 256 MB or one-fourth of the available real storage, whichever is smaller. If the amount of central storage is 64 MB or less, the amount of storage used is limited to one-eighth of that available.

The PDSE1_HSP_SIZE parameter can be used to request up to 2047 MB for the PDSE1 Hiperspace. You can indicate that the Hiperspace is not to be created by setting PDSE1_HSP_SIZE to 0.

APAR OA07807 must be applied to increase the upper limit from 512 MB to 2047 MB. (See “OA07807” on page 332.)

If a valid value is specified for PDSE1_HSP_SIZE, the system uses it to create the PDSE1 Hiperspace at IPL time. The PDSE1_HSP_SIZE cannot be altered after IPL.

If there is not enough storage resource available to satisfy both the PDSE_HSP_SIZE and PDSE1_HSP_SIZE values, then the system uses some portion, up to the full amount, of what is available and distributes it between PDSE and PDSE1 Hiperspaces depending on the amount of caching activity. PDSE and/or PDSE1 member caching will cease if the storage resources in use for the Hiperspaces become fully utilized in the system.

The value for this parameter should be chosen with the utmost care. Too low a size will degrade PDSE performance, and too large a value can potentially have a serious impact on system wide performance.

Attention: Inappropriate use of PDSE_HSP_SIZE might have a serious impact on your system’s performance. Although the impact of inappropriate specification of the Hiperspace size is less than before the SMSPDSE address spaces were implemented, care is still required to avoid overloading the real and auxiliary storage managers.

The DISPLAY SMS,OPTIONS command can be used to see the current value in use for PDSE1_HSP_SIZE, and its effectiveness may be evaluated by examining SMF type 42 subtype 1 records.

Refer to 3.7, “Hiperspace considerations” on page 47 for setting PDSE1_HSP_SIZE.

PDSE1_BMFTIME (nnn | 3600)

This specifies the interval in seconds between which statistics are recorded and SMF type 42 records are written for Buffer Management Facility (BMF) cache use for the SMSPDSE1 address space. You can specify a value from 1 to 86399 (24 hours minus 1 second).

According to the DFSMS documentation, if the SMF_TIME keyword is set to YES (the default), it overrides this keyword and SMF records are created at the expiration of SMF intervals.This has been found to be not the case. Specify a value the corresponds to the SMF INTVAL. Note that SMF INTVAL is expressed in minutes, and PDSE1_BMFTIME is expressed in seconds.

Default: 3600 (1 hour).

The SETSMS command can be used to change the value of PDSE1_BMFTIME dynamically. Note that the new value takes effect after the previous value expires.

3.2 Restartable address space considerations

The reason for creating the restartable address space (SMSPDSE1) was to reduce the impact of problems in the use of PDSE data sets if they stopped functioning properly, particularly in a sysplex. The original SMSPDSE address space was created to alleviate problems in the amount of ECSA used to support the PDSE data set format.

PDSE restartable address space support server 1 address space separates the function into two, a server address space and the new restartable address space SMSPDSE1. As many functions as possible are handled by SMSPDSE1.

SMSPDSE1 is not intended to be routinely restarted. A certain amount of storage will be lost each time it is restarted, and depending on the nature of the problem, an IPL may still be required to completely resolve a problem. However, the IPL may be scheduled for a convenient time.

Enhanced diagnostic procedures are available to help determine what may be wrong. Before the SMSPDSE1 restart capability is used, the information contained in 10.3, “Diagnosis guidelines” on page 212 should be reviewed.

3.2.1 Operational changes

The following changes apply if the restartable address space SMSPDSE1 has been initiated and started during Nucleus Initialization Program (NIP) processing during IPL. If it has not started, PDSE operation remains unchanged.

There will be two system address spaces: SMSPDSE and SMSPDSE1. The SMSPDSE1 address space is restartable.

- **T**he original SMSPDSE address space

  The original address space, SMSPDSE, will remain as a non-restartable address space for global connections. These connections cannot tolerate interruption and these connections include those made for PDSEs in the LINKLIST concatenations.

- The new SMSPDSE1 address space

  The new SMSPDSE1 address space is restartable. It will provide all non-global connections, For SMSPDSE1 to be active, both PDSESHARING(EXTENDED) and PDSE_RETSTARTABLE_AS(YES) must be specified in the IGDSMSxx PARMLIB member.

  While the SMSPDSE1 address space is restartable, it should only be restarted when necessary, such as when there are problems with PDSEs associated with this address space.

The VARY SMS,PDSE[1],ANALYSIS command shows which address space is affected. Before issuing the ANALYSIS command, review Chapter 10, “PDSE problem determination” on page 209.

Each time the SMSPDSE1 address space is stopped or terminated, a small amount of ECSA is lost. This small amount can be several megabytes. In addition to this, an ASID will be lost, which is typical for address spaces that provide cross-memory services. A few restarts will not affect the system resource pool overall, but there are some limitations on repetitive restarts.

If the SMSPDSE1 address space is restarted (see the RESTART command in 3.2.3, “Operator commands” on page 42), previously connected PDSEs will be reconnected transparently. Some known incompatibilities are detailed in 3.2.2, “Considerations for restarting SMSPDSE1” on page 42.

3.2.2 Considerations for restarting SMSPDSE1

Before terminating or restarting the SMSPDSE1 address space, consider the likely effect that it will have. Where there are problems with PDSEs, terminating or restarting SMSPDSE1 can result in failures of jobs or TSO users that are accessing PDSEs. Not all connections are restartable. Refer to 10.5.2, “VARY SMS,[PDSE,PDSE1]” on page 224 for command format and considerations that should be reviewed before attempting to restart the SMSPDSE1 address space.

3.2.3 Operator commands

The DISPLAY SMS,OPTIONS command shows what options were either specified or defaulted for all SMS options, and in particular for the SMSPDSE address space and the SMSPDSE1 restartable address space.

Commands are available to activate and restart the SMSPDSE1 address space, which must have been started at IPL time. These commands should not be used until diagnosis has been completed, which will indicate which commands and options can be used. Refer to 10.3, “Diagnosis guidelines” on page 212 for more about using these commands.

To restart SMSPDSE1 if it has been forced from the system (it is not available to start SMSPDSE1 if it had not been started at IPL time), use the command:

```
VARY SMS,PDSE1,ACTIVATE [,COMMONPOOLS ( NEW | REUSE )]
```

To recycle SMSPDSE1 if problems are indicated, use the command:

```
VARY SMS,PDSE1,RESTART [,QUIESCE(n|3) [,COMMONPOOLS ( NEW | REUSE )]]
```

3.2.4 Diagnostic and monitoring commands

Commands for diagnostic and monitoring purposes can be used against SMSPDSE, SMSPDSE1, or both. Refer to 10.2, “Overview of diagnosis tools” on page 211 for detailed problem determination information.

```
VARY SMS,PDSE[1],ANALYSIS [,DSNAME(dsname) [,VOLSER(volser)]] [,RETRIES( n | 1500 ) ]
VARY SMS,PDSE[1],FREELATCH(latchaddr,asid,tcbaddr) [,RETRIES(n|1500)]
VARY SMS,PDSE[1],MONITOR [,ON | OFF | RESTART] [,interval,duration]
```

These commands will be available when the appropriate APARs have been implemented:

```
D SMS,PDSE[1],LATCH(aaaaaaaa),DETAILED|SUMMARY
D SMS,PDSE[1],MODULE(mmmmmmmm)
```

For current implementation status of these APARs, see the fixing PTF for OA09265 in “OA09265” on page 330 and “OA08941” on page 329.

3.3 Link Pack Area (LPA) considerations

Modules that are stored in PDSE format may be stored in the LPA only if they are added after the system has been activated.

- The LPALSTxx PARMLIB member may not list any PDSE data sets for inclusion in the LPA.
- The SET PROG=xx command may be used to specify modules to be added to the LPA in the same way as for PDS data sets. This can only be done after an IPL, so modules that are loaded during IPL are not good candidates for PDSEs. This includes PDSE code. You may automate this at IPL by including the SET PROG command in the COMMNDxx member of PARMLIB.
- It is not possible to refresh LPA resident PDSE support modules after applying maintenance. To implement any PDSE support code modifications for modules in the LPA, an IPL is required.
- There are no issues to consider as far as SMSPDSE1 is concerned as the data set used with a dynamic LPA update is only opened, and therefore connected to the PDSE tasks for the time it takes to load the modules. Once the load has completed, the data set will be closed, and the connection established for this purpose, will be freed. If SMSPDSE1 was restarted, there would be no affect on modules loaded in to the LPA.

3.4 Library lookaside (LLA) considerations

LLA is a system facility that retains a copy of the directory entries for PDS and PDSE data sets. The content of the LLA is specified by the system programmer. The minimum (and usual) definition is to manage the directory entries for all members in all data sets that are included in all system linklist concatenations. However, LLA can be directed to manage other PDS or PDSE data sets and specific members.

By default, LLA manages all data sets in all LINKLIST concatenations whether current or from previously used lists. To add additional data sets to LLA, the START LLA command must be issued with the suffix of the required CSVLLAxx member name (see Figure 3-5). This command would normally be part of PARMLIB member COMMNDxx or IEACMDxx.

```
S LLA,LLA=XX,SUB=MSTR
```

Figure 3-5 Example of command to start LLA using PARMLIB member CSVLLAXX

Multiple PARMLIB members, prefixed CSVLLA, may be set up to allow changes to the LLA configuration. If the member is intended to be used at system startup time, then it should include specification of how to handle the LINKLIST definitions, as well as which data sets to add to the LLA.

A PARMLIB member may be used to alter the LLA definition. In order to add the two data sets PDSERES.LOADING.PDS and PDSERES.LOADED.PDSE to the LLA, a member named CSVLLAxx would be created (where xx is a unique suffix). Figure 3-6 shows member CSVLLAPD.

```
LIBRARIES(PDSERES.LOADED.PDS)
LIBRARIES(PDSERES.LOADED.PDSE)
```

Figure 3-6 Example showing member CSVLLAPD to add two data sets to LLA management

To implement the CSVLLAPD member, issue the command F LLA,UPDATE=PD. It produces the output shown in Figure 3-7. There will be a varying number of data sets listed between the **modify** command and the response, depending on the system configuration.

```
F LLA,UPDATE=PD
IEE252I MEMBER CSVLLAPD FOUND IN SYS1.PARMLIB
IEF196I IEF285I   SYS1.SYSPROG.PARMLIB
IEF196I IEF285I   VOL SER NOS= SBOX01.
IEF196I IEF285I   SYS1.PARMLIB
IEF196I IEF285I   VOL SER NOS= 037CAT.
IEF196I IEF285I   CPAC.PARMLIB
IEF196I IEF285I   VOL SER NOS= Z16CAT.
IEF196I IEF285I   SYS1.IBM.PARMLIB
IEF196I IEF285I   VOL SER NOS= Z16CAT.
IEF196I IEF237I 2677 ALLOCATED TO SYS00201
IEF196I IEF237I 2677 ALLOCATED TO SYS00202
CSV210I LIBRARY LOOKASIDE UPDATED
```

Figure 3-7 Example of the output from modify LLA command to implement member CSVLLAPD

For additional confirmation that the update has been completed, the DISPLAY LLA command can be issued. Partial output from the D LLA command (up to the point where it can be seen that PDSERES.LOADING.PDS has been added) is shown in Figure 3-8.

```
CSV600I 19.51.07 LLA DISPLAY 238
EXITS: CSVLLIX1 - ON   CSVLLIX2 - OFF
VLF: ACTIVE  GET LIB ENQ: YES  SEARCH FAIL COUNT: 0
LNKLST SET: LNKLSTQ4
90 LIBRARY ENTRIES FOLLOW
ENTRY   L F R P  LIBRARY NAME
    1   L        EQQ.SEQQLMD0
    2   L        TCPIP.SEZALNK2
    3   L        FPE.V2R1M0.SFPELINK
    4   L        GIM.SGIMLMD0
    5   L        FFST.V120ESA.SEPWMOD2
    6   L        FMN.SFMNMOD1
    7   L        DVG.NFTP230.SDVGLMD2
    8   L        CSF.SCSFMOD0
    9   L        IXM.SIXMMOD1
   10   L        SYS1.SANDBOX.LINKLIB
   11   L        SYS1.SIOALMOD
   12   L        EJES2.V2R3M0.SEJWLINK
   13   L        ISF.SISFLOAD
   14            PDSERES.LOADED.PDS
```

*Figure 3-8 Example of output from the **D LLA** command*

Data sets added to the LLA can be removed.

If the two data sets added in member CSVLLAPD were to be removed, a member CSVLLAPX might be coded as shown in Figure 3-9.

```
REMOVE(PDSERES.LOADED.PDS)
REMOVE(PDSERES.LOADED.PDSE)
```

Figure 3-9 Example showing member CSVLLAPX to remove two data sets from LLA management

Figure 3-10 shows the command F LLA,UPDATE=PX and its output.

```
F LLA,UPDATE=PX
IEE252I MEMBER CSVLLAPX FOUND IN SYS1.PARMLIB
IEF196I IEF285I   SYS1.SYSPROG.PARMLIB
IEF196I IEF285I   VOL SER NOS= SBOX01.
IEF196I IEF285I   SYS1.PARMLIB
IEF196I IEF285I   VOL SER NOS= 037CAT.
IEF196I IEF285I   CPAC.PARMLIB
IEF196I IEF285I   VOL SER NOS= Z16CAT.
IEF196I IEF285I   SYS1.IBM.PARMLIB
IEF196I IEF285I   VOL SER NOS= Z16CAT.
CSV210I LIBRARY LOOKASIDE UPDATED
IEF196I IEF285I   PDSERES.LOADED.PDS
IEF196I IEF285I   VOL SER NOS= SBOX20.
IEF196I IEF285I   PDSERES.LOADED.PDSE
IEF196I IEF285I   VOL SER NOS= SBOX20.
```

Figure 3-10 Example of the output from modify LLA to implement member CSVLLAPX

3.5 LNKLST and LLA considerations

PDSE data sets may be used in the LINKLIST, and the LLA will manage their directories in much the same way as for a standard PDS. This means that for data sets that are not changed for the life of an IPL, there are no significant considerations.

However, when a data set is in the LINKLIST, and the data set might be modified from another system in a GRSPLEX, it is essential that the GRS parameters are correctly set up across all members of the group. Failure to ensure that the GRS parameters are correctly set up could result in damage to the PDSE.

When PDSESHARING(EXTENDED) is specified, it is assumed that a correctly configured GRSPLEX is defined and that all systems sharing the PDSEs are in the same sysplex. We recommend that PDSESHARING(EXTENDED) be specified even if the SMSPDSE1 address space is not being used because EXTENDED enables users to share read and write access to PDSEs across systems in the sysplex.

We recommend that PDSE_RESTARTABLE(YES) be specified to obtain the advantages available with the restartable SMSPDSE1 address space. PDSESHARING(EXTENDED) must be specified to be able to implement SMSPDSE1.

Management of the LINKLIST members that are allocated at IPL time or using the SETPROG LNKLST is in the SMSPDSE address space, which is not restartable, so if the server becomes inoperable an IPL may be required. This also restricts the opportunity to remove a PDSE from the LINKLIST. See APAR OW40072 in “OW40072” on page 326 for more about the restrictions that apply when PDSE data sets are in the LINKLIST at IPL time.

Some alleviation of this problem is provided for data sets that can be added to the LINKLIST after IPL in the resolution of APARs OW57609 (see ““OW57609” on page 328) and OW57671 (see “OW57671” on page 329).

If there is a potential requirement to be able to remove PDSE data sets from the LINKLIST without an IPL, consideration should be given to adding them after IPL.

Attention: This change in PDSE handling makes the function from this point of view similar to PDS data sets added to the LINKLIST. This means that the data set to be removed must be removed from the LLA and the active LINKLIST and from any other LINKLISTs that may have been in use before it can be deleted

3.6 Virtual lookaside facility (VLF) considerations

The LLA function can (and in most installations does) use VLF. LLA is only applicable to the LINKLIST. VLF is more general and can manage items other than program objects and load modules.

PDSE data sets will also be subject to Hiperspace considerations. For more about the performance implications of the interaction of LLA/ VLF and Hiperspace, see Chapter 9, “Performance considerations” on page 179.

Program objects being housed in PDSEs are eligible to be managed by LLA/VLF as well as the PDSE Hiperspaces. In general, a program object that is in use by more than one address space concurrently may benefit from the use of the PDSE Hiperspace. However, if there is only one user, use of Hiperspace may result in a waste of real storage or CPU cycles.

3.6.1 LLA registration with VLF

From the system programming point of view, the PARMLIB COFVLFxx member must have been set up with appropriate classes for it to be effective for use with LLA.

For VLF to manage LLA, along with any other VLF management classes, the active COFVLFxx member must contain the major and minor specifications as shown in Figure 3-11. This is as supplied with the system. All members being managed by LLA are made eligible for VLF services through this simple definition.

```
CLASS NAME(CSVLLA)          /* Class name for Library Lookaside @P2C*/
      EMAJ(LLA)             /* Major name for Library Lookaside @P2C*/
```

Figure 3-11 PARMLIB COFVLFxx requirements for VLF management of LLA

The activity of LLA and VLF can be displayed using the undocumented command:

```
D LLA,STATS
```

Refer to 9.6.7, “LLA and VLF usage display” on page 199 for additional information.

3.6.2 VLF use with REXX EXECs

Data stored in PDS or PDSE data sets can be loaded into the VLF address space. This may be beneficial if there are multiple users of a particular PDS or PDSE, or, for example, a single user needs to swap between many members in an editing session. An end user does not have to do anything in order to use the VLF copy of a member after the PDS or PDSE has been registered with VLF.

From the system programming point of view, the PARMLIB COFVLFxx member must have been set up with appropriate classes for it to be effective for use with REXX EXECs.

Unlike registering VLF to manage LLA, the active COFVLFxx member must contain major and minor specifications as shown in Figure 3-12 to name the PDS or PDSE data sets that are to be managed. No sample is supplied with the system as delivered.

```
CLASS NAME(IKJEXEC) /* Class name for Library Lookaside @P2C*/
      EDSN(rexx_members_dsname) /* data set to be managed by VLF */
```

Figure 3-12 PARMLIB COFVLFxx requirements for VLF management of specific dataset

Even when a PDS or PDSE name has been registered with VLF, the member contents will not be copied to the VLF address space until a cutoff level has been reached. This prevents VLF from being filled up with infrequently used members.

To use a PDS or PDSE member name from VLF, the implicit form must be used whereby the member name is issued as a command name, rather than by using the EXEC processor. If the EXEC processor is used, the member will be fetched from the PDS or PDSE directly.

Attention: When a data PDS or PDSE is to be managed by VLF, there is no operational difference between the two if the PDS or PDSE contents are not altered. If a member is altered, then VLF must be notified by STOW or alternative means, and this is correctly handled for PDS data sets.

When the managed data set is a PDSE and a member is updated using IEBCOPY, then an APAR fix is needed to make IEBCOPY (which uses FAMS internally) notify VLF because STOW is not used. The APAR fix for this is OA09955. (See “OA09955” on page 332.)

If this situation should arise before the APAR fix is available and installed, it would be necessary to stop and restart VLF to pick up the new definitions.

3.7 Hiperspace considerations

PDSE Hiperspace is a function created to augment PDSE processing that is most beneficial for repeated use of the contents of a member.

Whether benefits will be obtained from using PDSE Hiperspace depends on the application. There are two separate functions related to a PDSE data set that could use Hiperspace: the directory and the member content.

There will not be any benefit from using PDSE Hiperspace if there is only one user of the data set at any one time. Only heavily used PDSEs, and those that stay open long enough to take advantage of it, should use PDSE Hiperspace member caching. Therefore, in general, it is better to avoid using PDSE Hiperspace unless there is a demonstrated improvement from using it.

PDSE system data sets (for example, those in LNKLST) are often managed by LLA/VLF. Any module that is cached to Hiperspace that is also managed by LLA/VLF will, in time, be dropped from Hiperspace, so it is better to avoid having these program objects go to the Hiperspace in the first place. This happens automatically for program objects in system non-SMS-managed PDSEs after APAR OA06884 (see “OA06884” on page 330) or a fix to that, OA08991 (see “OA08991” on page 331), and OA09162 (see “OA09162” on page 331) are installed.

For SMS-managed PDSE data sets, the use of Hiperspace is controlled by the Direct MSR value in the associated Storage Class. The MSR values have wider applicability to the DASD

storage units, and in the case of PDSE data sets, it is only the *must cache* and *do not cache* settings that are of interest. A storage class with Direct MSR set at less than 10 indicates must cache, between 10 and 999 means may cache, and 999 means do not cache. To be sure of getting expected cache activity, ensure that SMS-managed PDSEs are associated with storage classes that have appropriate MSR settings. For PDSE data sets delivered as part of the operating system, or applications such as DB2®, the data sets are generally not SMS-managed.

If the PDSE Hiperspace is to be used, then its use of CPU time, real storage, or both must be balanced against the benefit obtained.

The PDSE_HSP_SIZE and PDSE1_HSP_SIZE parameters affect the size of the Hiperspace associated with the SMSPDSE and SMSPDSE1 address spaces respectively. It is possible to prevent the creation of Hiperspaces by setting one or both of the HSP_SIZE parameters to 0. This requires an IPL to implement.

Hiperspaces use system storage, and the must be tuned to avoid excessive use at the expense of other system functions. See 9.5, “PDSE use of processor storage” on page 188 for information about the PDSE buffering techniques, and 3.1, “PARMLIB requirements” on page 34 for information about specifying the PARMLIB statements for PDSE operation. The PDSE implementation of Hiperspace does not have the pages written to the page data sets. If a page is stolen by the real storage manager, it will be refreshed from the original PDSE if and when required. This is functionally transparent to the user, but it may make response time erratic, so it is better to have the size of the Hiperspace as small as possible to be consistent with overall system performance. Hiperspace pages are the last to be stolen in a real storage constrained situation.

When the fix for APAR OA08991 or its predecessor OA06884 is installed (see “OA08991” on page 331) all program objects, whether SMS-managed or not, are eligible to be cached in the PDSE Hiperspaces. This can increase CPU time and use of real storage in the Hiperspaces associated with the SMSPDSE and SMSPDSE1 address spaces. A follow-on APAR, OA09162, removes any non-SMS managed PDSE data sets from consideration, which improves the situation because many of the modules that became eligible were non-SMS managed system datasets.

The three Hiperspace parameters, HSP_SIZE, LRUCYCLES, and LRUTIME, can be adjusted to values that provide a trade-off between the increased CPU time and real storage use and the benefits that can be obtained by using Hiperspace.

The *HSP_SIZE* parameter specifies the amount of storage in the Hiperspace. If there are many actively referenced modules, real storage usage can increase up to the HSP_SIZE value. A smaller value limits real storage contention between the PDSE Hiperspace and other parts of the system.

The *LRUCYCLES* parameter controls the length of time an unreferenced module may stay in the Hiperspace. A reduction in the LRUCYCLES value reduces the number of modules retained in Hiperspace.

The *LRUTIME* parameter controls the frequency with which the checks related to LRUCYCLES are carried out.

When APARs OA06884/OA08991 are installed, system operation can be restored to the state that was in effect before it was installed by setting HSP_SIZE to 0. To make such a change, however, requires an IPL. Removing Hiperspace may also affect data-only PDSE data sets, which might not be desirable.

Note: Adjusting either LRUCYCLES or LRUTIME may be done without an IPL.

Therefore, if problems of increased CPU time or real storage commitment are experienced, we recommend a change to the LRUCYCLES and LRUTIME as follows. Note that if the restartable address space is in use, both the PDSE and PDSE1 versions of these parameters must be changed, because modules in the LINKLIST at IPL time will be cached in the PDSE Hiperspace, and modules added after IPL, or in STEPLIBs, will be cached in the PDSE1 Hiperspace.

This is halving the value from the default value of 240:

```
SETSMS PDSE_LRUCYCLES(120)
SETSMS PDSE1_LRUCYCLES(120)
```

This is increasing the value to the maximum:

```
SETSMS PDSE_LRUTIME(60)
SETSMS PDSE1_LRUTIME(60)
```

If these changes do not result in sufficient improvement in resource consumption, then we recommend reducing the HSP_SIZE values from the default 256 to 128.

These recommendations have been used to resolve increased CPU and real storage usage issues, but for particular installations other values may have better results.

Note: Modules cached in Hiperspace may be aged out faster if the Hiperspace fills up or if there is a shortage of real storage (z/OS mode) or expanded storage (ESA mode). The LRU algorithm adjusts if the system has been casting out Hiperspace pages because of a storage shortage.

3.8 SMS considerations

PDSE format data sets may be managed by SMS or unmanaged. In general, it is better to use SMS to manage PDSE user data sets. For certain system data sets the unmanaged mode may be necessary.

For SMS to function correctly, for any type of data set, there must be ACS (Automatic Class Selection) routines, as well as Storage Group, Storage Class, Data Class, and Management Class tables.The SMS ACS routines work on PDSE data sets in a manner similar to any other type of managed data set.

3.8.1 PDSE data sets that are not SMS-managed

When the PDSE is not SMS-managed, the allocation parameters are as specified or defaulted. The data set placement and attributes, other than those that are specified in an appropriate data class, are not altered by SMS.

3.8.2 PDSE data sets that are SMS-managed

When the PDSE is SMS-managed, the same rules apply as for any other SMS managed data set. The data set will be placed on volumes indicated by the results from the ACS analysis of the request.

In some cases, the attributes of the data set may be different from what was intended. In most cases the PDSE can be used as if it were a standard PDS, and as such it is shown in its data set attributes as PO-E. However, the way the directory and the member space is managed is significantly different in the data set from the way it is done for a standard PDS.

Allocation of data sets intended to be in PDSE format are no different from any other data set in being subject to the ACS priority rules for deciding on attributes.

3.8.3 Automatic Class Selection (ACS) routine considerations

PDSE data sets are initially treated the same way as any other data sets, so the ACS selection routines must be coded to distinguish between those intended to be SMS-managed and those that are not. If a data set is unexpectedly allocated as SMS-managed, some of the allocation parameters may be overridden.

Through ISMF, you can create and maintain four ACS routines in an SCDS, one for each Data Class, Storage Class, Management Class, and Storage Group. Before we look at a real case described in 3.8.4, “How to set up an SMS environment for PDSE” on page 55, we describe some guidelines to be used when you code ACS routines.

Figure 3-13 shows the order in which ACS routines are processed. Data will become system-managed if the storage class routine assigns a storage class storage class to the data set or if a user-specified storage class is assigned to the data set. (Your storage administrator can override this.)

A storage group must be assigned by the ACS routine.

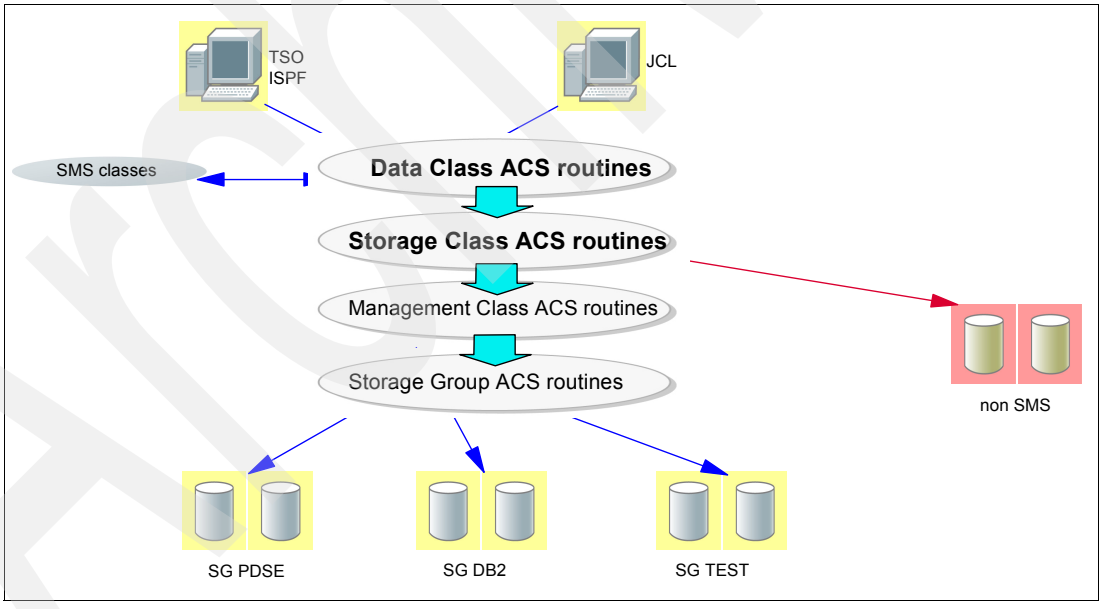


Figure 3-13 Overview of ACS routines

Read-only variables

The ACS language contains a number of read-only variables, which you can use to analyze new data allocations. For example, you can use the read-only variable &DSN to make classes and group assignments based on data set name, or &LLQ to make assignments based on the low-level qualifier of the data set. Table 3-1 on page 51 shows the complete list of read-only variables that are available to your ACS routines.

Table 3-1 Read-only variables

&ACCT_JOB &ACCT_STEP &ACSENVIR &ALLVOL &ANYVOL &APPLIC &BLKSIZE &DD &DEF_DATACLAS &DEF_MGMTCLAS &DEF_STORCLAS	&DSN &DSNTYPE &DSORG &DSOWNER &DSTYPE &EXPDT &FILENUM &GROUP &HLQ &JOB &LABEL	&LIBNAME &LLQ &MAXSIZE &MEMHLQ &MEMLLQ &MEMN &MEMNQUAL &MSPDEST &MSPARM &MSPOLICY &MSPOOL &MSVGP	&NQUAL &NVOL &PGM &RECORG &RETPD &SECLABL &SIZE &SYSNAME &SYSPLEX &UNIT &USER &XMODE

You cannot alter the value of read-only variables. Use the four read-write variables (&DATACLAS, &STORCLAS, &MGMTCLAS, and &STORGRP) to assign the class or storage group you determine for the data set based on the routine you are writing. You may only set a read-write variable in its own ACS routine.

Mask rules

A mask is a character string, such as SYS1.*LIB, that is *not* enclosed in single quotation marks. You can use a mask to represent job names, volume serial numbers, or other system values that have a common string of characters, such as all volume serial numbers that begin with SMS. You can also use a mask to represent data set or collection names that have a common string of characters.

An asterisk, (*), means that zero or more characters can be present in its place.
A percent sign (%) represents exactly one non-blank character. '%%%' represents three character positions.

Examples of using masks include:

TSO* All names, of any length, beginning TSO

ABC All names of any length having three adjacent characters, ABC

SMS%%% All six-character names beginning SMS

***%WK%%** All names where the second and third character of the last five (or only five) are WK

ACS language statements

This section describes the function and syntax of the ACS language statements that you can use when writing ACS routines.

The statement types are defined as follows:

PROC Start of an ACS routine

FILTLIST Definition of filter criteria

SET Assigns a value to a read-write variable

DO Start of statement group

IF Provides conditional statement execution

SELECT Defines a set of conditional execution statements

EXIT Causes immediate termination of the ACS routine and can be used to force allocation failures

WRITE	Sends a message to the end user
END	End of statement group (DO or SELECT) or ACS routine (PROC)

PROC statement

PROC must be the first statement of each ACS routine as it identifies the ACS routine. You can precede the PROC statement with blank lines or comments, but not with other statements. You must also place an END statement at the end of each ACS routine.

Figure 3-14 shows an example: You should replace STORGRP with MGMTCLAS, DATACLAS, or STORCLAS for each ACS routine.

```
/* any comments are allowed before PROC, but NO STATEMENT! */
PROC STORGRP
.......
END
```

Figure 3-14 Example of a PROC and END statement

FILTLIST statement

The FILTLIST statement is a definition list that you can use when testing variables in an ACS routine. You can do multiple tests using one name rather than repeating the whole set. Define the information that you want to include and exclude in the list using the INLCUDE and EXCLUDE keywords. It helps you to compare read-only variables to more than one item (for example, data set name, user ID, volumes) included in this list. Figure 3-15 shows an example of FILTLIST usage.

```
PROC STORCLAS
.......
FILTLIST SCDBMED  INCLUDE(DB2P.DSNDB%.*.M*.**)

FILTLIST SCDBCRIT INCLUDE(DB2P.DSNDB%.*.C0*.**)

FILTLIST SCDBFAST INCLUDE(DB2P.DSNDB%.*.F0*.**)
                  EXCLUDE(DB2P.DSNDB0.*.F0000.OUT)

FILTLIST SCDBTEST INCLUDE(DB2D.DSNDB%.*.T*.**,
                          DB2T.DSNDB%.*.T*.**)

FILTLIST HSMDSET  INCLUDE('HSM.HMIGTAPE.DATASET',
                          'HSM.COPY.HMIGTAPE.DATASET',
                          'HSM.BACKTAPE.DATASET',
                          'HSM.COPY.BACKTAPE.DATASET',
                          HSM.DMP.** )

FILTLIST UNIT_3590 INCLUDE('3590','3590-1')

FILTLIST UNIT_ATL2 INCLUDE('ATL2')
.....
IF &DSN = &SCDBFAST THEN
(some action)
......

END
```

Figure 3-15 Example of a FILTLIST statement

SET statement

The SET statement assigns values to the read-write variables. You can assign one name *only* to these variables:

- &DATACLAS
- &STORCLAS
- &MGMTCLAS

The following read-write variable can have more than one name assigned to it:

- &STORGRP

The names can be from one to eight characters long and they must be enclosed in single quotation marks. You can assign a null value to any of the read-write variables except for &STORGRP. Figure 3-16 shows an example of read-write variable assignments.

```
SET &STORCLAS EQ ‘SCSMS’
....
SET &STORCLAS = ‘SCSMS’
....
SET &DATACLAS = ‘DCPDSE’
....
SET &STORCLAS = ‘’
....
SET &STORGRP = ‘SG1’, ‘SG2’, ‘SG3’
```

Figure 3-16 SET statement sample

DO statement

You can group a collection of ACS language statements using a DO statement paired with an END statement. The DO statement can follow an IF-THEN clause, an ELSE clause, or a SELECT-WHEN group.

Figure 3-17 shows an example of a DO statement.

```
PROC STORCLAS
....
FILTLIST VALID_HLQ INCLUDE('DAVIDE','MATTEO','ALESSIA')
....
   IF &HLQ NE &VALID_HLQ THEN
      DO
        WRITE 'HLQ NOT ALLOWED!'
        EXIT CODE(1)
      END
....
END
```

Figure 3-17 Example of a DO statement

SELECT statement

Use the SELECT statement to write conditional statements in sequential form. A SELECT statement consists of a SELECT keyword, one or more WHEN clauses, an optional OTHERWISE clause, and an END statement.

Important: The first true WHEN condition is executed and the remaining WHEN conditions are ignored. If none of the WHEN conditions is true and there is an OTHERWISE clause, then the OTHERWISE action is taken.

For this reason, it is important to consider where to add a new conditional test in an existing ACS routine. Usually you have to put more specific SELECT groups at the top of the ACS routine, and more general ones at the bottom.

Figure 3-18 shows the controlling variable named after the SELECT keyword.

```
PROC STORCLAS
....
   SELECT (&USER)
      WHEN ('IBMUSER')  SET &STORCLAS = 'SCSMS'
      WHEN ('OPERATOR') SET &STORCLAS = ''
      WHEN ('ADMIN')    SET &STORCLAS = 'SCHIGH'
      WHEN ('IBMUSER')  SET &STORCLAS = 'SCSMS'
      OTHERWISE         SET &STORCLAS = 'SCLOW'
   END
....
END
```

Figure 3-18 Example of a SELECT keyword

Figure 3-19 shows the three variables being tested after WHEN keywords, enabling you to test different variables and different conditions under the same SELECT.

```
PROC DATACLAS
/*....*/
   SELECT
      WHEN (&DSNTYPE  = 'LIBRARY') SET &DATACLAS = 'DCPDSE'
      WHEN (&HLQ      = 'GDS'    ) SET &DATACLAS = 'DCGDG'
      WHEN (&DSN      = NOSMS.** )
       DO
            WRITE 'ALLOCATION WILL BE DIRECTED TO NON SMS VOLUMES'
            SET &DATACLAS = ''
       END
   END
/*....*/
END
```

Figure 3-19 Example of a SELECT statement

EXIT statement

The EXIT statement immediately terminates the operation of an ACS routine. It forces the completion and sets an exit code. Figure 3-20 shows an example of an EXIT statement.

```
PROC DATACLAS
/*....*/
   SELECT
      WHEN (&DSNTYPE  = 'LIBRARY') SET &DATACLAS = 'DCPDSE'
      WHEN (&HLQ      = 'GDS'    ) SET &DATACLAS = 'DCGDG'
      WHEN (&DSN      = NOSMS.** )
       DO
            WRITE 'ALLOCATION WILL BE DIRECTED TO NON SMS VOLUMES'
            SET &DATACLAS = ''
            EXIT CODE(0)
       END
      WHEN (&LLQ      = 'CNTL'   ) SET &DATACLAS = 'DCJCL'
   END
/*....*/
END
```

Figure 3-20 Example of an EXIT statement

WRITE statement

Use the WRITE statement to issue a message to an end user at execution and allocation.

END statement

The END statement concludes an ACS routine, a DO group, or a SELECT statement.

3.8.4 How to set up an SMS environment for PDSE

In this section we describe how to create a minimum SMS environment for PDSE data sets. We created the following classes:

- DCPDSE
- SCPDSEHI
- SCPDSELO
- SCGSWYES

SMS Data Class considerations

When SMS is implemented in an environment, the Data Class in conjunction with the ACS routines affects most of the data set allocations.

In order to cause a data set to be allocated as a PDSE, the DATA SET NAME TYPE field must be specified as LIB. The DATA SET NAME TYPE that is used in the SMS definition panels, and the DSNTYPE that is used in TSO and JCL allocation, can be provided by TSO or JCL, and can be changed or set in the SMS ACS routines.

The ISMF application DATA CLASS DEFINITION is required to define a data class. There may be many data classes in an installation to allow for different requirements. The following example covers the values that affect PDSE definitions.

There are four panels to complete a data class definition, as shown in figure Figure 3-21 through Figure 3-24 on page 57. The Data Class name, DCPDSE in our example, is defined on panel 1 of 4.

```
------------------------------------------------------------------------------
                              DATA CLASS DEFINE                Page 1 of 4
Command ===>

SCDS Name . . . : SYS1.SMS.SCDS
Data Class Name : DCPDSE

To DEFINE Data Class, Specify:
  Description ==>
              ==>
  Recfm . . . . . . . . . .            (any valid RECFM combination or blank)
  Lrecl . . . . . . . . . .            (1 to 32761 or blank)
  Space Avgrec  . . . . . .            (U, K, M or blank)
        Avg Value . . . . .            (0 to 65535 or blank)
        Primary . . . . . .            (0 to 999999 or blank)
        Secondary . . . . .            (0 to 999999 or blank)
        Directory . . . . .            (0 to 999999 or blank)
  Retpd or Expdt  . . . . .            (0 to 9999, YYYY/MM/DD or blank)
  Volume Count  . . . . . . 1          (1 to 59 or blank)
  Add'l Volume Amount . . .            (P=Primary, S=Secondary or blank)

Use ENTER to Perform Verification; Use DOWN Command to View next Panel;
```

Figure 3-21 Data class definition for PDSE (1 of 4)

For PDSE format, panel 2 of 4 contains the critical setting of LIB for the Data Set Name Type (Figure 3-22).

```
 DATA CLASS DEFINE                Page 2 of 4
Command ===>

SCDS Name . . . : SYS1.SMS.SCDS
Data Class Name : DCPDSE

To DEFINE Data Class, Specify:

  Data Set Name Type  . . . . . . LIB  (EXT, HFS, LIB, PDS or blank)
    If Ext  . . . . . . . . . . .      (P=Preferred, R=Required or blank)
    Extended Addressability . . . N    (Y or N)
    Record Access Bias  . . . . .      (S=System, U=User or blank)
  Space Constraint Relief . . . . N    (Y or N)
    Reduce Space Up To (%)  . . .      (0 to 99 or blank)
    Dynamic Volume Count  . . . .      (1 to 59 or blank)
  Compaction  . . . . . . . . . .      (Y, N, T, G or blank)
  Spanned / Nonspanned  . . . . .      (S=Spanned, N=Nonspanned or blank)



Use ENTER to Perform Verification; Use UP/DOWN Command to View other Panels;
. . . . . .
```

Figure 3-22 Data class definition for PDSE (2 of 4)

```
 DATA CLASS DEFINE                 Page 3 of 4
Command ===>

SCDS Name . . . : SYS1.SMS.SCDS
Data Class Name : DCPDSE

To DEFINE Data Class, Specify:

  Media Interchange
    Media Type . . . . . .          (1, 2, 3, 4, 5, 6, 7, 8 or blank)
    Recording Technology . .        (18, 36, 128, 256, 384, E1 or blank)
    Performance Scaling  . .        (Y, N or blank)
  Block Size Limit . . . . .        (32760 to 2GB or blank)
  Recorg . . . . . .                (KS, ES, RR, LS or blank)
  Keylen . . . . . .                (0 to 255 or blank)
  Keyoff . . . . . .                (0 to 32760 or blank)
  CIsize Data  . . . . . .          (1 to 32768 or blank)
  % Freespace CI . . . . . .        (0 to 100 or blank)
              CA . . . . . .        (0 to 100 or blank)

Use ENTER to Perform Verification; Use UP/DOWN Command to View other Panels;
. . . . . .
```

Figure 3-23 Data class definition for PDSE (3 of 4)

```
 DATA CLASS DEFINE                 Page 4 of 4
Command ===>

SCDS Name . . . : SYS1.SMS.SCDS
Data Class Name : DCPDSE

To DEFINE Data Class, Specify:

  Shareoptions Xregion  . . .       (1 to 4 or blank)
               Xsystem  . . .       (3, 4 or blank)
  Reuse . . . . . . . . . . . N     (Y or N)
  Initial Load  . . . . . . . R     (S=Speed, R=Recovery or blank)
  BWO . . . . . .           .       (TC=TYPECICS, TI=TYPEIMS, NO or blank)
  Log . . . . . .           .       (N=NONE, U=UNDO, A=ALL or blank)
  Logstream Id  . . . . . .
  FRlog . . . . . .         .       (A=ALL, N=NONE, R=REDO, U=UNDO or blank)
  RLS CF Cache Value  . . . . A     (A=ALL, N=NONE, U=UPDATESONLY)



Use ENTER to perform Verification; Use UP Command to View previous Panel;
. . . . . .
```

Figure 3-24 Data class definition for PDSE (4 of 4)

Figure 3-25 on page 58 shows a sample ACS routine for this Data Class.

```
PROC DATACLAS
/*....*/
   SELECT
      WHEN (&DSNTYPE  = 'LIBRARY') SET &DATACLAS = 'DCPDSE'
      WHEN (&HLQ      = 'PDSE'   ) SET &DATACLAS = 'DCPDSE'
   END
/*....*/
END
```

Figure 3-25 Data Class sample ACS routine

SMS Storage Class considerations

The storage class defines the way the disk subsystem is to handle operations on the data set. The required response time affects whether a PDSE data set will use the PDSE Hiperspaces. The performance objectives that can be specified include the Direct Millisecond Response and Sequential Millisecond Response. The response performance objective is set on panel one of the two Storage Class definition panels. The Direct Millisecond Response (generally referred to as MSR) controls whether modules from a PDSE in this class will be eligible for Hiperspace caching. MSR values of <9, <50, and 999 correspond to *must cache*, *may cache*, and *never cache*. Values between 10 and 999 have effects that are dependent on the system load. Table 3-2 summarizes the effects of MSR settings.

Table 3-2 Effects of MSR settings for PDSE Storage Class

	Hiperspace cache	**Disk control unit cache**
MSR 1	Used	Must
MSR 50 or blank	*not used*	Maybe
MSR 999	*not used*	Never

Although program objects from non-SMS managed PDSE data sets are not associated with a storage class, their MSR value is taken to be low, and they are in the never cache category.

Note: Non-SMS managed PDSEs have no Storage Class, so they do not use Hiperspace.

The MSR values have broader meaning for DASD storage units, but in the case of PDSE data sets, they can only result in must cache or do not cache.

This example is a definition for a storage class named SCPDSEHI. This class is intended to ensure *must cache*, so the Direct Millisecond Response value is set to 1. Figure 3-26 on page 59 shows the first of the two panels.

```
 Panel  Utilities  Scroll  Help
------------------------------------------------------------------------------
                              STORAGE CLASS DEFINE             Page 1 of 2
Command ===>

SCDS Name . . . . . : SYS1.SMS.SCDS
Storage Class Name  : SCPDSEHI
To DEFINE Storage Class, Specify:
  Description ==>
              ==>
  Performance Objectives
   Direct Millisecond Response . . . . 1          (1 to 999 or blank)
   Direct Bias . . . . . .            .          (R, W or blank)
   Sequential Millisecond Response . . 1          (1 to 999 or blank)
   Sequential Bias . . . . . .        .          (R, W or blank)
   Initial Access Response Seconds . .           (0 to 9999 or blank)
   Sustained Data Rate (MB/sec)  . . .           (0 to 999 or blank)
  Availability . . . . . . . . . . . N           (C, P ,S or N)
  Accessibility  . . . . . . . . . . N           (C, P ,S or N)
   Backup  . . . . . .               .          (Y, N or Blank)
   Versioning  . . . . . .           .          (Y, N or Blank)

Use ENTER to Perform Verification; Use DOWN Command to View next Page;
```

Figure 3-26 Storage class definition for PDSE

If an application requires that any output directed to a data set be verified as transferred to the disk, the Guaranteed Synchronous Write (GSW) option can be specified in the Storage Class definition. This ensures that all write I/O to that PDSE (or any other form of data set using the same storage class) is synchronized between disk and the buffers.

If a PDSE is allocated to a storage class that does not specify GSW, the system does not synchronize between disk and data set buffers until the CLOSE or STOW macro is issued. PDSE buffering attempts to reduce the number of separate physical operations, thus giving better performance. However, if an error occurs on the disk, any data still in the buffer and not yet successfully recorded on disk may be lost.

Storage administrators specify the Guaranteed Synchronous Write option on the SMS storage class definition panel. Figure 3-27 on page 60 shows an example of panel 2 from the definition sequence to create a Storage Class SCGSWYES.

```
 STORAGE CLASS DEFINE               Page 2 of 2
Command ===>

SCDS Name . . . . . : SYS1.SMS.SCDS
Storage Class Name  : SCGSWYES

To DEFINE Storage Class, Specify:

  Guaranteed Space  . . . . . . . . . N          (Y or N)
  Guaranteed Synchronous Write  . . . Y.          (Y or N)
  Multi-Tiered SG . . . . . . . . . .            (Y, N, or blank)
  Parallel Access Volume Capability   N          (R, P, S, or N)
  CF Cache Set Name . . . . . . . . .            (up to 8 chars or blank)
  CF Direct Weight  . . . . . . . . .            (1 to 11 or blank)
  CF Sequential Weight  . . . . . . .            (1 to 11 or blank)




Use ENTER to Perform Verification; Use UP Command to View previous Page;
```

Figure 3-27 Storage Class definition for PDSE

For completeness, we should also define a Storage Class SCPDSELO that uses an MSR of 999 milliseconds, as shown in Figure 3-28.

```
 Panel  Utilities  Scroll  Help
------------------------------------------------------------------------------
                           STORAGE CLASS DEFINE             Page 1 of 2
Command ===>

SCDS Name . . . . . : SYS1.SMS.SCDS
Storage Class Name  : SCPDSELO
To DEFINE Storage Class, Specify:
  Description ==>
              ==>
  Performance Objectives
   Direct Millisecond Response . . . . 999        (1 to 999 or blank)
   Direct Bias . . . . . . . . . . . .            (R, W or blank)
   Sequential Millisecond Response . . 999        (1 to 999 or blank)
   Sequential Bias . . . . . . . . . .            (R, W or blank)
   Initial Access Response Seconds . .            (0 to 9999 or blank)
   Sustained Data Rate (MB/sec)  . . .            (0 to 999 or blank)
  Availability . . . . . . . . . . . . N          (C, P ,S or N)
  Accessibility  . . . . . . . . . . . N          (C, P ,S or N)
   Backup  . . . . . . . . . . . . . .            (Y, N or Blank)
   Versioning  . . . . . . . . . . . .            (Y, N or Blank)

Use ENTER to Perform Verification; Use DOWN Command to View next Page;
```

Figure 3-28 Storage Class definition for PDSE

Figure 3-29 shows a sample ACS routine for these Storage Classes.

```
PROC STORCLAS
/*....*/
   FILTLIST HIGH_PDSE INCLUDE (SYS1.**, VIP.**, PDSEHI.**)
   FILTLIST GSW_PDSE  INCLUDE (PDSE*.**.GSW)
/*....*/
   SELECT
      WHEN (&DSN      = &HIGH_PDSE) SET &STORCLAS = 'SCPDSEHI'
      WHEN (&DSN      = &GSW_PDSE ) SET &STORCLAS = 'SCGSWYES'
      WHEN (&DATACLAS = 'DCPDSE'  ) SET &STORCLAS = 'SCPDSELO'
   END
/*....*/
END
```

Figure 3-29 Storage Class sample ACS routine

SMS Management Class considerations

There are no unique considerations for PDSEs.

SMS Storage Group considerations

There are no particular requirements for PDSEs regarding SMS Storage Group considerations. An exception is that one of the reasons for converting to PDSE may have been to facilitate sharing of data, in which case the storage group must be appropriately shared.

You could assign any existing Storage Group to PDSE. A sample ACS routine for Storage Group is shown in Figure 3-30.

```
PROC STORGRP
/*....*/
   FILTLIST PDSE_SG  INCLUDE(SCPDSE*, 'SCGSWYES')
/*....*/
   SELECT
      WHEN (&STORCLAS = &PDSE_SG  ) SET &STORGRP  = 'SGPROD'
   END
/*....*/
END
```

Figure 3-30 Storage Group sample ACS routine

3.8.5 How to determine the current ACS routines

The easiest way to understand where the ACS source routines are located is to use ISMF.

From the main ISMF menu, chose `option 7 Automatic Class Selection`. Select `option 5` and enter '`ACTIVE`' in the CDS Name field as shown in Figure 3-31.

```
 Panel  Utilities  Help
------------------------------------------------------------------------------
                          ACS APPLICATION SELECTION
Command ===>

Select one of the following options:
5  1. Edit            - Edit ACS Routine source code
   2. Translate       - Translate ACS Routines to ACS Object Form
   3. Validate        - Validate ACS Routines Against Storage Constructs
   4. Test            - Define/Alter Test Cases and Test ACS Routines
   5. Display         - Display ACS Object Information
   6. Delete          - Delete an ACS Object from a Source Control Data Set



If Display Option is Chosen, Specify:

  CDS Name  . . 'ACTIVE'
                                (1 to 44 Character Data Set Name or 'Active')



Use ENTER to Perform Selection;
```

Figure 3-31 How to determine the current ACS routines

Figure 3-32 on page 63 shows sample output. For each ACS routine the library name, active member name, last user ID, and last date translated are shown. You can edit each ACS routine using ISPF as usual.

```
 Panel  Utilities  Help
------------------------------------------------------------------------------
                              ACS OBJECT DISPLAY
Command ===>

CDS Name  : ACTIVE

ACS Rtn   Source Data Set ACS      Member    Last Trans  Last Date  Last Time
Type      Routine Translated from  Name      Userid      Translated Translate
--------  -----------------------  --------  ----------  ---------- ---------
DATACLAS  EVEBYE.JCL.CNTL          OMVSDC1   PAOLOR2     2004/07/28  16:54


MGMTCLAS  EVEBYE.JCL.CNTL          HGMC      MHLRES3     2003/11/07  19:19


STORCLAS  EVEBYE.JCL.CNTL          SCACTIVE  HAIMO       2004/06/23  15:20


STORGRP   EVEBYE.JCL.CNTL          SGACTIVE  MHLRES3     2004/11/10  11:55
```

Figure 3-32 How to find out which are the current ACS routines

3.9 JES2 PROCLIB considerations

JES2 supports the use of PDSEs to hold PROCLIBs. This may be by definition in the JES2 STC procedure itself, by using the dynamic PROCLIBs definition capability, or by use of the JCLLIB JCL statement.

There are no considerations relating to the use of PDSEs in JCLLIB. A PDSE would be a good choice when a procedure library is being updated frequently, because compression is not needed.

If the JES2 procedure library is shared across systems sysplex, problems with the data set may necessitate a sysplex-wide restart if the PROCLIB data sets are allocated in the JES2 STC procedure. The JES2 instances must all be restarted without the problem data set.

The JES2 dynamic PROCLIB feature was introduced with z/OS V1R5, and it provides a potential solution to the problem of damaged PROCLIB data sets by enabling additional data sets to be made available to JES2. Because the dynamic PROCLIB function supports PDSE data sets, issues relating to PDS data sets expanding and requiring compressing may be reduced. JES2 in combination with the z/OS converter/interpreter can resolve some PROCLIB issues by closing and opening a particular DDNAME, but cannot recover from corruption within the data set.

One advantage of setting up the JES2 PROCLIB structure using dynamic PROCLIBs is that the $D PROCLIB(xxxxxxxx) command can be used to display the structure; however, there is no defined way of displaying the PROCLIB data sets that were allocated in the JES2 STC procedure. $D PROCLIB(xxxxxxxx) displays the PROCLIB data sets that were added dynamically, and it should be remembered that this is not a complete list of active PROCLIB data sets unless they were all allocated dynamically. If all PROCLIB data sets are added dynamically, the $D PROCLIB command can be used to display them.

3.9.1 JES2 dynamically-allocated PROCLIBs

Dynamically defined PROCLIB data sets may be altered and deleted by command without restarting JES2.

Figure 3-33 shows sample JCL that might be included in the JES2 STC procedure.

```
//MHLRES1 DD DISP=SHR,DSN=PDSERES.PROCLIB1
//        DD DISP=SHR,DSN=PDSERES.PROCLIB2
//        DD DISP=SHR,DSN=SYS1.PROCLIB
```

Figure 3-33 Example of JES2 PROCLIB data sets as defined in a JES2 STC procedure

The JCL as defined in the STC procedure may be overridden by dynamic PROCLIB definition or new data sets will be defined, depending on whether the name matches an existing one. Figure 3-34 shows the functionally equivalent dynamic PROCLIB definition for the JCL example in Figure 3-33.

```
PROCLIB(MHLRES1) DD(1)=(PDSERES.PROCLIB1)
                 DD(2)=(PDSERES.PROCLIB2)
                 DD(3)=(SYS1.PROCLIB1)
```

Figure 3-34 Adding JES2 PROCLIB data sets to JES2 STC dynamically via JES2 PARMLIB

To dynamically add a set of PROCLIB data sets through command, Figure 3-35 shows the syntax for the examples in Figure 3-34. If the DDNAME MHLRES1 already exists, the command will fail.

```
$ADD PROCLIB(MHLRES1),DD1=dsn=PDSERES.PROCLIB1,
DD2=dsn=PDSERES.PROCLIB2,
DD3=dsn=SYS1.PROCLIB1
```

Figure 3-35 Example of adding JES2 proclib data sets to JES2 dynamically by command

To modify a set of PROCLIB data sets on DDNAME MHLRES1, the syntax is very similar to that for the $ADD PROCLIB command, but using the SET command. The example in Figure 3-36 shows the command to change the HLQ PDSERES to MHLRES1 on the first two data sets, but the third one is left as it was.

```
$T PROCLIB(MHLRES1),DD1=dsn=MHLRES1.PROCLIB1,
DD2=dsn=MHLRES1.PROCLIB2,
DD3=dsn=SYS1.PROCLIB1
```

Figure 3-36 Example of altering JES2 proclib data sets defined to JES2 dynamically by command

3.9.2 JES2 using PDSE data sets as procedure libraries

When a PDSE is shared across multiple systems in a sysplex, PDSE establishes connections between the data set and the SMSPDSE1 address space. A PROCLIB data set can be deleted from the system (which would include recovery activity) only if it is not connected anywhere.

When JES2 is running, it works with the converter/interpreter functions of z/OS. The number of converter/interpreters is defined by the JES2 PCEDEF parameter CNVTNUM operand.

The significance of this is that the JES2 task retains the connection for a PDSE PROCLIB data set until all jobs using that data set have completed (which may be as many jobs as defined by PCEDEF CNVTNUM).

In order to delete a PDSE data set from JES2 control:

- All JES2 instances must be modified to no longer allocate the data set.
- All converter interpreters must have ceased using the data set.

To determine the number of converters, issue the $D PCEDEF command. Figure 3-37 shows an example of the command being issued and the response. The number shown next to CVNTNUM is the number of active converters.

```
$DPCEDEF
$HASP849 PCEDEF 002
$HASP849 PCEDEF  CNVTNUM=2,PURGENUM=2,PSONUM=2,OUTNUM=2,
$HASP849         STACNUM=2,SPINNUM=3
```

Figure 3-37 Example of $D PCEDEF with results

To satisfy the deletion conditions, it is necessary to:

- Issue PROCLIB **delete** or **modify** commands on all JES2 instances so that the PDSE data set in question is no longer defined to any JES2 instance.
- All converter/interpreters must have processed a job other than the ones running when the PDSE data set was removed from the JES2 setup. For example, if CNVTNUM was defined as 10, then each one of the 10 converter/interpreters would have to process a job in order for the PDSE connection to be dropped.
- The number of connections to a PROCLIB data set can be found by issuing the $D PROCLIB,DEBUG command. Remember that only dynamically allocated PROCLIB data sets will be displayed. Figure 3-38 shows an example of the command being issued, using (*) as the PROCLIB DDNAME specification (the wildcard (*) means display them all), together with the response. Note that in this case, the USECOUNT=0 indicates that any of the data sets allocated to DDNAME MHLRES1 may be deleted.

```
$D PROCLIB(*),DEBUG
$HASP319 PROCLIB(MHLRES1) 032
$HASP319 PROCLIB(MHLRES1)   USECOUNT=0,DDNAME=SYS00008,
$HASP319                    CREATED=2004.321,
$HASP319                    19:33:19.48,
$HASP319                    DD(1)=(DSNAME=
$HASP319                    MHLRES1.PROCLIB2),
$HASP319                    DD(2)=(DSNAME=
$HASP319                    MHLRES1.PROCLIB1),
$HASP319                    DD(3)=(DSNAME=SYS1.PROCLIB1)
```

Figure 3-38 Example of $D PROCLIB(*),DEBUG with response

- If the USECOUNT is not equal to zero, then some jobs are still using the PROCLIB. As they finish, the usecount will decrease.
- Knowing the number of converters using the CNVTNUM value as discussed previously, it is possible to cause all converters to be exercised. If the same number, or more, of dummy jobs were submitted to the system, then all converters will be exercised after the affected data set has been removed from the JES2 instances. When that has happened, all PDSE connections should have been released, and the data sets can then be deleted.

4

Converting to PDSE format

This chapter discusses the types of data sets that should and should not be converted to PDSEs. It presents methods of converting between PDSs and PDSEs.

4.1 When to use a PDSE

A data set should be considered for PDSE allocation or conversion when one or more of the characteristics listed in Table 4-1 apply.

Table 4-1 PDS and PDSE conversion table

When to use PDSE or PDS	
PDSE	► Frequent compresses are needed for the data set.
	► Out-of-space abends occur often.
	► A large directory leads to lengthy directory searches. (PDS directory searches are sequential while PDSE searches are indexed.)
	► The directory size is unknown at allocation time.
	► The data set is shared for output among many users, even in different systems.
	► The directory size will grow considerably.
	► Members will be shared and reused many times.
	► Protection is needed to prevent users from overwriting the directory or changing DCB attributes for members.
	► You need more than 16 extents (up to 123).
	► You need more than 32767 external symbols in a program.
	► You want a CSECT or load module that is more than 16 MB in size.
	► Common system data sets: See 4.1.1, “Common system data sets” on page 69.
	► ISPF data sets: See 4.1.2, “ISPF data sets” on page 70.

When to use PDSE or PDS	
PDS	► Data sets that use PDS note lists must remain PDSs. One such example is the IMS™ ACBLIB.
	► Block boundaries are not retained in members with imbedded short blocks (that is, blocks other than the last) in PDSEs. In effect, the system automatically reblocks to the block size that you have specified.
	► Null segments are discarded from PDSE members.
	► You should checkpoint/restart jobs with open partitioned data sets at the time of a checkpoint (because PDSEs cannot be open during a checkpoint).
	► If a data set must be aware of the physical disk, a PDS should be used. Some examples are: – An application using the TRKCALC macro – An unmovable data set
	► A partitioned data set that is listed in the master scheduler JCL (either in SYS1.LINKLIB(MSTRJCLxx) or SYS1.PARMLIB(MSTJCLxx) and therefore used during IPL must be a PDS. Examples of such data sets are: – SYS1.LPALIB – SYS1.NUCLEUS – SYS1.SVCLIB – SYS1.PARMLIB – Initial LNKLST data set *SYS1.PROCLIB must be a PDS although other procedure libraries can be PDSEs.*
	► If the OPTCD=W DCB parameter is required, the data set must be a PDS.
	► If EXCP, EXCPVR, or XDAP access methods are used, the data set must be a PDS. These access methods are not supported for PDSEs.
	► SYS1.IMAGELIB cannot be a PDSE because a DEB is built without going through allocation and without a test for a PDSE.
	► A program such as a debugger that reads load modules directly from a PDS will not be able to do so for PDSEs. However, the program management binder provides special services for this purpose.
	► In some specific circumstances (for example, if you have thousands of members in your PDS or PDSE and you need to access it via ISPF OPT 3.4), the first open PDSE will have an overhead because directory pages are not in cache for the first access. Every subsequent access should be much faster as long as the PDSE is not closed in between. The overall performances may be better using a PDS. Assuming that you access the PDSE more than once, sharing it among systems or users, the overall performances should be better using PDSE compared to PDS.

4.1.1 Common system data sets

Many partitioned data sets serve as common system data sets to all system users. Large system data sets will take advantage of the indexed directory in a PDSE to reduce the search time for a member. Some examples of common system data sets that may be converted to PDSEs are:

- JES macro and source libraries
- SYS1.AMODGEN
- SYS1.MACLIB
- SYS1.FONT38PP
- SYS1.FONT3820

4.1.2 ISPF data sets

ISPF panel libraries, ISPPLIB, are frequently accessed and can use PDSE data spaces and Hiperspaces to reduce I/O. This can result in a significant I/O reduction on a system with high TSO activity.

However, ISPF profile data sets are better left as PDSs because ISPF frequently issues requests to update the members in place, which is extremely efficient. Update in place is not used for PDSE profile data sets.

4.1.3 z/OS products using PDSE

The number of products shipped with PDSE data sets is increasing. They include:

- Java LE
- WebSphere®
- DB2

4.1.4 When to use a PDS

Not all partitioned data sets should be converted to PDSEs. A PDS consisting of many small members will occupy more space if converted to a PDSE because each member will occupy a 4 KB page.

Refer to 5.9, “PDSE restrictions” on page 114, for more about data sets that cannot be in PDSE format.

4.2 Converting a PDS to a PDSE

You can convert data sets implicitly or explicitly. An implicit conversion uses SMS functions to create some, or all, new partitioned data sets in PDSE format. An explicit conversion converts existing data sets from one format to another through commands and job streams. This section discusses both methods of conversion.

All conversions to or from a PDSE are in the form of an allocate and copy. There is no way to convert-in-place. It is possible to make the conversion appear as a convert-in-place as shown in 4.2.2, “Explicit conversion” on page 72. The PDSEs are given the same names as the PDSs, and the PDSs are then deleted.

4.2.1 Implicit conversion

As applications and users allocate new partitioned data sets, the installation can implicitly create them as PDSEs. The implicit migration can be controlled through installation defaults or through SMS data class definitions and ACS routines.

Installation default

The installation can define the default library type to be used in the IGDSMSxx member of SYS1.PARMLIB. If DSNTYPE=LIBRARY is specified, all allocations for a partitioned data set will default to creating PDSEs. If DSNTYPE=PDS is specified, or left blank, an allocation for a partitioned data set becomes a PDS. An allocation for a partitioned data set is one where DSORG=PO is specified or a directory block allocation request is made in the SPACE parameter (Figure 4-1 on page 71).

```
SPACE=(CYL,(5,2,10))
```

Figure 4-1 PDS allocation

If no other options are specified, the system default determines the type of data set. The data class request for a PDS or PDSE overrides the installation default selection. If PDSE or PDS is selected in JCL, ISPF, TSO/E, or dynamically with SVC 99, it overrides both the data class and installation default.

Figure 4-2 shows how to allocate a PDSE using ISPF panel option 3.2. See 5.2, “Allocating a PDSE” on page 88 for more information.

```
Command ===>
                                                                More:
Data Set Name  . . . : PDSERES.SMS.LIBRARY

Management class . . .                (Blank for default management class)
Storage class  . . . .                (Blank for default storage class)
 Volume serial . . . .                (Blank for system default volume) **
 Device type . . . . .                (Generic unit or device address) **
Data class . . . . . .                (Blank for default data class)
 Space units . . . . . TRACK          (BLKS, TRKS, CYLS, KB, MB, BYTES
                                       or RECORDS)
 Average record unit                  (M, K, or U)
 Primary quantity  . . 2              (In above units)
 Secondary quantity    1              (In above units)
 Directory blocks  . . 0              (Zero for sequential data set) *
 Record format . . . . FB
 Record length . . . . 80
 Block size  . . . . .
 Data set name type  : LIBRARY        (LIBRARY, HFS, PDS, or blank)  *
                                      (YY/MM/DD, YYYY/MM/DD
 Expiration date . . .                 YY.DDD, YYYY.DDD in Julian form
Enter "/" to select option             DDDD for retention period in days
   Allocate Multiple Volumes           or blank)

( * Specifying LIBRARY may override zero directory block)

( ** Only one of these fields may be specified)
```

Figure 4-2 ISPF example to allocate a PDSE using OPT 3.2

Data class definition

A data class can be defined to allocate PDSEs. The DSNTYPE field identifies a data set as a PDSE.

ACS routines

A read-only variable, &DSNTYPE, is available to identify a partitioned data set as a PDSE. If &DSNTYPE is LIBRARY, the data set is a PDSE. The &DSNTYPE variable can be used with the &DSORG variable. Figure 4-3 on page 72 shows an example.

```
/*********************************************************************/
/* SC SELECTION ROUTINE FOR PDSE DATASETS                            */
/*********************************************************************/

    IF &DSNTYPE = 'LIBRARY'
    THEN DO
    SET &STORCLAS = 'SCSMS'
    EXIT
    END
```

Figure 4-3 Sample ACS routine for Storage Class

For SMS-managed PDSEs, the storage class ACS routine should be modified to assign GSW (guaranteed synchronous write) and MSR (millisecond response) time parameters properly.

For more information about establishing an SMS environment for PDSE data sets, refer to 3.8, “SMS considerations” on page 49.

4.2.2 Explicit conversion

You can use an explicit conversion to convert existing data sets with IEBCOPY or DFSMSdss. You can convert groups of data sets with DFSMSdss according to the filter criteria you specify. If the PDS characteristics have to be changed, IEBCOPY can convert the data set and change the attributes you select.

Using DFSMSdss

When you convert a PDSE to a PDS with DFSMSdss, the data set may be reblocked. If the PDSE has a block size of 32720 and if you use DFSMSdss JCL in Figure 4-4 on page 73, the resulting PDS is reblocked to system-determined block size. The PDS block size is a physical block size. For more about PDSE block sizes, see 5.1, “PDSE space usage” on page 80.

You can code the DFSMSdss reblock exit, ADRREBLK, to override the PDS block size selected by DFSMSdss. For more information about the reblock exit, see *z/OS V1R6.0 DFSMSdss Storage Administration Reference,* SC35-0424.

When the data set is filtered by DSORG, you can select PDSE as a keyword in these cases:

- When `BY(DSORG,EQ,PDS)` is specified, only PDSs are selected.
- When `BY(DSORG,EQ,PDSE)` is specified, only PDSEs are selected.
- When `BY(DSORG,EQ,PAM)` is specified, both PDSs and PDSEs are selected.

Figure 4-4 on page 73 shows an example of using the filtering statements to convert all of a user’s PDSs that do not have a low level qualifier (LLQ) of LOAD.

```
//USERID   JOB ...
//*
//*  CONVERT ALL PDSS FOR USER TO PDSES,
//*  WITH THE EXCEPTION OF PDSS WITH A LOW LEVEL QUALIFIER OF '.LOAD'
//*
//STEP1    EXEC  PGM=ADRDSSU
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD    *
 COPY DATASET( -
        INCLUDE(USER.**) -
        EXCLUDE(**.LOAD) -
        BY(DSORG,EQ,PDS) -
                ) -
      CONVERT(PDSE(**)) -
      CATALOG -
      DELETE PURGE -
            REPLACE -
            WAIT(2,2)
```

Figure 4-4 Sample JCL to convert PDSs to PDSEs

If a PDSE is updated frequently and is on a volume with high activity, the physical pages may be placed in noncontiguous space. You can use DFSMSdss or IEBCOPY to reorganize the pages into contiguous space. Figure 4-5 shows an example of a DFSMSdss job stream to reorganize the PDSEs on a volume. See 6.3, “IEBCOPY” on page 131 for additional considerations when using IEBCOPY.

```
//USERM     JOB ...
//*
//*  COPY PDSEs TO REORGANIZE THE DATA SET
//*
//REORG    EXEC  PGM=ADRDSSU
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD    *
 COPY DATASET( -
        INCLUDE(USER.**) -
        EXCLUDE(**.LOAD) -
        BY(DSORG,EQ,PDSE) -
                ) -
      CATALOG -
      DELETE PURGE -
      REPLACE -
      CANCELERROR -
      WAIT(2,2)
/*
```

Figure 4-5 Using DFSMSdss to reorganize a PDSE

These job streams can be invoked automatically on a volume basis by using the DFSMShsm space management volume exit, ARCMVEXT. When this exit is used, DFSMSdss must use the LOGINDDNAME or LOGINDYNAM parameter to execute on a volume basis (Figure 4-6 on page 74).

```
//USERM     JOB ...
//*
//*  COPY PDSES TO REORGANIZE THE DATA SET
//*
//REORG    EXEC  PGM=ADRDSSU
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD    *
 COPY DATASET( -
        INCLUDE(USER.**) -
        EXCLUDE(**.LOAD) -
        BY(DSORG,EQ,PDSE) ) -
      LOGINDY(SMS012) -
      CATALOG -
      DELETE PURGE -
      REPLACE -
      CANCELERROR -
      WAIT (2,2)
/*
```

Figure 4-6 Sample parameters for DFSMSdss invoked through DFSMShsm exit

The *z/OS V1R6.0 DFSMShsm Implementation and Customization Guide,* SC35-0418, has more information about the ARCMVEXT exit.

Using IEBCOPY

When you convert a data set, you can change the attributes in the DCB with IEBCOPY. This can be done individually by specifying the new attributes on the DD definition. You can define all attributes (except BLKSIZE and EXPDT) by using the LIKE keyword to refer to a model data set. Specific attributes can be overridden by defining the new attribute with the LIKE keyword. Usually, there is no advantage in coding BLKSIZE because the system selects an optimal value.

In Figure 4-7, the first data set defined has all of the attributes of the model data set. The second data set defined has the attributes of the model data set, except that the record format is variable-blocked.

```
//OUTONE    DD DSN=USERID.PDSE.DATA1,DISP=(NEW,KEEP),
//          LIKE=USERID.PDSE.MODEL
//OUTTWO    DD DSN=USERID.PDSE.DATA2,DISP=(NEW,KEEP),
//          LIKE=USERID.PDSE.MODEL,
//          RECFM=VB
```

Figure 4-7 Examples of the LIKE keyword

Because a PDS and PDSE use space differently, you might consider using IEBCOPY to:

- Reduce the secondary allocation for a PDSE.
- Let the system determine the block size.
- Increase the number of directory blocks when converting from a PDSE to a PDS.
- Change the allocation units from tracks or cylinders to device-independent units of KB, MB, GB, or a number of records.
- Assign new data class, storage class, or management class.

Figure 4-8 shows an example of using IEBCOPY to change the secondary allocation units. This job:

- Creates a PDSE with new space allocations.
- Keeps all other attributes (*except logical block size)* the same as the PDS.
- Copies the data set from the PDS to PDSE.
- Deletes the original PDS.

```
//PDSERESY  JOB  ...
//S1  EXEC  PGM=IEBCOPY
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD DSN=PDSERES.PDS.CNTL,DISP=(OLD,DELETE,KEEP)
//SYSUT2    DD DSN=PDSERES.PDSECOP.CNTL,DISP=(NEW,KEEP),
//          LIKE=PDSERES.PDS.CNTL,UNIT=3390,
//          SPACE=(80,(8,8,15)),AVGREC=K,
//          DSNTYPE=LIBRARY
```

Figure 4-8 Converting a PDS to PDSE with IEBCOPY

Refer to 9.4.4, “IEBCOPY performance considerations” on page 187 for more considerations.

ISPF considerations

You may want to convert a PDS to a PDSE on an individual basis with ISPF. Such a conversion would consist of allocating a new data set as a PDSE and copying the members of the PDS to the PDSE.

The allocation quantities and units can be changed easily on the new data set to allow for functional differences between PDSs and PDSEs. For example, because a PDSE can contain up to 123 extents, you may want to reduce the secondary allocation quantity.

4.3 Converting a PDSE to a PDS

You may need to convert a PDSE to a PDS in the *unlikely* situation that you need to use an application that does not run against a PDSE.

4.3.1 Using DFSMSdss

Figure 4-9 illustrates the JCL to convert a single PDSE to a PDS with DFSMSdss.

```
//PDSERESD JOB  ...
//CONVERT  EXEC  PGM=ADRDSSU
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD    *
     COPY DS(INC(PDSERES.SMS.PDSE3)) -
          CONVERT(PDS(PDSERES.SMS.PDSE3)) -
          RENAMEU(PDSERES.SMS.PDSE3, -
                  PDSERES.PDS.CONVERT.CNTL )
/*
```

Figure 4-9 Converting a PDSE to a PDS using DFSMSdss

If you need to convert a set of PDSEs, you should use generic filtering with the conversion. Figure 4-10 shows an example of a two-step job that uses generic filtering. This job can be used to dump a set of PDSEs. This job:

1. Selects the PDSEs to be converted, by DSORG and data set name.
2. Converts the selected data sets to PDS.
3. Deletes and replaces the source.
4. Dumps the PDSs to tape.

```
//USERIDV  JOB   ...
//CVTPDS   EXEC  PGM=ADRDSSU,REGION=6M
//SYSPRINT DD  SYSOUT=*
//TAPE     DD UNIT=3590,VOL=SER=TAPE01,
//         LABEL=(1,SL),DISP=(NEW,CATLG),DSNAME=PDSERES.PDS.DUMP
//SYSIN    DD    *
      COPY DS(INC(PDSERES.**) -
              EXC(**.LOAD)  -
              BY(DSORG,EQ,PDSE)) -
           CONVERT(PDS(PDSERES.**)) -
           REPLACE -
           CATALOG -
           DELETE PURGE -
           CANCELERROR -
           WAIT(2,2)
      DUMP DS(INC(PDSERES.**)) -
           OUTDD(TAPE) -
           TOL(ENQF)
```

Figure 4-10 Converting and dumping a set of PDSEs

The job in Figure 4-11 selects a group of PDSs to be converted and converts them to PDSEs.

```
//USERA    JOB ...
//CVTPDSE  EXEC  PGM=ADRDSSU,REGION=6M
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD    *
     COPY DS(INC(PDSERES.**) -
             EXC(**.LOAD)  -
             BY(DSORG,EQ,PDS)) -
          CONVERT(PDSE(PDSERES.**)) -
          CATALOG  -
          DELETE PURGE -
          REPLACE -
          CANCELERROR -
          WAIT(2,2)
```

Figure 4-11 Converting a group of PDSs to PDSE

4.3.2 Using IEBCOPY

To use IEBCOPY to convert a PDSE to a PDS, you can use a technique similar to the opposite conversion that was described in 4.2.2, “Explicit conversion” on page 72. You might consider:

- Increasing the secondary allocation amount for the PDS.
- Letting the system determine the block size. The optimum values for a PDS and for a PDSE are different.
- The default number of directory blocks from the PDSE will probably be too low for the PDS.

4.4 Non-SMS PDSEs

You can convert an existing managed PDSE to a non-managed PDSE by means similar to those shown earlier in this chapter. You can use DFSMSdss, DFSMShsm, or IEBCOPY to create a new copy of the PDSE on a non-SMS volume.

Depending on how your ACS routines are written, you may have to supply a dummy storage class that will be converted to a null storage class in your storage class ACS routine. You should specify a non-SMS volume, as well.

4.5 How to convert load modules to program objects

You can use IEBCOPY to convert between program objects and load modules. For details, refer to 7.2.7, “How to convert load modules to program objects” on page 150.

Conversion from program objects to load modules is not possible if the program object exploits facilities not available with load modules; for example, more than 16 MB of executable code, or alias name is longer than eight characters.

5

Using PDSEs

This chapter begins by describing the space requirements for PDSE and compares several sample PDSEs to corresponding PDSs. It then discusses the various ways to create a PDSE, and how PDSEs are used or supported by other IBM software products. The last section summarizes the restrictions on using PDSEs.

This information is intended mainly for end users as an overview of how to use and manage PDSE.

5.1 PDSE space usage

A PDSE differs from a PDS in the way space is allocated and used. This section discusses the way a PDSE uses space, describes how to allocate space for a PDSE, and compares the space requirements of some sample PDSs and PDSEs.

Several factors affect space utilization in a PDSE. Compared to a PDS, there are both advantages and disadvantages in the way a PDSE uses space. However, when considering the space differences, remember that a PDSE has these functional advantages: it does not have to be compressed, it has an expandable directory so that space planning is less critical, and it can be shared more widely.

This section covers some areas to consider when determining the space requirements for a PDSE. See *DFSMS Using Data Sets*, SC26-7410, for additional information about space requirements for PDSEs.

5.1.1 Use of noncontiguous space

When members are created, the member pages use contiguous space whenever possible to avoid fragmenting the members. However, noncontiguous space may be used if contiguous space is not available. This may be reclaimed space from deleted members. This is a clear advantage over PDSs, which require all member space to be contiguous and do not reclaim space from deleted members without explicit intervention (such as compress).

5.1.2 Integrated directory

All PDSE space is available for either directory or member use (except the first five pages, which are directory pages). Within a data set, there is no difference between pages used for the directory and pages used for members. As the data set grows, the members and directory will have the same space available for use. The directory, or parts of it, may be placed in secondary extents.

Directory pages are a factor in determining space requirements; they are a separate calculation.There is no longer a separate space requirement for directory pages. A PDSE does not have preallocated, unused directory pages. Directory space is automatically expanded as needed.

The format of a PDSE enables the directory to contain more information. This information may take more space than a PDS directory block. The PDSE directory contains keys (member names) in a compressed format. The insertion or deletion of new keys may cause the compression of other directory keys to change. Therefore the change in the directory size may be different from the size of the inserted or deleted directory record.

5.1.3 Full block allocation

A PDSE member is allocated in full-page increments. If a short logical block is created, the member is maintained on full-page boundaries, and any remaining space in the page will be unused. PDSE members are stored in 4 KB pages, and each member takes at least one page.

5.1.4 PDS gas and PDSE unused space

PDS gas is the unreclaimed space in a PDS that was vacated when members were deleted or rewritten. Users often overallocate their PDSs to allow for the inevitable amount of PDS gas

that would develop over time. With PDSEs, you do not need to add as much additional space to the allocation to allow for growth of the data set.

Studies show that a typical installation has 18% to 30% of its PDS space in the form of gas. This space is unusable to the data set until it has been compressed. A PDSE dynamically reuses all of the allocated space according to a first-fit algorithm. You do not need to make any allowance for gas when you allocate space for a PDSE.

If a deleted or updated member still has an existing connection from another task, the member space is not reclaimed until the connection is released, the data set is opened for output, and that OPEN for OUTPUT is the only one on that system.

ABEND D37 can occur on a PDSE indicating that it is FULL, but another member can still be saved in the data set. Recovery processing from an ABEND D37 in ISPF closes and reopens the data set. This new open of the data set may enable PDSE code to reclaim space so a member can be saved. Note that the OPEN for OUTPUT must be the only one, so the space reclamation may not work.

5.1.5 Frequency of data set compression

Data set compression is the process by which gas is removed from a PDS.

A PDSE dynamically reuses all the allocated space, so data set compression cannot be run with a PDSE. Because there is no gas accumulation in a PDSE, there is no need to compress the data set.

Tip: PDSEs can become fragmented depending on the access pattern. This does not normally occur when the adding and deleting of members is balanced, but might occur when members are deleted and new members are not added to reclaim the space. This can affect *performance*. To reorganize the PDSE, use IEBCOPY or DFSMSdss COPY to back up all the members. You can either delete and restore all members, or delete and reallocate the PDSE. It is preferable to delete and reallocate the PDSE because it usually uses less processor time and does less I/O than deleting every member.

5.1.6 Extent growth

A PDSE may have up to 123 extents. Because a PDSE can have more secondary extents than a PDS, you can get the same total space allocation with a smaller secondary allocation. A PDS requires a secondary extent about eight times larger than a PDSE to have the same total allocation. Conversely, for a given secondary extent value, PDSEs can grow about eight times larger before having to be condensed. Condensing is the process by which multiple small extents are consolidated into fewer large extents. This operation can be performed directly from the ISMF data set list, using the CONDENSE line operator command.

For example, the two DD cards in Figure 5-1 would produce data sets that each have a potential maximum space of about 1.82 MB.

```
//DDCARD1 DD  ...,SPACE=(80,(20,8,5)),AVGREC=K,DSNTYPE=PDS

//DDCARD2 DD ...,SPACE=(80,(20,1,5)),AVGREC=K,DSNTYPE=LIBRARY
```

Figure 5-1 PDSE and PDS allocation sample

5.1.7 Logical block size

The logical block size may affect the performance of a PDSE. In general, a large block size improves performance. It is recommended that you use a system-determined block size for allocating data sets. The system chooses the optimal block size for each data set. For variable-length records, a PDSE is given the maximum block size of 32760 bytes. For fixed-length records, the block size is the largest multiple of the record size that is less than or equal to 32760.

Different applications can use different logical block sizes to access the same PDSE. The block size in the DCB is logical and has no effect on the physical block (page) size being used. See Figure 9-2 on page 183 for more information about the effect that logical block size has on performance.

5.1.8 Physical block size

PDSEs use a physical block size of 4 KB, the same size as an MVS page. These 4 KB physical blocks are also referred to as pages.

The DCB block size does not affect the physical block size, and all members of a PDSE are assumed to be reblockable. For example, if you code your DCB with BLKSIZE=6160, the data set has physical blocks of 4 KB pages but the user program still sees 6160-byte logical blocks.

5.1.9 Partial release, free space

The space for any library can be overallocated. This excess space can be released manually with the FREE command in ISPF or, to prevent over-allocation at allocation time, you could code the release (RLSE) parameter on your JCL, which will cause release when the data set is opened for output.

For managed data sets, RLSE is complemented by the partial release parameter in the management class, which calls DFSMShsm to release the space during the daily space management cycle. This function works the same way for PDSEs as it does for PDSs or sequential data sets.

The partial release parameter frees the never-used space between the HFRFN (high formatted relative frame number) and the “high allocated” space in a PDSE data set, as shown in Figure 5-2 on page 83. Figure 2-5 on page 21 and “High formatted relative frame number” on page 21 also include information regarding the HFRFN.

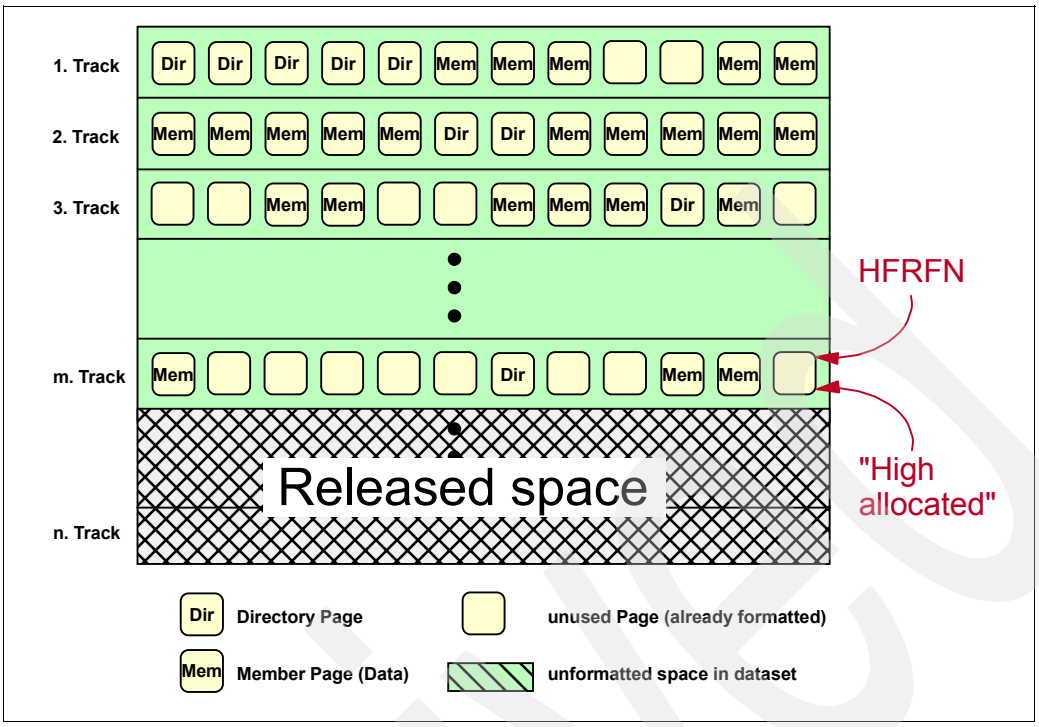


Figure 5-2 Partial release for a PDSE

The partial release function does not free unused space from deleted members (occupying space below the HFRFN) nor does it consolidate extents.

5.1.10 Fragmentation

Most allocation units are approximately the same size. This is because of the way members are buffered and written in groups of multiple pages. There is very little, if any, fragmentation in a PDSE.

If there is fragmentation, copy the data set with IEBCOPY. The fragmented members are combined in the new copy. Unused space is not copied. (See 5.1.5, “Frequency of data set compression” on page 81.)

5.1.11 Directory blocks on the JCL SPACE parameter

The syntax of the JCL SPACE keyword parameter is the same for both a PDS and a PDSE. A typical DFSMS allocation looks similar to Figure 5-3.

```
SPACE=(reclgth,(primary-qty,secondary-qty,directory)),AVGREC=({U|K|M})
```

Figure 5-3 SPACE parameter for PDSE allocation

When the data set is identified as a PDSE with the JCL keyword DSNTYPE=LIBRARY, the directory blocks field in the SPACE parameter is optional and not used for the PDSE allocation. Directory blocks are added dynamically as the number of members increases.

Note: This is not always true when using IEBCOPY. See 6.3.3, “IEBCOPY Reload” on page 133, for details about IEBCOPY’s behavior with and without the directory blocks subparameter.

For compatibility reasons, you may want to continue to specify directory blocks in PDSE allocations in case the PDSE might be converted to a PDS. The directory blocks to be used at conversion time are determined by the number requested at PDSE allocation time. If the actual number used exceeds the number requested or if none was specified, the total number actually required by the number of members in the data set will be allocated.

The two examples in Figure 5-4 are based on the following assumptions:

- Data set attributes are copied using: `LIKE=HLQ.PDSE.DATA`.
- Space is allocated as: `SPACE=(80,(24,1,5)),AVGREC=K`.
- Conversion to a PDS is made with: DSNTYPE=PDS.
- One PDS directory block holds seven members.

```
Example 1: With 20 members in the data set, three directory blocks are used. Five
directory blocks are allocated in the conversion. There is room in the directory for 15
more members to be added.

Example 2: With 50 members in the data set, eight directory blocks are used. Eight
directory blocks are allocated in the conversion. You have space to add six more members.
```

Figure 5-4 Example for directory blocks

5.1.12 Sample comparisons of PDSs to PDSEs

This section compares PDS and PDSE versions of sample partitioned data sets from a production system. Both the PDSs and PDSEs were allocated with the same primary quantities. These started with initial allocations with free space removed and no PDS gas. Members were then edited, added, deleted, and copied to both the PDS and the PDSE versions of the data sets. This usage of the data sets resulted in approximately 15% to 20% gas for the PDSs. This amount of gas is not atypical of the amount we observe on many production systems. The PDSE, by its very nature, did not develop any gas.

For the purpose of these comparisons, if the data set had not been allocated in tracks on the production system, it was converted to tracks for the PDSE testing. This enabled the data sets to be compared at the lowest level of physical allocation.

The directory block field does not apply to PDSE, but it is included for comparison. Likewise, the number of pages used refers to the physical allocation of the PDSE, but it is not relevant to a PDS.

JCL library

The first data set is a system programmer’s JCL library. It was allocated with these characteristics:

- There are 84 members in the data set.
- Each member averages less than 40 records.
- The block size is system-determined.
- The record format is fixed block (RECFM=FB).
- The logical record length is 80 bytes (LRECL=80).

Table 5-1 shows the data set characteristics for the two versions of the data set.

Table 5-1 PDS to PDSE comparison for a JCL library

Data set name	Type	Directory blocks	Tracks	Pages	Block size
USER.PDS.CNTL	PDS	15	10	-	23440
USER.PDSE.CNTL	LIBRARY	-	12	115	32720

Figure 5-5 and Figure 5-6 show the ISPF 3.4 Data Set Information panels for these data sets.

```
-------------------------  Data Set Information  -----------------------------
Command ===> _

Data Set Name: USER.PDS.CNTL

General Data:                              Current Allocation:
   Management class:          PRIMARY          Allocated tracks:             10
   Storage class:             NORMAL           Allocated extents:             2
      Volume:                 SYS107           Maximum dir. blocks:          15
      Device type:            3380
   Data class:
      Organization:           PO            Current Utilization:
      Record format:          FB               Used tracks:                  10
      Record length:          80               Used extents:                  2
      Block size:             23440            Used dir. blocks:             15
      1st extent tracks:      7                Number of members:            84
      Secondary tracks:       3
      Data set name type:     PDS
```

Figure 5-5 ISPF information screen for JCL library as PDS

```
-------------------------  Data Set Information  -----------------------------
Command ===> _

Data Set Name: USER.PDSE.CNTL

General Data:                              Current Allocation:
   Management class:          PRIMARY          Allocated tracks:             12
   Storage class:             NORMAL           Allocated extents:             6
      Volume:                 SYS100           Maximum dir. blocks:      NOLIMIT
      Device type:            3380
   Data class:
      Organization:           PO            Current Utilization:
      Record format:          FB               Used pages:                  115
      Record length:          80               % Utilized:                   96
      Block size:             32720            Number of members:            84
      1st extent tracks:      7
      Secondary tracks:       1
      Data set name type:     LIBRARY
```

Figure 5-6 ISPF information screen for JCL library as PDSE

SYS1.MACLIB

The macro library SYS1.MACLIB was allocated with the following characteristics:

- There are 1742 members in the data set.
- Each member averages 250 records.
- The block size is 3120 bytes.
- The record format is fixed block (RECFM=FB).
- The logical record length is 80 bytes (LRECL=80).

Table 5-2 shows the data set characteristics for the two versions of the data set.

Table 5-2 PDS to PDSE comparison for a macro library

Data set name	Type	Directory blocks	Tracks	Pages	Block size
USER.PDS.MACLIB	PDS	113	1300	-	3120
USER.PDSE.MACLIB	LIBRARY	-	1200	11851	3120

Figure 5-7 and Figure 5-8 show the ISPF 3.4 information screens for these data sets.

```
------------------------  Data Set Information  -----------------------------
Command ===> _

Data Set Name: USER.PDS.SYS1.MACLIB

General Data:                          Current Allocation:
   Management class:          LARGE        Allocated tracks:          1,300
   Storage class:             NORMAL       Allocated extents:             4
      Volume:                 SYS208       Maximum dir. blocks:         184
      Device type:            3380
   Data class:
      Organization:           PO        Current Utilization:
      Record format:          FB           Used tracks:               1,263
      Record length:          80           Used extents:                  4
      Block size:             3120         Used dir. blocks:            113
      1st extent tracks:      1000         Number of members:          1742
      Secondary tracks:       100
      Data set name type:     PDS
```

Figure 5-7 ISPF information screen for macro library as PDS

```
------------------------  Data Set Information  -----------------------------
Command ===> _

Data Set Name: USER.PDSE.MACLIB

General Data:                          Current Allocation:
   Management class:          LARGE        Allocated tracks:          1,200
   Storage class:             NORMAL       Allocated extents:             3
      Volume:                 SYS207       Maximum dir. blocks:      NOLIMIT
      Device type:            3380
   Data class:
      Organization:           PO        Current Utilization:
      Record format:          FB           Used pages:                11851
      Record length:          80           % Utilized:                   98
      Block size:             3120         Number of members:          1742
      1st extent tracks:      1000
      Secondary tracks:       100
      Data set name type:     LIBRARY
```

Figure 5-8 ISPF information screen for macro library as PDSE

SYS1.FONT3820

This comparison is between copies of the font library SYS1.FONT3820, which has fonts for IBM Advanced Function Printing™ (AFP™) devices. It was allocated with these characteristics:

- There are 2515 members in the data set.
- Each member averages 20 records.
- The block size is 23476 bytes.
- The record format is variable block (RECFM=VB).
- The logical record length is 23472 bytes (LRECL=23472).

Table 5-3 shows the data set characteristics for the two versions of the data set.

Table 5-3 PDS to PDSE comparison for an AFP font library

Data set name	Type	Directory blocks	Tracks	Pages	Block size
USER.PDS.FONT3820	PDS	143	1138	-	23476
USER.PDSE.FONT3820	LIBRARY	-	1200	11851	23476

Figure 5-9 and Figure 5-10 on page 88 show ISPF 3.4 information screens for these data sets.

```
------------------------  Data Set Information  -----------------------------
Command ===> _

Data Set Name: USER.PDS.FONT3820

General Data:                          Current Allocation:
   Management class:      LARGE            Allocated tracks:          1138
   Storage class:         NORMAL           Allocated extents:            4
      Volume:             SYS213           Maximum dir. blocks:        184
      Device type:        3380
   Data class:
      Organization:       PO         Current Utilization:
      Record format:      VBM              Used tracks:               1138
      Record length:      23472            Used extents:                 4
      Block size:         23476            Used dir. blocks:           143
      1st extent tracks:  900              Number of members:         2515
      Secondary tracks:   100
      Data set name type: PDS
```

Figure 5-9 ISPF information screen for AFP font library as PDS

```
------------------------  Data Set Information  -----------------------------
Command ===> _

Data Set Name: USER.PDSE.FONT3820

General Data:                              Current Allocation:
   Management class:          LARGE           Allocated tracks:          1,102
   Storage class:             NORMAL          Allocated extents:             4
      Volume:                 SYS214          Maximum dir. blocks:      NOLIMIT
      Device type:            3380
   Data class:
      Organization:           PO           Current Utilization:
      Record format:          VBM             Used pages:               10776
      Record length:          23472           % Utilized:                  97
      Block size:             23476           Number of members:         2515
      1st extent tracks:      900
      Secondary tracks:       100
      Data set name type:     LIBRARY
```

Figure 5-10 ISPF information screen for AFP font library as PDSE

5.2 Allocating a PDSE

A new PDSE can be allocated by batch, TSO/E, ISPF, or by using dynamic allocation services that call SVC 99. These allocations can be used with an installation default and data class definitions.

A keyword, DSNTYPE, is available in each case to identify a data set as a PDSE. If DSNTYPE is specified as LIBRARY at allocation, the data set is created in PDSE format. If DSNTYPE is specified as PDS, the data set is created in PDS format.

A PDSE does not use predefined directory blocks, so the specification of directory blocks at allocation is optional. See 5.1, “PDSE space usage” on page 80 for more information.

5.2.1 Batch allocation

The DSNTYPE JCL keyword specifies that a data set should be a PDSE or a PDS. If DSNTYPE=LIBRARY is specified, the data set is a PDSE. If DSNTYPE=PDS is specified, the data set is a PDS. If DSNTYPE is not specified and a DSORG=PO is specified, the data set is created in the format specified by the data class. If the data class does not specify DSNTYPE, the installation default in SYS1.PARMLIB(IGDSMSxx) is used.

All the parameters you use to allocate a PDS are valid for allocating a PDSE. In Figure 5-11, a PDSE is allocated by adding the keyword DSNTYPE=LIBRARY.

```
//USERA    JOB  ....
//         EXEC  PGM=IEFBR14
//SYSIN    DD    DSN=PDSERES.PDSE.PDSE01,DISP=(NEW,KEEP),
//         DSNTYPE=LIBRARY,
//         RECFM=FB,LRECL=80,
//         SPACE=(80,(10,2,5)),AVGREC=K
```

Figure 5-11 Defining a PDSE with JCL

When directory blocks are not specified, DSORG=PO must be included to identify the data set as partitioned. In Figure 5-12, the directory blocks in the SPACE parameter are omitted.

```
//USERA    JOB   ....
//         EXEC  PGM=IEFBR14
//SYSIN    DD    DSN=USER.PDSE.PDSE02,DISP=(NEW,KEEP),
//         DSNTYPE=LIBRARY,
//         DSORG=PO,
//         RECFM=FB,LRECL=80,
//         SPACE=(80,(10,2)),AVGREC=K
```

Figure 5-12 Omitting directory blocks in JCL to define a PDSE

5.2.2 TSO/E allocation

When allocating a PDSE with the TSO/E ALLOCATE command, you can specify DSNTYPE to identify the data set as PDSE or PDS. If a data set exists with the DCB attributes that you need, you can use the LIKE parameter to copy the attributes to the new PDSE.

Figure 5-13 shows an example of allocating a PDSE using the LIKE parameter to copy the attributes of an existing PDSE.

```
ALLOC DS('USER.PDSE.TSO01') LIKE('USER.PDSE.PDSE01')
```

Figure 5-13 Defining a PDSE with TSO/E using the LIKE parameter

Figure 5-14 shows an example of allocating a PDSE using the LIKE parameter to copy the attributes of an existing PDS. Adding the DSNTYPE(LIBRARY) parameter to the allocation makes the data set a PDSE.

```
ALLOC DS('USER.PDSE.TSO02')
       LIKE('USER.PDS.PDS01') DSNTYPE(LIBRARY)
```

Figure 5-14 Defining a PDSE with TSO/E using the DSNTYPE parameter

See *z/OS V1R6.0 TSO/E Command Reference,* SA22-7782, for more information about allocating a PDSE with TSO/E.

5.2.3 ISPF allocation

A PDSE can be defined through ISPF/PDF by using the Data Set Utility menu. Access this menu by selecting **3.2** from the ISPF/PDF primary option menu. You can also access this menu by entering DSUTIL at the command line of the ISMF Data Set List panel.

Figure 5-15 on page 90 shows the Data Set Utility screen.

```
                              Data Set Utility
Option ===> A

    A Allocate new data set                C Catalog data set
    R Rename entire data set               U Uncatalog data set
    D Delete entire data set               S Short data set information
blank Data set information                 V VSAM Utilities

ISPF Library:
   Project  . . MHLRES3          Enter "/" to select option
   Group  . . . DAMIAN           /  Confirm Data Set Delete
   Type . . . . CNTL

Other Partitioned, Sequential or VSAM Data Set:
   Data Set Name . . . 'PDSERES.SMS.PDSE2'
   Volume Serial . . .           (If not cataloged, required for option "C")

Data Set Password  . .           (If password protected)
```

Figure 5-15 ISPF/PDF Data Set Utility screen option 3.2

Figure 5-16 shows the screen that is displayed when you select option A from the Data Set Utility panel.

```
 Menu  RefList  Utilities  Help
--------------------------------------------------------------------------------
                            Allocate New Data Set
Command ===>
                                                                More:     +
Data Set Name  . . . : PDSRES.SMS.PDSE2

Management class . . .                (Blank for default management class)
Storage class  . . . .                (Blank for default storage class)
 Volume serial . . . . SBOX79         (Blank for system default volume) **
 Device type . . . . .                (Generic unit or device address) **
Data class . . . . . .                (Blank for default data class)
 Space units . . . . . TRACK          (BLKS, TRKS, CYLS, KB, MB, BYTES
                                       or RECORDS)
 Average record unit                  (M, K, or U)
 Primary quantity  . . 3              (In above units)
 Secondary quantity    15             (In above units)
 Directory blocks  . . 45             (Zero for sequential data set) *
 Record format . . . . FB
 Record length . . . . 80
 Block size  . . . . . 27920
Data set name type  : LIBRARY        (LIBRARY, HFS, PDS, or blank)  *
                                      (YY/MM/DD, YYYY/MM/DD
 Expiration date . . .                YY.DDD, YYYY.DDD in Julian form
Enter "/" to select option            DDDD for retention period in days
   Allocate Multiple Volumes          or blank)

( * Specifying LIBRARY may override zero directory block)

( ** Only one of these fields may be specified)
```

Figure 5-16 ISPF/PDF allocate new data set panel

The DATA SET NAME TYPE field specifies whether the new data set is to be a PDSE or PDS. Specify DATA SET NAME TYPE equal to `LIBRARY` to define a PDSE. With a PDSE allocation, the DIRECTORY BLOCKS field is optional. See 5.1, “PDSE space usage” on page 80 for more information about directory blocks.

If you know which Data Class to use, you may allocate a PDSE specifying its name. Figure 5-17 shows an example, starting from ISPF panel option 3.2. See also 5.3.1, “How to find a Data Class for PDSE” on page 96.

```
 Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
                           Allocate New Data Set
Command ===>
                                                                      More:     +
Data Set Name  . . . : PODSERES.SMS.PDSE

Management class . . .                (Blank for default management class)
Storage class  . . . .                (Blank for default storage class)
 Volume serial . . . . MHLV03         (Blank for system default volume) **
 Device type . . . . .                (Generic unit or device address) **
Data class . . . . . . DCPDSE         (Blank for default data class)
 Space units . . . . . CYLINDER       (BLKS, TRKS, CYLS, KB, MB, BYTES
                                       or RECORDS)
 Average record unit                  (M, K, or U)
 Primary quantity  . . 10             (In above units)
 Secondary quantity    0              (In above units)
 Directory blocks  . . 10             (Zero for sequential data set) *
 Record format . . . . FB
 Record length . . . . 80
 Block size  . . . . .
 Data set name type  :                (LIBRARY, HFS, PDS, or blank)  *
                                      (YY/MM/DD, YYYY/MM/DD
 Expiration date . . .                 YY.DDD, YYYY.DDD in Julian form
```

Figure 5-17 How to request a Data Class at allocation time

If you want to request a specific Storage Class at allocation time, specify the LIBRARY parameter to create the PDSE, or request the proper Data Class. However, according to your ACS routines, the requested Storage Class might or might not be assigned to your data set. Selecting option `3.2` from ISPF panel, you may allocate the new PDSE as shown in Figure 5-18.

```
 Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
                           Allocate New Data Set
Command ===>
                                                                    More:     +
Data Set Name  . . . : PDSERES.SMS.JUVE

Management class . . .                (Blank for default management class)
Storage class  . . . . SCGSWYES       (Blank for default storage class)
 Volume serial . . . . MHLV03         (Blank for system default volume) **
 Device type . . . . .                (Generic unit or device address) **
Data class . . . . . .                (Blank for default data class)
 Space units . . . . . CYLINDER       (BLKS, TRKS, CYLS, KB, MB, BYTES
                                       or RECORDS)
 Average record unit                  (M, K, or U)
 Primary quantity  . . 10             (In above units)
 Secondary quantity    1              (In above units)
 Directory blocks  . .                (Zero for sequential data set) *
 Record format . . . . FB
 Record length . . . . 80
 Block size  . . . . .
 Data set name type  : LIBRARY        (LIBRARY, HFS, PDS, or blank)  *
                                      (YY/MM/DD, YYYY/MM/DD
```

Figure 5-18 How to request a Storage Class at allocation time

See 5.3.2, “How to find a Storage Class for a PDSE” on page 97.

5.2.4 How to verify whether a data set is PDSE

If you have an allocated data set and you want to verify whether it is a PDSE, there are several ways to determine this.

First, you could use ISMF panel DGTLGP11, selecting `DATA SET LIST`, as shown in Figure 3-19 on page 54. Column 30, DATA SET NAME TYPE, shows whether the data set is PDSE, specifying it as type of `LIBRARY`.

```
 Panel  List  Dataset  Utilities  Scroll  Help
------------------------------------------------------------------------------
DGTLGP11                          DATA SET LIST
Command ===>                                                 Scroll ===> HALF
                                                        Entries 4-12 of 12
Enter Line Operators below:                             Data Columns 29-31 of 39

     LINE                                    DATA SET     DATA SET   NUM OF
     OPERATOR          DATA SET NAME         ENVIRONMENT  NAME TYPE  STRIPES
    ---(1)----  ------------(2)------------ ---(29)----  --(30)---  -(31)--
                PDSERES.CEE.SCEERUN2.       UNMANAGED    LIBRARY        --
                  RELOADED.CDAEED
                PDSERES.CEE.SCEERUN2.       UNMANAGED    LIBRARY        --
                  RELOADED.CRTEC128
                PDSERES.CEE.SCEERUN2.       UNMANAGED    LIBRARY        --
                  RELOADED.C128
                PDSERES.PDS.CONVERT.CNTL    MANAGED      OTHERS         --
                PDSERES.PDSECOP.CNTL        UNMANAGED    LIBRARY        --
                PDSERES.SCEERUN2.UNLOADED   UNMANAGED    OTHERS         --
                PDSERES.SMS.DIRK.PDSE1      MANAGED      LIBRARY        --
                PDSERES.SMS.PDSE3           MANAGED      OTHERS         --
                PDSERES.SMS.PDSE33          MANAGED      OTHERS         --
```

Figure 5-19 ISMF data set list; column 30 indicates whether you have a PDSE

Another way is to use ISPF. Using OPT 3.4, and moving right with PF11, you could see the DSORG for your listed data sets (Figure 5-20).

```
 Menu  Options  View  Utilities  Compilers  Help
------------------------------------------------------------------------------
ISRUDSLO Data Sets Matching PDSERES.**                        Row 1 of 12
Command ===>                                               Scroll ===> PAGE

Command - Enter "/" to select action              Dsorg  Recfm  Lrecl  Blksz
------------------------------------------------------------------------------
        PDSERES
        PDSERES.CEE.SCEERUN2.RELOADED              PO-E  U          0  32760
        PDSERES.CEE.SCEERUN2.RELOADED.BAD          PS    VS     32760  32760
        PDSERES.CEE.SCEERUN2.RELOADED.CDAEED       PO-E  U          0  32760
        PDSERES.CEE.SCEERUN2.RELOADED.CRTEC128     PO-E  U          0  32760
        PDSERES.CEE.SCEERUN2.RELOADED.C128         PO-E  U          0  32760
        PDSERES.PDS.CONVERT.CNTL                   PO    FB        80  27920
        PDSERES.PDSECOP.CNTL                       PO-E  FB        80  32720
        PDSERES.SCEERUN2.UNLOADED                  PS    VS     32768  32760
        PDSERES.SMS.DIRK.PDSE1                     PO-E  FB        80  32720
        PDSERES.SMS.PDSE3                          PO    FB        80  27920
        PDSERES.SMS.PDSE33                         PO    FB        80  27920
****************************** End of Data Set list ****************************
```

Figure 5-20 ISPF OPT 3.4; DSORG column shows a PDSE as PO-E

Always from an ISPF DATA SET list, you could verify whether you have a PDSE in Data Set Information, specifying “I” in front of your file. The DATA SET NAME TYPE shown in the output tells you the status of your data set, as shown in Figure 5-21.

```
ISRUAILE                    Data Set Information
Command ===>

Data Set Name  . . . : PDSERES.PDSECOP.CNTL

General Data                          Current Allocation
 Management class . . : **None**       Allocated kilobytes : 2,684
 Storage class  . . . : **None**       Allocated extents . : 4
  Volume serial . . . : SBOXEF         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . : **None**
  Organization  . . . : PO            Current Utilization
  Record format . . . : FB             Used pages  . . . . : 599
  Record length . . . : 80             % Utilized  . . . . : 89
  Block size  . . . . : 32720          Number of members . : 3
  1st extent kilobytes: 671
  Secondary kilobytes : 640
  Data set name type  : LIBRARY

  Creation date . . . : 2004/11/08     Referenced date . . : 2004/11/08
  Expiration date . . : ***None***
```

Figure 5-21 ISPF data set information

5.2.5 Dynamic allocation

A PDSE can be allocated through dynamic allocation using SVC 99. Refer to *z/OS V1R6.0 MVS Programming Assembler Services Guide,* SA22-7605, and *z/OS V1R6.0 MVS Programming Authorized Assembler Services Reference Vol 1 (ALESERV-DYNALLOC),* SA22-7609, for information about dynamic allocations of PDSEs.

5.3 SMS definitions and settings

This section shows how to determine the Data Class and Storage Class for a PDSE. In an SMS managed environment the storage administrator usually sets up the classes for a PDSE.

In a DFSMS environment, use SMS classes and groups to set service requirements, performance goals, and data definition models for your installation. Storage administrators use ISMF to create the appropriate classes and groups; ACS routines are provided to assign them to data according to your installation’s policies. Figure 5-22 on page 95 provides an overview of allocation in an SMS environment.

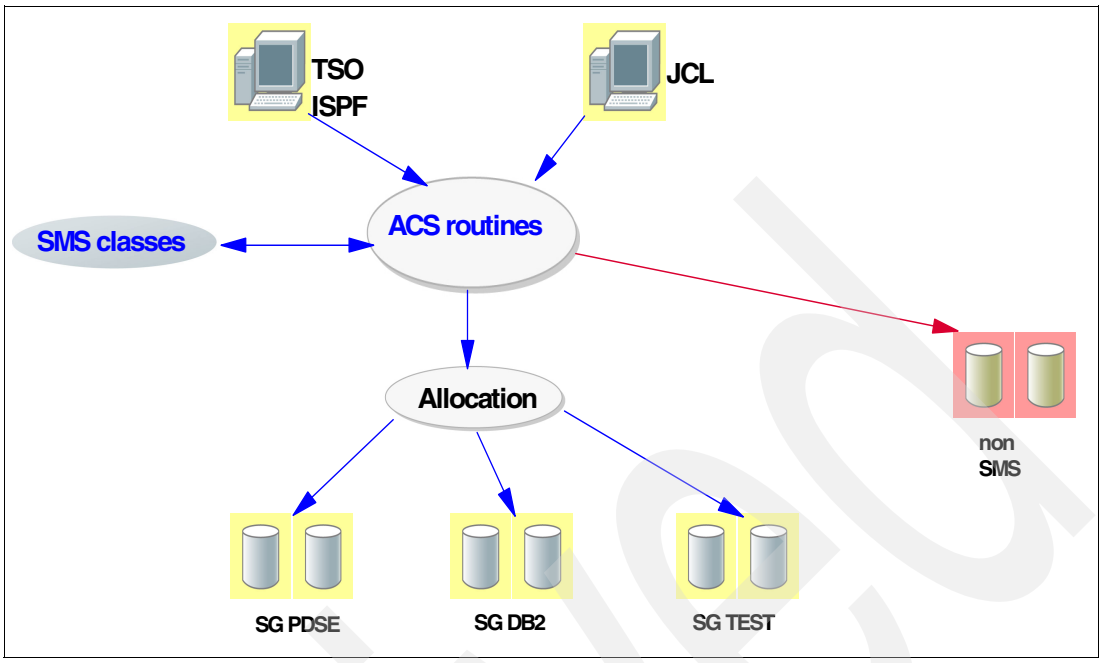


Figure 5-22 Overview of allocation

You should run all of the ISMF LIST commands against your ACTIVE CDS name to query against the active configuration. Otherwise, you could use the SCDS name, but you must be sure to specify the current one. The D SMS command can be used to determine the ACDS and SCDS, as shown in Figure 5-23.

```
Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF OUTPUT DISPLAY MHLRES3Z JOB11648  DSID   102 LINE  COMMAND ISSUED
COMMAND INPUT ===> /D SMS                                  SCROLL ===> CSR
RESPONSE=SC64
 IGD002I 20:32:30 DISPLAY SMS 282
 SCDS = SYS1.SMS.SCDS
 ACDS = SYS1.SMS.ACDS
 COMMDS = SYS1.SMS.COMMDS
 DINTERVAL = 150
 REVERIFY = NO
 ACSDEFAULTS = NO
     SYSTEM     CONFIGURATION LEVEL    INTERVAL SECONDS
     SC63       2004/11/15 20:32:23           15
     SC64       2004/11/15 20:32:23           15
     SC65       2004/11/15 20:32:22           15
     SC70       2004/11/15 20:32:17           15
```

Figure 5-23 D SMS command output to get SCDS name

5.3.1 How to find a Data Class for PDSE

Data Class is an optional class that can be used in an SMS managed or a non-SMS managed environment. A Data Class is a policy used at allocation time to supply defaults for certain allocation options.

The Data Set Name Type value of LIBRARY creates a PDSE. This may override what has been specified in the JCL and depends on how your Storage Administrator wrote the ACS routines.

If you want the Data Set Name Type parameter set to a value of LIBRARY at allocation time by the Data Class, use ISMF to determine which Data Class you have to use. Figure 5-24 and Figure 5-25 on page 97 show the steps to do this.

From the ISMF main panel, choose the `4 Data Class` option. Specify `ACTIVE` for the CDS name, as shown in Figure 5-24.

```
 Panel  Utilities  Help
------------------------------------------------------------------------------
                        DATA CLASS APPLICATION SELECTION
Command ===>

To perform Data Class Operations, Specify:
  CDS Name . . . . . . 'ACTIVE'
                              (1 to 44 character data set name or 'Active' )
  Data Class Name  . . *        (For Data Class List, fully or partially
                                specified or * for all)

Select one of the following options  :
  1  1. List    - Generate a list of Data Classes
     2. Display - Display a Data Class
     3. Define  - Define a Data Class
     4. Alter   - Alter a Data Class

If List Option is chosen,
   Enter "/" to select option      Respecify View Criteria
                                   Respecify Sort Criteria


Use ENTER to Perform Selection;
```

Figure 5-24 ISMF Data Class list

You will see the output list as shown in Figure 5-25 on page 97. From this panel move to the right by pressing the PF11 key multiple times, until you see column 26.

```
 Panel  List  Utilities  Scroll  Help
------------------------------------------------------------------------------
DGTLGP21                        DATA CLASS LIST
Command ===>                                                  Scroll ===> CSR
                                                            Entries 21-28 of 28
                                                            View in Use
CDS Name : ACTIVE

Enter Line Operators below:

     LINE       DATACLAS                        EXTENDED                    MEDIA
     OPERATOR   NAME     DATA SET NAME TYPE     ADDRESSABILITY  COMPACTION  TYPE
    ---(1)----  --(2)--- -------(26)-------     -----(27)-----  ---(28)---  -(29)
                KEYEDEXG EXTENDED REQUIRED     NO              ----        -----
                MHLSTRIP EXTENDED REQUIRED     NO              ----        -----
                MULTVSAM EXTENDED REQUIRED     YES             ----        -----
                BIGHFS   HFS                   NO              ----        -----
                HFSMULTI HFS                   NO              ----        -----
                HFS      HFS                   NO              ----        -----
                DCPDSE   LIBRARY               NO              ----        -----
                MMDC     LIBRARY               NO              ----        -----
    ----------  -------- ------  BOTTOM  OF    DATA  ------  --------  ---------
```

Figure 5-25 Column 26 identifies which Data Class forces the allocation of a LIBRARY PDSE

When you have found the proper Data Class, specify it at allocation time. See 5.2.3, “ISPF allocation” on page 89 for details.

5.3.2 How to find a Storage Class for a PDSE

There are two Storage Class parameters related to PDSEs.

MSR *Millisecond response* time is used as a performance objective for selecting candidate volumes for new data set placement. SMS searches for a volume that meets or closely matches this objective. MSR values are also used to cache PDSE members.

GSW You can use the *Guaranteed Synchronous Write* attribute to indicate whether your system should return from a BSAM CHECK (or WAIT) issued for a WRITE against a PDSE member before (unsynchronized) or after (synchronized) the data has actually been written to DASD. This attribute does not apply to objects. Specify `Y` for synchronized write or `N` for no synchronization.

The Storage Administrator usually sets up a Storage Class for this purpose. To determine which Storage Classes have the parameters specified use ISMF. From the ISMF Primary panel choose option `5 Storage Class`. Then, specify `ACTIVE` for the CDS name as shown in Figure 5-26 on page 98.

```
 Panel  Utilities  Help
------------------------------------------------------------------------------
DGTSCSC2             STORAGE CLASS APPLICATION SELECTION
Command ===>

To perform Storage Class Operations, Specify:
  CDS Name  . . . . . . . 'ACTIVE'
                              (1 to 44 character data set name or 'Active' )
  Storage Class Name  . . *         (For Storage Class List, fully or
                                    partially specified or * for all)
Select one of the following options  :
  1  1. List          - Generate a list of Storage Classes
     2. Display       - Display a Storage Class
     3. Define        - Define a Storage Class
     4. Alter         - Alter a Storage Class
     5. Cache Display - Display Storage Classes/Cache Sets
If List Option is chosen,
   Enter "/" to select option      Respecify View Criteria
                                   Respecify Sort Criteria

If Cache Display is Chosen, Specify Cache Structure Name . .

Use ENTER to Perform Selection;
```

Figure 5-26 ISMF Storage Class list

You will get the output list shown in Figure 5-27; from this panel look for columns 3, 5, and 12. You can easily select them by running VIEW 2 3 4 5 12. To go back to the default view, simply enter VIEW *.

```
 Panel  List  Utilities  Scroll  Help
------------------------------------------------------------------------------
DGTLGP21                      STORAGE CLASS LIST
Command ===>                                                Scroll ===> HALF
                                                          Entries 26-35 of 57
                                                          View in Use
CDS Name : ACTIVE

Enter Line Operators below:

     LINE       STORCLAS DIR RESP  DIR   SEQ RESP  SEQ   GUARANTEED
     OPERATOR   NAME       (MSEC)  BIAS   (MSEC)   BIAS  SYNC WRITE
    ---(1)----  --(2)--- --(3)---  (4)-  --(5)---  (6)-  ---(12)---
                SCDBCRIT         5  W            5  -     NO
                SCDBFAST         5  -            5  -     NO
                SCDBMED         10  -           10  -     NO
                SCDBTEST        20  -           20  -     NO
                SCDUMP         ---  -          ---  -     NO
                SCGDG          ---  -          ---  -     NO
                SCGSWNO        999  -          999  -     NO
                SCGSWYES       999  -          999  -     YES
                SCGUSYWR       ---  -          ---  -     YES
                SCLIB1         ---  -          ---  -     NO
```

Figure 5-27 Columns 2 and 12 identify which Storage Class could be used for PDSE GSW

When you have found the proper Storage Class, you may specify it at allocation time. See 5.2.3, “ISPF allocation” on page 89 for details.

Note: If you leave all MSR fields blank (direct and sequential), SMS ignores device performances during volume selection and PDSE will not use Hiperspace caching.

5.3.3 Creating PDSE members

Member creation for a PDSE is functionally the same as for a PDS. Members can be moved between a PDS and a PDSE in the same way as between two PDSs. There is no functional difference between the two formats. All members can be accessed using BSAM and QSAM. Keyed BSAM is not supported for member processing.

A PDSE member is allocated in full-page increments. If a short logical block is created, the member is maintained on full-page boundaries (4K), and any remaining space in the page is unused.

For members with a variable record format, the block descriptor word (BDW) is functionally unchanged. The BDW is not stored in the PDSE but is calculated by DFSMSdfp and presented to the application.

All data records handled by PDSE services contain a gap between records that contains a record prefix of 6 bytes when they are stored on a DASD device. Fixed length files contain the same 6-byte record prefix as variable length files. This enables a fixed-length file to be converted to a variable-length file without data movement. This prefix is only a physical record hidden to the end user.

5.3.4 Deleting PDSE members

A member of a PDSE can be deleted through the ISPF library panel or Access Method Services. With a JCL disposition of delete (DISP=(OLD,DELETE)) the entire data set is deleted. This is the same as for a PDS.

When an alias is deleted, the alias name and its directory entry are deleted.

When a primary name is deleted, the primary name, all alias names, and the directory entries are deleted. The pointers from the directory entries are removed so the member can no longer be accessed by the application. The pointer and the physical space from a deleted member can be reused by another member.

5.4 Accessing PDSEs

In this section, we describe the main system macros used in Assembler programs to access PDSEs. Although most access to PDSEs will be from programs written in high-level languages, we include this information to give you some insight into what facilities are available.

Basic Partitioned Access Method (BPAM) macros are used to process partitioned data sets at a data set level. Table 5-4 on page 100 shows these macros.

Table 5-4 BPAM macros

Macro	Function provided
BLDL	Build a list of entries from the directory containing: ► Member or alias name ► Location of the start of the member ► Concatenation number ► Type of library (private, link, job, task, or step library) ► Type of entry (member or alias) ► User data The NOCONNECT option is provided for PDSEs to avoid excessive numbers of connections to members. See also 9.5.3, “Pending delete considerations” on page 193.
DESERV	Directory Entry Services that: ► Delete members and aliases ► Get directory information ► Rename members and aliases ► Update program object attributes
FIND	Establish the first record of a specified member as the starting point for the next READ macro. A connection is made to a PDSE member.
STOW	Update partitioned data set directory by adding, changing, replacing, or deleting a directory entry. You can clear an entire PDSE directory.

Members of a partitioned data set are accessed by using normal BSAM (Basic Sequential Access Method, providing access to physical blocks) or QSAM (Queued Sequential Access Method, providing access at the logical record level) macros. A member is read or written as a sequential data set. Table 5-5 provides a summary of these macros.

Table 5-5 BSAM and QSAM macros

Macro	Function provided
DCB	Build a Data Control Block (DCB) to hold information about a data set so that it can be opened.
DCBE	Build a DCB extension (DCBE) that, for QSAM access to partitioned data set members, is used to specify I/O buffers above the 16 MB line and to specify optimization of I/O requests.
OPEN/ CLOSE	Open a member or PDSE by completing the DCB entries and (optionally) connecting to the member; or close a PDSE by synchronizing member data to disk.
GET/PUT	Read or write next logical record (QSAM).
READ/ WRITE	Read or write next block (BSAM).
NOTE/ POINT	NOTE returns the position within an open member. POINT establishes a connection to a member (which is maintained until the PDSE is closed) and establishes the position for the next operation.

Control blocks are built for each connection established to a PDSE member. Connections are discussed in more detail in “Connections” on page 17.

Finally, the ISITMGD macro is used to determine the attributes of a data set. It returns a bit string that you can test. The bit setting enables you to determine whether a data set is managed, whether it is a PDSE, and whether the PDSE contains data members, program objects, or is empty. Find details for these macros in *z/OS V1R3.0-V1R6.0 DFSMS Macro Instructions for Data Sets*, SC26-7408.

5.5 Non-SMS PDSEs

System management of PDSEs implies that the PDSEs must be cataloged in the standard catalog search order. This does not present a problem for user PDSEs but it can be very inconvenient if you wish to use PDSEs to store system software where the natural location is the system residence volume. It makes maintenance of systems difficult and it hinders the cloning of systems.

As PDSEs began to be required for data sets in the LNKLIST and used for new types of executable code such as dynamic link libraries, this restriction became too inconvenient.

Non-SMS PDSEs need not be cataloged. Indirect cataloging (that is, specifying `volume(******)`) can be used. You can have more than one PDSE with the same name, all of which can be accessed concurrently. You may share non-SMS PDSEs in the same way as sharing managed PDSEs in either normal or extended sharing modes.

5.5.1 How to implement non-SMS PDSEs

First, install any required maintenance. We recommend that you contact your local Support Center for the latest details. You should also contact the vendors of other products that may have awareness of PDSEs. Examples might include non-IBM storage management products and program code version control software.

If you are already using PDSEs, change your Storage Class ACS routines so that a Storage Class is not assigned to the PDSEs that you wish to allocate as non-SMS. We suggest that you do this by checking high-level qualifiers or by allowing an explicit storage class in the allocation which is translated to a null storage class in your ACS routine.

5.5.2 Non-SMS PDSE characteristics

These are the main characteristics of unmanaged PDSEs that distinguish them from managed PDSEs:

- The SMS bit is not set in the Format-1 DSCB (data set control block) for the PDSE.
- The PDSE bit is set in the Format-1 DSCB.
- Cataloged unmanaged PDSEs do not have an SMS cell in the catalog entry.
- Unmanaged PDSEs do not have an NVR in the VVDS.
- A call to the ISITMGD service does not return SMS information.

5.5.3 PDSEs in LNKLST

SMS-managed and non-SMS PDSEs can be specified in LNKLST. PDSEs may be added with their volume serial number specified to avoid a catalog lookup and to avoid them being cataloged. Before “Non-SMS PDSE support,” a PDSE had to be cataloged when it was specified in LINKLIST, in order to enable the PDSE connect token to extract the Storage Class that was used to determine the caching attributes for the data set. NIP OPEN always tries to get the Storage Class name from the catalog. When a PDSE is added to the LINKLIST with a volser specified, the catalog is bypassed. Users would like to be able to specify an unmanaged PDSE without cataloging it in the master catalog. Because of LLA processing, the unmanaged PDSE still must be cataloged in a user catalog when LLA starts because LLA will allocate them by name. PDSE data sets that are in the LNKLST concatenation are processed by the non-restartable address space SMSPDSE.

It may be convenient to use PDSEs in the LINKLIST concatenation because there is a limit of 255 extents across all of the libraries in the concatenation. However, a PDSE counts as only one extent even though it can physically have up to 123 extents on disk.

With z/OS DFSMS 1.4 or later with OW57609 PTFs applied, PDSE support will be modified to enable a PDSE that is globally connected by LINKLIST to be disconnected. The PDSE can then be deleted in the same way that PDSs are. Note that even when a LINKLIST is undefined, LLA will continue to be allocated to the PDSEs and PDSs in the prior LINKLIST and you will have to shut down and restart LLA or activate a new LLA PARMLIB member that includes a delete for the specified PDSE. This is the same process that would be required for a PDS. The fixes provided by OW57609 are applicable to PDSE libraries added to LNKLST at some point after IPL. For PDSE libraries included in LNKLST at IPL time, the permanent restriction described by OW40072 is still valid: If a PDSE is added to a dynamic LINKLST, you can remove it from the LINKLST but you cannot scratch the PDSE.

For additional information, see 3.5, “LNKLST and LLA considerations” on page 45. For a detailed and updated list of APARs, refer to Appendix B, “PDSE APARs” on page 325.

5.5.4 Considerations

If you need to tell from a program whether a system can support non-managed PDSEs, you can examine the data facilities area (DFA) control block that is mapped by the IHADFA macro. The DFAUPDSE bit will be set if non-managed PDSEs can be used. The communication vector table (CVT) points to the DFA. See *z/OS DFSMSdfp Advanced Services*, SC26-7400, for more information.

The support is intended to be exploited by system software installation. It enables PDSE program libraries to be placed on a system residence volume, which will become increasingly important.

Important: We do not recommend that user PDSEs be allocated as non-SMS data sets. The use of SMS still provides benefits for storage management, enabling management class controls to be used and allocation to be made to storage groups rather than specific volumes.

5.6 Guaranteed Synchronous Write

Guaranteed Synchronous Write is an optional attribute that can be specified in an SMS storage class. If you give a PDSE a storage class that specifies `GUARANTEED SYNCHRONOUS WRITE ===> YES`, then when you open the PDSE or one of its members for update, all of the write I/Os to that PDSE are synchronized between disk and the PDSE buffers.

When processing PDSEs that have storage classes with `GUARANTEED SYNCHRONOUS WRITE ====> NO`, the system does not synchronize between disk and PDSE buffers until the CLOSE macro is issued or another member is processed. The CHECK macro, used for checking successful I/O operation, does not synchronize the PDSE buffers to disk. PDSE buffering attempts to reduce physical I/O operations, giving you better performance. However, if an error occurs on the disk, any data you updated at that time may be lost. The GSW function makes the system start a physical I/O for each 4 KB page immediately after issuing the PUTX macro (QSAM) for adding the last record in a page, or the CHECK macro (BSAM) coded after the WRITE macro for a PDSE opened for update.

Storage administrators can set up storage classes with GSW set to YES with the storage class application in ISMF. Figure 5-28 on page 103 shows an example of the storage class definition screen.

```
 Panel  Utilities  Scroll  Help
ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
DGTDCSC3                   STORAGE CLASS DEFINE              Page 2 of 2
Command ===>

SCDS Name . . . . . : SYS1.SMS.SCDS
Storage Class Name  : SCGUSYWR

To DEFINE Storage Class, Specify:

  Guaranteed Space  . . . . . . . . . N          (Y or N)
  Guaranteed Synchronous Write  . . . Y          (Y or N)
  Multi-Tiered SG . . . . . . . . . .            (Y, N, or blank)
  Parallel Access Volume Capability   N          (R, P, S, or N)
  CF Cache Set Name . . . . . . . . .            (up to 8 chars or blank)
  CF Direct Weight  . . . . . . . . .            (1 to 11 or blank)
  CF Sequential Weight  . . . . . . .            (1 to 11 or blank)
```

Figure 5-28 Defining a DFSMS storage class with Guaranteed Synchronous Write

Note: You can also synchronize between disk and the buffers of the PDSE opened for output or update by issuing the SYNCDEV macro in your program. See *z/OS V1R3.0-V1R6.0 DFSMS Macro Instructions for Data Sets,* SC26-4913, for more information about SYNCDEV macro usage.

However, this *could degrade performance*. See 9.4.1, “Effect of Guaranteed Synchronous Write” on page 184 for information about the performance of PDSEs that use GSW.

5.7 Concatenating PDSEs

Two different types of concatenation are supported: sequential and partitioned. You can concatenate PDSEs with sequential data sets or with PDSs.

5.7.1 Sequential concatenation

To process sequentially concatenated data sets, use a DCB with DSORG=PS. Each DD statement in the concatenation must specify one of the following:

- Sequential data set
- UNIX® file
- PDS member
- PDSE member

A sequential concatenation can include sequential data sets, PDS members, PDSE members, and UNIX files. With sequential concatenation, the system treats a PDS, PDSE, or UNIX member as if it were a sequential data set. The system treats a striped extended-format data set as if it were a single-volume data set.

See *z/OS V1R6.0 DFSMS Using Data Sets,* SC26-7410 for additional information and restrictions.

5.7.2 Partitioned concatenation

When you use partitioned concatenation, the system treats each PDSE as only one extent even if the PDSE actually has more than one extent on disk. There is a limit to how many DD statements are allowed in a partitioned concatenation. The total number of PDS extents, PDSEs, and UNIX directories must not exceed the concatenation limit of 255.

Figure 5-29 and Table 5-6 show how to calculate the number of extents for concatenation. If you concatenate as in the figure, the total number of extents for the concatenation is 19, as shown in Table 5-6.

```
...
//CONCAT DD DSN=USERID.PDSE01,DISP=SHR
//       DD DSN=USERID.PDS001,DISP=SHR
//       DD DSN=USERID.PDSE02,DISP=SHR
//       DD DSN=USERID.PDS002,DISP=SHR
...
```

Figure 5-29 Concatenating PDSEs mixed with PDSs

Table 5-6 Example to calculate the number of extents for concatenation

Data set name	Type of data set	Number of extents on disk	Number of extents for concatenation
USERID.PDSE01	PDSE	25	1
USERID.PDS001	PDS	6	6
USERID.PDSE02	PDSE	13	1
USERID.PDS002	PDS	11	11
Total		55	19

5.8 DFSMS and z/OS facilities used with PDSE

This section describes how z/OS components and utilities support PDSEs.

5.8.1 ISMF filtering

You can use the PDSE indicator on the Data Set List panel.

Generate a new list from Catalog

You can select PDSEs according to the filtering criteria shown in Figure 5-30 on page 105 and Figure 5-31 on page 105. This is valid for all cataloged PDSEs, whether they are SMS-managed or not. Figure 5-32 on page 106 shows the resulting ISMF list. The list will not show uncataloged non-SMS PDSEs. To generate a list of cataloged and uncataloged PDSEs, see the procedure listed in “Generate a new list from VTOC” on page 106.

```
 Panel  Defaults  Utilities  Scroll  Help
------------------------------------------------------------------------------
DGTDDDS1               DATA SET SELECTION ENTRY PANEL            Page 1 of 5
Command ===>

For a Data Set List, Select Source of Generated List  . . 2  (1 or 2)

  1  Generate from a Saved List         Query Name To
       List Name  . .                   Save or Retrieve

  2  Generate a new list from criteria below
       Data Set Name . . . 'PDSERES.**'
       Enter "/" to select option      Generate Exclusive list
       Specify Source of the new list  . . 2    (1 - VTOC, 2 - Catalog)
       1 Generate list from VTOC
          Volume Serial Number . . .            (fully or partially specified)
       2 Generate list from Catalog
          Catalog Name . . .
          Volume Serial Number . . .            (fully or partially specified)
          Acquire Data from Volume . . . . . . . Y  (Y or N)
          Acquire Data if DFSMShsm Migrated  . . Y  (Y or N)

Use ENTER to Perform Selection; Use DOWN Command to View next Selection Panel;
```

Figure 5-30 Selecting PDSEs in ISMF data set application via catalog (Part 1 of 2)

```
 Panel  Defaults  Utilities  Scroll  Help
------------------------------------------------------------------------------
DGTDDDS3               DATA SET SELECTION ENTRY PANEL            Page 4 of 5
Command ===>

To limit the List, Specify a value or range of values in any of the following:
                           Rel Op   Value      Value      Value     Value
                           ------  --------   --------   --------  --------
   Allocation Unit  . . . .
   CF Cache Set Name  . . .
   CF Monitor Status  . . .
   CF Status Indicator  . .
   Change Indicator . . . .
   Compressed Format  . . .
   Data Class Name  . . . .
   Data Set Environment . .
   Data Set Name Type . . . EQ      LIBRARY
   Data Set Organization    EQ      PO
   (1 to 8 Values)     ===>
   DDM Attributes . . . . .
   Device Type  . . . . . .
   (1 to 8 Values)     ===>
Use ENTER to Perform Selection; Use UP/DOWN Command for other Selection Panel;
```

Figure 5-31 Selecting PDSEs in ISMF data set application via catalog (Part 2 of 2)

```
 Panel  List  Dataset  Utilities  Scroll  Help
------------------------------------------------------------------------------
DGTLGP11                          DATA SET LIST
Command ===>                                                  Scroll ===> HALF
                                                          Entries 1-6 of 6
Enter Line Operators below:                               Data Columns 29-31 of 39

     LINE                                    DATA SET     DATA SET   NUM OF
     OPERATOR          DATA SET NAME         ENVIRONMENT  NAME TYPE  STRIPES
    ---(1)----  ------------(2)------------ ---(29)----  --(30)---  -(31)--
                PDSERES.CEE.SCEERUN2.       UNMANAGED    LIBRARY        --
                  RELOADED
                PDSERES.CEE.SCEERUN2.       UNMANAGED    LIBRARY        --
                  RELOADED.CDAEED
                PDSERES.CEE.SCEERUN2.       UNMANAGED    LIBRARY        --
                  RELOADED.CRTEC128
                PDSERES.CEE.SCEERUN2.       UNMANAGED    LIBRARY        --
                  RELOADED.C128
                PDSERES.PDSECOP.CNTL        UNMANAGED    LIBRARY        --
                PDSERES.SMS.DIRK.PDSE1      MANAGED      LIBRARY        --
    ----------  ------  -----------  BOTTOM OF  DATA  -----------  ------  ----
```

Figure 5-32 Using ISMF data set application to retrieve PDSEs via catalog

Generate a new list from VTOC

You can select PDSEs specifying the volume serial numbers you want to look into (partially or fully qualified), as shown in Figure 5-33 and the parameters shown in Figure 5-34 on page 107. The results shown in Figure 5-35 on page 107 contain cataloged and uncataloged PDSEs allocated on those volumes.

```
 Panel  Defaults  Utilities  Scroll  Help
------------------------------------------------------------------------------
DGTDDDS1               DATA SET SELECTION ENTRY PANEL    OTHER VALUES PRESENT
Command ===>

For a Data Set List, Select Source of Generated List  . . 2  (1 or 2)

  1  Generate from a Saved List         Query Name To
       List Name  . .                   Save or Retrieve

  2  Generate a new list from criteria below
       Data Set Name . . . '**'
       Enter "/" to select option      Generate Exclusive list
       Specify Source of the new list  . . 1    (1 - VTOC, 2 - Catalog)
       1 Generate list from VTOC
          Volume Serial Number . . . MHL*       (fully or partially specified)
       2 Generate list from Catalog
          Catalog Name . . .
          Volume Serial Number . . .            (fully or partially specified)
          Acquire Data from Volume . . . . . Y  (Y or N)
          Acquire Data if DFSMShsm Migrated  . . Y  (Y or N)

Use ENTER to Perform Selection; Use DOWN Command to View next Selection Panel;
```

Figure 5-33 Selecting PDSE in ISMF data set application via VTOC (Part 1 of 2)

```
 Panel  Defaults  Utilities  Scroll  Help
------------------------------------------------------------------------------
DGTDDDS3               DATA SET SELECTION ENTRY PANEL            Page 4 of 5
Command ===>

To limit the List, Specify a value or range of values in any of the following:
                            Rel Op   Value      Value      Value     Value
                            ------  --------   --------   --------  --------
    Allocation Unit  . . . .
    CF Cache Set Name  . . .
    CF Monitor Status  . . .
    CF Status Indicator  . .
    Change Indicator . . . .
    Compressed Format  . . .
    Data Class Name  . . . .
    Data Set Environment . .
    Data Set Name Type . . . EQ      LIBRARY
    Data Set Organization    EQ      PO
    (1 to 8 Values)     ===>
    DDM Attributes . . . . .
    Device Type  . . . . . .
    (1 to 8 Values)     ===>
Use ENTER to Perform Selection; Use UP/DOWN Command for other Selection Panel;
```

Figure 5-34 Selecting PDSE in ISMF data set application via VTOC (Part 2 of 2)

```
 Panel  List  Dataset  Utilities  Scroll  Help
------------------------------------------------------------------------------
DGTLGP11                        DATA SET LIST
Command ===>                                                Scroll ===> HALF
                                                      Entries 13-20 of 20
Enter Line Operators below:                           Data Columns 29-31 of 39

     LINE                                  DATA SET     DATA SET   NUM OF
     OPERATOR         DATA SET NAME        ENVIRONMENT  NAME TYPE  STRIPES
    ---(1)----  ------------(2)------------ ---(29)----  --(30)---  -(31)--
                MHLRES6.TESTCASE.LIBRARY    MANAGED      LIBRARY         --
                MHLRES7.CMD.CLIST           MANAGED      LIBRARY         --
                MHLRES7.SACBCNTL.ISPTABL    MANAGED      LIBRARY         --
                MHLRES7.SACBCNTL.SAMPLE     MANAGED      LIBRARY         --
                MHLRES7.SACBCNTL.TAILOR     MANAGED      LIBRARY         --
                MHLRES7.SACBCNTL.TESTBED    MANAGED      LIBRARY         --
                PDSERES.PDSE.ALESSIA9       UNMANAGED    LIBRARY         --
                PDSERES.SMS.DAVIDE9         MANAGED      LIBRARY         --
    ----------  ------  -----------  BOTTOM OF  DATA  -----------  ------  ----
```

Figure 5-35 Using ISMF data set application to retrieve PDSEs via VTOC

5.8.2 DFSMSdfp utilities: IEBCOPY

Figure 5-36 is an example of using IEBCOPY to convert a PDS to a PDSE. In this example, IEBCOPY retains the original PDS, creates a PDSE, and copies the members of the PDS into the PDSE. The space is explicitly defined for the PDSE. You may use the LIKE parameter to get the DCB information from a PDS or a PDSE.

For further information and examples of how to use IEBCOPY for PDSE, see:

- 4.3.2, “Using IEBCOPY” on page 77
- 6.3, “IEBCOPY” on page 131
- 9.4.4, “IEBCOPY performance considerations” on page 187

```
//PDSERESY  JOB  ...
//*
//*
//*     CONVERT PDS TO PDSE
//*     WITH EXPLICIT SPACE DEFINED FOR PDSE
//*
//*
//STEP1     EXEC  PGM=IEBCOPY
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD DSN=PDSERES.PDS.CONVERT.CNTL,DISP=OLD
//SYSUT2    DD DSN=PDSERES.SMS.PDSE.MATTEO,DISP=(NEW,KEEP),
//          SPACE=(TRK,(7,2,15)),LRECL=80,
//          DSNTYPE=LIBRARY
//SYSIN     DD    *
       COPY   INDD=SYSUT1,OUTDD=SYSUT2
/*
```

Figure 5-36 Converting a PDS to a PDSE with IEBCOPY

5.8.3 DFSMSdfp utilities: IEHLIST

Two useful commands for PDSE provided by IEHLIST are LISTVTOC and LISTPDS.

LISTVTOC or VTOCLIST

The output is the same with either command. You can run LISTVTOC from an IEHLIST JCL job. You can run a VTOCLIST simply from the ISMF command line.

You can run the IEHLIST utility from an ISMF panel by entering `VTOCLIST` in the line operator column, as shown in Figure 5-37 on page 109.

The output in Figure 5-38 on page 109 shows that the data set is a PDSE. (It has an “I” in the SMS.IND field.)

```
 Panel  List  Dataset  Utilities  Scroll  Help
------------------------------------------------------------------------------
DGTLGP11                          DATA SET LIST
Command ===>                                                  Scroll ===> HALF
                                                           Entries 4-13 of 13
Enter Line Operators below:                                Data Columns 3-6 of 39

     LINE                                      ALLOC    ALLOC   % NOT  COMPRESSED
     OPERATOR          DATA SET NAME           SPACE    USED    USED   FORMAT
    ---(1)----  ------------(2)------------ --(3)--  --(4)--  -(5)-  ---(6)----
                PDSERES.CEE.SCEERUN2.          8522     8466      1  ---
                  RELOADED.C128
                PDSERES.CEE.SCEERUN2.       1549406   703043     54  ---
                  RELOADED.STOP
                PDSERES.LOADED.PDSE             111       55     50  ---
                PDSERES.LOADING.PDSE            111       55     50  ---
                PDSERES.PDSECOP.CNTL          60427    54285     10  ---
                PDSERES.SMS.DAVIDE9          829762   821960      1  ---
                PDSERES.SMS.DIRK.PDSE1          277      111     59  ---
    vtoclist    PDSERES.SMS.PDSE.MATTEO      276624       55     99  ---
                PDSERES.SYS1.MACLIB             830      166     80  ---
                PDSERES.SYS1.MACLIB.COPYGRP     830      166     80  ---
    ----------  ------  -----------  BOTTOM OF  DATA  -----------  ------  ----
```

Figure 5-37 ISMF data set list panel and VTOCLIST command

```
1                          SYSTEMS SUPPORT UTILITIES---IEHLIST
-DATE: 2004.313  TIME: 20.50.44
                 CONTENTS OF VTOC ON VOL SBOX26  <THIS IS AN SMS MANAGED
0---------------DATA SET NAME---------------  SER NO  SEQNO  DATE.CRE
 PDSERES.SMS.PDSE.MATTEO                     SBOX26      1  2004.313
0SMS.IND   LRECL  KEYLEN  INITIAL ALLOC  2ND ALLOC    EXTEND       LAST
 S R I        80           TRKS              2499
0           EXTENTS  NO  LOW(C-H)   HIGH(C-H)
                      0    1494 13 1494 13
                               ----UNABLE TO CALCULATE EMPTY SPACE.
0
```

Figure 5-38 VTOCLIST (IEHLIST) command output from an ISMF data set list

These are the System-managed storage attributes you could find in the SMS.IND field. For more details, refer to *z/OS V1R3.0-V1R6.0 DFSMSdfp Utilities*, SC26-7414:

B Optimal block size selected by DADSM create
C Compressed format
E Extended format
I Data set is a PDSE or USS data set (not a USS file)
R Data set is reblockable
S SMS-managed data set
U No BCS entry exists for data set

LISTPDS

IEHLIST can list up to 10 partitioned data set or PDSE directories at a time. Each directory block can contain one or more entries that reflect member or alias names and other attributes of the partitioned members. IEHLIST can list these blocks in edited and unedited format. The directory of a PDSE, when listed, will have the same format as the directory of a partitioned data set.

Figure 5-39 shows JCL and sample output to get a DUMP format.

```
//PDSERESY  JOB  ...
//S1  EXEC  PGM=IEHLIST
//SYSPRINT  DD SYSOUT=*
//DD1       DD UNIT=3390,VOL=SER=SBOXEF,DISP=OLD
//SYSIN     DD *
  LISTPDS DSNAME=(PDSERES.PDSECOP.CNTL),VOL=3390=SBOXEF,DUMP
-------------------------------------------------------------------
DIRECTORY INFO FOR SPECIFIED PDS ON VOL SBOXEF
 PDSERES.PDSECOP.CNTL
0           MEMBERS     TTRC     VARIABLE USER DATA ---(USER DATA AND TTRC ARE I
           MEME7     00000A0F  0100003901 04315F0104 315F12026D 486D480000 D4C8D
           MEM10     00000E0F  0100001801 04315F0104 315F1253FF FFFFFF0000 D4C8D
           MEM11     00000F0F  0100003501 04315F0104 315F1254FF FFFFFF0000 D4C8D
           MEM3      0000060F  0102004601 04309F0104 310F18326D 4800646CE4 D4C8D
           MEM4      0000070F  0100002601 04315F0104 315F12016D 486D480000 D4C8D
           MEM6      0000090F  0100001601 04315F0104 315F1202DA 90DA900000 D4C8D
           MEM9      00000D0F  0100000201 04315F0104 315F1219FF FFFFFF0000 D4C8D
---OF THE      52416 KBYTES ALLOCATED TO THIS PDSE,      47076 KBYTES ARE IN USE
******************************** BOTTOM OF DATA ********************************
```

Figure 5-39 JCL and output of LISTPDS for a PDSE in a DUMP format

Figure 5-40 shows JCL and sample output to get a FORMAT output.

```
//PDSERESY  JOB  ...
//S1  EXEC  PGM=IEHLIST
//SYSPRINT  DD SYSOUT=*
//DD1       DD UNIT=3390,VOL=SER=SBOXEF,DISP=OLD
//SYSIN     DD *
  LISTPDS DSNAME=(PDSERES.PDSECOP.CNTL),VOL=3390=SBOXEF,FORMAT
-------------------------------------------------------------------
DIRECTORY INFO FOR SPECIFIED PDS ON VOL SBOXEF
 PDSERES.PDSECOP.CNTL
0                                          OTHER INFORMATION INDEX
0    SCATTER FORMAT   SCTR=SCATTER/TRANSLATION TABLE TTR IN HEX,LEN OF SCTR LIST
                           ESDID OF FIRST TEXT RCD IN DEC,ESDID OF CSECT CONTAIN
0    OVERLAY FORMAT   OVLY=NOTE LIST RCD TTR IN HEX,NUMBER OF ENTRIES IN NOTE LI
0    ALIAS NAMES      ALIAS MEMBER NAMES WILL BE FOLLOWED BY AN ASTERISK IN THE
-                                                ATTRIBUTE INDEX
-       BIT  ON     OFF          BIT  ON      OFF          BIT    ON       OFF
0        0  RENT  NOT RENT        4  OL     NOT OL          8  NOT DC    DC
         1  REUS  NOT REUS        5  SCTR   BLOCK           9  ZERO ORG  NOT Z
         2  OVLY  NOT OVLY        6  EXEC   NOT EXEC       10  EP ZERO   NOT Z
         3  TEST  NOT TEST        7  1 TXT  MULTI RCD      11  NO RLD    RLD
-  MEMBER       ENTRY    ATTR    REL ADDR-HEX     CONTIG    LEN 1ST   ORG 1ST
    NAME        PT-HEX   HEX    BEGIN  1ST TXT   STOR-DEC   TXT-DEC   TXT-HEX

  MEME7        00486D48  0104   ******  ******  0003235602   *****    ******
  MEM10        00FFFFFF  0104   ******  ******  0003235602   *****    ******
  MEM11        00FFFFFF  0104   ******  ******  0003235602   *****    ******
  MEM3         40400000  0104   ******  ******  3319984960   *****    ******
  MEM4         00486D48  0104   ******  ******  0003235602   *****    ******
  MEM6         0090DA90  0104   ******  ******  0003235602   *****    ******
  MEM9         00FFFFFF  0104   ******  ******  0003235602   *****    ******
---OF THE      52416 KBYTES ALLOCATED TO THIS PDSE,      47076 KBYTES ARE IN USE
```

Figure 5-40 JCL and output of LISTPDS for a PDSE in a FORMAT output

5.8.4 IDCAMS LISTCAT command

The CATLIST command can be issued from an ISMF Data Set List panel to invoke the IDCAMS LISTCAT command. The DSNTYPE field identifies the data set as a PDSE or a PDS. A PDSE is shown as LIBRARY. Figure 5-41 shows output from the CATLIST command.

```
1IDCAMS  SYSTEM SERVICES                                          TIME
ONONVSAM ------- PDSERES.SMS.PDSE.MATTEO
      HISTORY
        DATASET-OWNER-----(NULL)     CREATION--------2004.313
        RELEASE----------------2     EXPIRATION------0000.000
        ACCOUNT-INFO-----------------------------------(NULL)
        DSNTYPE----------LIBRARY
      SMSDATA
        STORAGECLASS -----SCTEST     MANAGEMENTCLASS---MCDB22
        DATACLASS --------(NULL)     LBACKUP ---0000.000.0000
      VOLUMES
        VOLSER-----------SBOX26     DEVTYPE------X'3010200F'     FSEQN
      ASSOCIATIONS--------(NULL)
      ATTRIBUTES
      IN-CAT --- UCAT.VSBOX01
```

Figure 5-41 Catalog listing of a PDSE

5.8.5 Distributed FileManager/MVS

Distributed FileManager/MVS (DFM/MVS), a DFSMSdfp component, provides a way for users on Microsoft® Windows®, AIX®, or OS/400® systems to access data in S/390 data sets across an SNA LU 6.2 connection as if it were local.

A PDS or PDSE will appear as a directory and the members will appear as files within the directory. When accessed by DFM/MVS, PDSEs offer advantages compared to PDSs:

- Attributes can be stored for each member in the PDSE directory.
- The CCSID attribute identifies the character set used for a PDSE member.
- PDSE members can be used to hold stream files, which have no inherent record structure.

For more information, see *z/OS V1R1.0-V1R6.0 DFSMS DFM Guide and Reference*, SC26-7395.

5.8.6 DFSORT™

Members of a PDSE can be used as input and output of SORT, MERGE, and COPY.

DFSORT Release 11, and later releases, fully support PDSEs. If a previous release of DFSORT is used, you can access PDSE data sets, but you must specify the DEBUG parameter to force use of BSAM as the access method instead of EXCP.

5.8.7 ISPF

All ISPF functions support PDSEs. You can concatenate a mixture of PDSs and PDSEs as long as the record format and record length are compatible. All of the library management functions (option 3.1) work, with the exception that ISPF will not compress a PDSE. If you try to do that, you will see the message shown in Figure 5-42.

```
 Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
                              Library Utility           Compress not allowed
Option ===> C

blank Display member list     I Data set information          B Browse member
    C Compress data set       S Short data set information    D Delete member
    X Print index listing     E Edit member                   R Rename member
    L Print entire data set   V View member                   P Print member

                                  Enter "/" to select option
ISPF Library:                     /  Confirm Member Delete
   Project . . . PDSERES             Enhanced Member List
   Group . . . . SMS      . . .          . . .         . . .
   Type  . . . . DAVIDE9
   Member  . . .             (If B, D, E, P, R, V, or blank selected)
   New name  . .             (If R selected)

Other Partitioned or Sequential Data Set:
   Data ------------------------------------------------------------
   Volu - The specified library is a PDSE and cannot be compressed. -
        ------------------------------------------------------------
Data Set Password  . .          (If password protected)
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

Figure 5-42 Error message trying to COMPRESS a PDSE

5.8.8 TSO/E

The TSO/E TRANSMIT command may be used to send a complete PDSE or selected members to another TSO user. TRANSMIT first calls IEBCOPY to unload the PDSE to a sequential data set, then adds control information, and finally sends the data set.

It is reloaded automatically when the RECEIVE command is used.

5.8.9 LISTDSI TSO/E

The TSO/E LISTDSI service may be called from CLISTs or REXX execs to provide information about a data set. It returns information in variables such as those listed in Table 5-7.

Table 5-7 Some of LISTDSI variables for PDSEs

Variable name	Values
SYSDSORG	PO for PDS or PDSE.
SYSUSED	N/A returned for PDSE.
SYSUSEDPAGES	Number of 4 KB pages used in a PDSE.
SYSADIRBLK	Directory blocks allocated — NO_LIM returned for PDSE.

Variable name	Values
SYSUDIRBLK	Directory blocks used — N/A returned for PDSE.
SYSMEMBERS	Number of members.
SYSDSSMS	PDS or PDSE. If the SMSINFO option was used on the call, it will contain different values for a PDSE: LIBRARY for an empty PDSE, PROGRAM_LIBRARY or DATA_LIBRARY for a PDSE with members in it.

The information from the directory is present only if the DIRECTORY option was added to the LISTDSI call.

For more information and complete syntax for LISTDSI, see *z/OS V1R6.0 TSO/E RESS Reference*, SA22-7790, and *z/OS V1R4.0-V1R6.0 TSO/E CLISTs, SA22-7781*.

5.9 PDSE restrictions

This section describes the functions that you can use with PDSs but *not* with PDSEs.

5.9.1 Load module libraries

A minor restriction of PDSEs is that you cannot use them to store load modules although PDSEs may contain program objects. To create a load module, you can use the Program Management binder and use a PDS as a target.

The binder creates the PDS member in which the module is stored. For efficiency reasons, it does this directly without using normal access method macros. There have been no changes to the linkage editor component to support PDSEs.

You can also use the program management binder to produce a program object that is stored in a PDSE. The improvements you gain by storing executable code as program objects within a PDSE are described in Chapter 7, “Program management and PDSEs” on page 145.

5.9.2 LPALSTxx and PDSE

z/OS does support PDSE modules in LPA; however, PDSEs cannot be in the LPALST concatenation. They have to be loaded into LPA dynamically after DFSMS services become active (after IPL), using the ET PROG=xx command.

5.9.3 Member size and number of members

The simulated TTR addressing scheme for PDSEs limits the size of each member to 15,728,639 (hexadecimal EFFFFF) records and the number of members in a PDSE to 524,236. The section “Member and record addressing” on page 18 explains how these numbers are calculated.

5.9.4 Other restrictions

Additional PDSE processing restrictions that you are very unlikely to encounter include:

- The JCL keyword DSNTYPE cannot be specified with the JCL keyword RECORG.
- You cannot use PDSEs if your application is dependent on physical block size or expects null record segments.

- You cannot write or update the PDSE directory using the WRITE or PUT macros. To write or update the directory, you must use the STOW or DESERV FUNC=UPDATE macros.
- You cannot add or replace members of a PDSE program library using the STOW macro.
- Aliases for data members must point to the beginning of the member.
- The deletion of the primary member name causes all aliases to be deleted.
- EXCP, EXCPVR, and XDAP macros are not supported for PDSEs.
- If you allocate a PDSE, it cannot be read on an earlier version or release of DFSMSdfp that does not support PDSEs.
- Note lists are not supported for PDSEs. When using STOW with a PDSE, do not supply a list of TTRs in the user data field of the directory.
- The CHECK macro does not guarantee that the data has been synchronized to DASD. Use the SYNCDEV macro or the storage class parameter GSW=YES to guarantee synchronizing data when open for update.
- DS1LSTAR field of the format 1 DSCB is unpredictable for PDSEs.
- The OPTCD=W DCB parameter (write-check) is ignored for PDSEs.
- A checkpoint data set cannot be a PDSE. Checkpoint fails if the checkpoint data set is a PDSE. Also, a failure occurs if a checkpoint is requested when a DCB is opened to a PDSE, or if a PDS was opened at checkpoint time but was changed to a PDSE by restart time.
- Do not use the TRKCALC macro because results could be inaccurate. (However, no error indication is returned.)
- The system procedure library, SYS1.PROCLIB, must be a PDS. The installation can have other procedure libraries, which can be PDSEs.
- When a cataloged procedure library is used, the library is opened for input and stays open for extended periods. If another user or system attempts to open the procedure library for output in a non-XCF (sysplex) environment, the second user receives an abend. This could prevent procedure library updates for extended periods.

5.10 PDSE usage differences

This section highlights PDSE usage differences caused by PDSE improvements over PDS.

5.10.1 Device independence

Because PDSEs are device-independent, several macros that depend on the physical geometry of a storage device do not work with them:

- TRKCALC, other TTR calculation algorithms, and the sector convert routine are device-dependent and should not be used with PDSEs.
- When used with PDSEs, NOTE returns X'7FFF' for the track balance and track capacity. This should enable most programs that use these macros to continue to work.

PDSEs are always movable and always reblockable.

5.10.2 Changed macros

NOTE, POINT, and BSP are not supported in the PDSE directory.

Because of the TTR addressing scheme, you cannot use the POINT macro across members. You must position to the first record in a member before using POINT to records within that member. If you do this, your program will also work correctly with PDSs.

5.10.3 Buffering changes

The CHECK macro does not guarantee that PDSE data is actually synchronized to disk. Use the SYNCDEV macro or guaranteed synchronous write if you require this function.

5.10.4 Access method changes

You cannot process PDSEs directly with EXCP or XDAP. You should use BPAM, BSAM, or QSAM as appropriate. Because of improvements to the way records are physically stored in a PDSE, keyed BSAM is supported only for reading the directory.

5.10.5 Aliases

Deleting a primary member name deletes all aliases to that name, so programs cannot use old aliases to find members that have been deleted. This improves integrity but makes it more important to have proper backup for all data sets.

Because of the addressing structure of a PDSE, aliases of data members must point to the beginning of the member.

5.10.6 Block size and record segments

Because PDSEs are stored in a fixed physical block (page) size regardless of the logical block size, they are all assumed to be reblockable. Short logical blocks can be created, but they will not be honored on read.

Record segment boundaries are not maintained. Null record segments are not saved on output.

5.10.7 Last track indicator

For a PDS, the last track used is stored in DS1LSTAR in the Format-1 DSCB for the data set. DS1LSTAR is always zero for a PDSE. See 2.5.2, “Space usage on disk” on page 19 for more details about this.

5.11 Limitations common to PDSs and PDSEs

- CLOSE type T is not supported for PDSs or PDSEs.
- Neither a PDSE nor a PDS can extend beyond a volume boundary.
- PDS and PDSE members that have different security access requirements from other members in the data set should be stored in separate data sets.

See 2.6, “Summary of PDSE characteristics and limits” on page 27 for additional information.

6

Backup and recovery of PDSEs

In this chapter, we give a brief overview of the tools and techniques that you can use to protect and manage PDSEs. We discuss DFSMSdss and DFSMShsm in particular. There are few considerations specific to PDSEs.

There are different considerations for data set backup, particularly for recovery depending on whether a data set belongs to a user or to the system, and whether it is or is intended to be shared among several users across different parts of a sysplex. Where there are different considerations, these are noted.

Note: The storage management products described below and the DFSMSdfp utilities such as IEBCOPY work at the data set level. So when a member of a PDSE is changed, the entire data set will be backed up. Similarly, if there is a request to open a member in a migrated PDSE, the entire data set will be recalled.This is the same as for PDSs.

6.1 DFSMSdss

DFSMSdss provides several functions to help you manage PDSEs. DFSMSdss is usable on all types of data sets, so take care to use appropriate keywords when the intent is only to affect PDSE or PDS format data sets.

DFSMSdss provides four kinds of back-up processing:

- Logical data set dumps
- Physical data set dumps
- Logical volume dumps
- Physical volume dumps

DFSMSdss can perform either logical or physical processing. If you dump a data set logically, DFSMSdss restores it logically; if you dump it physically, DFSMSdss restores it physically.

Logical processing operates against data sets independently of physical device format. The data sets are located by searching either the catalog or the VTOC. If input volumes are specified through the LOGINDD, LOGINDYNAM, or STORGRP keywords, then data sets are located by searching the VTOCs. Otherwise, data sets are located by searching the catalog.

Physical processing moves data at the track-image level and operates against volumes, tracks, and data sets. The data sets are located by searching the VTOC.

The processing method is determined by the keywords specified on the command. For example, LOGINDYNAM indicates a logical dump and INDYNAM a physical dump. Each type of processing offers different capabilities and advantages.

Physical processing is required to process a PDSE that has been corrupted. Such a condition could be indicated when accessing a PDSE results in an ABEND 0F4. The physical dump processing should complete successfully, compared to logical dump processing, because DFSMSdss is not using PDSE services to access the PDSE. Instead, the physical tracks related to the PDSE are dumped as an image. An IBM service representative may ask you to capture a DFSMSdss physical dump of a PDSE to preserve the PDSE for later analysis.

You can select data sets for DFSMSdss processing by filtering on specified criteria. DFSMSdss can filter on fully qualified or partially qualified data set names (by using the INCLUDE or EXCLUDE keyword) and on various data set characteristics (by using the BY keyword; for example BY(DSORG,EQ,PDSE)). DFSMSdss can automatically select data sets from volumes. To control the selection process, the BY keyword, as well as the EXCLUDE and INCLUDE keywords, can be used. See 6.1.1, “DFSMSdss filtering using the BY keyword” on page 119.

Both logical and physical operations are supported for PDSEs. Conversion between PDS and PDSE requires that the copy operation be logical. In general, we recommend the use of logical operations to avoid accidentally creating uncataloged data sets.

DFSMSdss is the primary data mover for PDSEs (as was the predecessor product, DFDSS); DFSMShsm calls DFSMSdss to handle PDSEs. PDSE support was first provided with DFDSS Version 2.5. Support for non-SMS PDSEs is available in DFSMSdss 1.4 and higher, and there are toleration APARs for DFSMSdss 1.2 and 1.3 that issue descriptive messages when a non-SMS PDSE is encountered. You may see either ADR285E or ADR778E with a new reason code of 14 informing you that a data set has not been processed because it is an unmanaged PDSE or HFS data set.

DFSMSdss can:

- Copy PDSEs (described in this and other sections with specific additional operands)
- Dump PDSEs (6.1.8, “DFSMSdss DUMP” on page 124)
- Restore PDSEs (6.1.9, “DFSMSdss RESTORE” on page 125)
- Release free space (6.1.3, “DFSMSdss RELEASE” on page 121)
- Convert between PDS and PDSE (4.2, “Converting a PDS to a PDSE” on page 70)
- Convert between PDSE and PDS (4.3, “Converting a PDSE to a PDS” on page 75)
- Reorganize PDSEs (6.1.6, “DFSMSdss PDSE content reorganization” on page 121)

6.1.1 DFSMSdss filtering using the BY keyword

You can easily select PDSEs by using the BY keyword on DFSMSdss commands. DFSMSdss uses its DSORG keyword differently from when specified on JCL, ISPF/PDF Panels, or in the ISMF data class application. Figure 6-1 shows an example.

```
...
COPY data set(INCLUDE(**) BY(DSORG,EQ,PDSE)) -
...
```

Figure 6-1 DFSMSdss BY keyword

You can also use DSORG,EQ,PDS to select only PDSs, or DSORG,EQ,PAM to select any partitioned data set, whether PDS or PDSE.

6.1.2 Maintaining PDSE space allocation

By default, DFSMSdss data set level COPY and DUMP operations will process a PDSE up to the high-formatted relative frame number. There may be allocated but unformatted space after the HFRFN. If you want to maintain the space allocation during a COPY or a DUMP, you must specify the ALLDATA(*) and ALLEXCP keywords.

Figure 6-2 on page 120 shows an example for DUMP processing; the same considerations apply for COPY processing. *All* allocated space will be dumped and later restored if you specify ALLDATA(*) ALLEXCP in your DFSMSdss DUMP job.

By default, DFSMSdss dumps only the used space instead of all of the allocated space if you do *not* use the ALLDATA or ALLEXCP keywords. Because DFSMSdss moves tracks and not individual pages, DFSMSdss cannot dump only the used pages in a PDSE.

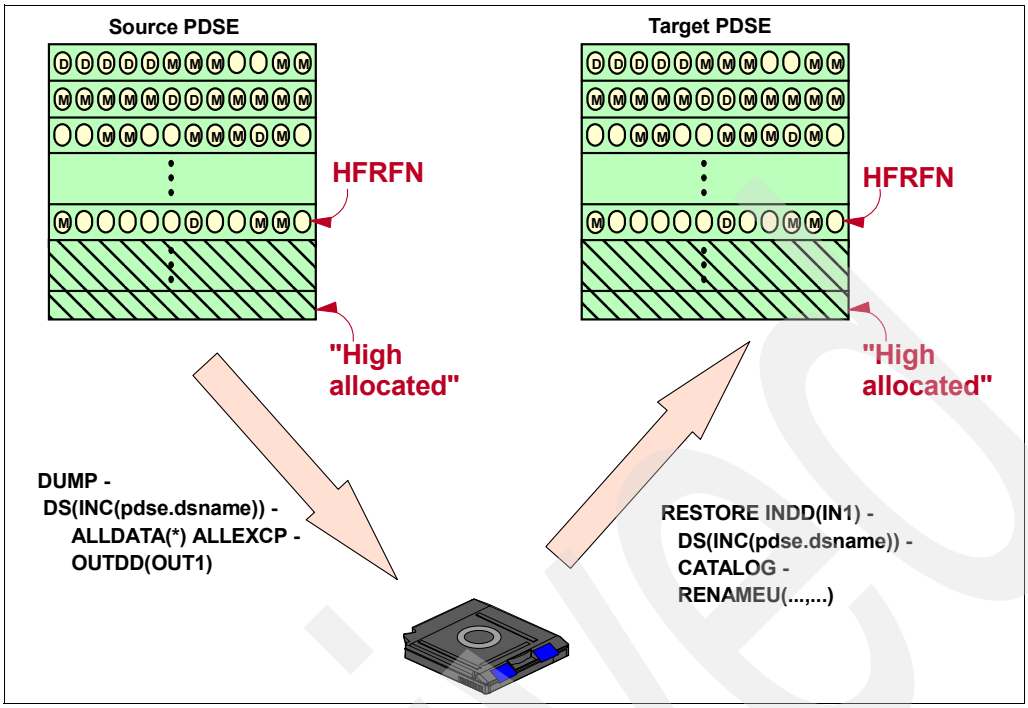


Figure 6-2 DFSMSdss DUMP/RESTORE with ALLDATA(*) ALLEXCP

All space up to the HFRFN value will be dumped and later restored if you do *not* specify the ALLDATA or ALLEXCP keywords. Figure 6-3 illustrates the significance of the HFRFN for DFSMSdss DUMP/RESTORE processing for PDSEs when the ALLDATA or ALLEXCP keywords are not specified.

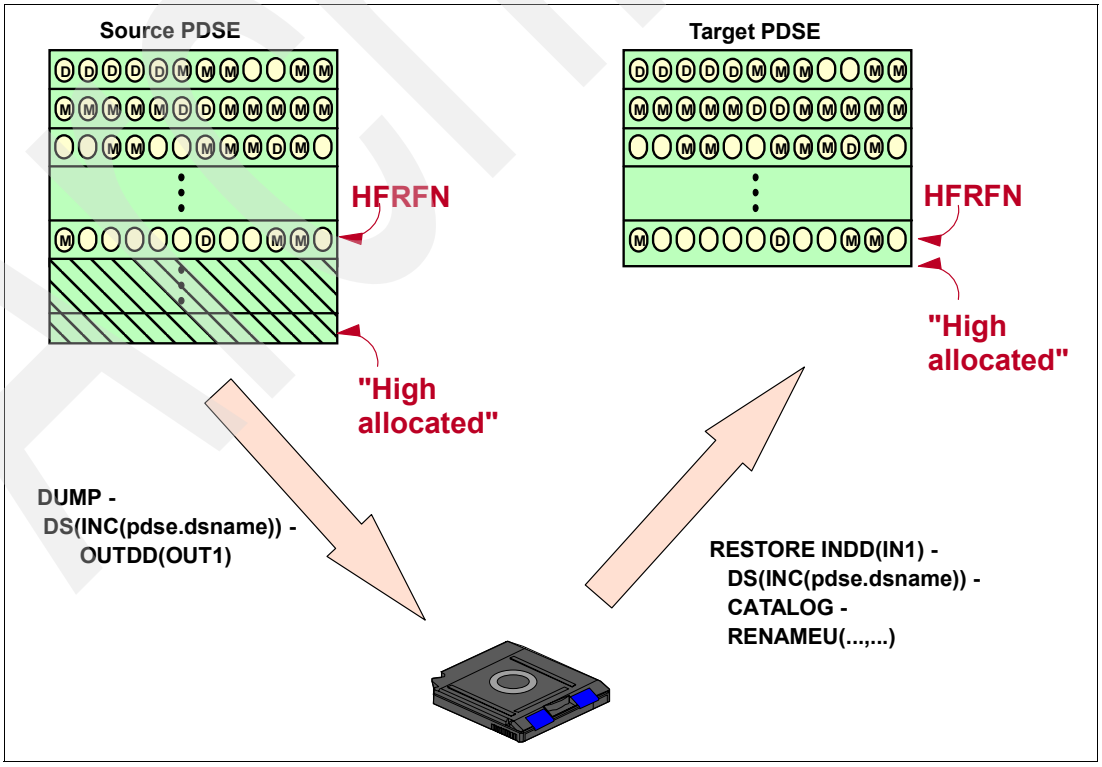


Figure 6-3 DFSMSdss DUMP/RESTORE without ALLDATA(*) ALLEXCP

6.1.3 DFSMSdss RELEASE

By default DFSMSdss only dumps the used part of a data set. If only the used part of the data set is required, then rather than use dump and restore, the excess space can be dropped using the RELEASE command.

The DFSMSdss RELEASE function can be used to release overallocated space from a PDSE. It frees the never-used tracks between the HFRFN and the high-allocated space in a PDSE data set, as shown in Figure 6-4. See also 5.1.9, “Partial release, free space” on page 82, and “High formatted relative frame number” on page 21.

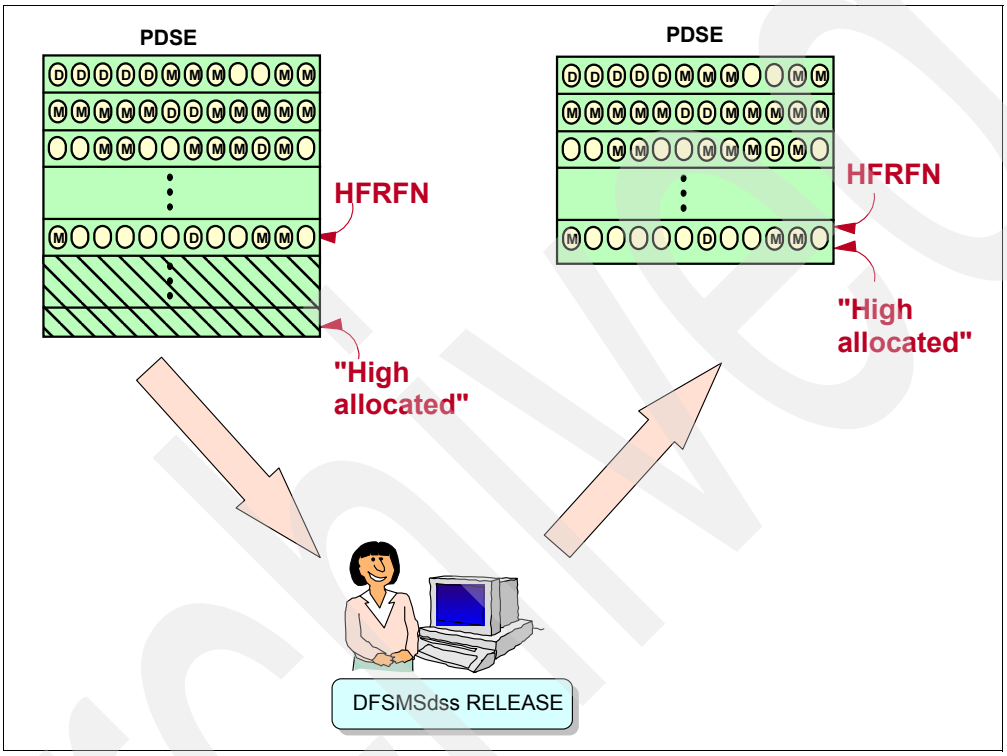


Figure 6-4 DFSMSdss RELEASE

6.1.4 DFSMSdss COMPRESS

The DFSMSdss COMPRESS function calls IEBCOPY to compress partitioned data sets.

It will not compress PDSEs because space is automatically reused anyway. If you run DFSMSdss COMPRESS against a list of data sets, the PDSs will be compressed but the PDSEs will not. If your job only selects PDSEs, you will receive message ADR327W, which tells you that no data sets were selected.

6.1.5 DFSMSdss PDS and PDSE conversion

Refer to Chapter 4, “Converting to PDSE format” on page 67 for conversion options.

6.1.6 DFSMSdss PDSE content reorganization

If a PDSE is updated frequently, the pages of a member may be placed in noncontiguous space. You can use DFSMSdss or IEBCOPY to reorganize the pages into contiguous space. DFSMSdss appears to do the work in one operation, but it performs a copy-delete-rename

sequence internally. To use IEBCOPY the copy, delete, and rename steps have to be carried out explicitly.

Figure 6-5 shows an example of a DFSMSdss job stream to reorganize the PDSEs on a volume. To make this operation applicable only to PDSEs, the COPY function should be qualified with the BY filter statement to select only PDSE.

PDSE reorganization is not essential for backups to be successful, but the better organized a PDSE when a backup is taken, the easier it will be to restore a data set. DFSMShsm (using DFSMSdss) does not reorganize data sets when it takes a backup, so if it was not well organized when the backup was made, it will remain that way when restored.

When the data set is filtered by DSORG, PDSs and PDSEs are selected as follows:

- When `BY(DSORG,EQ,PDS)` is specified, only PDSs are selected.
- When `BY(DSORG,EQ,PDSE)` is specified, only PDSEs are selected.
- When `BY(DSORG,EQ,PAM)` is specified, both PDSs and PDSEs are selected.

In the example shown in Figure 6-5, data sets to be processed have an HLQ of PDSERES, and data sets with LLQ of LOAD are to be excluded. However, load module data sets with other LLQs are not excluded. The DSORG associated with the data set reflects only the organization and not the content. Any data sets that are in use will not be processed.

Restriction: The data sets cannot be in use because they will be allocated and deleted.

```
//MHLRES1D JOB (999,POK),MSGLEVEL=1,NOTIFY=MHLRES1
//DFDSS EXEC PGM=ADRDSSU
//*
//* COPY PDSES TO REORGANIZE THE data set
//*
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 COPY DATASET( -
        INCLUDE(PDSERES.**) -
        EXCLUDE(**.LOAD) -
        BY(DSORG,EQ,PDSE) -
                ) -
      CATALOG -
      DELETE PURGE -
      REPLACE -
      CANCELERROR -
      WAIT(2,2)
```

Figure 6-5 Example of a job to reorganize PDSEs

Figure 6-6 shows the error messages resulting from running the job shown in Figure 6-5 when data set PDSERES.SMS.DIRK.PDSE1 was in use. In particular, note ADR410E, ADR801I, and ADR455W.

```
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'COPY '
ADR109I (R/I)-RI01 (01), 2004.310 17:23:45 INITIAL SCAN OF USER CONTROL STATEMENTS
COMPLETED.
ADR016I (001)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR006I (001)-STEND(01), 2004.310 17:23:45 EXECUTION BEGINS
ADR410E (001)-DDFLT(01), data set PDSERES.SMS.DIRK.PDSE1 IN CATALOG UCAT.VSBOX01 ON
VOLUME SBOX26 FAILED SERIALIZATION FOR DELETE.
ADR801I (001)-DDDS (01), data set FILTERING IS COMPLETE. 0 OF 1 data setS WERE SELECTED:
1 FAILED SERIALIZATION AND 0 FAILED FOR
               OTHER REASONS.
ADR455W (001)-DDDS (02), THE FOLLOWING data setS WERE NOT SUCCESSFULLY PROCESSED
               PDSERES.SMS.DIRK.PDSE1
ADR006I (001)-STEND(02), 2004.310 17:23:50 EXECUTION ENDS
ADR013I (001)-CLTSK(01), 2004.310 17:23:50 TASK COMPLETED WITH RETURN CODE 0008
ADR012I (SCH)-DSSU (01), 2004.310 17:23:50 DFSMSDSS PROCESSING COMPLETE. HIGHEST RETURN
CODE IS 0008 FROM:
                          TASK    001
```

Figure 6-6 Example of the error messages received when a data set is in use

These job streams can be invoked automatically on a volume basis by using the DFSMShsm space management volume exit, ARCMVEXT. When this exit is used, DFSMSdss must use the LOGINDDNAME or LOGINDYNAM parameters to execute on a volume basis. Figure 6-7 illustrates this. The code in the ARCMVEXT module would supply the value to be used in the LOGINDYNAM operand shown as `smsvol` in the example.

```
//MHLRES1D JOB (999,POK),MSGLEVEL=1,NOTIFY=MHLRES1
//DFDSS EXEC PGM=ADRDSSU
//*
//* COPY PDSES TO REORGANIZE THE data set
//*
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 COPY data set( -
        INCLUDE(PDSERES.**) -
        EXCLUDE(**.LOAD) -
        BY(DSORG,EQ,PDSE) -
                ) -
      LOGINDYNAM(smsvol) -
      CATALOG -
      DELETE PURGE -
      REPLACE -
      CANCELERROR -
      WAIT(2,2)
```

Figure 6-7 Sample parameters for DFSMSdss invoked through DFSMShsm exit

See *DFSMShsm Implementation and Customization,* SH21-1078, for more about this exit.

Note: When DFSMSdss is used to move a data set, it creates a new data set whose Format-1 DSCB contains a new creation date and last reference date. The migrations and backups with DFSMShsm are based on the format 1 DSCB of the new data set.

For more about using DFSMSdss, refer to *DFSMSdss Storage Administration Guide,* SC26-4930, and *DFSMSdss Storage Administration Reference*, SC26-4929.

6.1.7 Concurrent copy and SnapShot

WIth DFSMSdss 1.3 and later, the PTFs for APAR OW38672 add support for concurrent copy and SnapShot to DFSMSdss logical copies.

If, as part of the copy operation, there is a conversion between PDS and PDSE, SnapShot will not be used.

6.1.8 DFSMSdss DUMP

Refer to the introduction to this chapter for an explanation of the differences between physical and logical processing in 6.1, “DFSMSdss” on page 118.

DFSMSdss can perform either logical or physical processing, and can operate at a data set level or a full volume level. If you dump a data set logically, DFSMSdss restores it logically; if you dump it physically, DFSMSdss restores it physically.

Figure 6-8 shows an example of a job to dump all data sets on the system with HLQ PDESRES that are in PDSE format, and excluding any with an LLQ of LOAD. The output is defined in the JCL on DDNAME DDDUMP and then referred to in the control statements.

```
//MHLRES1D JOB (999,POK),MSGLEVEL=1,NOTIFY=MHLRES1
//DFDSS EXEC PGM=ADRDSSU
//*
//* DUMP PDSES
//*
//SYSPRINT DD SYSOUT=*
//DDDUMP DD DISP=(,CATLG,DELETE),UNIT=SYSALLDA,
//       DSN=PDESRES.DSSD1.DUMP,
//       SPACE=(TRK,(28000,2800))
//SYSIN DD *
 DUMP data set( -
        INCLUDE(PDSERES.**) -
        EXCLUDE(**.LOAD) -
        BY(DSORG,EQ,PDSE) -
                ) -
      CANCELERROR -
      OUTDDNAME(DDDUMP) -
      WAIT(2,2)
```

Figure 6-8 Example of a job to dump data sets in PDSE format

Figure 6-9 shows part of the output from a dump job. The job did not specify which data sets to dump, so the catalog was used to find all of those with HLQ PDSERES, then any that were not in PDSE format were excluded along with any with LLQ of LOAD. This results in a logical form of dump.

```
PAGE 0001  5695-DF175 DFSMSDSS V1R06.0 data set SERVICES  2004.313 17:24
 DUMP data set( -
    INCLUDE(PDSERES.**) -
    EXCLUDE(**.LOAD) -
    BY(DSORG,EQ,PDSE) -
                ) -
     CANCELERROR -
    OUTDDNAME(DDDUMP) -
    WAIT(2,2)
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'DUMP '
ADR109I (R/I)-RI01 (01), 2004.313 17:24:13 INITIAL SCAN OF USER CONTROL
                         STATEMENTS COMPLETED.
ADR016I (001)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR006I (001)-STEND(01), 2004.313 17:24:13 EXECUTION BEGINS
ADR801I (001)-DTDSC(01), data set FILTERING IS COMPLETE. 4 OF 4 data setS WERE SELECTED:
0 FAILED SERIALIZATION AND 0 FAILED FOR
                OTHER REASONS.
ADR454I (001)-DTDSC(01), THE FOLLOWING data setS WERE SUCCESSFULLY
                                                      PROCESSED
            PDSERES.CEE.SCEERUN2.RELOADED
            PDSERES.CEE.SCEERUN2.RELOADED.CDAEED
            PDSERES.CEE.SCEERUN2.RELOADED.CRTEC128
            PDSERES.CEE.SCEERUN2.RELOADED.C128
ADR006I (001)-STEND(02), 2004.313 17:24:51 EXECUTION ENDS
ADR013I (001)-CLTSK(01), 2004.313 17:24:51 TASK COMPLETED WITH RETURN
                                                        CODE  0000
ADR012I (SCH)-DSSU (01), 2004.313 17:24:51 DFSMSDSS PROCESSING COMPLETE.
                                      HIGHEST RETURN CODE IS 0000
```

Figure 6-9 Part of the output from the job to dump PDSE format data sets

If the system has a restartable address space SMSPDSE1, and it is restarted while the dump job is running, the job is suspended until SMSPDSE1 is operational again. No special requirements are needed to use this support.

The only special PDSE consideration for DFSMSdss dump is that when a PDSE is open for update, only a logical dump may be taken, only using concurrent copy, and TOLERATE(ENQFAILURE) must be specified.

6.1.9 DFSMSdss RESTORE

The only special PDSE consideration for DFSMSdss restore is that you may only restore a data set in the same mode that it was dumped. If you dump a data set logically, DFSMSdss restores it logically; if you dump it physically, DFSMSdss restores it physically. The data set is not reorganized on backup or restore, so if it is in need of reorganization this should be carried out first.

The data set will not change from PDS to PDSE or vice versa on restore, but depending on the SMS ACS processing, it may be restored in SMS or non-SMS mode.

Figure 6-10 shows an example of a job to restore data sets from a logical dump. The filter keyword BY, and the data set INCLUDE and EXCLUDE keywords, specify the same values as in the job shown in Figure 6-8 on page 124, although these are not required in this case because the only data sets in the dump data set correspond with these values. If there had been other data sets in the dump, then the BY, INCLUDE, and EXCLUDE keywords would restrict the restore to those data sets that pass the criteria.

```
//MHLRES1D JOB (999,POK),MSGLEVEL=1,NOTIFY=MHLRES1
//DFDSS EXEC PGM=ADRDSSU PARM='TYPRUN=SCAN'
//*
//* RESTORE PDSES
//*
//SYSPRINT DD SYSOUT=*
//DDREST DD DISP=SHR,
//       DSN=PDESRES.DSSD1.DUMP
//SYSIN DD *
 RESTORE data set( -
        INCLUDE(PDSERES.**) -
        EXCLUDE(**.LOAD) -
        BY(DSORG,EQ,PDSE) -
                ) -
      RENAMEUNC(MHLRES5) -
      CATALOG -
      CANCELERROR -
      INDDNAME(DDREST) -
      WAIT(2,2)
```

Figure 6-10 Example job to restore data sets from a logical dump

Figure 6-11 on page 127 shows partial output from the job to restore data sets from a logical dump. Note that all the data sets that meet the BY, INCLUDE, and EXCLUDE keywords are restored. In addition, the HLQ is changed from PDSERES to MHLRES5. Note message ADR780I, which indicates that the restore is from a logical dump.

```
PAGE 0001   5695-DF175 DFSMSDSS V1R06.0 data set SERVICES   2004.313 17:58
 RESTORE data set( -
      INCLUDE(PDSERES.**) -
      EXCLUDE(**.LOAD) -
      BY(DSORG,EQ,PDSE) -
           ) -
    RENAMEUNC(MHLRES5) -
      CANCELERROR -
CATALOG -
'INDDNAME(DDREST) -
    WAIT(2,2)
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'RESTORE '
ADR109I (R/I)-RI01 (01), 2004.313 17:58:48 INITIAL SCAN OF USER CONTROL STATEMENTS
COMPLETED.
ADR016I (001)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR006I (001)-STEND(01), 2004.313 17:58:48 EXECUTION BEGINS
ADR780I (001)-TDDS (01), THE INPUT DUMP data set BEING PROCESSED IS IN LOGICAL data set
                                          FORMAT AND WAS CREATED BY DFSMSDSS VERSION
                        1 RELEASE 6 MODIFICATION LEVEL 0
ADR395I (001)-NEWDS(01), data set PDSERES.CEE.SCEERUN2.RELOADED ALLOCATED WITH NEWNAME
          MHLRES5.CEE.SCEERUN2.RELOADED, ON VOLUME(S):
                SBOXEE
ADR474I (001)-TDNVS(01), data set MHLRES5.CEE.SCEERUN2.RELOADED CONSISTS OF 00012705
        TARGET TRACKS AND 00012705 SOURCE TRACKS
ADR489I (001)-TDLOG(01), data set MHLRES5.CEE.SCEERUN2.RELOADED WAS RESTORED
ADR395I (001)-NEWDS(01), data set PDSERES.CEE.SCEERUN2.RELOADED.CDAEED ALLOCATED WITH
      NEWNAME MHLRES5.CEE.SCEERUN2.RELOADED.CDAEED,
               ON VOLUME(S): SBOXEE
ADR474I (001)-TDNVS(01), data set MHLRES5.CEE.SCEERUN2.RELOADED.CDAEED CONSISTS OF
      00000027 TARGET TRACKS AND 00000027 SOURCE
               TRACKS
ADR489I (001)-TDLOG(01), data set MHLRES5.CEE.SCEERUN2.RELOADED.CDAEED WAS RESTORED
ADR395I (001)-NEWDS(01), data set PDSERES.CEE.SCEERUN2.RELOADED.CRTEC128 ALLOCATED WITH
          NEWNAME
          MHLRES5.CEE.SCEERUN2.RELOADED.CRTEC128, ON VOLUME(S): SBOXEE
ADR474I (001)-TDNVS(01), data set MHLRES5.CEE.SCEERUN2.RELOADED.CRTEC128 CONSISTS OF
         00000154 TARGET TRACKS AND 00000154 SOURCE
                TRACKS
ADR489I (001)-TDLOG(01), data set MHLRES5.CEE.SCEERUN2.RELOADED.CRTEC128 WAS RESTORED
ADR395I (001)-NEWDS(01), data set PDSERES.CEE.SCEERUN2.RELOADED.C128 ALLOCATED WITH
        NEWNAME MHLRES5.CEE.SCEERUN2.RELOADED.C128, ON
              VOLUME(S): SBOXEE
ADR474I (001)-TDNVS(01), data set MHLRES5.CEE.SCEERUN2.RELOADED.C128 CONSISTS OF
        00000154 TARGET TRACKS AND 00000154 SOURCE TRACKS
ADR489I (001)-TDLOG(01), data set MHLRES5.CEE.SCEERUN2.RELOADED.C128 WAS RESTORED
ADR454I (001)-TDLOG(01), THE FOLLOWING data setS WERE SUCCESSFULLY PROCESSED
             PDSERES.CEE.SCEERUN2.RELOADED
             PDSERES.CEE.SCEERUN2.RELOADED.CDAEED
             PDSERES.CEE.SCEERUN2.RELOADED.CRTEC128
             PDSERES.CEE.SCEERUN2.RELOADED.C128
ADR006I (001)-STEND(02), 2004.313 18:00:07 EXECUTION ENDS
ADR013I (001)-CLTSK(01), 2004.313 18:00:07 TASK COMPLETED WITH RETURN CODE 0000
ADR012I (SCH)-DSSU (01), 2004.313 18:00:07 DFSMSDSS PROCESSING COMPLETE. HIGHEST RETURN
CODE IS 0000
```

Figure 6-11 Partial output from job to restore data sets from a logical dump

DFSMSdss does not permit a data set that already exists to be overwritten unless the REPLACE keyword is issued. Figure 6-12 shows an example of the output from a restore job that selects an existing data set, but without the REPLACE keyword being specified.

```
RESTORE data set( -
     INCLUDE(PDSERES.CEE.SCEERUN2.RELOADED) -
      EXCLUDE(**.LOAD) -
      BY(DSORG,EQ,PDSE) -
            ) -
    RENAMEUNC(MHLRES5) -
    CANCELERROR -
    INDDNAME(DDREST) -
    WAIT(2,2)
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'RESTORE '
ADR109I (R/I)-RI01 (01), 2004.313 18:26:32 INITIAL SCAN OF USER CONTROL STATEMENTS
COMPLETED.
ADR016I (001)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR006I (001)-STEND(01), 2004.313 18:26:32 EXECUTION BEGINS
ADR780I (001)-TDDS (01), THE INPUT DUMP data set BEING PROCESSED IS IN LOGICAL data set
FORMAT AND WAS CREATED BY DFSMSDSS VERSION
               1 RELEASE 6 MODIFICATION LEVEL 0
ADR392E (001)-FRLBO(01), PDSERES.CEE.SCEERUN2.RELOADED EXISTS ON SBOXEE WITH NEWNAME
                         MHLRES5.CEE.SCEERUN2.RELOADED
ADR415W (001)-TDLOG(01), NO data sets  WERE COPIED, DUMPED, OR RESTORED FROM  ANY
                         VOLUME
ADR480W (001)-TDLOG(01), THE FOLLOWING data setS WERE NOT PROCESSED FROM THE LOGICALLY
FORMATTED DUMP TAPE DUE TO ERRORS:
               PDSERES.CEE.SCEERUN2.RELOADED
ADR006I (001)-STEND(02), 2004.313 18:26:32 EXECUTION ENDS
ADR013I (001)-CLTSK(01), 2004.313 18:26:32 TASK COMPLETED WITH RETURN CODE 0008
```

Figure 6-12 Partial output from job restoring a data set that already exists

If the system has a restartable address space SMSPDSE1, and it is restarted while the restore job is running, the job is suspended until SMSPDSE1 is operational again. No special requirements are needed to use this support.

6.1.10 DFSMSdss restore considerations for system or shared user data sets

Most system or shared user data sets are already allocated to some task, and so cannot be replaced as whole data sets.

However, it may be possible to restore a specific member of a data set from a backup if the data set is first restored with a different name. After a copy of the data set has been loaded, and the member concerned has been verified as a suitable replacement, then IEBCOPY can be used to copy it, with any aliases, into the allocated data set. System and shared user data sets should not have any exclusive ENQs.

Important: It is essential that all aliases be copied at the same time as the primary member name.This can be done in general by specifying all alias names, or for PDSEs by using the COPYGRP statement.

Refer to 6.3.9, “IEBCOPY copying PDSE to PDSE” on page 138 for examples of correctly set up IEBCOPY jobs.

6.2 DFSMShsm

DFSMShsm manages PDSEs by using DFSMSdss as the data mover. It provides availability management and space management services for PDSEs. PDSE support was first provided with DFSMShsm Version 2.5 together with DFSMSdss Version 2.5. All space and availability management functions are supported.

DFSMSdss, and therefore DFSMShsm, moves space up to the HFRFN, but not all the space that is allocated. So, when a PDSE is backed up or migrated, unused space is released down to the HFRFN. If the data set is allocated with secondary extents, a recall or recover will result in the data set being allocated with only the space needed to hold the data.

More detail is available in *DFSMShsm Storage Administration Guide*, SH21-1076, and *DFSMShsm Storage Administration Reference*, SH21-1075.

Attention: The automatic backup and migration functions provided by DFSMShsm are suitable for user data sets as the user is able to manage their allocation. For system data sets and shared user data sets, DFSMShsm may be able to take backups, but may be restricted in the extent to which data sets can be migrated or restored. Users should verify that DFSMShsm is making backups as expected.

If the system has a restartable address space SMSPDSE1, and it is restarted while a DFSMShsm operation is running on PDSEs, the operation is suspended until SMSPDSE1 is operational again. No special requirements are needed to use this support.

6.2.1 DFSMShsm availability management

DFSMShsm availability management provides automatic and command backup functions and the ability to restore from backup copies. It also automates DFSMSdss volume dumps.

For SMS-managed PDSEs, the management class attributes determine how the PDSE will be managed.

If you have a non-SMS PDSE, it might not cataloged in the standard catalog search order. DFSMShsm will not back up uncataloged data sets. If it encounters uncataloged PDSEs during daily backup, it will issue message ARC1399I with reason code 50.

If you are starting to use non-SMS PDSEs as well as SMS-managed PDSEs, remember that they will be managed according to these parameter specifications in ARCCMDxx or by operator command. To control the number of backup versions kept, use:

```
SETSYS VERSIONS
```

To determine the interval between backups, use:

```
SETSYS FREQUENCY
```

6.2.2 DFSMShsm space management

DFSMShsm space management provides automatic and command migration functions and automatic recall upon reference.

During migration and recall operations, DFSMShsm does not reorganize PDSEs, but it does release unused space down to the HFRFN and reduce extents.

Extent reduction is done by migrating the PDSE and immediately scheduling a recall. Obviously this cannot be done if the data set is allocated somewhere. The recall brings the PDSE back with an initial allocation large enough to contain the space up to the HFRFN. Extent reduction is controlled by the ARCCMDxx parameter:

```
SETSYS MAXEXTENTS(n)
```

For SMS-managed PDSEs, space management is controlled by management class attributes. For non-SMS PDSEs, the space management actions are determined by ARCCMDxx parameters. The main parameters are:

SETSYS DAYS(n) Specifies how many days since last use should elapse before a data set is eligible for migration from a primary volume to ML1.

SETSYS ML1DAYS(n) Specifies how many days since last use should elapse before a data set is eligible for migration from an ML1 volume to an ML2 volume.

SETSYS RECALL ... Specifies where recalled data sets should go.

Free space release is controlled by management class attributes for SMS-managed PDSEs. The management class must have the partial release attribute set to one of these choices:

Y Yes. Space is released as part of the daily space management cycle.

YI Yes immediate. Space is released at close time when the data set is open for output and as part of the daily space management cycle.

C Conditional. Space is released as part of the daily space management cycle if the data set was allocated with a non-zero secondary allocation quantity.

CI Conditional immediate. Space is released at close time when the data set is open for output and as part of the daily space management cycle if the data set was allocated with a non-zero secondary allocation quantity.

For non-SMS PDSEs, space is released by DFSMShsm only if the data set becomes eligible for extent reduction.

6.2.3 DFSMShsm commands

If you specify a partitioned data set with a member name in a DFSMShsm command, DFSMShsm rejects the request. This is true for PDSs and PDSEs.

You can force a recall of a PDSE to an unmanaged volume using this DFSMShsm authorized command:

```
RECALL data set FORCENONSMS VOLUME(xxxxxx)
```

This provides a simple way to convert a PDSE from managed to non-managed: migrate the PDSE, then recall it as shown. You must use the authorized command. The user version of the command, HRECALL, does not provide the ability to force a non-SMS allocation.

6.2.4 DFSMShsm ABARS

The ABARS selection data set may be a PDSE, a PDS, or a sequential data set.

An ABARS ABACKUP will fail if it encounters an open PDSE. You may use the ARCBEEXT exit to continue the ABACKUP.

6.3 IEBCOPY

The IEBCOPY utility can also be used to perform several functions on PDSEs. We give a brief overview in this section; for details, see *DFSMS/MVS Utilities*, SC26-4926.

When IEBCOPY is used for PDSs alone, a minimum region size of 1 MB is recommended. When it is used for PDSEs, the minimum region size should be 2 MB.

If you copy a PDSE with IEBCOPY, the PDSE is processed at a member level and the HFRFN is reset. This means that any fragmentation within the PDSE will be removed.

After you have used IEBCOPY in this way, either PARTREL or DFSMSdss functions such as DUMP/RESTORE, COPY, and RELEASE can be used to reduce the size of PDSEs with a low utilization but a large allocation quantity.

This is shown in Figure 6-13, where we assume that the target PDSE is allocated with the same size as the source PDSE.

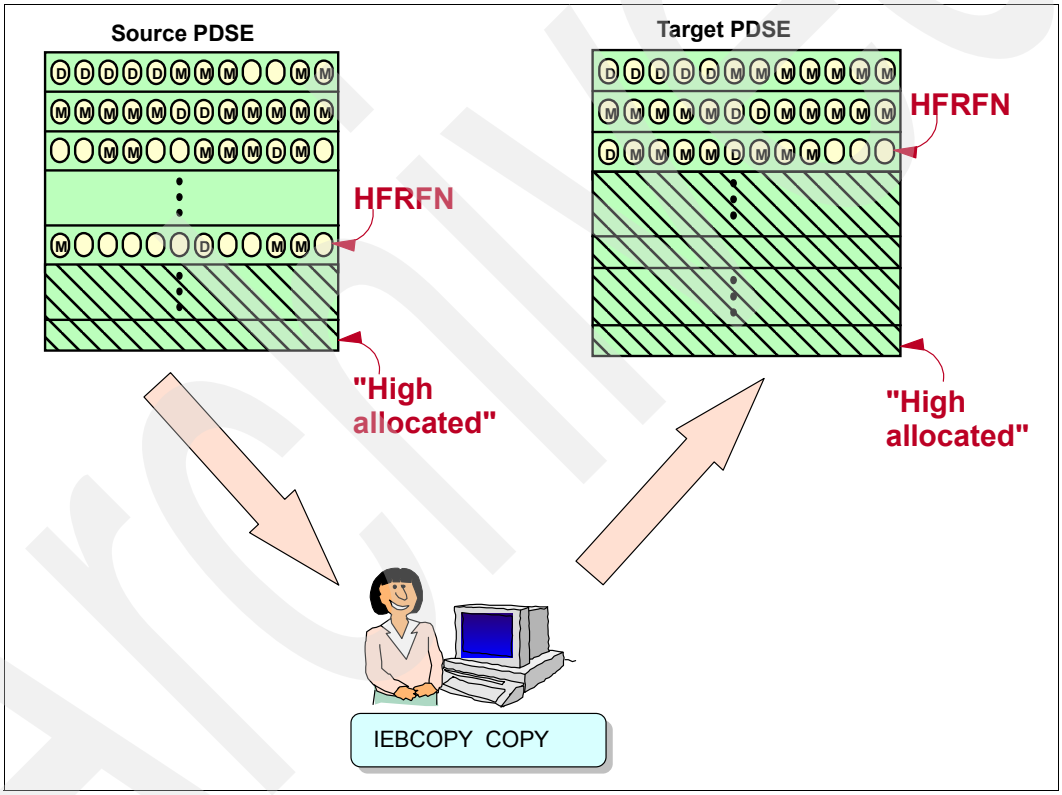


Figure 6-13 IEBCOPY and PDSE space usage

6.3.1 IEBCOPY compression

IEBCOPY is used only to compress PDSs. If you try to use IEBCOPY to compress a PDSE, IEBCOPY will ignore the request and issue this message:

```
IEB1019I SKIPPING COMPRESS OF PDSE ...
```

6.3.2 IEBCOPY Unload

IEBCOPY can be used to unload a PDS or a PDSE to a special sequential format. The sequential data set produced by the unload can be on disk or tape. An unload is done with an

IEBCOPY COPY operation to a sequential output data set. No control statements are required, but SYSIN must be provided as DD DUMMY or with a null input stream. The sequential data set will be approximately the same size as the input.

For IEBCOPY Unload purposes, it makes no difference whether the input is a PDS or PDSE. However, the content of some of the messages does change depending on whether the input is a PDS or PDSE and whether the output is a PDS, PDSE, or PDSU. PDSU indicates that an unloaded PDS is being processed.

Figure 6-14 shows a job to unload a PDSE into a sequential data set. The input data set specified on SYSUT1 is a system data set in PDSE format.

```
//MHLRES1D JOB (999,POK),MSGLEVEL=1,NOTIFY=MHLRES1
//*
//* IEBCOPY JOB TO UNLOAD A PDS OR PDSE TO A SEQUENTIAL data set
//*
//IEBCOPY EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=SHR,
//         DSN=CEE.SCEERUN2
//SYSUT2   DD DISP=(,CATLG,DELETE),UNIT=SYSALLDA,
//         SPACE=(TRK,(15000,1500),RLSE),
//         DSN=PDSERES.SCEERUN2.UNLOADED
//SYSIN DD DUMMY
```

Figure 6-14 Example job to unload a PDSE into a sequential data set

Figure 6-15 shows part of the output of the job showing a copy being performed, and also that alias members were unloaded. This information may be required for successful selective member reload. In particular, message IEB014I indicates that the output is a PDSU format data set, and note that any occurrences of message IGW01553I provides alias names.

```
IEBCOPY MESSAGES AND CONTROL STATEMENTS                          PAGE     1
IEB1135I IEBCOPY  FMID HDZ11H0  SERVICE LEVEL NONE     DATED 20040319 DFSMS 01.06.00
z/OS    01.06.00 HBB7709  CPU 2084
IEB1035I MHLRES1D IEBCOPY 18:29:47 FRI 05 NOV 2004 PARM=''
IEBCOPY COPY   INDD=SYSUT1,OUTDD=SYSUT2     GENERATED STATEMENT
IEB1013I COPYING  FROM PDSE INDD=SYSUT1  VOL=Z16RB1 DSN=CEE.SCEERUN2
IEB1014I         TO PDSU OUTDD=SYSUT2 VOL=SBOXEC DSN=PDSERES.SCEERUN2.UNLOADED
IGW01553I ALIAS CCNELDEF OF COPIED PRIMARY CEL4ELDF HAS BEEN UNLOADED
IGW01553I ALIAS CCNLMSGS OF COPIED PRIMARY CEL4ELMS HAS BEEN UNLOADED
IGW01551I MEMBER CDAEED  HAS BEEN UNLOADED
IGW01551I MEMBER CDAEQED  HAS BEEN UNLOADED
```

Figure 6-15 Partial output from running the example to unload a PDSE

6.3.3 IEBCOPY Reload

An advantage of using IEBCOPY Unload is that a selective reload of either the whole PDS or a particular member can be specified, which is not an option with DFDSS.

Notes:

- In general, specification of DSNTYPE=LIBRARY with either DSORG=PO or directory blocks not equal to 0 should cause a PDSE to be allocated, whatever form of the SPACE parameter is used. To allocate a PDSE using JCL, the third parameter, which is the number of directory blocks, is normally ignored. However, when the output data set is specified without a number of directory blocks, IEBCOPY assumes that the output is to be a sequential data set. Specification of at least 1 directory block alleviates this problem.

  `SPACE=(trk,(1,1))` signifies that a sequential unload data set is to be created.

  `SPACE=(trk,(1,1,1))` will allocate a PDS or PDSE depending on SMS ACS; whether DSNTYPE is specified as PDS or LIBRARY; or both.
- You must specify the primary name and all aliases to reload a complete duplicate of a member. If just the primary name is specified, IEBCOPY issues a warning message but continues with a 0 return code, but the resulting structure is unlikely to work properly. An exception to this is provided with the COPYGRP control statement to use with PDSEs.
- IEBCOPY has a PARM option RC4NOREP that is intended to cause a return code 4 if a copy is attempted to an output data set that already contains the same member names. For PDS format data set this option works, but for PDSE it currently has no effect.

6.3.4 IEBCOPY reload effect when directory blocks omitted

Figure 6-16 shows an example of a job to reload a complete unloaded PDSE, without any directory blocks on the SPACE parameter. This is provided as a demonstration of what happens, and is not intended to be used under normal circumstances.

```
//MHLRES1D JOB (999,POK),MSGLEVEL=1,NOTIFY=MHLRES1
//*
//* IEBCOPY JOB TO RELOAD A PDS OR PDSE FROM A SEQUENTIAL data set
//*
//IEBCOPY EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=OLD,
//         DSN=PDSERES.SCEERUN2.UNLOADED
//SYSUT2   DD DISP=(,CATLG,DELETE),
//         SPACE=(TRK,(14000,1500)),
//         UNIT=SYSALLDA,
//         DSNTYPE=LIBRARY,
//         DSN=PDSERES.CEE.SCEERUN2.RELOADED.BAD
//SYSIN DD DUMMY
```

Figure 6-16 Example of an attempt to reload a PDSE without any directory blocks specified

Figure 6-17 on page 134 shows part of the output from the reload, including the messages IEB1013I for the input and IEB1014I for the output. The output was another unloaded (PDSU) form of the data set.

```
 IEBCOPY MESSAGES AND CONTROL STATEMENTS                                 PAGE     1
IEB1135I IEBCOPY  FMID HDZ11H0  SERVICE LEVEL NONE     DATED 20040319 DFSMS 01.06.00
z/OS    01.06.00 HBB7709  CPU 2084
IEB1035I MHLRES1D IEBCOPY 18:38:05 FRI 05 NOV 2004 PARM=''
IEBCOPY COPY   INDD=SYSUT1,OUTDD=SYSUT2     GENERATED STATEMENT
IEB1013I COPYING FROM PDSU  INDD=SYSUT1   VOL=SBOXEC DSN=PDSERES.SCEERUN2.UNLOADED
IEB1014I           TO PDSU OUTDD=SYSUT2   VOL=SBOXEF DSN=PDSERES.CEE.SCEERUN2.RELOADED
IEB167I FOLLOWING MEMBER(S) COPIED FROM INPUT data set REFERENCED BY SYSUT1
IEB154I CCNELDEF HAS BEEN SUCCESSFULLY COPIED
IEB154I CCNLMSGS HAS BEEN SUCCESSFULLY COPIED
IEB154I CDAEED  HAS BEEN SUCCESSFULLY COPIED
IEB154I CDAEQED  HAS BEEN SUCCESSFULLY COPIED
```

Figure 6-17 Partial output from running the job without directory blocks specified

6.3.5 IEBCOPY reload with directory blocks specified

Figure 6-18 shows an example of a job to reload a complete unloaded PDSE, with the directory blocks specified on the SPACE operand.

```
//MHLRES1D JOB (999,POK),MSGLEVEL=1,NOTIFY=MHLRES1
//*
//* IEBCOPY JOB TO RELOAD A PDS OR PDSE FROM A SEQUENTIAL data set
//*
//IEBCOPY EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=OLD,
//         DSN=PDSERES.SCEERUN2.UNLOADED
//SYSUT2   DD DISP=(,CATLG,DELETE),
//         SPACE=(TRK,(14000,1500,1)),
//         UNIT=SYSALLDA,
//         DSNTYPE=LIBRARY,
//         DSN=PDSERES.CEE.SCEERUN2.RELOADED
//SYSIN DD DUMMY
```

Figure 6-18 Example of a job to reload an unloaded PDSU data set with directory blocks specified

Figure 6-19 on page 135 shows part of the output from the reload, including the message that indicates that the output was a PDSE as intended. Message IEB1013I indicates that the input data set was in PDSU format, and IEB1014I indicates that the output is in PDSE format. Also note messages IGW01553I that list alias members that have been reloaded. These should correspond with similar messages in the unload job.

```
 IEBCOPY MESSAGES AND CONTROL STATEMENTS                                 PAGE     1
IEB1135I IEBCOPY  FMID HDZ11HO  SERVICE LEVEL NONE     DATED 20040319 DFSMS 01.06.00
z/OS    01.06.00 HBB7709  CPU 2084
IEB1035I MHLRES1D  IEBCOPY  18:45:47 FRI 05 NOV 2004 PARM=''
IEBCOPY COPY   INDD=SYSUT1,OUTDD=SYSUT2     GENERATED STATEMENT
IEB1013I COPYING FROM PDSU INDD=SYSUT1  VOL=SBOXEC DSN=PDSERES.SCEERUN2.UNLOADED
IEB1014I      TO PDSE OUTDD=SYSUT2  VOL=SBOXEE DSN=PDSERES.CEE.SCEERUN2.RELOADED
IGW01553I ALIAS CCNELDEF OF COPIED PRIMARY CEL4ELDF HAS BEEN LOADED
IGW01553I ALIAS CCNLMSGS OF COPIED PRIMARY CEL4ELMS HAS BEEN LOADED
IGW01551I MEMBER CDAEED  HAS BEEN LOADED
IGW01551I MEMBER CDAEQED  HAS BEEN LOADED
IGW01551I MEMBER CEH$AAAR HAS BEEN LOADED
```

Figure 6-19 Partial output from a job that correctly reloads an unloaded PDSE

6.3.6 IEBCOPY reload with aliases specified

The job in Figure 6-20 shows the selection of the primary member name CRTC128 and its alias C128.

```
//MHLRES1D JOB (999,POK),MSGLEVEL=1,NOTIFY=MHLRES1
//*
//* IEBCOPY JOB TO RELOAD A PDS OR PDSE FROM A SEQUENTIAL data set
//*
//IEBCOPY EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=OLD,
//         DSN=PDSERES.SCEERUN2.UNLOADED
//SYSUT2   DD DISP=(,CATLG,DELETE),
//         SPACE=(TRK,(14000,1500,1),RLSE),
//         UNIT=SYSALLDA,
//         DSNTYPE=LIBRARY,
//         DSN=PDSERES.CEE.SCEERUN2.RELOADED.C128
//SYSIN DD *
 COPY INDD=SYSUT1,OUTDD=SYSUT2
 S M=CRTEC128
 S M=C128
```

Figure 6-20 Example job to reload a member with an alias

Figure 6-21 on page 136 shows an example of a job to reload a specific member (that does have an alias) from an unloaded PDSE.

```
 IEBCOPY MESSAGES AND CONTROL STATEMENTS                                PAGE     1
IEB1135I IEBCOPY  FMID HDZ11HO  SERVICE LEVEL NONE     DATED 20040319 DFSMS 01.06.00
z/OS    01.06.00 HBB7709  CPU 2084
IEB1035I MHLRES1D IEBCOPY 18:59:14 FRI 05 NOV 2004 PARM=''
 COPY INDD=SYSUT1,OUTDD=SYSUT2
 S M=CRTEC128
 S M=C128
IEB1013I COPYING FROM PDSU INDD=SYSUT1 VOL=SBOXEC DSN=PDSERES.SCEERUN2.UNLOADED
IEB1014I           TO PDSE OUTDD=SYSUT2   VOL=SBOXEE
DSN=PDSERES.CEE.SCEERUN2.RELOADED.C128
IGW01551I MEMBER CRTEC128 HAS BEEN LOADED
IGW01553I ALIAS C128 OF COPIED PRIMARY CRTEC128 HAS BEEN LOADED
IGW01550I 2 OF 2 SPECIFIED MEMBERS WERE LOADED
IEB147I END OF JOB - 0 WAS HIGHEST SEVERITY CODE
```

Figure 6-21 Partial output from job that correctly reloads a member with alias from an unloaded PDSE

6.3.7 IEBCOPY reload selecting a member with an alias but omitting the alias

Figure 6-22 shows an example of a job to reload a specific member (that has an alias) without including the member's alias (or aliases) on the select member statement.

```
//MHLRES1D JOB (999,POK),MSGLEVEL=1,NOTIFY=MHLRES1
//*
//* IEBCOPY JOB TO RELOAD A PDS OR PDSE FROM A SEQUENTIAL data set
//*
//IEBCOPY EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=OLD,
//         DSN=PDSERES.SCEERUN2.UNLOADED
//SYSUT2   DD DISP=(,CATLG,DELETE),
//         SPACE=(TRK,(14000,1500,1),RLSE),
//         UNIT=SYSALLDA,
//         DSNTYPE=LIBRARY,
//         DSN=PDSERES.CEE.SCEERUN2.RELOADED.CRTEC128
//SYSIN DD *
 COPY INDD=SYSUT1,OUTDD=SYSUT2
 S M=CRTEC128
```

Figure 6-22 Example job to reload a member that has an alias which is not specified

Figure 6-23 on page 137 shows part of the output from the job including the warning message that indicates that one or more aliases exist but were not included on the select member statement. Note message IGW01569W indicating that aliases exist but were not specified, and IEB1130I from IEBCOPY warning that there was a potential problem. Note that the return code is set to 0 even though warning messages were issued.

```
 IEBCOPY MESSAGES AND CONTROL STATEMENTS                                PAGE     1
IEB1135I IEBCOPY  FMID HDZ11HO  SERVICE LEVEL NONE     DATED 20040319 DFSMS 01.06.00
z/OS    01.06.00 HBB7709  CPU 2084
IEB1035I MHLRES1D IEBCOPY 18:56:27 FRI 05 NOV 2004 PARM=''
 COPY INDD=SYSUT1,OUTDD=SYSUT2
 S M=CRTEC128
IEB1013I COPYING FROM PDSU INDD=SYSUT1 VOL=SBOXEC DSN=PDSERES.SCEERUN2.UNLOADED
IEB1014I           TO PDSE OUTDD=SYSUT2   VOL=SBOXEE
DSN=PDSERES.CEE.SCEERUN2.RELOADED.CRTEC128
IGW01569W MEMBER CRTEC128 WAS SPECIFIED FOR LOAD BUT ONLY 0 OF 1
 ALIASES WERE SPECIFIED
IGW01551I MEMBER CRTEC128 HAS BEEN LOADED
IGW01550I 1 OF 1 SPECIFIED  MEMBERS WERE LOADED
IEB1130I A WARNING MESSAGE FROM PDSE PROCESSING APPEARS ABOVE -- DIAGNOSTIC INFORMATION
IS X'28100239'
IEB147I END OF JOB - 0 WAS HIGHEST SEVERITY CODE
```

Figure 6-23 Partial output from a job that specified a member name but not its aliases

6.3.8 IEBCOPY reload selecting a member that does not have an alias

Figure 6-24 shows an example of a job that specifies a member to be reloaded that does not have an alias.

```
//MHLRES1D JOB (999,POK),MSGLEVEL=1,NOTIFY=MHLRES1
//*
//* IEBCOPY JOB TO RELOAD A PDS OR PDSE FROM A SEQUENTIAL data set
//*
//IEBCOPY EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=OLD,
//         DSN=PDSERES.SCEERUN2.UNLOADED
//SYSUT2   DD DISP=(,CATLG,DELETE),
//         SPACE=(TRK,(14000,1500,1),RLSE),
//         UNIT=SYSALLDA,
//         DSNTYPE=LIBRARY,
//         DSN=PDSERES.CEE.SCEERUN2.RELOADED.CDAEED
//SYSIN DD *
 COPY INDD=SYSUT1,OUTDD=SYSUT2
 S M=CDAEED
```

Figure 6-24 Example of a job to reload a member that does not have an alias

Figure 6-25 shows partial output from a correctly completing job selecting a member name that does not have an alias. Note messages IEB1013I indicating that the input was a PDSU format data set, and IEB1014I indicating that the output was a PDSE format data set.

```
 IEBCOPY MESSAGES AND CONTROL STATEMENTS                                PAGE     1
IEB1135I IEBCOPY  FMID HDZ11H0  SERVICE LEVEL NONE     DATED 20040319 DFSMS 01.06.00
z/OS    01.06.00 HBB7709  CPU 2084
IEB1035I MHLRES1D IEBCOPY 20:42:26 FRI 05 NOV 2004 PARM=''
 COPY INDD=SYSUT1,OUTDD=SYSUT2
 S M=CDAEED
IEB1013I COPYING FROM PDSU INDD=SYSUT1 VOL=SBOXEC DSN=PDSERES.SCEERUN2.UNLOADED
IEB1014I           TO  PDSE OUTDD=SYSUT2   VOL=SBOXEE
DSN=PDSERES.CEE.SCEERUN2.RELOADED.CDAEED
IGW01551I MEMBER CDAEED  HAS BEEN LOADED
IGW01550I 1 OF 1 SPECIFIED  MEMBERS WERE LOADED
IEB147I  END OF JOB - 0 WAS HIGHEST SEVERITY CODE
```

Figure 6-25 Partial output from job selecting a member that does not have an alias

6.3.9 IEBCOPY copying PDSE to PDSE

You can use IEBCOPY COPY or COPYGRP control statements to copy either complete data sets or selected members. For copying a limited number of members, no special considerations are required other than for members with aliases longer than eight characters.

IEBCOPY control statements allow member names of up to eight characters. Program objects stored in PDSEs can alias names longer than eight characters, so they cannot be selected with the standard IEBCOPY COPY-related statements.

You must use COPYGRP for PDSEs containing program objects with alias names longer than eight characters. The advantage of using the COPYGRP control statement is that the alias members need not be specified, as all aliases will be copied, but this only works when the input data set is a PDSE or an unloaded PDSE.

When copying a complete data set, if the target data set is in PDSE format, performance could be slower than copying to a PDS. You can improve performance by using an IEBCOPY unload-reload sequence with a sequential intermediate data set.

6.3.10 IEBCOPY selecting members without REPLACE

IEBCOPY REPLACE control statement options must be used to replace a member that already exists in the output data set.

The COPY control statement may be used for PDS, PDSE, and PDSU format data sets if member names are eight characters or less. Figure 6-26 on page 139 is an example of a job that will copy a primary member name and its alias from one data set to another if the members do not exist in the output data set. Neither member name is longer than eight characters.

```
//MHLRES1D JOB (999,POK),MSGLEVEL=1,NOTIFY=MHLRES1
//*
//* IEBCOPY JOB TO COPY A MEMBER OF A PDSE FROM ANOTHER data set
//*
//IEBCOPY EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//INDD1    DD DISP=OLD,
//         DSN=MHKRES5.SCEERUN2.UNLOADED
//OUTDD1   DD DISP=SHR,
//         DSN=PDSERES.CEE.SCEERUN2.RELOADED
//SYSIN DD *
 COPY I=INDD1,O=OUTDD1
 S M=CRTEC128
 S M=C128
```

Figure 6-26 IEBCOPY job to copy selected members

See Figure 6-27 for partial output from the attempt to copy two members without specifying the REPLACE format of the COPY statement. The important thing to note is that IEBCOPY returns a non-zero return code even though no members were copied.

```
 IEBCOPY MESSAGES AND CONTROL STATEMENTS                          PAGE     1
IEB1135I IEBCOPY  FMID HDZ11HO  SERVICE LEVEL NONE     DATED 20040319 DFSMS 01.06.00
z/OS    01.06.00 HBB7709  CPU 2084
IEB1035I MHLRES1D  IEBCOPY  20:28:33 MON 08 NOV 2004 PARM=''
 COPY I=INDD1,O=OUTDD1
 S M=CRTEC128
 S M=C128
IEB1013I COPYING FROM PDSE INDD=INDD1  VOL=SBOXEE DSN=MHLRES5.CEE.SCEERUN2.RELOADED
IEB1014I     TO PDSE OUTDD=OUTDD1  VOL=SBOXEE DSN=PDSERES.CEE.SCEERUN2.RELOADED
IGW01584W MEMBER CRTEC128 WILL NOT BE COPIED DUE TO NO-REPLACE
IGW01592W ALIAS C128  NOT COPIED BECAUSE THE PRIMARY CRTEC128
 ALREADY EXISTS IN THE OUTPUT data set CAUSING A NO-REPLACE CONFLICT
IGW01550I 0 OF 2 SPECIFIED  MEMBERS WERE COPIED
IEB1130I A WARNING MESSAGE FROM PDSE PROCESSING APPEARS ABOVE -- DIAGNOSTIC INFORMATION
IS X'28100248'
IEB147I END OF JOB - 0 WAS HIGHEST SEVERITY CODE
```

Figure 6-27 Partial output from IEBCOPY job without replace form of COPY

6.3.11 IECOPY selecting and replacing members that exist in the output

Figure 6-28 on page 140 shows a sample job that copies a primary member name and its alias from one data set to another when the members might or might not exist in the output data set. The difference is that the COPY statement is specified in the required form to allow the replace.

As an alternative to using this form of the COPY statement, a form of the select member statement is available that allows the replace. Refer to *z/OS DFSMSdfp Utilities*, SC26-7414 for details. Neither member name is longer than eight characters.

```
//MHLRES1D JOB (999,POK),MSGLEVEL=1,NOTIFY=MHLRES1
//*
//* IEBCOPY JOB TO COPY A MEMBER OF A PDSE FROM ANOTHER data set
//*
//IEBCOPY EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//INDD1    DD DISP=OLD,
//         DSN=MHLRES5.CEE.SCEERUN2.RELOADED
//OUTDD1   DD DISP=SHR,
//         DSN=PDSERES.CEE.SCEERUN2.RELOADED
//SYSIN DD *
 COPY I=((INDD1,R)),O=OUTDD1
 S M=CRTEC128
 S M=C128
```

Figure 6-28 Example job to copy members that might already exist in the output data set

Figure 6-29 shows the expected output from a job that is intended to replace a member with an alias.

```
 IEBCOPY MESSAGES AND CONTROL STATEMENTS                          PAGE     1
IEB1135I IEBCOPY  FMID HDZ11HO  SERVICE LEVEL NONE     DATED 20040319 DFSMS 01.06.00
z/OS    01.06.00 HBB7709  CPU 2084
IEB1035I MHLRES1D  IEBCOPY  20:39:24 MON 08 NOV 2004 PARM=''
 COPY I=((INDD1,R)),O=OUTDD1
 S M=CRTEC128
 S M=C128
IEB1013I COPYING FROM PDSE INDD=INDD1  VOL=SBOXEE DSN=MHLRES5.CEE.SCEERUN2.RELOADED
IEB1014I     TO PDSE OUTDD=OUTDD1  VOL=SBOXEE DSN=PDSERES.CEE.SCEERUN2.RELOADED
IGW01552I MEMBER CRTEC128 HAS BEEN COPIED  AND REPLACED
 A PRIMARY WITH ALIAS(ES)
IGW01554I ALIAS C128  OF COPIED PRIMARY CRTEC128  HAS BEEN COPIED
 AND REPLACED
IGW01550I 2 OF 2 SPECIFIED  MEMBERS WERE COPIED
IEB147I END OF JOB - 0 WAS HIGHEST SEVERITY CODE
```

Figure 6-29 Partial output of job to copy members with aliases that exist in the output data set

6.3.12 IEBCOPY COPYGRP

The COPYGRP control statement, for use with PDSEs, manages the inclusion of aliases automatically. The COPYGRP statement replaces the COPY statement in this case, and may be followed by SELECT MEMBER statements, which only need specify the primary member name. The use of the replace option format on SELECT MEMBER statements is not available, but it can be specified on the COPYGRP statement.

6.3.13 IEBCOPY COPYGRP selecting a member without its aliases

COPYGRP, unlike COPY, will find and automatically include alias members. See Figure 6-30 on page 141 for a job to select a member of a PDSE containing program objects, without including the alias name.

```
//MHLRES1D JOB (999,POK),MSGLEVEL=1,NOTIFY=MHLRES1
//*
//* IEBCOPY JOB TO COPY A MEMBER OF A PDSE FROM ANOTHER data set
//*         using COPYGRP and not specifying any alias names
//*
//IEBCOPY EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//INDD1    DD DISP=OLD,
//         DSN=MHLRES5.CEE.SCEERUN2.RELOADED
//OUTDD1   DD DISP=SHR,
//         DSN=PDSERES.CEE.SCEERUN2.RELOADED
//SYSIN DD *
 COPYGRP I=((INDD1,R)),O=OUTDD1
 S M=CRTEC128
```

Figure 6-30 Example of use of COPYGRP

Figure 6-31 shows output from the job using COPYGRP without specifying alias names. Note that the alias name has been detected and copied and replaces the primary and alias name in the output data set.

```
 COPYGRP I=((INDD1,R)),O=OUTDD1
 S M=CRTEC128
IEB1013I COPYING FROM PDSE  INDD=INDD1   VOL=SBOXEE DSN=MHLRES5.CEE.SCEERUN2.RELOADED
IEB1014I       TO PDSE OUTDD=OUTDD1  VOL=SBOXEE DSN=PDSERES.CEE.SCEERUN2.RELOADED
IGW01552I MEMBER CRTEC128 HAS BEEN COPIED   AND REPLACED
 A PRIMARY WITH ALIAS(ES)
IGW01554I ALIAS C128  OF COPIED PRIMARY CRTEC128   HAS BEEN COPIED
 AND REPLACED
IGW01550I 2 OF 2 SPECIFIED  MEMBERS WERE COPIED
IEB147I END OF JOB - 0 WAS HIGHEST SEVERITY CODE
```

Figure 6-31 Partial output from job using COPYGRP without selecting alias names

6.3.14 IEBCOPY to reorganize a data set

It is not necessary to reorganize a PDSE before making a backup copy, but restoration will be more efficient if it is well organized at the time the backup was taken. If the backup is being done using IEBCOPY to unload to a sequential data set, then the data set will be well organized when reloaded.

Reorganization is necessary only for highly active data sets where the automatic space reuse function has been used and parts of the data set have become spread over a volume, making access inefficient.

DFSMSdss is easier to use than IEBCOPY for reorganization, but if IEBCOPY is to be used, the most efficient method is to unload to a sequential data set in PDSU format, then reload from it to another PDSE, then complete the deletes and renames that are necessary to have the data set available with the original name.

6.4 TSO/E TRANSMIT

The TSO/E TRANSMIT command first uses IEBCOPY to unload a PDS or a PDSE before adding control information and sending the data set.

When the data set is received, regardless of whether a PDS or PDSE was transmitted, the result will depend on the receiving system's SMS ACS routines. If PDS or PDSE is required, it should be pre-allocated.

If a PDSE or PDS contains program objects or load modules you can copy only PDS to PDS or PDSE to PDSE.

6.5 ISMF data set application

The ISMF (Interactive Storage Management Facility) data set application presents a list of data sets based on criteria that you specify. These criteria may include full or partial data set names and matches on data set attributes such as size or organization. For details about using ISMF, refer to *DFSMS: Using the Interactive Storage Management Facility*, SC26-7411.

There are five Panels in the set to build up a data set selection list. Panel 1 affects the other Panels when they apply varying levels of qualification. In particular, in order to restrict the selection to PDSE format data sets, use the Data Set Name Type entry on Panel 4 of 5, and specify EQ LIBRARY.

Figure 6-32 shows an example of the initial settings on Panel 1. If values are specified on subsequent Panels, the top line is annotated with OTHER VALUES PRESENT; otherwise this area shows the Panel number that you are working with. The specification in this example says to obtain the list from the catalog, option 2, select all data sets that match `'PDSERES.**'`, obtain additional data from the catalog (2), and obtain additional data from the volume.

```
data set SELECTION ENTRY PANEL    OTHER VALUES PRESENT
Command ===>

For a data set List, Select Source of Generated List  . . 2  (1 or 2)

  1  Generate from a Saved List         Query Name To
       List Name  . .                   Save or Retrieve

  2  Generate a new list from criteria below
       data set Name . . . 'PDSERES.**'
       Enter "/" to select option   /  Generate Exclusive list
       Specify Source of the new list  . . 2    (1 - VTOC, 2 - Catalog)
       1 Generate list from VTOC
          Volume Serial Number . . .            (fully or partially specified)
       2 Generate list from Catalog
          Catalog Name . . .
          Volume Serial Number . . .            (fully or partially specified)
          Acquire Data from Volume . . . . . . . Y  (Y or N)
          Acquire Data if DFSMShsm Migrated  . . N  (Y or N)

Use ENTER to Perform Selection; Use DOWN Command to View next Selection Panel;
```

Figure 6-32 ISMF Panel 1 initial data set selection criteria

To further qualify the selection list, use Panels 2 through 5 using the DOWN command (PF 8). In particular, to limit the selection to PDSE data sets, go to Panel 4. Figure 6-33 shows an example of Panel 4 set for this purpose.

```
data set SELECTION ENTRY PANEL             Page 4 of 5
Command ===>

To limit the List, Specify a value or range of values in any of the following:
                            Rel Op   Value      Value      Value     Value
                            ------  --------   --------   --------  --------
    Allocation Unit  . . . .
    CF Cache Set Name  . . .
    CF Monitor Status  . . .
    CF Status Indicator  . .
    Change Indicator . . . .
    Compressed Format  . . .
    Data Class Name  . . . .
    data set Environment . .
    data set Name Type . . . EQ      LIBRARY
    data set Organization
    (1 to 8 Values)     ===>
    DDM Attributes . . . . .
    Device Type  . . . . . .
    (1 to 8 Values)     ===>
Use ENTER to Perform Selection; Use UP/DOWN Command for other Selection Panel;
```

Figure 6-33 ISMF Panel 4 showing selection of PDSEs - Data Set Name Type of LIBRARY

At this stage, if no other qualifications are wanted, press Enter. Figure 6-34 shows an example of the resulting list of just PDSE data sets.

When presented with a data set list, you can issue line commands against individual data sets or list commands against the entire list.

```
 Panel  List  data set  Utilities  Scroll  Help
------------------------------------------------------------------------------
                              data set LIST               REDISPLAYED LIST
Command ===>                                              Scroll ===> HALF
                                                       Entries 1-7 of 7
Enter Line Operators below:                            Data Columns 3-6 of 39

    LINE                                  ALLOC    ALLOC   % NOT  COMPRESSED
    OPERATOR         data set NAME        SPACE    USED    USED   FORMAT
   ---(1)----  ------------(2)------------ --(3)--  --(4)--  -(5)-  ---(6)----
               PDSERES.CEE.SCEERUN2.       774703   703043     9  ---
                 RELOADED
               PDSERES.CEE.SCEERUN2.         1494     1494     0  ---
                 RELOADED.CDAEED
               PDSERES.CEE.SCEERUN2.         8522     8466     1  ---
                 RELOADED.CRTEC128
               PDSERES.CEE.SCEERUN2.         8522     8466     1  ---
                 RELOADED.C128
               PDSERES.PDSECOP.CNTL          3099     2767    10  ---
               PDSERES.SMS.DIRK.PDSE1         277      111    59  ---
               PDSERES.SMS.PDSE.MATTEO        387       55    85  ---
   ----------  ------  -----------  BOTTOM OF  DATA  -----------  ------  ----
```

Figure 6-34 ISMF data set list selecting PDSEs with HLQ PDSERES

We do not cover all of these commands, but we note cases in Table 6-1 where there are specific PDSE considerations, largely caused by the programs that are called to process the ISMF commands.

Table 6-1 ISMF line and list commands

Line command	Processed by	Notes
`compress`	DFSMSdss	Line command and list command. A DFSMSdss batch COMPRESS job is generated. Normal DFSMSdss rules apply; PDSEs will not be processed.
`condense`	DFSMShsm	Line command only. The number of extents is reduced.
`release`	DFSMSdss	Line command and list command. A DFSMSdss batch RELEASE job is generated. Normal DFSMSdss rules apply; space will be released in PDSEs to the HFRFN.

6.6 Backup considerations for large PDSE data sets

The DFSMSdss and DFSMShsm functions described above are for logical operations where the data set is considered as a number of members.

For very large PDSEs and system non-SMS managed data sets, physical DFSMSdss backups may be required of the volumes that they are on. If such physical dumps are done they are point-in-time copies, and depending on other activity may not fully reflect the content. Therefore, these should be considered as last-resort recovery options if the logical level backups cannot be used to restore the PDSE. This is not a situation specific to PDSEs, but may arise because of the potential for very large data sets to be allocated, or for data sets initially allocated at one size to grow significantly.

There is no facility to manage data in a PDSE at other than the data set level. As mentioned above, this means that any update activity anywhere in the data set will cause the change indicator to be turned on, and then DFSMShsm will try to back up the entire data set.

This contrasts with the situation for HFS or zFS data sets where data backup can be managed in a much more granular manner by using a product such as Tivoli Storage Manager.

6.7 IEBCOPY performance note

As stated in 6.3, “IEBCOPY” on page 131, IEBCOPY can copy to and from PDSs and PDSEs. When the number of members becomes large, particularly when a PDSE has had many updates, using IEBCOPY to make a PDSE backup by copying it to another PDSE may take an excessive elapsed time, use more CPU time than expected, or both.

However, using IEBCOPY to create a sequential copy of the data set is a viable method of making a backup. The process of restoring from the sequential IEBCOPY format to either PDS or PDSE creates the data set more efficiently. Even if the sequential data set is on magnetic tape, the overall unload-reload may take less elapsed time than a DASD-to-DASD copy.

Further information about PDSE performance can be found in Chapter 9, “Performance considerations” on page 179.

7

Program management and PDSEs

In this chapter we give an overview of program management using the z/OS program management binder (PM binder) and program management loader (PM Loader). Next, we explain how they lift many of the limitations of the linkage editor and batch loader and provide the infrastructure for new capabilities. Finally, we explain the role of PDSEs in program management.

7.1 Program management overview

Program management is concerned with the creation and management of executable programs. This consists of creating, loading, modifying, listing, reading, transporting, and copying programs.

Before the PM binder and PM loader were available, the linkage editor, loader, and batch loader were used for program management.

7.1.1 Linkage editor

Programs are written in various languages. The programs are translated to object modules by the assembler or by compilers such as C++, COBOL, or PL/I. Object modules are not executable, mainly because they refer to variables in other object modules. These variables are called external references. Before execution, the object modules must be processed by the linkage editor to resolve these references.

The linkage editor also combines multiple object modules and creates load modules ready for execution. These load modules are stored in partitioned data sets (PDS).

Although the binder is now used by default, the linkage editor still exists with a different alias.

7.1.2 Batch loader

The batch loader combines the basic editing and loading functions of the linkage editor and program fetch into one job step. The loader loads object modules produced by a language processor, and load modules produced by the linkage editor, into the processor virtual storage for execution. Optionally, it searches a call library (SYSLIB), a resident link pack area, or both to resolve external references.

Unlike the linkage editor, the batch loader does not produce load modules for program libraries.

Figure 7-1 on page 147 shows an overview of program management using the linkage editor and the batch loader.

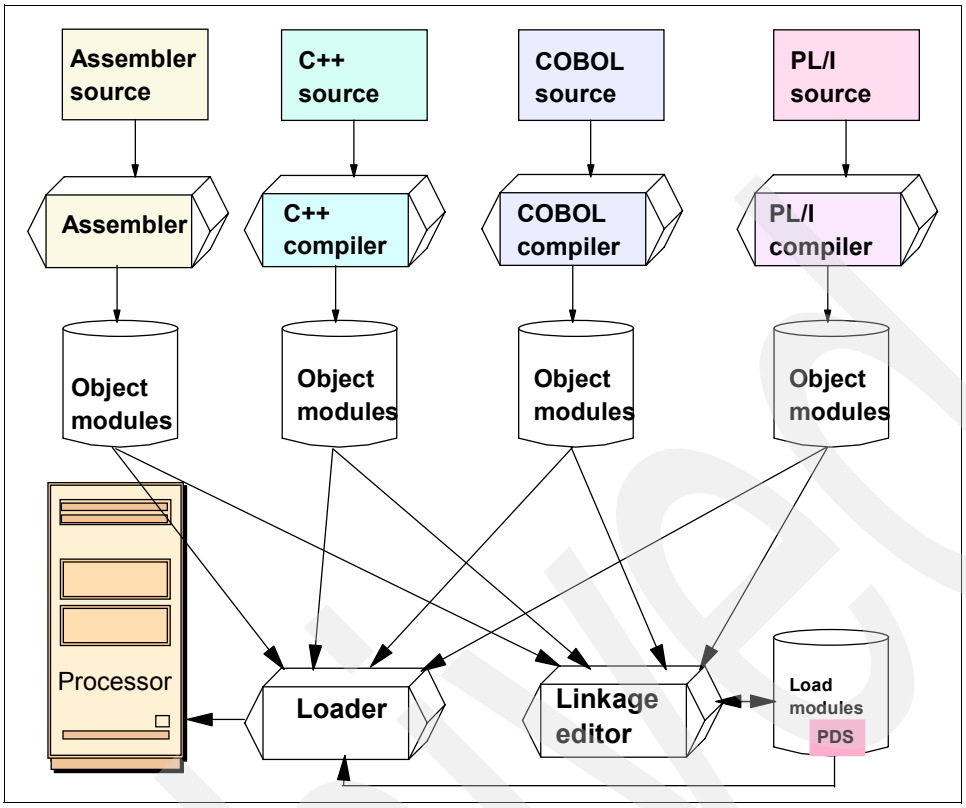


Figure 7-1 Program management overview

Using the linkage editor and batch loader for program management has a number of limitations, some due to the linkage editor and some due to the use of PDSs.

7.1.3 Linkage editor restrictions

The linkage editor has the following restrictions:

- The size of program load modules is limited to 16 MB.
- There is no facility to split the load module into smaller segments when loading into virtual storage.
- The number of aliases is limited to 64.
- External names (entry points in one section that can be referenced from another section) are limited to 32767, which can become a constraint for large programs.
- External names, program entry points, aliases, and section names are limited to eight characters.

7.1.4 Limitations imposed by using partitioned data sets

There are also some limitations imposed because load modules are stored in PDSs:

- All names are limited to eight characters.
- The aliases are loosely linked to the load modules. Copying the load modules without the aliases has been the cause of many system outages in the past. Also, if you delete a primary member name, DFSMSdfp does not delete its aliases and your PDS can contain orphaned or *dangling* aliases. These dangling aliases point to the old location of the

member, which may be the start of a new, different member or, worse, point into the middle of a new member.

- The directory structure is prone to corruption. It is easy to overwrite the directory structure by allocating the PDS as a sequential data set for write. This often happens when the member name is missed out of the JCL.
- If proper care is not taken while copying load modules, it is possible to truncate load module records.
- Load libraries cannot be used by UNIX System Services.

7.2 Program management using the z/OS binder and loader

z/OS program management lifts these limitations of the linkage editor and batch loader, and enhances program management by:

- Providing the program management binder to replace the linkage editor and batch loader
- Providing the program management loader to replace the program fetch
- Introducing program objects and the Generalized Object File Format (GOFF)
- Providing a Transport Utility to convert program objects to a form understood by other programs

Figure 7-2 shows an overview of program management using the program management binder.

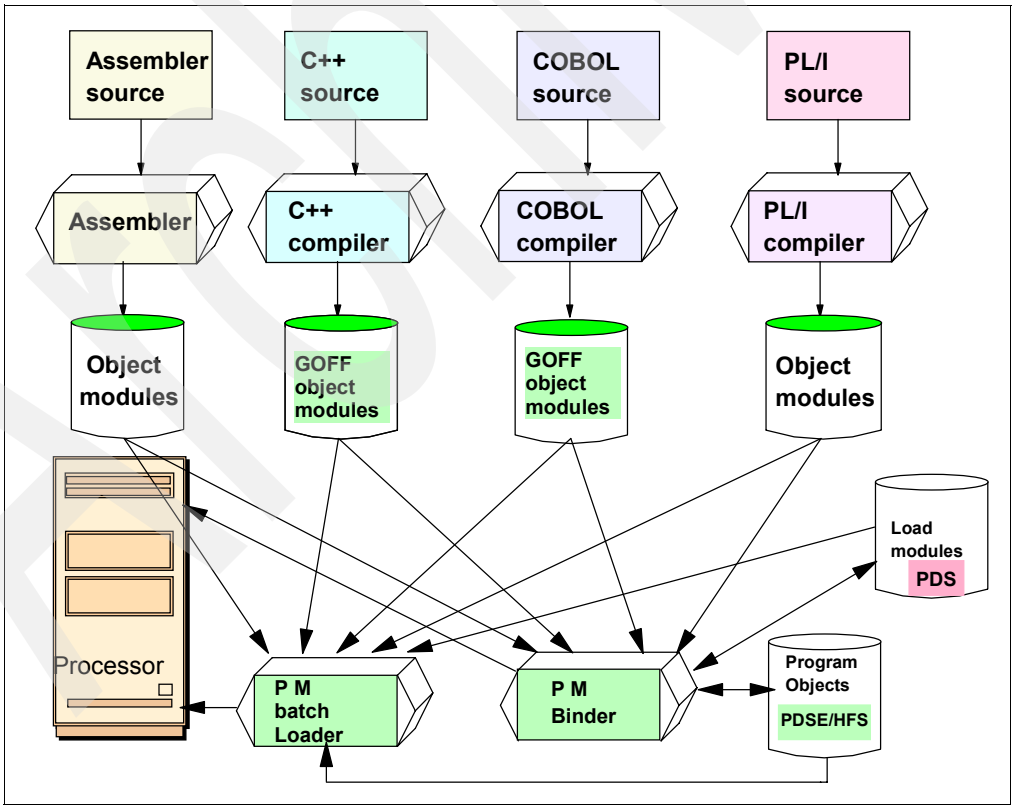


Figure 7-2 Program management binder

7.2.1 Differences between the PM binder and the linkage editor

The binder can do everything the linkage editor and batch loader can do. It eliminates the restrictions imposed by the linkage editor as described in 7.1.3, “Linkage editor restrictions” on page 147. In addition, it has the following new features:

- The output of the binder can be either load modules or program objects. Program objects are similar to load modules but have a number of new features. Program objects are described in 7.2.3, “What are program objects?” on page 149. The binder stores the program objects in PDSE or z/OS UNIX file system. Like the linkage editor, it stores the load modules in PDS data sets.
- The binder supports a modified extended object module, produced by the C and C++ compilers, and an entirely new object module format, called GOFF. This new object module type is described in 7.2.5, “GOFF” on page 150.
- The binder can load the programs into processor storage for execution. The linkage editor cannot load programs directly into processor storage for execution; the batch loader can.
- The binder performs the functions of the C-pre-linker, eliminating the need to run it.

7.2.2 Differences between the PM loader and program fetch

The program management loader supports the loading of program objects. Program fetch loads standard load modules.

All program objects are page-mapped into virtual storage. When loading program objects from a PDSE or z/OS UNIX file system, the loader selects a loading mode based on the module characteristics and parameters specified to the binder when you created the program object.

The program management loader provides an option to read and relocate program objects into virtual storage only when they are referenced during execution. The program fetch loads the whole program into virtual storage before execution.

7.2.3 What are program objects?

In simple terms, program objects are enhanced load modules with extended capability. Program objects cannot be stored in PDS libraries. Either a PDSE library or a z/OS UNIX file system is required to store program objects.

A program object can have a text size of up to 1 GB, whereas the text size of a load module is limited to 16 MB.

The block size of a program object is fixed, eliminating the need to reblock when you copy programs between devices.

Segments of executable code may be loaded into discontiguous storage ranges.

You can have different data classes inside the program object representing control information, executable code, or data. (This is a completely different use of data class than the SMS usage.) Program objects support an unlimited number of data classes. The program object module can be divided into class segments, each of which can be loaded into separate storage locations.

Virtually any type of data that is associated with a module or its constituent sections can be saved in a program object.

The linkage editor limited the number of aliases to 64 and the number of external names to 32767. With the binder and the program objects, the number of aliases and external names for programs stored in a PDSE or z/OS UNIX file system file is limited only by the space available to store them. Programs stored in a partitioned data set rather than a PDSE program object are still limited to 32767 external names.

For program objects, external names (entry points in one section that can be referenced from another section or module) can be up to 1024 bytes in length. Long names can be used for section names, external labels and references, pseudo-registers and common areas, and aliases and alternate entry points for the module. Primary or member names are still limited to 8 bytes, however, as are member names appearing in JCL or system macros.

7.2.4 Dynamic link libraries

Dynamic link libraries, or DLLs, were introduced in OS/390 2.4. Using DLLs enables improvements in the packaging and sharing of modules and performance improvements within UNIX System Services.

Binding is deferred until execution, in contrast to static binding done by the binder or explicit binding at execution time as a result of calls to LINK, LOAD, or XCTL system services. DLLs may be stored in either PDSs or PDSEs.

7.2.5 GOFF

GOFF stands for *Generalized Object File Format*. It is an enhanced type of object module introduced by the binder in DFSMS/MVS 1.3 and produced by the High Level Assembler, COBOL, and C++.

GOFF supports long names and multipart modules, and provides some other new features. These object modules can be stored as sequential files, as members of PDS or PDSE libraries, or as z/OS UNIX file system files.

7.2.6 XOBJ format

The XOBJ format is used by C and C++ compilers. It required a pre-linking step to convert the object file to the normal format prior to the Program Management 3 changes introduced in DFSMS/MVS 1.4.

7.2.7 How to convert load modules to program objects

You can use IEBCOPY to convert between program objects and load modules. For details, refer to the section “Using Utilities for program management” in *z/OS: MVS Program Management: User's Guide and Reference*, SA22-7643.

Conversion from program objects to load modules is not possible if the program object exploits facilities that are not available with load modules, such as more than 16 MB of executable code.

7.2.8 Program management levels

There have been several distinct sets of changes to program management and it is helpful to identify each level, especially to make clear what migration and compatibility considerations there might be.

There are corresponding changes in program object format. The changes are upward compatible.

Program management 1

Program management 1 (PM1) is the level introduced by DFSMS/MVS 1.1.

Program management 2

Program management 2 (PM2) is the level introduced by DFSMS/MVS 1.3. The changes were largely in support of new compilers such as those for C++, FORTRAN 90, and the High Level Assembler. A new form of object module, known as GOFF, was introduced.

If a program contained large amounts of uninitialized data, the linkage editor would not store that data in the load module. However, the program object format did, meaning that program objects could be much larger than load modules. Additionally, the linkage editor would initialize the data with binary zeroes, although this was undocumented. The program object format was updated to omit uninitialized data.

The format of program objects was extended to include associated data (ADATA), a place for information to be held that is not executable but may be of use during execution. The primary users of ADATA are compilers, who use it for storing data such as symbol tables and source statements to allow interactive debugging facilities.

The binder was enhanced to allow long external names, up to 1024 characters.

Program management 3

Program management 3 (PM3) is the level introduced by DFSMS/MVS 1.4.

The changes introduced in PM3 were mainly in support of C and C++ programs and the UNIX System Services environment.

Binder enhancements remove the need for a pre-linking step for C and C++ compilers.

The binder and loader were enhanced to support OS/390 2.4 dynamic linking and DLLs.

Program management 4

Program management 4 (PM4) is delivered in z/OS V1.3.

PM4 is the minimum level that can be specified if any of the following features are used:

- Input modules contain 8-byte adcons.
- Any ESD record is AMODE 64.
- Input contains symbol names longer than 1024, unless EDIT=NO.
- A value of 64 is specified on the AMODE option or control statement.

A *variant of PM4* format is introduced in z/OS Version 1.5 to support deferred load data segments if RMODE64 is specified. It cannot be rebound or inspected (by Fast Data, the binder API, or AMBLIST) on earlier releases, but it can be loaded and executed on other systems supporting PO4 format.

7.2.9 Migration and compatibility considerations

The different program object formats are upward compatible.

Downward compatibility is not guaranteed. However, this binder option allows the generation of a program object in an earlier format:

```
COMPAT={MIN|LKED|CURRENT|CURR
       |PM1
       |PM2
       |PM3|OSV2R8|OSV2R9|OSV2R10|ZOSV1R1|ZOSV1R2
       |PM4|ZOSV1R3|ZOSV1R4
       |ZOSV1R5|ZOSV1R6}
```

7.3 Value of PDSEs for executable code

There are now many cases where you must use PDSEs because of the need to exploit new function in program objects.

However, even if you have a choice, the use of PDSEs rather than PDSs for storing executable code offers these advantages:

- PDSE sysplex sharing allows updates from any system in a sysplex.
- The PDSE directory can grow to accommodate new members.
- There is no need to compress program libraries regularly to avoid space abends. PDSEs reuse space used by deleted members without compression.
- The number of aliases and external names for programs stored in a PDSE file is limited only by the space available to store them. Programs stored in a partitioned data set are limited to 32767 external names.
- Long names (up to 1024 bytes in length) may be used for section names, external labels and references, pseudo-registers, and common areas, and aliases and alternate entry points for the module.

8

PDSE sharing and serialization

Unlike PDSs, PDSEs can be shared at both data-set and member levels. Users on one or more systems can access a PDSE concurrently to create, read, or replace members.

Your installation can define the extent of PDSE sharing. PDSEs open for output can be shared among users on a single system (normal sharing) or among users on multiple systems in a sysplex (extended sharing). You select the type of sharing by using the PDSESHARING keyword in the IGDSMSxx member in SYS1.PARMLIB.

Extended sharing is available only when PDSESHARING(EXTENDED) has been specified on all systems. Extended sharing applies when a PDSE is allocated with DISP=SHR; it cannot be used with DISP=OLD, DISP=MOD, or DISP=NEW.

With extended sharing, any number of users or systems can concurrently share a PDSE or members within it for input (read) or output (write). While multiple members can be updated (in place) by different users on the same system, only one user can update a member (in place) at a given time. In addition, only one system can access the PDSE when update in place is being done on any member.

In this chapter we explain:

- Serialization techniques used by PDSE
- Different modes of sharing a PDSE
- How to set up PDSE for sharing
- Results of tests we conducted to compare PDS and PDSE sharing
- PDSE sharing considerations

Our tests were run using a sysplex environment. We were able to test PDSE sharing in normal mode and extended mode. The programs used were specially written in Assembler for these tests.

Before we explain the different sharing options that are provided for PDSs and PDSEs, we clarify some important terms.

8.1 Resource serialization

In a multitasking, multiprocessing environment, resource serialization refers to the techniques used to coordinate access to resources that are used by more than one program. When multiple users share data, a way to control access to that data is needed. For example, users who update data need exclusive access to that data; if several users tried to update the same data at the same time, the result would be a data integrity exposure—the possibility of incorrect or damaged data.

The techniques discussed in this section are internal to PDSE processing and do not require changes to application programs. This information is useful only for diagnostic efforts.

8.1.1 RESERVE macro

The RESERVE macro serializes access to a resource (a data set on a shared DASD volume) by obtaining control of the *volume* on which the resource resides to prevent jobs on other systems from using any data set on the entire volume. When a task on one system has issued a RESERVE macro to obtain control of a data set, no programs on other systems can access any other data set on that volume.

Combining the systems that access shared resources into a global resource serialization complex can solve the serialization granularity problems caused by using the RESERVE macro.

8.1.2 ENQ and DEQ macros

The ENQ and DEQ macros enable programs to serialize use of resources, such as data sets, at the *resource level* in a single system and in a multisystem environment.

The ENQ macro assigns control of a resource to the current task. The DEQ macro releases the resource from the task.

The ENQ and DEQ macros identify the resource and the sharing request by:

- Two names, known as the qname (major name) and the rname (minor name)
- A scope value
- The type of control

If a task has exclusive control of a resource, it is the only one that can use that resource; all other programs that issue ENQs for the resource (either for exclusive or shared control) must wait until the resource is released.

If a task has shared control of a resource, other programs can have concurrent use of the resource. If another program requests exclusive control over the resource during the time the resource is held, that program must wait until all current users have issued DEQ to release the resource.

8.1.3 Global resource serialization (GRS)

In a global resource serialization complex, programs can serialize access to data sets on shared disk volumes at the *data set level* rather than at the disk volume level. A program on one system can access one data set on a shared volume while other programs on any system can access other data sets on the same volume.

Global resource serialization overcomes the major drawbacks of using the RESERVE macro. Because it enables jobs to serialize their use of resources at the data set level, it can reduce

contention for these resources and minimize the chance of an interlock occurring between systems.

An ENQ with a scope of SYSTEMS might be used to synchronize processing in multisystem applications. The ENQ, DEQ, and RESERVE macros identify a resource by its symbolic name. The symbolic name has three parts: major name (qname), minor name (rname), and scope (which can be STEP, SYSTEM, or SYSTEMS).

See *z/OS: MVS Planning: Global Resource Serialization*, SA22-7600, for further details.

Resource name lists (RNL)

When an application uses the ENQ, DEQ, and RESERVE macros to serialize resources, global resource serialization uses resource name lists (RNLs) and the scope on the ENQ, DEQ, or RESERVE macro to determine whether a resource is local or global.

A *local resource* is a resource requested with a scope of STEP or SYSTEM. It is serialized only within the job or system that is processing the request for the resource. If a system is not part of a global resource serialization complex, all resources (with the exception of resources serialized with the RESERVE macro), regardless of scope (STEP, SYSTEM, SYSTEMS), are local resources.

A *global resource* is a resource requested with a scope of SYSTEMS. It is serialized among all systems in the complex.

Note: An ENQ or DEQ request can be excluded from RNL processing by specifying RNL=NO on the request. GRS is the only serialization product that can be used to serialize PDSE because PDSE services specifies RNL=NO for their ENQs. Therefore, RNLs do not apply to PDSEs.

When RNL=NO is specified, the ENQ request will be ignored by alternative serialization products, and this affects protection of the resource to systems outside of the current global resource serialization environment using alternative serialization products.

8.1.4 DISP keyword on the DD statement

The disposition (DISP) keyword on the DD statement is used to serialize an entire data set. The system issues the ENQ macro for each data set at the data set's allocation time. PDSEs work like PDSs.

If you allocate a PDSE that is already allocated by another job specifying DISP=SHR, the system issues the ENQ macro with the sharing use option, so that you can share the data set with other jobs.

If you allocate a PDS or PDSE that is allocated to any other jobs specifying anything other than DISP=SHR, you have to wait until all of the other jobs deallocate it. This is because the system issues the ENQ macro with the exclusive use option for the data set.

To verify the enqueue status requested by the system, you can use the system command in Figure 8-1 even if the system is not running in the global resource serialization (GRS) environment.

```
D GRS,RES=(SYSDSN,dsname)
```

Figure 8-1 Display GRS command (SYSDSN)

The system uses the name SYSDSN as a qname (major name), and the data set name as an rname (minor name) at the data set's allocation time.

See *z/OS: MVS System Commands,* SA22-7627, for more information.

8.1.5 OPEN types

The OPEN macro connects a program or task to a data set.You can open a PDS or PDSE for input, output, or update processing.

Open for input

When you open a PDSE for input, it is to read one or more members. Multiple users within the system, or across multiple systems in the sysplex, can normally share members for input.

Open for output

You open a PDSE for output when you want to create new members, rename members, delete members, or replace existing members. PDSE creates new members in new space without disturbing existing members, so multiple users could be creating new members at the same time.

When you start writing a member, the system does not know the member name. When you issue the STOW macro, the member logically replaces a member already created with the same name.

Open for update

In this case, you update a member in place. Normally this is done by an assembly language program. When a member is being updated in place, the system locks it out for input or output by other users to ensure data integrity.

An update is done by positioning to the correct member using either FIND or POINT; issuing READ, WRITE, and CHECK macros; and then issuing STOW to perform the update. STOW is the system service that updates a PDS or PDSE directory.

Note that CLOSE does an automatic STOW if the DD statement has a member name and at least one WRITE has occurred.

8.2 Modes of sharing a PDSE

A PDSE can be shared at the data set level or at the member level. Sharing a PDSE at the data set level is equivalent to sharing the PDSE directory. A PDSE can be shared by users on a single system or across multiple systems. In multiple system sharing, there are two modes of sharing: normal sharing and extended sharing.

8.2.1 Sharing a PDSE within a single system

Users within a single MVS system can share a PDSE at either the data set level or the member level.

8.2.2 Normal sharing across multiple systems

Users across multiple MVS systems can share a PDSE. In the *normal* mode of sharing, users can share a PDSE only at the data set level. The system uses GRS to provide the necessary serialization across systems to ensure data integrity.

8.2.3 Extended sharing across multiple systems

In *extended* sharing mode, users across multiple systems can share a PDSE at either the data set level or the member level. The system uses XCF signalling and GRS to ensure data set integrity in this mode. The systems must be set up in a sysplex for this mode of sharing.

8.3 Sharing within a single system

We now look at different sharing scenarios in general before we move on to see how these are handled by PDS, PDSE normal sharing, and PDSE extended sharing.

You can share PDSEs or PDSE members within a single system. With PDSE sharing, an entire data set is shared by multiple users.

PDSE uses an improved sharing mechanism compared with PDS to ensure data integrity. The system uses the ENQ/DEQ macro instruction to serialize its resources. Your programs do not have to issue any ENQ/DEQ macros. ISPF/PDF has its own member-level sharing mechanism for PDSs and PDSEs.

8.3.1 OPEN macro

With PDS, if you specify DISP=SHR on the DD statement, you may need to protect the data set yourself for your exclusive use. Some operations may destroy directories or create data integrity problems. You must issue an ENQ macro before the OPEN macro and a DEQ macro after the CLOSE macro in your program to serialize the PDS so that it is protected from corruption.

With PDSs, OPEN attempts to detect improper serialization. If you issue OPEN for output with DISP=SHR while another program has the PDS open for output (not update), OPEN issues ABEND 213-30.

However, in the case of an open for a PDSE, the system does the proper serialization for you, based on the options chosen on the OPEN macro.

8.3.2 Single system data-set-level sharing

The objective of our first test was to show what happens when two jobs share a PDSE and how this differs from two jobs sharing a PDS. Each job opened its PDS or PDSE member for input, output, or update.

Figure 8-2 on page 158 illustrates our test scenario. Each job opens a *different* member in the PDSE as a physical sequential data set. Hence, we are sharing the PDSE directory, but we are not sharing any members.

1. The first job opens and processes MEMBER01.
2. The second job opens, processes, and closes MEMBER02.
3. The first job closes MEMBER01.

Note: The following apply to all the test cases:

- DISP=SHR was specified.
- The second job is the first one to complete.

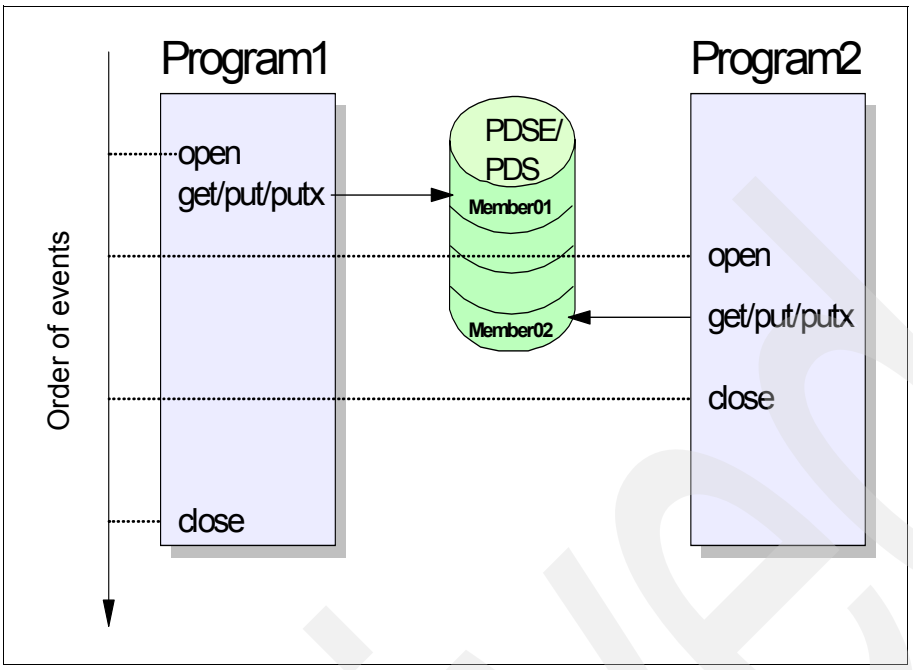


Figure 8-2 Data set level sharing test scenario

Table 8-1 summarizes the results for a PDS, and Table 8-2 for a PDSE. Note that there is no data set level sharing control for PDSs, so when both jobs open a PDS for output, one job gets an abend.

With a PDSE, each job can write its data to different members at the same time. Thus you can create multiple members simultaneously.

Table 8-1 PDS single system data set level sharing

Second job attempts open for	**First job has PDS currently open for**		
	Input	**Output**	**Update**
Input	Successful	Successful	Successful
Output	Successful	**ABEND 213-30**	Successful
Update	Successful	Successful	Successful

Table 8-2 PDSE single system data set level sharing

Second job attempts open for	**First job has PDSE currently open for**		
	Input	**Output**	**Update**
Input	Successful	Successful	Successful
Output	Successful	Successful	Successful
Update	Successful	Successful	Successful

8.3.3 Single system member-level sharing

Member-level sharing implies that one member of the PDSE is shared by multiple users. This section describes the results of a test to show how PDSE member-level sharing differs from PDS member-level sharing. It also details how ISPF/PDF controls sharing of PDSE members.

The objective of this test was to show what happens when two jobs share the same PDSE member, and how this differs from two jobs sharing a PDS member. Each job opened the same PDS or PDSE member for input, output or update. Figure 8-3 illustrates the test scenario for single system member-level sharing.

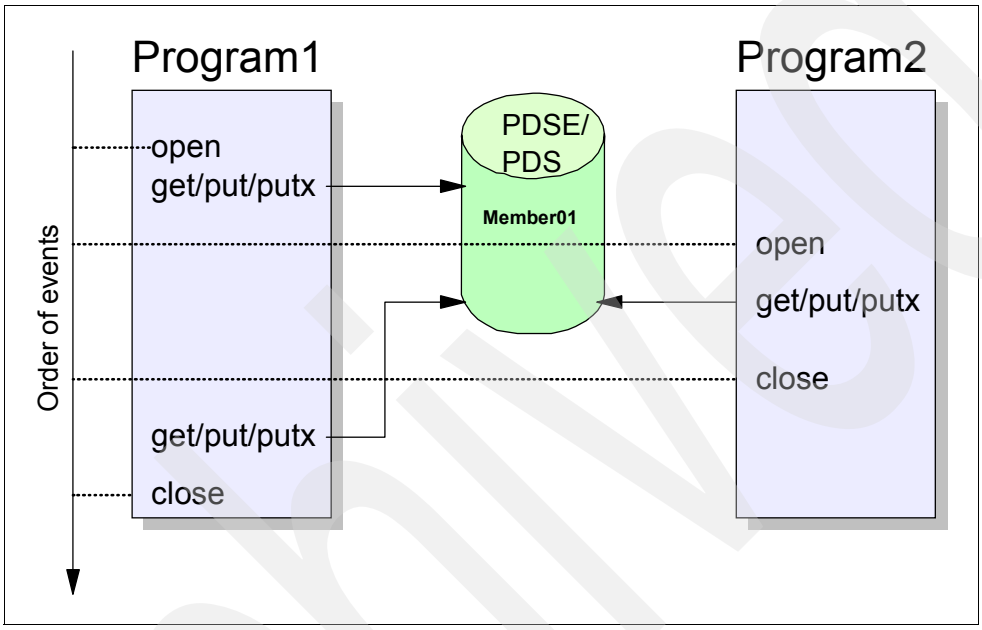


Figure 8-3 Single system member level sharing test scenario

1. The two jobs open MEMBER01 as a physical sequential data set at the same time.
2. The first job processes MEMBER01.
3. The second job processes MEMBER01 and then closes it.
4. The first job processes MEMBER01 again and closes it.

Note: The second job is the first one to complete.

Table 8-3 on page 160 summarizes the results for a PDS, and Table 8-4 on page 160 summarizes the results for a PDSE. There is no member-level sharing control for PDSs. If the first or second job opens a PDS for update and the other job opens it for input or update, data integrity problems may occur. This is because when both jobs open a PDS member for update, the two updates are written to the same area on disk. If the first job attempts to read its data again after the second job has updated it, the input operation of the first job will get unexpected data that has been updated by the second job. In this case, both jobs end normally with no indication that the first job has read unexpected data.

If a PDSE member is opened for update, any other job attempting to open it for input or update will abend with an abend code 213-74 and console message IEC143I 213-74.

Also, if a PDSE member is opened for input, any other job attempting to open it for update will abend with an abend code 213-74. If you do not want to get an abend, you can specify DISP=OLD when allocating PDSEs for update.

The explanation for the message IEC143I with ABEND 213 is:

- ABEND 213: This abend code means that the error occurred during processing of an OPEN macro instruction for a data set on a direct access device.
- Reason code 74: This reason code means that an OPEN detected a member-sharing option conflict for the PDSE.

With a PDSE, each job writes data to a different location. Each job finishes each member by issuing a STOW macro. If the programs simultaneously write members and give them the same name, the last STOW macro wins. That STOW macro replaces an existing member with the same name, no matter which member was opened first.

See *z/OS: MVS System Messages Volume 7 (IEB - IEE)*, SA22-7637 for more information.

Table 8-3 PDS single system member-level sharing

Second job attempts open for	**First job has PDS member currently open for**		
	Input	**Output**	**Update**
Input	Successful.	Successful. The member was replaced by the first job. The second job read the previous image of the data.	Successful. The second job may get inconsistent data.
Output	Successful. The member was replaced by the second job. The first job read the previous image of the data.	OPEN ABEND 213-30	Successful. The member was replaced by the second job.
Update	Successful. The second job may get inconsistent data.	Successful. The member was replaced by the first job.	*Unpredictable.* Data integrity exposure

Table 8-4 PDSE single system member-level sharing

Second job attempts open for	**First job has PDSE member currently open for**		
	Input	**Output**	**Update**
Input	Successful.	Successful. The member was replaced by the first job. The second job read the previous image of the data.	OPEN ABEND IEC141I, RC=18
Output	Successful. The member was replaced by the second job. The first job read the previous image of the data.	*Successful.* The member was replaced by the first job. The member from the second job is lost.	Successful. The member was replaced by the second job. The updates from the first job are lost.
Update	OPEN ABEND IEC141I, RC=18	Successful. The member was replaced by the first job.	OPEN ABEND IEC141I, RC=18

8.3.4 ISPF/PDF considerations

When you try to select a member on the ISPF/PDF EDIT member selection screen, you may get the `Member in use` message as shown in the top-right corner of Example 8-4.

ISPF/PDF has its own sharing mechanism, which serializes the use of a PDS or PDSE member by multiple ISPF/PDF users.

```
EDIT              PDSERES.SMS.DIRK.PDSE1                          Member in use
Command ===>                                                    Scroll ===> CSR
           Name     Prompt       Size   Created          Changed          ID
E          MEM1                    100  2004/11/04  2004/11/04 19:03:14  MHLRES6
_________  MEM2     *Edited         100  2004/11/04  2004/11/04 19:03:05  MHLRES2
_________  MEM3                     100  2004/11/04  2004/11/04 18:41:12  MHLRES2
           **End**
```

Figure 8-4 ISPF/PDF member selection panel: Member in use

The following sequence of ENQ/DEQ activities occurs when you process a PDS using the ISPF/PDF edit function (Figure 8-5).

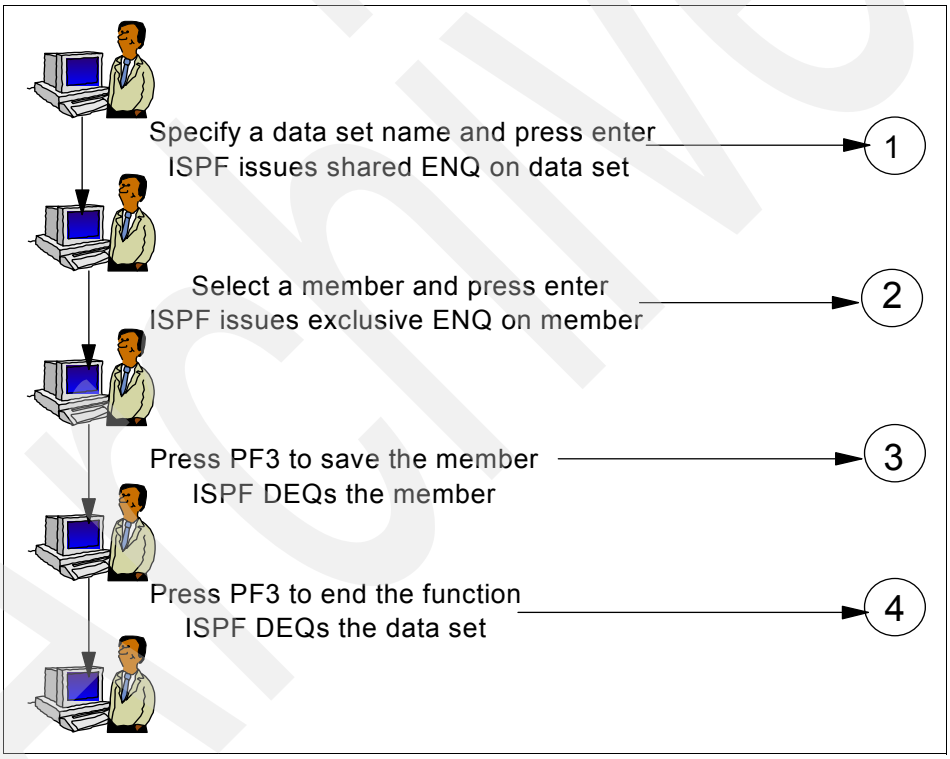


Figure 8-5 Member-level sharing by ISPF/PDF

1. When you pass the edit entry panel without specifying a member name, ISPF/PDF dynamically allocates and opens the data set to get all the directory entries. After that, the data set is closed but not deallocated. The system implicitly issues the ENQ macro for the data set with the sharing option when the data set is allocated.
2. When you select a member on the member selection panel, ISPF/PDF issues the ENQ macro for the member with the exclusive option, opens the member to get its data, and closes it.

3. When you press PF3 to end and save the member, ISPF/PDF reopens the member to write its data, closes it, and issues the DEQ macro for the member.
4. When you press PF3 again to return to the edit entry panel, ISPF/PDF dynamically deallocates the data set. The system implicitly issues the DEQ macro for the data set.

PDSEs are processed somewhat differently. The PDSE member is opened for output in step 2 on page 161 and closed in step 3.

You can use the system command in Figure 8-6 to verify the ENQ status requested by ISPF/PDF users even if the system is not running in a GRS environment.

```
D GRS,RES=(SPFEDIT,*)
```

Figure 8-6 Display GRS command (SPFEDIT)

ISPF/PDF uses SPFEDIT as a qname (major name), and the data set name and the member name as an rname (minor name) at data set allocation time.

8.4 PDSE normal sharing across multiple systems

In normal mode you can *share only at the data set level* and not at the member level. This means that when a user on one system has a member of the PDSE open for update/output, a user from another system cannot open any other member of the same PDSE (Figure 8-7).

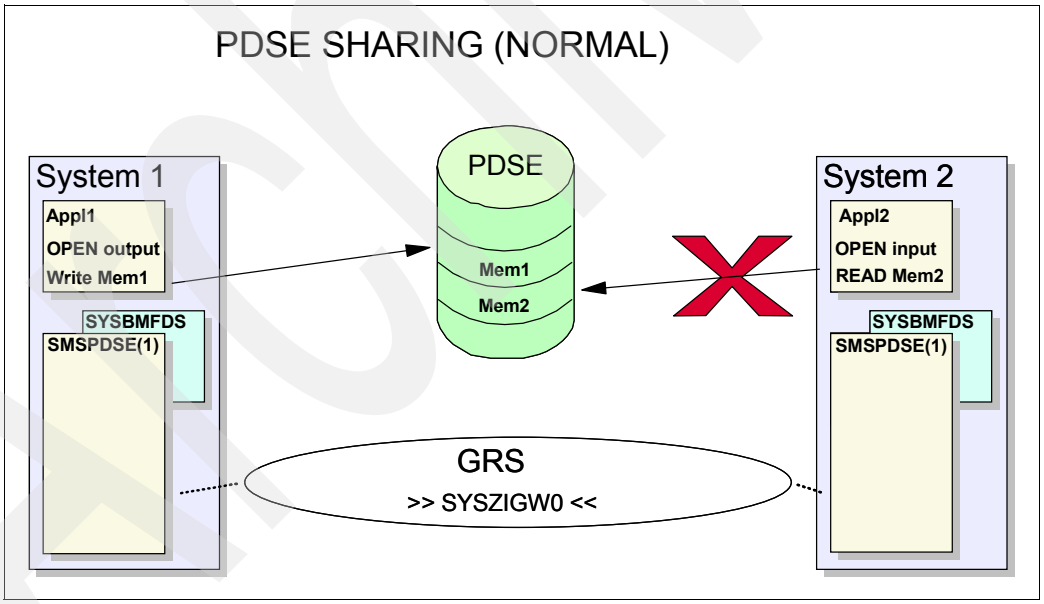


Figure 8-7 PDSE normal sharing across multiple systems

8.4.1 Test results of PDSE normal sharing across systems

Figure 8-8 illustrates the scenario for the cross-system data-set-level sharing test.

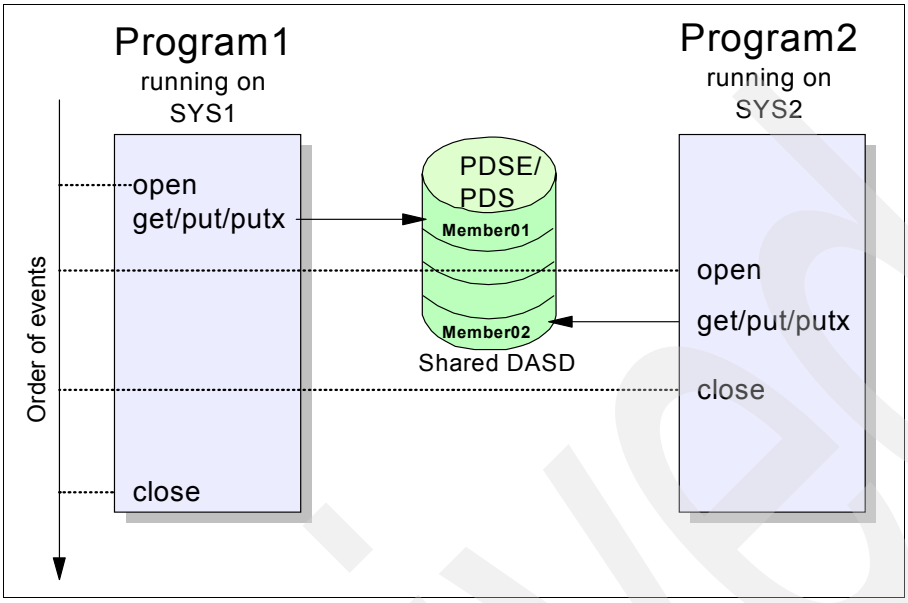


Figure 8-8 Cross system sharing test scenario

The test is the same as the single-system data-set-level sharing test in Figure 8-2 on page 158, except that:

- Program 1 is now running on SYS1, and Program 2 is running on another system (SYS2).
- The PDS or PDSE accessed by Program 1 and Program 2 resides on a shared DASD volume.
- The systems are coupled to each other by GRS.

Table 8-5 summarizes the results of the PDS test, and Table 8-6 on page 164 summarizes the PDSE test. Note that when both jobs have their respective members open for output, it can lead to data corruption of the PDS.

Table 8-5 PDS cross system data-set-level sharing

Second job attempts open for	**First job has PDS currently open for**		
	Input	**Output**	**Update**
Input	Successful	Successful	Successful
Output	Successful	Abend	Successful
Update	Successful	Successful	Successful

In the case of PDSE, the second job will abend if it tries to open the data set when there is a chance of corruption. The first job continues unaffected.

Table 8-6 PDSESHARING(NORMAL) cross system data-set-level sharing

Second job attempts open for	First job has PDSE currently open for		
	Input	**Output**	**Update**
Input	Successful.	OPEN ABEND for second job IEC143I 213-70	OPEN ABEND for second job IEC143I 213-70
Output	OPEN ABEND for second job IEC143I 213-70	OPEN ABEND for second job IEC143I 213-70	OPEN ABEND for second job IEC143I 213-70
Update	OPEN ABEND for second job IEC143I 213-70	OPEN ABEND for second job IEC143I 213-70	OPEN ABEND for second job IEC143I 213-70

The explanation for message IEC143I 213-70 is:

- Abend code 213: This abend code shows that the error occurred during the processing of an OPEN macro instruction for a data set on a direct access device.
- Reason code 70: This reason code states that an OPEN detected a cross-system share conflict for the PDSE.

See *z/OS: MVS System Messages Volume 7 (IEB - IEE)*, SA22-7637 for more information.

8.4.2 Setting up PDSE normal sharing across systems

You should specify `PDSESHARING(NORMAL)` in the SMS member IGDSMSxx in SYS1.PARMLIB.

You must have GRS running on your systems. DFSMS handles all of the enqueues and serialization for you. DFSMS opens the PDSE with `RNL=NO` in the ENQ macro. This tells GRS that DFSMS has specified all the right parameters in the ENQ macro and GRS should skip any additional RNL processing for the PDSE. Therefore, GRS will not even look in the GRSRNLxx PARMLIB member for that PDSE. This means that you do not have to set up anything in GRS for PDSE sharing. The ENQ macro issued by DFSMS will ensure that the resource names SYSZIGW0 and SYSZIGW1 are passed across to other systems by GRS for serialization of the PDSE.

Attention: GRS is the only serialization product that can be used to serialize PDSE. This is because PDSE services specifies RNL=NO for their ENQs.

When RNL=NO is specified, the ENQ request will be ignored by alternative serialization products. This will affect protection of the resource to systems outside of the current global resource serialization environment.

Also, these ENQs are not presented to any of the GRS exits. They are managed strictly by GRS.

8.5 PDSE extended sharing across multiple systems

Extended sharing enables you to share PDSEs at the member level among systems in the same sysplex. This means that a user on one system can open a member of a PDSE while a user from a different system has *any* member of the same PDSE open. (Refer to Figure 8-9.)

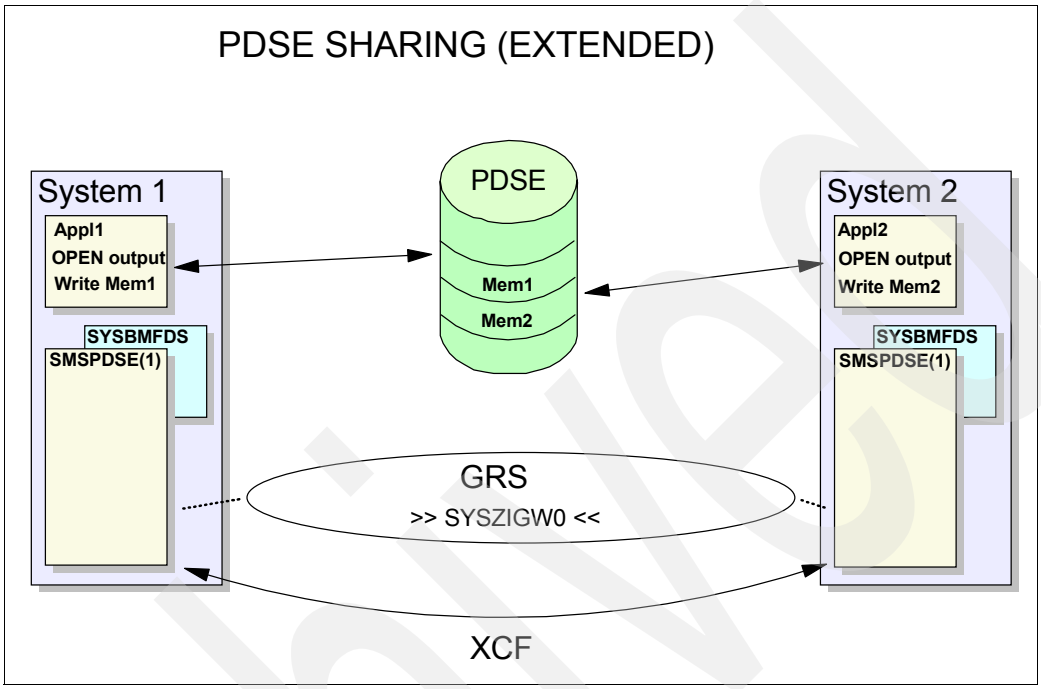


Figure 8-9 PDSE extended sharing

PDSE extended sharing uses GRS and XCF signaling for serialization. The major names SYSZIGW0 and SYSZIGW1 are the same as those used for normal sharing. Both of these ENQs are passed to all the systems in the sysplex as a SYSTEMS ENQ. PDSE code that is running in the SMSPDSE address spaces uses XCF to send contention messages to other systems to negotiate changes in the enqueues. Current versions of z/OS (including DFSMS and GRS) always pass on SYSZIGWx ENQs as SYSTEMS ENQs.

8.5.1 Scenarios for PDSE extended sharing across systems

Users on multiple systems can simultaneously share the same PDSE for input and output. When a member is open for update-in-place, it cannot be opened by any other user.

A sharer of a PDSE can read existing members and create new members (or copies of existing members) concurrently with other sharers on the same or other systems. But shared access to a PDSE, when update-in-place of a member is being done, is restricted to a single system. The update-in-place access restriction exists from the time the updater is positioned to the member (for example, by means of FIND or POINT macro) to the time the updater positions the directory or closes the PDSE.

Figure 8-10 on page 166 shows the overall scenario used to illustrate the sharing of PDSE members across two systems in a sysplex.

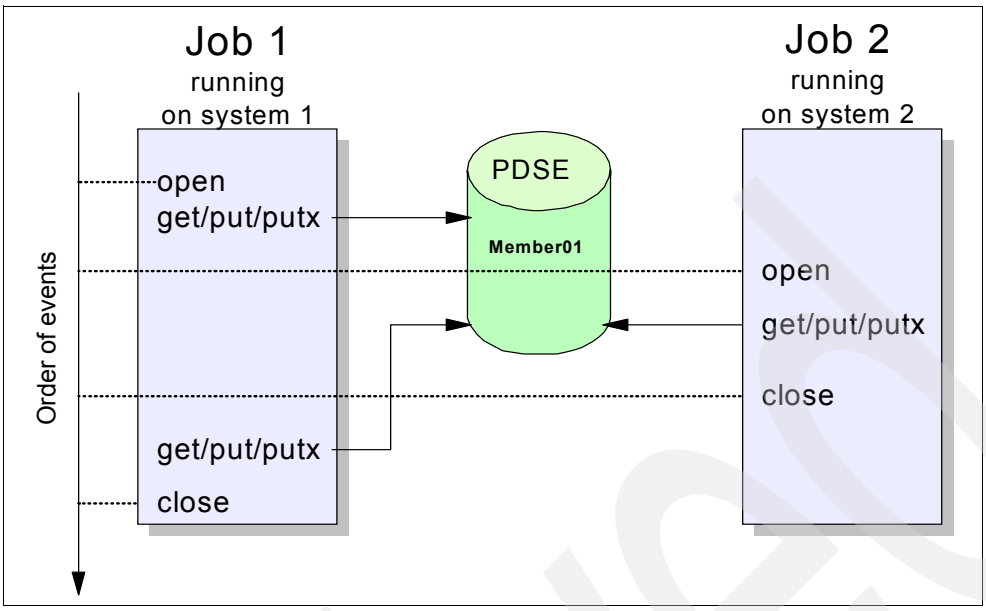


Figure 8-10 Sysplex sharing of a PDSE member, overall scenario

Table 8-7 shows the results of an OPEN when two systems share a PDSE across the sysplex.

Table 8-7 PDSE sysplex-level sharing

Job 2 on system 2 attempts open for	**Job 1 on system 1 has PDSE currently open for**			
	Input	**Output**	**Update - not positioned to a member**	**Update - positioned to a member**
Input	Successful.	Successful. Job 2 reads old data, new data after last close.	Successful.	OPEN ABEND IEC143I 213-74
Output	Successful. Job 1 reads old data, new data after last close.	Successful. Contains data from last close.	Successful.	OPEN ABEND IEC143I 213-74
Update	OPEN ABEND IEC143I 213-70	OPEN ABEND IEC143I 213-70	OPEN ABEND IEC143I 213-70	OPEN ABEND IEC143I 213-70

Note that PDSE member access to system 2 is denied in those cases where there is a possibility of data corruption during the update.

The explanation for message IEC143I ABEND 213 is:

- ABEND 213: The error occurred during processing of an OPEN macro instruction for a data set on a direct access device.
- RC70: This reason code means that an OPEN detected a cross-system share conflict for the PDSE.
- RC74: This reason code means that an OPEN detected a member share option conflict for the PDSE.

Refer to *z/OS: MVS System Messages Volume 7 (IEB - IEE)*, SA22-7637, for more information.

8.5.2 Setting up PDSE extended sharing across systems

PDSE extended sharing can be activated dynamically, without need for an IPL, as follows:

1. Verify that the system is in parallel sysplex mode and not XCF-local mode. Issue the system commands D XCF,SYSPLEX and D XCF,S,*sysname* (or D XCF,S,ALL), and make sure that the IXC335 I message does not indicate XCF-LOCAL MODE. Refer to *z/OS: MVS Setting Up a Sysplex,* SA22-7625 for details about setting up a sysplex.

Figure 8-11 shows an example for a system that is part of a sysplex.

```
D XCF,SYSPLEX
IXC334I  20.26.51  DISPLAY XCF
   SYSPLEX SANDBOX:    SC63              SC64              SC65
                       SC70

D XCF,S,SC64
IXC335I  20.26.55  DISPLAY XCF
SYSTEM   TYPE SERIAL LPAR STATUS TIME         SYSTEM STATUS
SC64     2084 6A3A   11   11/12/2004 20:26:54 ACTIVE
```

Figure 8-11 D XCF sysplex output

System MCEVSF in Figure 8-12 is running in XCL-LOCAL mode.

```
D XCF,SYSPLEX
IXC334I  02.30.45  DISPLAY XCF
   SYSPLEX VSFPLEX:    MCEVSF

D XCF,S,ALL
IXC335I  02.30.51  DISPLAY XCF
SYSTEM   TYPE SERIAL LPAR STATUS TIME         SYSTEM STATUS
MCEVSF   2064 0F5E   N/A  11/13/2004 02:30:50 MONOPLEX MODE
```

Figure 8-12 D XCF monoplex output

2. Modify SYS1.PARMLIB member IGDSMSxx to specify `PDSESHARING(EXTENDED)` on each system.
3. Issue the SET SMS=xx command, identifying the SYS1.PARMLIB member that starts the migration to the EXTENDED protocol. If you have a common SMS member, you can issue the command RO *ALL,SET SMS=xx on any system to route the SET command to all systems in the sysplex.

This SET SMS command establishes each system's preference, and negotiation among the sysplex members begins. When all systems members have agreed to extended sharing, the transition takes place.

You may see this message on each system:

```
IGW303I NORMAL PDSE SHARING FORCED, INCOMPATIBLE PROTOCOL FOUND
```

You may have to issue SET SMS=xx a second time to trigger the switch from NORMAL to EXTENDED sharing. All the systems will issue message IGW306I when they migrate to EXTENDED sharing:

```
IGW306I MIGRATION TO EXTENDED PDSE SHARING COMPLETE
```

Attention: GRS is the only serialization product that can be used to serialize PDSE. This is because PDSE services specifies RNL=NO for their ENQs.

When RNL=NO is specified, the ENQ request is ignored by alternative serialization products, and this affects protection of the resource to systems outside of the current global resource serialization environment using alternative serialization products.

Also, these ENQs are not presented to any of the GRS exits. They are managed strictly by GRS.

8.6 Sharing considerations

These sections describe the requirements and restrictions that exist for the different sharing modes.

8.6.1 Changing the sharing mode from extended to normal

You cannot change the sharing mode from extended to normal dynamically. You must IPL all of the systems in a sysplex at the same time.

Follow these steps for each system that is running with extended sharing:

1. Change SYS1.PARMLIB(IGDSMSxx) to specify `PDSESHARING(NORMAL)` or remove the PDSESHARING keyword to allow the system to default to normal sharing.
2. Re-IPL all systems sharing the SMS configuration concurrently.

8.6.2 PDSE extended sharing requirements

PDSE extended sharing is limited to a single sysplex because PDSE extended sharing uses the cross-system coupling facility (XCF) in addition to global ENQs to implement its sharing protocol. XCF only operates within a single sysplex.

If you are sharing PDSEs among multiple sysplexes or with a system outside of the sysplex, you must specify PDSESHARING(NORMAL) on each of the systems.

Therefore, the extended sharing protocol has these requirements:

- Every system that is sharing a PDSE must be a member of the same sysplex (base or parallel sysplex).

  In *z/OS: MVS Setting Up a Sysplex,* SA22-7625, the section “Characteristics of a Sysplex” provides further information about the requirements for setting up a sysplex, including:

  – z/OS systems
  – Signalling paths between MVS systems
  – Sysplex couple data set
  – Common time reference

  Note: Extended sharing cannot be used in XCF-local mode.

- XCF must be active.
- You must have GRS.

All systems sharing a PDSE must be communicating with each other and must be operating in the same sharing mode to prevent damage to the data set. If you share a PDSE outside a

GRSplex only read access is allowed. Any attempt to update the PDSE may result in damage to the PDSE structures.

The first member of the sysplex that IPLs determines the sharing mode and forces the other members to operate in the same mode. If the sysplex is running in extended mode and another system that can only run normal is added, the added system will not be able to use PDSEs.

Note: Systems in XCF-local mode cannot use PDSE extended sharing protocol. XCFLOCAL indicates that the system is to be a single, stand-alone MVS system that is not a member of a sysplex and cannot use couple data sets. In XCF-local mode, XCF does not provide signaling services between MVS systems.

8.6.3 PDSE locking services and the PDSE address spaces

PDSE locking services manages the serialization of PDSEs. Locking services runs in PDSE address spaces.

MVS ENQ and global serialization services (GRS) are used by PDSE locking services to serialize multitask and multisystem sharing for PDSEs, either in normal or extended sharing mode.

To perform locking in a multisystem environment using the extended sharing protocol, PDSE locking services (including event notification services and cross-system communication services) must be able to communicate with the other systems in a sysplex. These services are used to lock a resource and to negotiate the lock with other systems if resource contention exists. XCF is used solely as a transport facility; the negotiation is between PDSE servers.

The PDSE locking components provide hierarchical, single-system (local), and multisystem (global) locking services to serialize the different parts of a PDSE, such as PDSE directories and PDSE members.

In PDSE terms, a *lock* is a data structure that is maintained in common storage to represent a lockable resource. Several locks are maintained internally to serialize access to the PDSE as well as to individual members.

PDSE locking services also provides latches to serialize access to data structures (for example, control blocks) from different tasks in a system. These internal latches are not GRS latches. An example of such latches and their usage can be found in 10.5.2, “VARY SMS,[PDSE,PDSE1]” on page 224.

When you open a PDSE, locking services issues the ENQ macro using resource SYSZIGW0 for the PDSE, with the scope of Systems, and either the Shared or Exclusive option. The DEQ macro is issued when the PDSE is closed.

Figure 8-13 on page 170 shows the relationship between the different components.

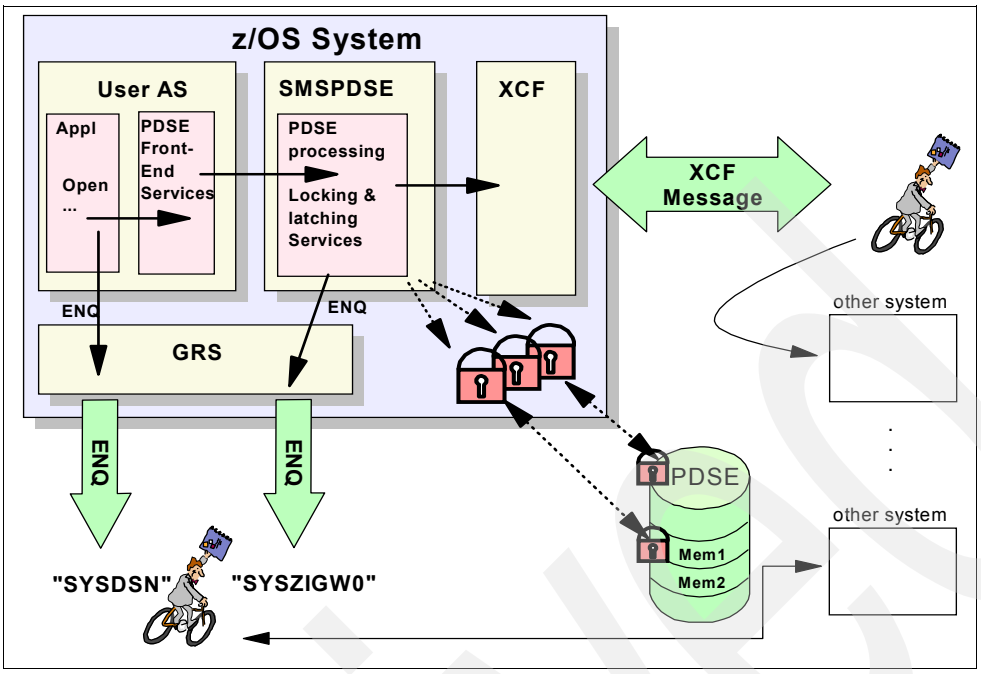


Figure 8-13 Extended sharing and locking (non-restartable)

If you decide to start SMSPDSE1, there will be two PDSE address spaces on that system:

- The non-restartable SMSPDSE address space is used for global connections (for example, PDSEs that are contained in the LNKLIST).
- The restartable SMSPDSE1 address space is used for all other PDSEs connections.

The restartable PDSE address space (SMSPDSE1) is optional and available only for systems that use PDSESHARING(EXTENDED). These PDSE address spaces operate independently of each other, so they both provide locking and latching services.

PDSE front-end services (in the user address space) decides which PDSE address space will process each request. Figure 8-14 on page 171 shows the relation between the user and PDSE address spaces.

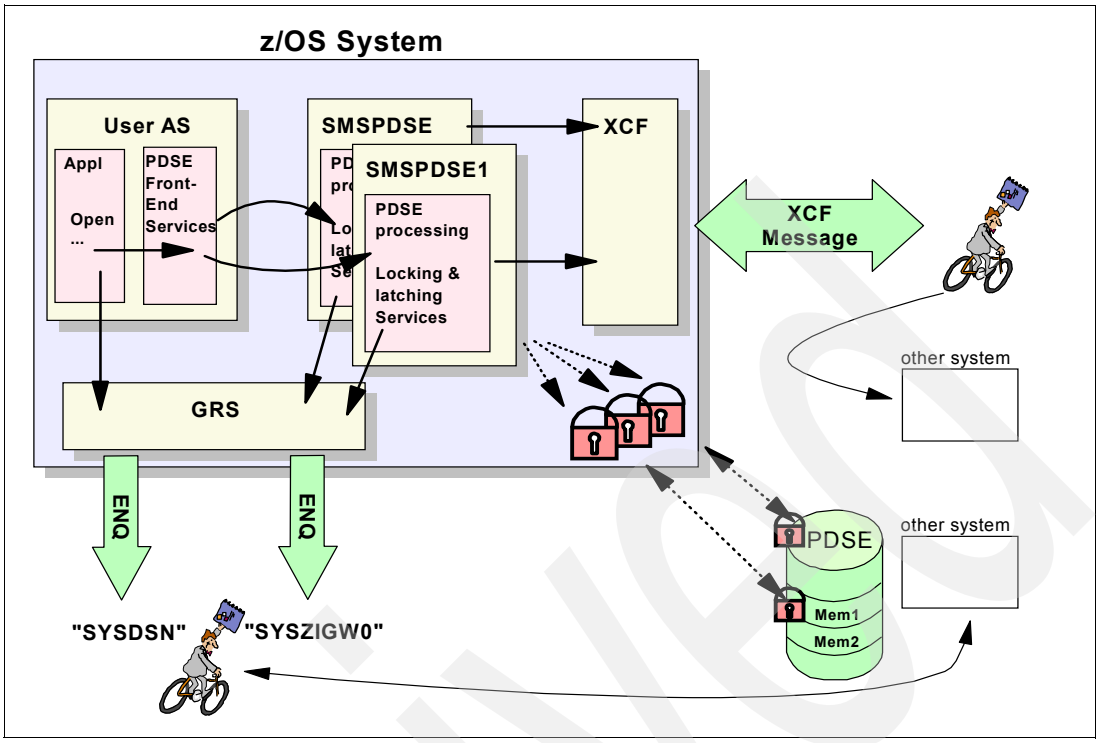


Figure 8-14 Extended sharing and locking (restartable)

8.6.4 PDSE extended sharing with MVS guests under VM

PDSE extended sharing cannot be used in a sysplex with a mix of native MVS machines and MVS guest machines under VM. This is because the coupling facility and sysplex timer are not supported in a mixed environment of MVS guests under VM and native MVS systems. PDSE extended sharing uses XCF, which requires the coupling facility and sysplex timer.

8.6.5 Detecting sharing mode

The DFSMSdfp callable service, IGWLSHR, can be used for a program to detect the sharing mode that is in operation. See *z/OS: DFSMSdfp Advanced Services*, SC26-7400, for information.

8.6.6 Summary of sharing considerations

You must follow these general rules to share PDSE data sets across a sysplex:

- All systems in the sysplex must use the same sharing protocol, as specified in the SMS PARMLIB member. You cannot have NORMAL on one system and EXTENDED on another.
- All systems sharing a PDSE must communicate using GRS. Systems scope enqueues are passed to all systems. These enqueues use major names SYSZIGW0 and SYSZIGW1.
- All systems using PDSESHARING(EXTENDED) must be in a sysplex and use XCF messaging.
- Extended sharing protocol cannot be used in XCF-local mode.
- If there is a mix of sysplex and non-sysplex machines sharing PDSEs, then PDSESHARING(NORMAL) must be used.

- If all systems sharing the PDSE are not part of the same sysplex, then PDSESHARING(NORMAL) must be used.
- The first MVS machine that IPLs establishes the preferred protocol and, as additional machines IPL, the protocol can be negotiated.
- If the sharing protocol is first established as normal, it can be migrated to extended without an IPL if all the systems are in a sysplex and they all have `PDSESHARING(EXTENDED)` specified.
- If the established protocol in a sysplex is extended, it cannot be migrated to normal sharing dynamically. An IPL of all systems is required to return to normal sharing.
- If a system that is only capable of normal sharing tries to join a group using extended sharing, the system is not allowed to join, and all PDSE support is disabled on that system (so that it cannot damage PDSEs).

8.7 PDSE serialization

This section describes the enqueues used by PDSE services.

8.7.1 Definitions

Definitions of terms in conjunction with PDSE serialization functions:

LATCH A serialization object that PDSE processing uses to maintain the integrity of its control block structures. These latches are not GRS latches. Other than the V SMS,PDSE|PDSE1 commands, there are no mechanisms that enable you to directly determine or modify the status of the PDSE latches.

Scope: *system*

LOCK The internal mechanism that is used to serialize access to the PDSE internal structures.

Scope: *sysplex*

8.7.2 ENQs used for PDSE sharing

To serialize access to PDSE data sets, PDSE locking services uses two ENQs with major names SYSZIGW0 and SYSZIGW1. These ENQs are used for both sharing protocols, normal and extended. Both ENQs are passed to all systems in the sysplex as global (with scope SYSTEMS) enqueues.

If global resource serialization was established properly, then you should see the same ENQs on all systems in the sysplex. Otherwise, the ENQs are not treated as global resources, which can result in PDSE damage, typically to the index structure.

In addition to these enqueues and to be compatible with other MVS data sets, an ENQ on SYSDSN also appears when a PDSE is allocated, either in JCL or by dynamic allocation.

8.7.3 ENQ on SYSZIGW0

An ENQ with a major name of SYSIGW0 represents an open to a PDSE. In PDSE terms, a PDSE (directory) is said to be *connected* to a task.

The minor name of a SYSZIGW0 ENQ contains a token plus an additional character. The token allows a unique identification of the associated PDSE within the sysplex. This token is called the *VSGT* (Virtual Storage Group Token).

You can display the token by using the hexadecimal view of the ENQ. See “Interpreting output of D GRS commands” on page 239 for a description.

The SYSZIGW0 ENQ is always obtained by a task in the PDSE address spaces. Therefore the SYSZIGW0 ENQ can be used to verify whether a PDSE is connected to SMSPDSE, SMSPDSE1, or both. If you see more than one user of SYSZIGW0 ENQ for the same PDSE from different systems, then the PDSE is shared between these systems.

To display the current SYSZIGW0 ENQs, you can use this command as in Figure 8-15:

```
D GRS,RES=(SYSZIGW0,*)
```

```
D GRS,RES=(SYSZIGW0,*)
ISG343I 11.50.23 GRS STATUS 844
S=SYSTEMS SYSZIGW0  SBOX26   I
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC65       SMSPDSE1           0009       007FD230   SHARE      OWN
SC64       SMSPDSE1           0009       007FD230   SHARE      OWN
SC63       SMSPDSE            0008       007FFD90   SHARE      OWN

S=SYSTEMS SYSZIGW0  SBOXB8   I
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC64       SMSPDSE            0008       007FD230   SHARE      OWN
SC64       SMSPDSE1           0009       007FD230   SHARE      OWN
```

Figure 8-15 D GRS,RES=(SYSZIGW0,) output*

Figure 8-16 shows the hexadecimal view of SYSZIGW0 ENQ.

```
D GRS,RES=(SYSZIGW0,*),HEX
ISG343I 11.54.09 GRS STATUS 860
S=SYSTEMS SYSZIGW0  SBOX26   I
          EEEECCEF 0ECDEFF031C
          28299760 12267260509
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC65       SMSPDSE1           0009       007FD230   SHARE      OWN
SC63       SMSPDSE            0008       007FFD90   SHARE      OWN
SC64       SMSPDSE1           0009       007FD230   SHARE      OWN

S=SYSTEMS SYSZIGW0  SBOXB8   I
          EEEECCEF 0ECDECF022C
          28299760 12267280D19
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC64       SMSPDSE            0008       007FD230   SHARE      OWN
SC64       SMSPDSE1           0009       007FD230   SHARE      OWN
```

Figure 8-16 D GRS,RES=(SYSZIGW0,),HEX output*

In this case, two PDSEs are open. One ENQ is obtained for a PDSE that resides on volume SBOX26; the PDSE is shared among three systems (SC63, SC64, and SC65). The other SYSZIGW0 ENQ represents a PDSE on volume SBOXB8, which is connected at least twice on system SC64: once for a global (LINKLIST) and another for a non-global connection.

There is no external association to the PDSE user task using the SYSZIGW0 resource. However, for diagnosis purposes, it is important for you to identify the users that are connected to the PDSE. The *VSGT*, which is shown in the hexadecimal format of the D GRS command, contains the *VOLSER* as well as a TTR pointer to the associated Format 1 DSCB of the PDSE. As mentioned earlier, these two pieces of information uniquely identify the PDSE within a multisystem environment.

See 10.5.4, “Interpreting output of D GRS commands” on page 239 for an example of how to identify PDSE user tasks starting with the display of a SYSZIGW0 ENQ.

“Which PDSE address space is used?” on page 245 describes how to determine the associated PDSE address space for a PDSE that is currently open or connected.

8.7.4 ENQ on SYSZIGW1

The ENQ with major name SYSZIGW1 is used to negotiate the different sharing protocols among the systems in a sysplex.

The minor names are CLM00000, CLM00001, or CLM00002 (Figure 8-17).

```
D GRS,RES=(SYSZIGW1,*)
ISG343I 12.20.36 GRS STATUS 902
S=SYSTEMS SYSZIGW1 CLM00002
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC65       SMSPDSE             0008       007FD230   SHARE      OWN
SC65       SMSPDSE1            0009       007FD230   SHARE      OWN
SC70       SMSPDSE             0008       007FD230   SHARE      OWN
SC70       SMSPDSE1            0009       007FD230   SHARE      OWN
SC64       SMSPDSE             0008       007FD230   SHARE      OWN
SC64       SMSPDSE1            0009       007FD230   SHARE      OWN
SC63       SMSPDSE             0008       007FFD90   SHARE      OWN
```

Figure 8-17 D GRS,RES=(SYSZIGW1,) output*

As shown, the ENQ with major name SYSZIGW1 will be obtained by every PDSE address space in the sysplex, including restartable (SMSPDSE1) and non-restartable (SMSPDSE) address spaces.

You can use the D SMS,OPTIONS command to identify the current sharing protocol (Figure 8-18).

```
D SMS,OPTIONS
IGD002I 12:21:48 DISPLAY SMS 904
...
PDSESHARING = EXTENDED
...
```

Figure 8-18 D SMS,OPTIONS output

8.7.5 ENQ on SYSDSN

An ENQ with the major name SYSDSN, and a minor name that represents the data set name, is also obtained while a PDSE is open to a task or a job.

This ENQ is required for the PDSE enqueue scheme to be consistent with the enqueues used for other MVS data sets. However, in contrast to the SYSZIGW0 ENQ, which is always obtained by a task in a PDSE address space, the SYSDSN ENQ is obtained by the task or job that allocated the PDSE. A D GRS,RES=(SYSDSN,pdse.dsn) command can be used to identify other tasks or jobs that share the PDSE at the same time. This might be helpful in diagnosing problems.

Figure 8-19 shows the SYSDSN ENQ information for PDSE PDSERES.SMS.DIRK.PDSE1 on volume SBOX26 and the associated job names. In this case, three users on three different systems are accessing the PDSE, by using the ISPF/PDF option 3.4 BROWSE function, at the same time.

In addition, we show the DISPLAY GRS output for the PDSE DB2R7.SDSNLOD2 on volume SBOXB8 (also see Figure 8-15 on page 173 and Figure 8-16 on page 173). This PDSE is currently connected to jobs XCFAS and LLA on system SC64. The LLA connection is a global connection and therefore handled by the non-restartable PDSE address space SMSPDSE. The access requested by XCFAS is managed by the restartable PDSE address space SMSPDSE1.

```
D GRS,RES=(SYSDSN,PDSERES.SMS.DIRK.PDSE1)
ISG343I 12.16.07 GRS STATUS 872
S=SYSTEMS SYSDSN   PDSERES.SMS.DIRK.PDSE1
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC65      MHLRES7            00A7      007FD230   SHARE      OWN
SC63      MHLRES2            008E      007FFD90   SHARE      OWN
SC64      MHLRES6            0076      007FD230   SHARE      OWN

D GRS,RES=(SYSDSN,DB2R7.SDSNLOD2)
ISG343I 12.16.20 GRS STATUS 874
S=SYSTEMS SYSDSN   DB2R7.SDSNLOD2
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC64      XCFAS              0006      007FD230   SHARE      OWN
SC64      LLA                0019      007FD230   SHARE      OWN
```

Figure 8-19 D GRS,RES=(SYSDSN,dsname) output

8.7.6 XCF and XCM

XCM is a subcomponent of PDSE processing that is responsible for communicating between systems using the XCF facilities.

XCF is used as a transport facility for the exchange of locking information among the sharing systems in a sysplex (extended sharing only).

Two XCF groups are created or joined by PDSE locking services running in PDSE address spaces. One group (SYSIGW00) is used for lock negotiations; the other group (SYSIGW01) is used for status monitoring. Both groups are created or joined as part of PDSE address space initialization.

Figure 8-20 on page 176 shows the XCF communication paths used by SMSPDSE1 address space on system SC64 in relation to systems SC63, SC64, and SC70.

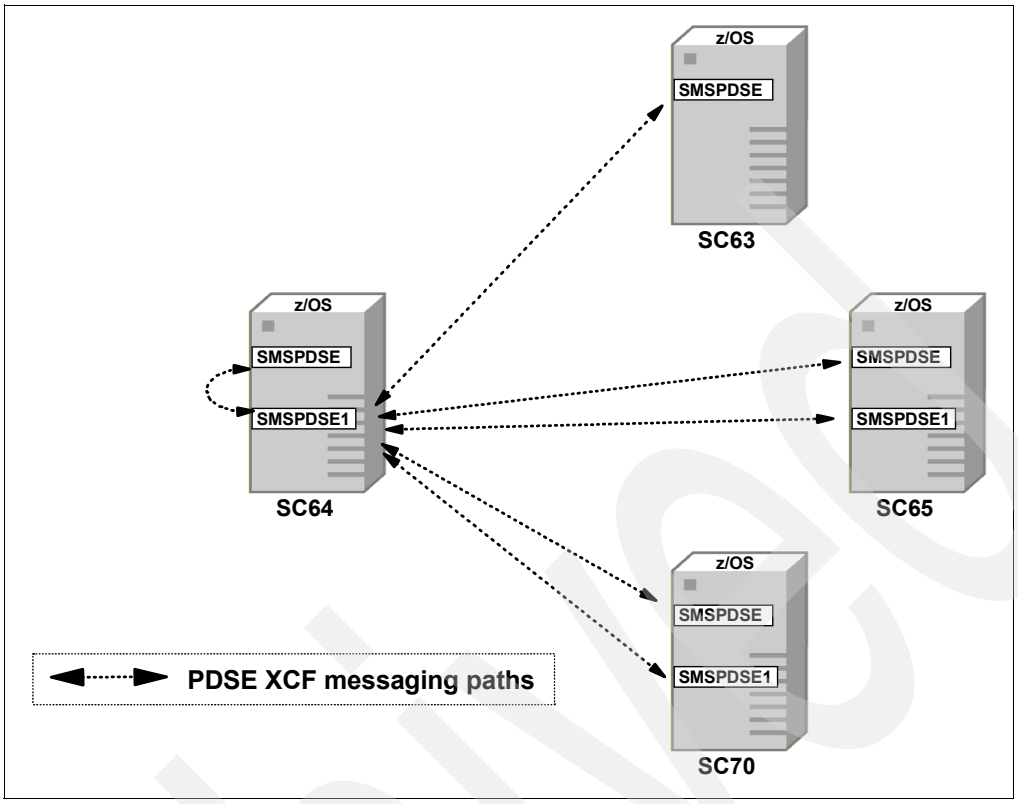


Figure 8-20 PDSE XCF messaging paths

Note: The restartable address space SMSPDSE1 either was not started on SC63 system (PDSE_RESTARTABLE_AS(NO) in IGDSMSxx parmlib member) or it is not z/OS V1.6 or later system and therefore does not provide the PDSE restart capability.

Various console commands can be used to verify the status of XCF:

- D XCF lists all systems currently participating in the sysplex in message IXC334I.
- D XCF,S,ALL displays detailed information for all systems in the sysplex (including the system status and the last recorded system status monitor time stamp for a system) in message IXC335I (Figure 8-21).

```
D XCF
IXC334I  22.58.40  DISPLAY XCF 486
   SYSPLEX SANDBOX:    SC63              SC64              SC65
                       SC70
D XCF,S,ALL
IEF196I IEF237I AF04 ALLOCATED TO SYS00139
IXC335I  22.58.55  DISPLAY XCF 489
SYSTEM   TYPE SERIAL LPAR STATUS TIME         SYSTEM STATUS
SC63     2084 6A3A   0C   11/04/2004 22:58:54 ACTIVE
SC64     2084 6A3A   11   11/04/2004 22:58:53 ACTIVE
SC65     2084 6A3A   12   11/04/2004 22:58:53 ACTIVE
SC70     2084 6A3A   13   11/04/2004 22:58:51 ACTIVE
```

Figure 8-21 D XCF output

- D XCF,GROUP,SYSZIGW0 displays the members of the specified group. You should see one member for each system in the sysplex that participates in the extended sharing mode in message IXC332I. See Figure 8-22.

```
D XCF,GROUP,SYSIGW00
IXC332I  23.02.48  DISPLAY XCF 523
 GROUP SYSIGW00:  IGWCLMR1SC64      IGWCLMR1SC65      IGWCLMR1SC70
                  IGWCLM01SC63      IGWCLM01SC64      IGWCLM01SC65
                  IGWCLM01SC70
```

Figure 8-22 D XCF,GROUP,SYSIGW00 output

- D XCF,GROUP,SYSIGW00,* displays detailed information—the system name, MVS job name, or current status—about the members of a particular group or all groups in message IXC333I. See Figure 8-23.

```
D XCF,GROUP,SYSIGW00,IGWCLMR1SC64
IXC333I  23.00.43  DISPLAY XCF 519
   INFORMATION FOR GROUP SYSIGW00
   MEMBER NAME:        SYSTEM:     JOB ID:    STATUS:
   IGWCLMR1SC64        SC64        SMSPDSE1   ACTIVE
D XCF,GROUP,SYSIGW00,*
IXC333I  23.00.57  DISPLAY XCF 521
   INFORMATION FOR GROUP SYSIGW00
   MEMBER NAME:        SYSTEM:     JOB ID:    STATUS:
   IGWCLMR1SC64        SC64        SMSPDSE1   ACTIVE
   IGWCLMR1SC65        SC65        SMSPDSE1   ACTIVE
   IGWCLMR1SC70        SC70        SMSPDSE1   ACTIVE
   IGWCLM01SC63        SC63        SMSPDSE    ACTIVE
   IGWCLM01SC64        SC64        SMSPDSE    ACTIVE
   IGWCLM01SC65        SC65        SMSPDSE    ACTIVE
   IGWCLM01SC70        SC70        SMSPDSE    ACTIVE
```

Figure 8-23 D XCF,GROUP,SYSIGW00, output*

Figure 8-24 on page 178 summarizes the D XCF,GROUP,SYSIGW00,* output.

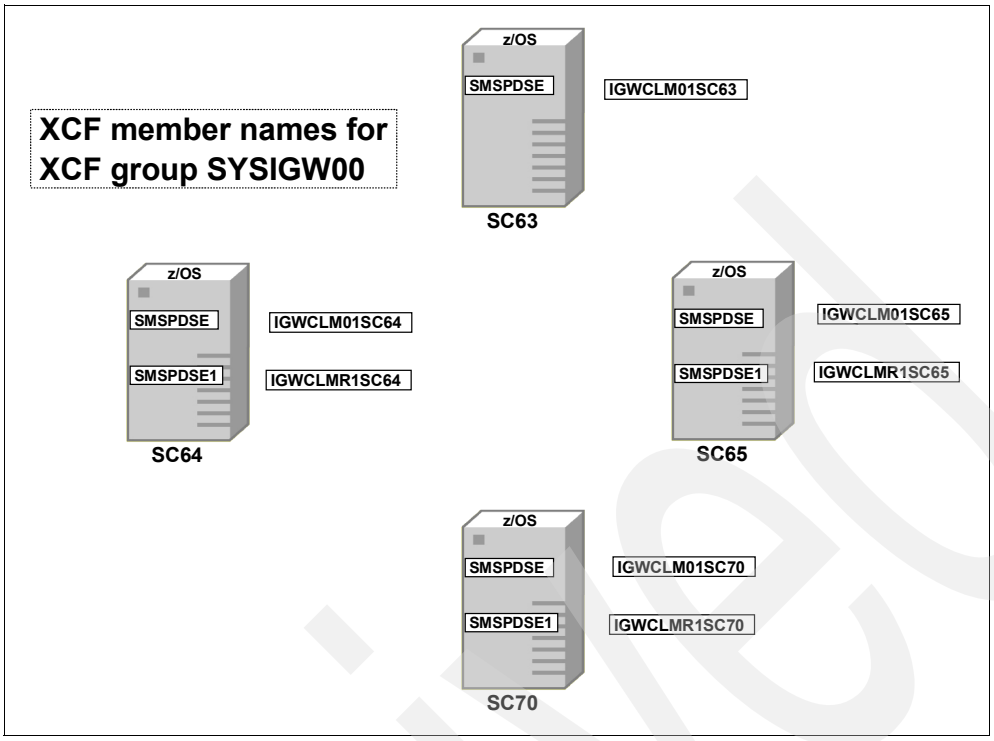


Figure 8-24 XCF member names

See *z/OS: MVS System Commands*, SA22-7627, for a description of the XCF commands to verify XCF status and to verify the connected (joined) clients (or members) in the XCF group.

Remember that systems in XCF-local mode cannot use the PDSE extended sharing protocol. XCFLOCAL mode indicates that the system is to be a single, stand-alone MVS system that is not a member of a sysplex and cannot use couple data sets. In XCF-local mode, XCF does not provide signaling services between MVS systems.

8.7.7 XQuiesce

During a restart of SMSPDSE1, a serialization mechanism called *XQuiesce* is invoked while serialization based on PDSE locks is temporarily abandoned. XQuiesce broadcasts messages to other systems in a sysplex, informing them that an SMSPDSE1 restart is beginning by using XCM and XCF. See “XCF and XCM” on page 175 and Figure 8-20 on page 176 for more information. The other systems are prevented from performing certain updating operations on PDSEs until the new SMSPDSE1 server instance is available.

9

Performance considerations

The first section of this chapter describes the buffering techniques used for PDSEs. The second section describes the way in which PDSEs take advantage of z/Architecture™ to cache directory and member pages in processor storage. Next, we summarize the tools that you can use to monitor the performance and resource use of PDSEs. Finally, we compare PDSE to other system functions that use z/OS facilities to improve performance.

This chapter addresses how PDSEs have been designed for high performance, but it does not present any measured results.

9.1 I/O activity and response time

The major element of transaction response time for systems that process standard PDSs tends to be I/O activity. So, in keeping with the general direction of z/OS performance improvements, the most important performance design goal for PDSEs is to reduce the physical I/O done to the directory and to the members.

This is achieved by keeping copies, in processor storage, of directory and member pages that are likely to be used again. I/O requests that would have caused a physical I/O when using PDSs can now be satisfied from processor storage, and the physical I/O can be avoided. DFSMSdfp manages these copies actively, copying directory and member pages into storage as they are needed, then removing them when they are unlikely to be used again. It also manages PDSE integrity, security of access, and sharing between different programs.

Performance improvements are seen in two areas: the overall I/O workload of the system is reduced, and the response time for access to individual PDSE pages is faster than access to PDS members. However, PDSEs tend to have a greater number of instructions executed for each transaction. When an I/O is done to a PDSE, DFSMSdfp looks for a copy in processor storage before it looks on disk, and this uses additional processing.

Thus, for a given workload with PDSEs, you see less I/O activity but slightly more CPU activity. This means that the workload runs faster overall in systems that are not CPU constrained.

9.2 PDS buffering

This section reviews how the sequential access methods manage I/O buffers for PDS members. Read *z/OS V1R3.0-V1R6.0 DFSMS Macro Instructions for Data Sets*, SC26-7408, for full details about using QSAM and BSAM macros to process PDSs and PDSEs.

9.2.1 QSAM buffering

QSAM manages blocking and unblocking of records and buffering of I/O automatically. It enables a program to process logical records without having to consider the physical block size or how the buffers are managed.

The two main QSAM macros are GET and PUT. GET retrieves the next sequential logical record from the I/O buffers, and PUT puts the next sequential logical record in the I/O buffers.

QSAM does a physical I/O to read data from disk when the I/O buffers are empty and to write data when the buffers are full. It usually reads or writes all buffers.

Figure 9-1 on page 181 illustrates QSAM buffering. The I/O to and from the I/O buffers is done by the QSAM code.

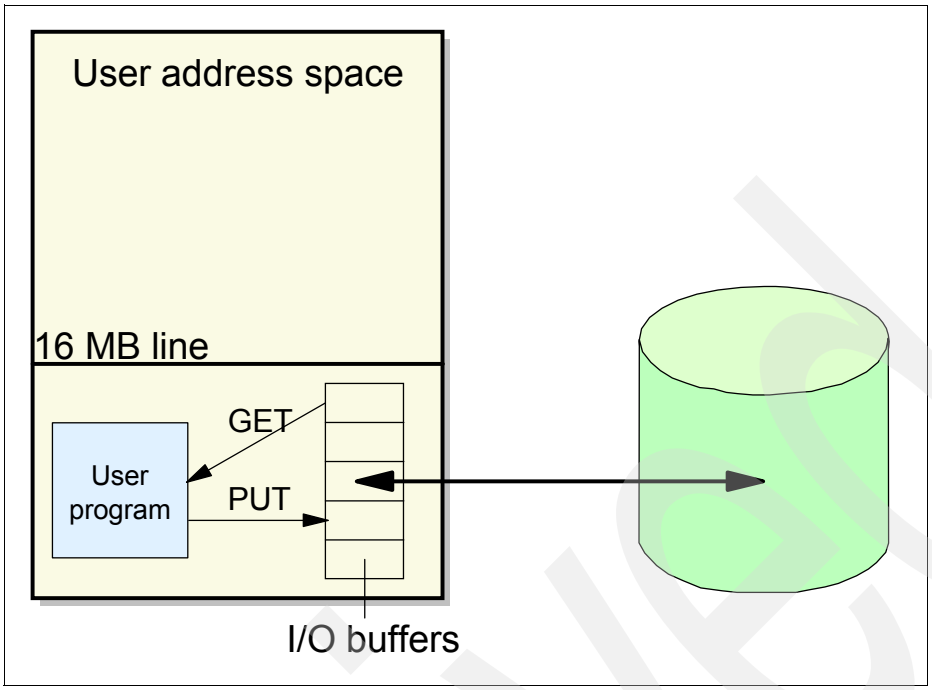


Figure 9-1 PDS buffering with QSAM

The amount of virtual storage allocated for an I/O buffer is generally equal to the block size of the data set. The default number of buffers for a PDS is five. The application program can use the DCBE RMODE31 parameter to control whether the virtual storage for them is allocated in the user's private area below the 16 MB line or above the 16 MB line. On the DCB for your data set, you can specify the block size with the BLKSIZE parameter, and the number of buffers with the BUFNO parameter. The total I/O buffer space is BLKSIZE x BUFNO. BLKSIZE must be at least as large as the largest block in the data set.

For PDSEs, BLKSIZE only has to meet the requirements of the data format because PDSE code will re-block the data on READ or GET operations.

9.2.2 BSAM and BPAM buffering

Programs that use BSAM or BPAM optionally define their own input and output areas in virtual storage as I/O buffers, and must block and unblock records. Your program can use BUILD or GETPOOL to build a buffer pool and GETBUF to acquire each buffer, but IBM suggests defining I/O areas with the GETMAIN, STORAGE, or CPOOL macro. The latter macros enable storage to be acquired above the 16 MB line. BSAM and BPAM allow more flexibility in how the I/O is managed at the cost of more complicated programming. It is used by applications such as the IMS logging function, which must use double buffering techniques to maximize throughput and to be certain when particular blocks have been written successfully to disk.

BPAM includes all of the BSAM macros plus BLDL, FIND, STOW, and DESERV. The three main BSAM macros are READ, WRITE, and CHECK. READ issues a request for data to be read from disk into an input area in virtual storage. WRITE issues a request for data to be written to disk from an output area in virtual storage. CHECK is used after each READ or WRITE to make the program wait until the physical I/O is completed. Optionally, you can use the TRUNC, WAIT, or EVENTS macros for special cases.

If every READ or WRITE is followed by a CHECK, a physical I/O is done for each block and, effectively, only one buffer can be used. You can use more than one buffer by issuing multiple READs or WRITEs before issuing the CHECK macro. This might not be as efficient as QSAM, which reads or writes all buffers on every physical I/O, but you can control exactly when to overlap I/O with other processing in your program, and when to wait for a physical I/O to complete. This enables you to do read-ahead processing to get records ready before you need them, and to do double buffering by processing data in one buffer while I/O is in progress to or from the other buffer. These techniques speed up I/O-intensive applications.

The number of READs or WRITEs that can be chained together is controlled by the NCP parameter on your DCB, or the MULTACC and MULTDSN parameters on your DCBE. The BUFNO parameter can be coded, but it is useful only if your BSAM or BPAM program uses the GETBUF macro. If your JCL or DCBE macro uses the BUFNO parameter, OPEN builds a buffer pool below the 16 MB line. GETBUF is the only way to use those buffers. The total I/O buffer space a program should allocate for a BSAM data set is BLKSIZE x NCP.

9.3 PDSE buffering techniques

The PDSE buffering techniques apply specifically to programs that use the sequential access methods, BSAM, QSAM, or BPAM, to read and write partitioned data set members. Such programs include most IBM subsystems and nearly all user-written programs.

The programs do not have to be rewritten to use these buffering techniques. Buffering is managed by the access methods.

In addition to improving performance, PDSE buffering also saves virtual storage in the user's region below 16 MB.

9.3.1 PDSE buffering in the Data Work Area

For PDSEs, the DFSMSdfp buffer manager creates a virtual storage area called Data Work Area (DWA), which acts as an intermediate buffer for I/O to and from PDSE data (not program object) members. Each user address space that processes PDSEs has its own DWA in the extended private region, above the 16 MB line. A separate DWA is used for each open DCB.

With a DWA acting as an intermediate buffer, a program does not need as many I/O buffers in the private region below the 16 MB line. The default number of buffers (BUFNO) used by QSAM is now 1 for a PDSE, but remains at 5 for a PDS. If you specify BUFNO greater than 1 for a PDSE, QSAM still allocates the number of buffers you specify but only uses one of them.

The user buffer is retained so that the access method interface remains the same. This means that you do not have to rewrite your programs in order to use PDSEs. As described above, with PDSs, QSAM writes to disk when the buffers are full and reads from disk when they are empty. For PDSEs processed with QSAM, the DFSMSdfp buffer manager copies data to the DWA when the user I/O buffer is full and from the DWA when the user buffer is empty. For BSAM, the buffer manager copies data to the DWA as the program writes it, and from the DWA as the program reads it.

This design enables the DFSMSdfp buffer manager to do read-ahead processing and double buffering for both QSAM and BSAM. Figure 9-2 on page 183 illustrates the DWA.

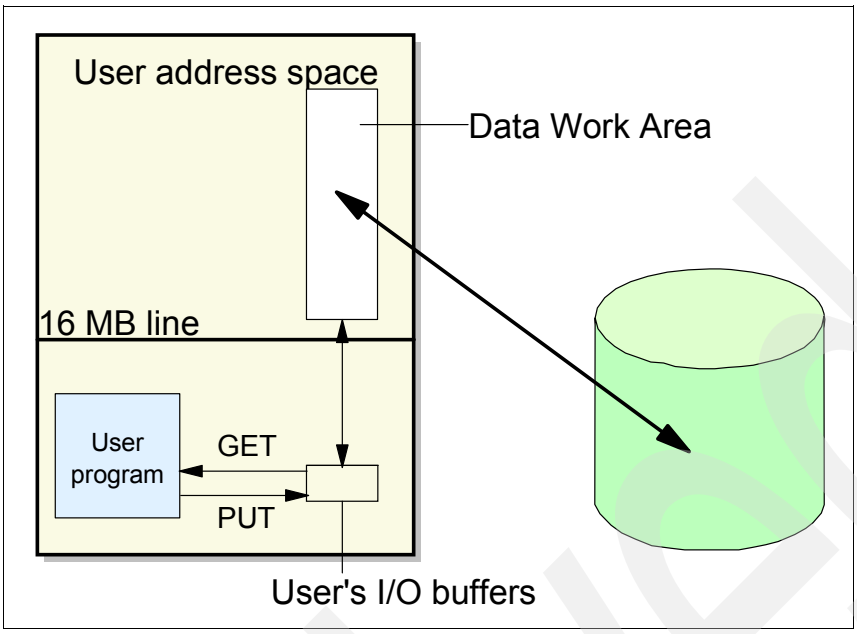


Figure 9-2 PDSE buffering with the Data Work Area

Controls

The size of the DWA is an important performance factor. A large DWA makes the I/O more efficient, because more data is transferred for each I/O request. But a large DWA also consumes more virtual storage (and real storage during the I/O). The size is determined by DFSMSdfp based on the existing parameters used to determine the buffer size for PDSs and the DFSMS response time objectives for the data set.

The response time objectives for a data set are the sequential and direct millisecond response (MSR) values specified in the DFSMS storage class that is assigned to the data set if it is created as a system-managed data set. DFSMSdfp uses these values to give data sets a sequential and a direct performance attribute. This attribute can be high, medium, or low performance. If the PDSE is not system-managed, it does not have a storage class, so DFSMSdfp assumes that medium performance is required.

The ranges of MSR values that DFSMSdfp maps to the three attributes depend on which disk device and storage controller you are using. In general, they are set as follows:

- If the MSR value can be met only by using cache in the storage controller (for example, if the MSR is 1 ms), the performance attribute is *high*.
- If the MSR value can be met by using disk without the storage controller cache (for example, if it is 40 ms), the performance attribute is *medium*. This is the objective set for a non-SMS PDSE.
- If the MSR is set to the special value of 999 ms, the performance attribute is *low*.

The system uses these performance attributes to determine the size of the DWA as well as the caching attributes. The sequential performance attribute is the only one used, and only to control the size of the DWA.

The system uses the direct and sequential MSR values to set cache usage attributes. See the discussion in 9.5.1, “PDSE member caching in Hiperspace” on page 188.

The size of the DWA is determined as follows:

- If the performance attribute of the PDSE is low, DFSMSdfp creates a DWA that is the same size as the PDS buffer area would have been (that is, BLKSIZE x BUFNO for QSAM, and BLKSIZE x NCP for BSAM).
- If the performance attribute of the PDSE is high, the DWA is 100% larger than the low performance DWA and consists of two segments of this larger size. The two segments enable DFSMSdfp to use double buffering.

In all cases, DFSMSdfp adds an extra page to the DWA for control information. Extra space is also added for variable length records, and a two-segment DWA is always used if open for update.

Benefits

These are the benefits of PDSE buffering:

- The DWA improves data integrity for PDSEs. With QSAM and BSAM buffering, a program may accidentally overwrite the buffer area before output data has been written to disk. If this happens, it can be very difficult to detect and correct. However, because the DWA is in protected storage, a user program cannot overwrite it. After the data is copied to the DWA, it is safe from user overlays.
- Having a DWA in the extended private region allows the user I/O buffer to be much smaller. This provides virtual storage constraint relief (VSCR) where it is needed in the private region below the 16 MB line.
- The way the size of the DWA is controlled enables the storage administrator to influence the allocation of virtual and central storage for I/O buffers. This extends the central policy control to all of the components of I/O response time.
- Even though you might set the BSAM or BPAM NCP value high to cause the DWA to be larger, there is no performance advantage in attempting I/O overlap with the CPU. You might as well issue CHECK after each READ or WRITE macro as you will still get the benefits of a large DWA.
- The DWA increases the efficiency of the physical I/O, as the amount of data transferred for each I/O might be greater.
- The DWA provides more efficient I/O to old applications that still use a single buffer or a small block size.
- The DWA enables you to do double buffering for QSAM as well as BSAM programs.

9.4 Performance influencers and exploiters

In this section, we cover the major influencers and exploiters related to PDSE performance.

9.4.1 Effect of Guaranteed Synchronous Write

As described in 5.6, “Guaranteed Synchronous Write” on page 102, GSW is a mechanism to ensure that data in the PDSE buffers is committed to disk immediately. It is an attribute given to a data set at allocation time and, therefore, applies to all I/Os to that data set when the data set is open for update.

Because a program updating a GSW data set waits for each block of data to be written out to disk, GSW cancels out the performance benefits of buffering PDSEs. Therefore, GSW should be used only where it is required.

GSW is a parameter specified in the SMS Storage Class definition (see Table 9-1).

Table 9-1 GSW SMS Storage Class definition

Guaranteed Synchronous Write	Performance	Buffer synchronization to disk
Yes	Lower	Immediately
No	Good	Later

9.4.2 Caching with LLA and VLF

This section covers PDSE directory and member caching functions for program objects, Clist, and REXX execs with LLA and VLF.

LLA and VLF

Library lookaside (LLA) is a z/OS function that may enable you to reduce the time it takes to search the directories of your load module libraries and load the code. You decide which libraries you want LLA to control, and you define them by listing them in a SYS1.PARMLIB member, CSVLLAxx. When you start LLA, the directories of the libraries you have defined are loaded into storage. When these directories are searched, they are searched in storage and no I/O is done. This reduces the search time.

When you use VLF and LLA together, you can keep in storage the directories of load modules as well as the load modules themselves. LLA makes the decision when to cache a module and it is stored by VLF. Later fetches are from virtual storage and, therefore, are much faster. There are considerations that affect whether a load module, program object, or data is stored by VLF as described in “LLA and VLF functional description” on page 186.

LLA and VLF work together to provide good performance for the directories and members of PDSs. The DFSMSdfp buffer manager provides a similar function for PDSEs. Directories are cached in a data space such as the LLA data space, and members are cached in a Hiperspace such as the VLF Hiperspace.

Load module PDSs can continue to benefit from LLA and VLF, and PDSs that contain data can be converted to PDSEs to get high performance.

Linklist load libraries and object libraries are included via parmlib member LNKLSTxx. LLA also manages libraries that are specified in any active CSVLLAxx members. This does not apply to PDSEs containing non-program objects.

PDSE program objects can be cached in VLF by LLA. TSO/E Clists and REXX execs can also have their objects cached in VLF using the TSO class IKJEXEC.

A group of related objects registered with VLF as a class (for example, LLA uses class name CSVLLA), and information about the class characteristics, is made known to VLF via member COFVLFxx. For each class the parameter MAXVIRT controls the maximum size available, up to the maximum data space size (2 GB).

Tip: LLA/VLF normally caches modules for PDSE program objects and Clists/REXX, so the usual recommendation is that these data sets not be put in a storage class that is eligible for member caching in Hiperspace.

Objects cached via SMSPDSE will be available only while the data set is open on that system; they will be purged when the final close is done.

PDSE code does not use VLF or LLA functions but uses its own mechanism to control its own data space and Hiperspace in which it stores the directories and members. PDSE BMF (Buffer Manager) caches the members based on the MSR value of the storage class.

LLA and VLF functional description

All LLA-managed libraries are eligible to be staged to VLF. LLA determines which modules of these libraries should be staged based on module statistics it keeps for all LLA-managed libraries. Staging evaluation runs asynchronously in the LLA address space and builds a staging candidate list. While building this list, staging calls the CSVLLIX2 exit for each module that it is analyzing (it only analyzes modules that have fetches against it). CSVLLIX2 can modify staging value information through its parameter list and return and reason codes. LLA sorts the candidate list by value and determines that a subset of the list should be staged. The whole list is not staged at once because we do not want to overload staging processing and I/O to library. The remainder of the list will be analyzed the next time staging runs for the library.

LLA then calls VLF to create an object for each module. If the VLF object create fails because storage is not available, VLF will trim the virtual storage in the data space to 90% of MAXVIRT minus the size of the object that did not fit. The next time staging runs, the same module candidate list or one similar will be built again, and we will try again to fit into VLF.

Note that LLA also reevaluates those modules that have been staged to VLF, and if the fetch statistics show that the module is no longer a good VLF candidate, LLA will not call VLF to remove these modules but rather mark them inactive (internally to LLA) and allow them to age in VLF, so that VLF trim will eventually get rid of them. VLF does not delete inactive load modules right away. If the module later becomes a good VLF candidate, the VLF version can be reactivated without the expense of rereading the module from DASD and recreating the VLF object.

The end result of all this is that the popular modules are identified by LLA and are eventually staged to VLF. However, depending on how full VLF is, it may take some iterations with trim to get them in. VLF controls trimming exclusively and there is no way to influence it externally besides changing the size of MAXVIRT. VLF will initiate trim whenever the virtual storage of the data space exceeds 90% of the MAXVIRT defined for that class, and whenever a create fails due to insufficient storage. VLF will generally try to maintain 90% of MAXVIRT, and it is based on a least recently used (LRU) method of elimination.

The LLA staging value is the expected value of utilizing LLA fetch over program fetch. It has four components: response time, contention, storage, and user-defined. The staging algorithm runs on a library basis. Each LLA-managed library is represented internally in data structures. An LLA value calculation routine is called for each library whenever an internal flag is turned on to indicate that staging functions should occur.

This bit or flag is set when:

- That library has exceeded the threshold (2000) for fetches on a library basis or after the initial 10 pgm fetches of a module.
- The CSVLLIX1 exit indicates that staging should be run for the library.

When staging is posted, the counts and flags are reset.

PDSE program object conditioning for LLA and VLF performance

PDSE program objects may be purged from VLF while PDS load modules stay in VLF. The caching of program objects to VLF works the same as PDS load modules if the PDSE program objects are bound as FETCHOPT=(PACK,PRIME) rather than page-mode-loaded.

Large program objects do not necessarily have to be fully loaded before execution starts, as they can be loaded page by page. This is called *page mode loading*. Page mode loading does not take full advantage of VLF. If the module is bound/linked for non-page mode loading (also known as *move mode loading*), they will be managed by VLF the same as PDS load modules, but the large size may itself affect VLF operation.

The LLA internal staging algorithm is not designed to handle page-mode-loaded program objects, causing LLA to remove the staged pages of the program objects from VLF. PDSE program objects have to be bound/linkedited as FETCHOPT=(PACK,PRIME) in order to fully utilize the benefits of LLA and VLF when using PDSE program objects.

This situation has been observed for IMS modules that experience a much higher I/O rate if loaded from PDSE when being page loaded compared to being loaded from PDS. Refer to APAR II14026 for more information.

9.4.3 SMSPDSE and SMSPDSE1

SMSPDSE and SMSPDSE1 are the two PDSE address spaces. They provide benefits when using PDSEs. Only the Hiperspace address space that belongs to the PDSE address spaces affects performance. There are special considerations during a restart of the SMSPDSE1 address space. See 3.2, “Restartable address space considerations” on page 41 for details.

9.4.4 IEBCOPY performance considerations

This section has general considerations for using IEBCOPY and PDSE, as well as tests run in our environment.

Copying from PDSE to PDSE

Copying from PDSE to PDSE can take longer than a copy from PDS to PDS. To determine whether you are experiencing a real performance problem, considerations include:

- Processing of the internal directory structures of a PDSE may require additional overhead when a PDSE-to-PDSE operation occurs.
- If you have a PDSE with a large quantity of *variable-length record* members, and those members are of a large size, then you could see a severe performance degradation in comparison to a PDS-to-PDS copy operation.
- Some physical characteristics of a PDSE could cause an undesirable performance impact (for example, internal data set fragmentation).

  PDSE space allocation occurs randomly within the PDSE. PDS data set members space is not random, and the deleted space is not reclaimed either. So, you can understand that a member whose space is fragmented will take more time to copy than a member whose space is contiguous.
- Another physical characteristic is data set extents.

  If a PDSE data set is in multiple extents, then because of the random nature of the internal allocation, it will take multiple channel programs to read from a PDSE or write to a PDSE. Therefore, the best-performing PDSE would be a new PDSE, whose allocation is within one single extent.

See 6.3.14, “IEBCOPY to reorganize a data set” on page 141 and 6.1.6, “DFSMSdss PDSE content reorganization” on page 121.

Copying from PDSE to PDS or from PDS to PDSE

In some cases, you may see poor performance using IEBCOPY from PDSE to PDS. This is documented as a known problem in APAR OA06701. For improved performance for the PDSE-to-PDS copy, the FAMS component of DFSMS requires a redesign of its code, but the size of the redesign would exceed the scope of an APAR fix. This APAR has been closed as a suggestion for future improvement.

Important: This applies mainly to large PDSEs and PDSs.

Currently, the most efficient way to use IEBCOPY to copy data members between PDSE and PDS data sets is a two-step process:

1. Use IEBCOPY UNLOAD to copy selected members or the entire PDSE or PDS to a sequential file.
2. Use IEBCOPY LOAD to copy these members or data set into a PDS or PDSE.

9.5 PDSE use of processor storage

This section describes how PDSE member and directory pages are cached in processor storage using z/OS facilities.

Table 9-2 describes the caching options for various PDSEs.

Table 9-2 PDSE comparison

	LLA	VLF	HSP	Data space
PDSE program object, SMS managed	Yes - via CSVLLAxx	Yes	See APAR OA08991, OA06884 - via storage class	Directory pages, as they are read
PDSE SMS-managed data members	No	Yes (clist and REXX)	Member caching, based on MSR (at open time)	Directory pages, as they are read
PDSE program object, non-SMS managed	Yes	Yes	No	Directory pages, as they are read
PDSE non-SMS managed data members	No	No	No	Directory pages, as they are read

9.5.1 PDSE member caching in Hiperspace

PDSE member caching is handled by BMF, and pages are placed into the SYSBMFHS Hiperspace.

Note: For non–LLA managed libraries, members will be cached only while the data set is open on a system.

In this example, program A opens the data set, loads a program from it, then closes the data set; then program B opens the data set and loads a program from it. Program B will not get the a program from the cache because the close from program A causes the member data in the cache to be purged.

Note that member pages may be stolen from the Hiperspace if real storage is overcommitted. PDSE code re-reads the page from DASD and repopulates it into the Hiperspace.

Member pages are purged from the Hiperspace if they “age out.” Time before age out is based on the LRU_TIME and LRU_CYCLES. Refer to 3.7, “Hiperspace considerations” on page 47 for information about setting these values.

The DFSMSdfp buffer manager caches the member pages of high-performance PDSEs in a shared Hiperspace, which is created as part of the DFSMS subsystem initialization. The Hiperspace is owned by the SMSPDSE or SMSPDSE1 address space.

A Hiperspace is a z/OS storage area that can contain up to 2 GB of contiguous virtual storage addresses. Your program can use a Hiperspace as a buffer for storing data, but you cannot run program instructions in it. To manipulate the data in a Hiperspace, use z/OS macros to move data between the Hiperspace and an address space in 4 KB pages.

When the DFSMSdfp buffer manager processes a request to read a PDSE member page that is eligible for caching into a user’s work area, it first checks the Hiperspace to see whether it has the page. If the buffer manager finds it, the page is copied into the user’s work area from the Hiperspace, and a physical I/O is avoided. If the buffer manager does not find the page, it retrieves the page from disk and copies it to the Hiperspace after reading it into the user’s work area. This process is shown in Figure 9-3.

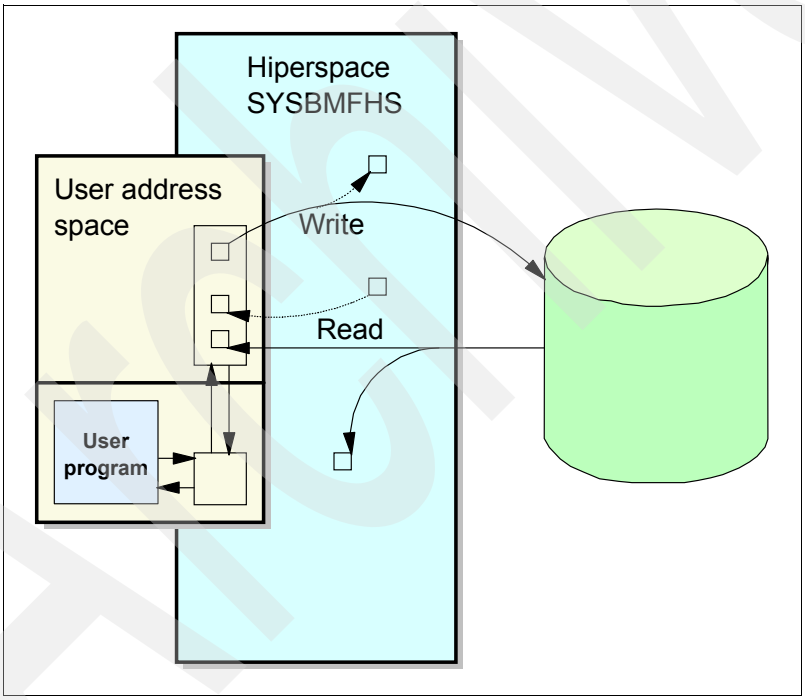


Figure 9-3 PDSE member caching in Hiperspace

When the buffer manager writes a PDSE member page that is eligible for caching to disk, it copies it to the Hiperspace when it writes it. In this way, the copy in Hiperspace is always a current copy, and any updates to it are made simultaneously to the copy on disk.

The buffer manager uses a least recently used (LRU) algorithm to remove the oldest pages from the Hiperspace.

When more than one program has a PDSE open for input they can share the member pages in the Hiperspace. When the last program closes the data set, all the pages are purged from

the Hiperspace to ensure the integrity of the PDSE member pages in a shared environment. If the pages were retained in storage after all programs had closed the PDSE, they would become invalid if a program on another system then updated the PDSE.

The Hiperspace cache is best used for those PDSEs that will be widely shared. Putting such data sets in Hiperspace reduces disk I/Os.

Controls

The DFSMSdfp buffer manager uses Hiperspace for PDSEs with a low direct *Millisecond Response Time (MSR)*.

The SMS storage class defines the way that the disk subsystem is to handle operations on the data set. The required response time affects whether a PDSE data set will use the PDSE Hiperspaces. The performance objectives that can be specified include the MSR and Sequential Millisecond Response. The MSR controls whether modules from a PDSE in this class will be eligible for Hiperspace caching. MSR values of <9, <50, and 999 correspond to must cache, may cache, and never cache. Values between 10 and 999 will have effects that are dependent on the system load. Table 9-3 summarizes the recommended value for MSR.

Table 9-3 MSR recommended values for PDSE Storage Class

	Hiperspace cache	**Disk control unit cache**
MSR 1	USED	MUST
MSR 50 or blank	*not used*	MAY BE
MSR 999	*not used*	NEVER

See 3.8.4, “How to set up an SMS environment for PDSE” on page 55 for additional information.

PDSE members are always accessed by a sequential access method, they are accessed in direct mode, because they tend to be small, and the direct mode caching algorithms are more appropriate. Therefore, the direct MSR value determines the cache usage attributes for a PDSE.

As described above, this is a read cache; writes are always done to disk and copied to the Hiperspace. Choosing the HSP_SIZE value involves a trade-off between CPU usage and PDSE member I/O.

PDSE member caching works best for PDSEs which are kept open and have members that are read multiple times. PDSE member caching works worst in an OPEN-READ-CLOSE pattern since the member cache is purged on the last close on a system. In this case, the members would be read into the cache but purged from the cache without ever being retrieved for a subsequent read.

We recommend that most IT environments that are not CPU constrained specify a nonzero value for HSP_SIZE and control PDSE member caching on an individual data set basis via the MSR time parameter in the data set Storage Class.

You can use SMF records to monitor Hiperspace usage by PDSE. See 9.6, “Monitoring performance” on page 195 for additional information.

Benefits

These are the benefits of Hiperspace caching:

- The Hiperspace cache provides very good I/O performance for selected PDSEs by avoiding physical I/O.
- The Hiperspace cache makes good use of the large processor storage available in an z/OS system.
- The Hiperspace cache allows PDSE members to be shared better by different users and jobs.
- The overall system I/O load will be reduced and processor utilization can be driven higher.
- Control over selecting Hiperspace cache candidates is integrated with DFSMSdfp Dynamic Cache Management for storage controllers.

9.5.2 PDSE directory caching in data space

Directory caching is done in the SYSBMFDS data space. Directory pages are cached as they are read. This applies to all PDSEs.

For non-SMS managed PDSEs only directory caching will be done. However, non-SMS managed PDSE program object libraries can obtain similarly good performance via LLA/VLF.

PDSE directory pages are cached in shared data spaces that are is created as part of the SMS initialization. This data spaces are owned by the SMSPDSE or SMSPDSE1 address spaces. PDSE directory pages are always cached regardless of the storage class of the PDSE.

A data space is a z/OS storage area that contains up to 2 GB of contiguous virtual storage addresses. Programs running in a user address space can access data stored in data spaces directly, without having to copy that data into the user's private area. This allows a user program to access large amounts of data without filling up its own virtual storage. Unlike an address space, a data space is used only for storing data; you cannot execute program instructions in a data space.

When a program opens a PDSE, DFSMSdfp reads the first five directory pages into the data space. The attribute and name directories (AD and ND) are implemented as B-trees (a balanced tree where balancing minimizes the tree depth and so speeds up searching) and they are index structured. See 2.5, "PDSE physical components" on page 19 for more information about the directory and index structure.

Directory searches are then done in storage. If the PDSE has more than five directory pages, additional I/Os will be needed to read them. At the time the PDSE is closed (last close on a system), the directory pages are purged to ensure integrity in case of updates from a second system, as data spaces cannot be shared across systems.

Directory pages for PDSEs can be shared by multiple programs. Figure 9-4 on page 192 illustrates how PDSE directories are cached in a data space.

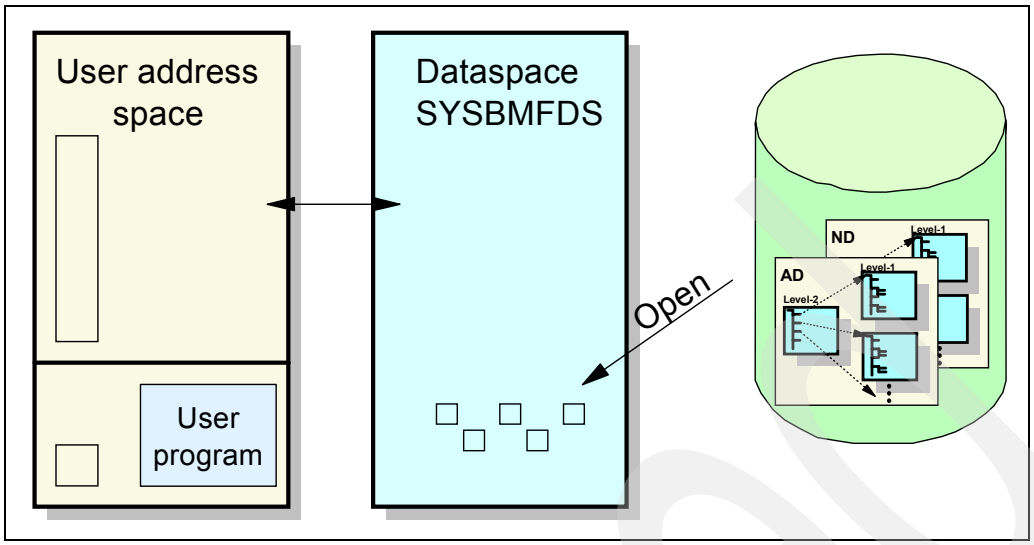


Figure 9-4 PDSE directory caching in a data space

The first OPEN reads the first directory pages into the data space. BLDL, FIND, and STOW are then executed against the directory pages in the data space. This is done by the code in BPAM that maps the BLDL, FIND, and STOW macros for PDSEs. When the last program sharing a PDSE issues a CLOSE, all of the directory pages are purged from the data space. The user's I/O buffer and the DWA, used for buffering PDSE members, are not used for the directory.

Reads to PDSE directories send disk controllers a request to inhibit loading of the disk controller cache.

Note: For non-LLA managed libraries, the directory pages are cached only while the data set is open on a system.

Assume that program A opens the data set, loads data from it, then closes the data set, and then program B opens the data set and loads data from it. Program B will not get the data from the cache because the close from program A causes the directory data in the cache to be purged.

Controls

If demand for storage used by the data space increases, DFSMSdfp uses an LRU algorithm to discard the oldest directory pages. This LRU processing is invoked more or less frequently depending on the setting of LRUTIME. See 3.1.3, "PARMLIB optional IGDSMSxx changes for SMSPDSE support" on page 35.

There are no other external parameters to control the caching of PDSE directories.

Benefits

The benefits of caching PDSE directories include:

- PDSE directory caching reduces directory search time. This is especially important for large or widely shared data sets. Frequently referenced directories benefit most, as they tend to stay in the data space.
- Accessing the directories in the data space provides virtual storage constraint relief (VSCR) by reducing the virtual storage required for I/O buffers in the user's private area.

- PDSE directory caching is dynamically managed and demand-driven. System programmers rarely need to set up or tune the directory-caching function.

9.5.3 Pending delete considerations

Operations on very large PDSEs occasionally have shown excessive storage use. Often the reason for this is that the most effective interfaces to system services have not been used.

BLDL is the service used to build a list of directory entries for a PDS or a PDSE. If issued against a PDSE, BLDL will connect to each member and the connection (and associated control blocks) will remain until the PDSE is closed. BLDL offers a NOCONNECT option to prevent this. This is because the application may use the TTR returned by BLDL to reference the member. The connection keeps the member from being physically deleted. It is considered to be “in use” by this application.

Similarly, the STOW macro also provides a DISCONNECT function for PDSEs.

If an application does not exploit BLDL NOCONNECT or STOW DISCONNECT, it may keep unnecessary connections to quite a few members, and each connection consumes storage.

This may result in one of the following problems:

- Common storage shortage (ABEND 878 or ABEND 80A)
- Space constraints on the volume (ABEND x37)
- PDSE system abends (ABEND 0F4)
- PDSE or PDSEs no longer accessible
- Performance impact to access to PDSEs, which might involve important subsystems or applications
- IPL

When a member is deleted but is still in use elsewhere, a pending delete condition arises where the name for the member is deleted only from the name directory (ND). Another record is created in the attribute directory (AD) to enable deletion of the member later, hence the return of allocated space. The data and the attributes of the member, as well as many control blocks, are kept in storage and on the disk volume.

When programs issue a BLDL with a very large member list against a PDSE data set, connections to each member are established and PDSE control blocks consume large amounts of storage. Programs should be changed to issue a BLDL NOCONNECT to process a large member list, and if the data set is a PDSE, issue a FIND against those members that need to be updated.

Note: PDSE serialization is not established when BLDL is used with the NOCONNECT option. The PDSE control blocks are the critical element that PDSE serialization code uses to manage sharing of PDSE data sets and members between address spaces. Without these control blocks, the user would have to be responsible for serialization of PDSE members and data sets. This restriction might prevent your program from using BLDL NOCONNECT.

Pending delete cleanup processing

Obviously, members in pending delete state should be deleted if they are no longer connected to a task. This is done by cleanup processing, which is part of OPEN processing for a PDSE data set. If an OPEN for output request is processed, PDSE services check to see

whether other connections for output exist. If not, then *delete pending destroy* processing is initiated. It releases the space on disk for members in pending delete state that are no longer connected to a task.

The control block structures for a connection to an individual member of a PDSE will be released when the last connection for that member is dropped.

Important: Delete pending cleanup processing is called only when the connection to the PDSE directory is the first and only open connect for output on the current system. Note that only members that are not currently connected on any system can be deleted.

Identifying pending delete

You may wonder how you can identify that a PDSE contains members in pending delete state.You might recognize an unusually high value in the current utilization field for a PDSE as shown in the ISPF/PDF Data Set Information panel (Figure 9-5). This could be an indication that many pages belong to members in pending delete state.

```
                          Data Set Information
Command ===>

Data Set Name  . . . : ONTOP.PDSE.PEND.DELETE

General Data                          Current Allocation
 Management class . . : ONTOP          Allocated cylinders : 30
 Storage class  . . . : ONTOP          Allocated extents . : 1
  Volume serial . . . : NC6E5E         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . :
  Organization  . . . : PO            Current Utilization
  Record format . . . : FB             Used pages  . . . . : 5372
  Record length . . . : 80             % Utilized  . . . . : 99
  Block size  . . . . : 32720          Number of members . : 15
```

Figure 9-5 Data set information for a PDSE

You can perform these steps to force pending delete cleanup processing on any system:

1. Stop all application programs that are holding the connections to the PDSE data set on that system. This closes the PDSE, and the connections from the application to the members of the PDSE are released during close processing.
2. Create a dummy member in the associated PDSE (OPEN for OUTPUT) by using ISPF EDIT or IEBCOPY to initiate the cleanup processing.

Afterwards, the current utilization values in the ISPF/PDF Data Set Information panel should be lower (Figure 9-6 on page 195).

```
                          Data Set Information
Command ===>

Data Set Name  . . . : ONTOP.PDSE.PEND.DELETE

General Data                          Current Allocation
 Management class . . : ONTOP          Allocated cylinders : 30
 Storage class  . . . : ONTOP          Allocated extents . : 1
  Volume serial . . . : NC6E5A         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . :
  Organization  . . . : PO            Current Utilization
  Record format . . . : FB             Used pages  . . . . : 362
  Record length . . . : 80             % Utilized  . . . . : 6
  Block size  . . . . : 32720          Number of members . : 16
```

Figure 9-6 Data set information for a PDSE

9.6 Monitoring performance

This section discusses the performance information provided for PDSEs and explains how to use the information to monitor PDSE performance.

9.6.1 SMF

The following SMF records contain PDSE information:

- SMF record type 14 and record type 15

  These records are written when the data set is opened or closed, and they indicate whether the data set is a PDSE. They contain the names of the data class, storage class, and management class.

- SMF record type 42, subtypes 1, 2, and 6.

  These records are written based on the specification of the IGDSMSxx member in SYS1.PARMLIB.

  Subtype 1 shows the statistics of the buffer manager on a storage class basis.

  Subtype 2 shows the statistics of the storage controller on an SMS-managed volume basis.

  Subtype 6 shows data set level I/O statistics. The records indicate whether the data set is a PDSE and contain information about the data set (name, volume serial number, storage class) and performance values and counters.

Note: APAR OA10091 should be applied to all z/OS V1R6 systems to fix a PDSE SMS recording error.

9.6.2 RMF

There is no special support for PDSEs in any RMF™ report, but there are several places where the effects of PDSEs may be observed.

Monitor I

The DASD device activity report provides feedback on the I/O rates and response times for each individual disk device and for each SMS storage group. It does not report the I/O rate or response times for individual data sets or SMS storage classes.

For those devices and storage groups used mainly for PDSEs, the I/O rate and the connect time to each device are reduced. The I/O rate is reduced because PDSE directories and members are cached in processor storage. The connect time is shorter because all directory searches for PDSEs are done in processor storage.

Monitors II and III

Monitor II or Monitor III can be used to observe how much expanded and central processor storage is being used for PDSEs.

In Monitor II, this information is presented in the Address Space State Data report. Processor storage used for PDSEs is reported under the name of the address space that owns the data spaces and Hiperspaces. These address spaces are SMSPDSE and SMSPDSE1.

In Monitor III, the same information can be found in the Storage Usage by Frames report. The default keywords for this are SF and STORF.

9.6.3 Block I/O counts

For data sets such as PDSs, which are processed with the BSAM, QSAM, and BPAM access methods, the EXCP (Execute Channel Program) count recorded in SMF record types 14, 15, and 30 is a count of the physical blocks of data read and written.

The EXCP count for a PDSE differs from the count for a corresponding PDS for both members and the directory.

When processing a PDS member, the amount of data actually represented by the block count depends on the DCB BLKSIZE parameter. However, for a PDSE, the count is of physical 4 KB pages.

The PDSE page count is independent of the Hiperspace cache hit ratio. When DFSMSdfp reads a page into a user's DWA, it adds one to the SMF page count, regardless of whether it gets that page from disk or from the Hiperspace cache.

For a PDSE directory, DFSMSdfp adds one to the SMF count for each directory page accessed independently of how many physical I/Os were done. This tends to make the count higher than for a PDS. On the other hand, during a BSAM sequential read of the directory, DFSMSdfp adds only one to the count for each physical 4 KB page, instead of one for each 256-byte logical block.

When you open a PDSE, the first five directory pages are read into the data space, but DFSMSdfp does not include these in the SMF count.

9.6.4 Buffer management statistics

The DFSMSdfp buffer manager also writes statistics to show how well the buffers are being used. It keeps count of how many PDSE directory and member pages are read and how many of these reads are hits in the PDSE buffers. It also counts how many pages of processor storage are in use for PDSEs.

You can control the length of the interval during which the buffer manager accumulates statistics. This is done with the BMFTIME parameter of the IGDSMSxx member that you code in SYS1.PARMLIB. The default is to write the statistics once an hour. See *z/OS MVS Initialization and Tuning Reference*, SC28-1752, for details about how to specify the BMFTIME parameter.

The statistics are written in SMF record type 42, subtype 1. Each PDSE address space writes its own record. Record type 42 is the DFSMSdfp record. The buffer manager writes overall totals for the reporting interval and subtotals for the reporting interval by each SMS storage class.

Note: The subtotals are by SMS storage *class*, not by SMS storage group.

The format of the SMF record is fully described in *z/OS MVS System Management Facilities*, GC28-1783. To use these numbers they have to be extracted from the SMF dump data set and formatted into reports or graphs. This can be done either by a user-written program or by a general-use statistical analysis program such as the Tivoli Performance Reporter.

Directory data space cache hit ratio

The hit ratio for the directory cache in the data space reflects how well PDSEs are being shared and how large they are. If there are many shared PDSEs in the data space, the hit ratio tends to be higher. If a PDSE does not have more than the five directory pages read into the data space when the PDSE is opened, it shows a directory page hit ratio of 100%. As larger directories will need additional I/O, the hit ratio will be lower.

The LRUCYCLES and LRUTIME parameters in IGDSMSxx indirectly control the use of the data space by PDSE directories.

Member Hiperspace cache hit ratio

The Hiperspace cache hit statistics are given by each storage class and as a total for all storage classes. Storage administrators should monitor the hit ratio in each storage class to make sure that they are assigning that storage class to the right data sets.

When a read miss occurs, the 4 KB page is read from disk and copied to the Hiperspace, using extra CPU cycles. For each read miss counted in SMF by DFSMSdfp, a corresponding page copy is done by the CPU. However, when a page of a PDSE member is output or updated, that page is also copied to the Hiperspace.

In deciding whether to allow a PDSE to use the Hiperspace cache, bear in mind the CPU cost of updates to it and how widely it is shared. The more widely a PDSE is shared, the more advantage there is to keep it in the Hiperspace cache.

The HSP_SIZE and PDSE1_HSP_SIZE parameter in IGDSMSxx controls the use of the Hiperspace for PDSE members.

9.6.5 IDCAMS LISTDATA command output

The LISTDATA command shows the status and the statistics of a storage controller on a subsystem or volume basis. When a volume dominated by PDSEs shows a falling cache hit ratio, you may need to reorganize the PDSEs, as this is probably due to increased fragmentation of the PDSEs into multiple extents, and of the PDSE members within the PDSEs. It is a good idea to choose a trigger value.

Figure 9-7 shows JCL and sample output from LISTDATA.

```
//PDSERESY  JOB  ...
//S1  EXEC  PGM=IDCAMS
//VOLUME1   DD UNIT=3390,DISP=SHR,VOL=SER=SBOX26
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  LISTDATA COUNTS FILE(VOLUME1) DEVICE
------------------------------------------------------------
0                         2105 STORAGE CONTROL
0                       SUBSYSTEM COUNTERS REPORT
0                    VOLUME SBOX26  DEVICE ID X'12'
0                         SUBSYSTEM ID X'8900'
0                                    CHANNEL OPERATIONS
                    ......SEARCH/READ.....  ............WRITE............
                         TOTAL  CACHE READ       TOTAL      DASDFW CACHE WRITE
 REQUESTS
   NORMAL                35734       35272      340182      340182      340182
   SEQUENTIAL           207487      207445      931836      931835      931835
   CACHE FAST WRITE          0           0           0         N/A           0
0TOTALS                 243221      242717     1272018     1272017     1272017
-REQUESTS                    CHANNEL OPERATIONS
0  INHIBIT CACHE LOADING                    0
   BYPASS CACHE                             0
0TRANSFER OPERATIONS                 DASD/CACHE   CACHE/DASD
0  NORMAL                                   968       145151
   SEQUENTIAL                              7356          N/A
0DASD FAST WRITE RETRIES                      0
-DEVICE STATUS     CACHING:          ACTIVE
                   DASD FAST WRITE: ACTIVE
                   DUPLEX PAIR:     NOT ESTABLISHED
 RAID RANK ID X'0000'
1IDCAMS  SYSTEM SERVICES                                        TIME: 13:57:4
0IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
0
0IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

Figure 9-7 LISTADATA JCL and sample output

RMF calculates and reports various values related to control unit activities in the Cache Subsystem Activity report and in SMF type 74 subtype 5 records. You can use it to monitor the cache hit ratios for volumes and storage groups in which you store PDSEs.

9.6.6 Limiting PDSE usage of expanded storage

DFSMS/MVS 1.3 introduced the ability to limit the amount of expanded storage obtained by PDSE for caching PDSE members in a Hiperspace.

You may set a limit by using the HSP_SIZE parameter in the IGDSMSxx member of SYS1.PARMLIB. You may set the value to between 0 and 2047 MB.

If you do not set a limit, the storage limit that is used defaults to the limit used in earlier releases: It will be set to the lesser of half of the available expanded storage or 256 MB. For z/Architecture, the algorithm is as follows:

```
If AvSt > 1 GB then Limit = 256 MB
Else If AvSt < 256 MB then Limit=AvSt/8
Else Limit = AvSt/4
```

`AvSt` is Available Real Storage.

9.6.7 LLA and VLF usage display

Modules become managed by VLF only when they reach a threshold of use that means that they are likely to be used again. Modules that are accessed only once are unlikely to be managed by VLF.

For the description of and considerations about LLA and VLF, refer to Chapter 3, “Managing the PDSE environment” on page 33.

The activity of LLA and VLF can be displayed using the undocumented command:

```
DISPLAY LLA,STATS
```

The simple form (Figure 9-8) lists all of the data sets being managed by LLA.

```
D LLA,STATS
```

Figure 9-8 Format of the DISPLAY LLA,STATS command

Figure 9-9 shows the more detailed form. Note that there can be a large amount of data from this command, so care should be taken when selecting the operands. The LIBRARY specification can list any LLA managed data set, and a member can be specified using the asterisk wildcard (*). Where possible, the MEMBER selection should be specific.

```
D LLA,STATS,LIBRARY=yourdsn,MEMBER=member,FETCHED
```

Figure 9-9 Format of the extended form of the DISPLAY LLA,STATS command

Attention: The DISPLAY LLA,STATS command, in either form, is an undocumented command, so either format may not be provided in future releases, and the format of the command and content of the output may change without notice by maintenance or future system releases.

Figure 9-10 on page 200 shows an example of issuing the command (with edited response). From this display, the data sets being managed by LLA and which have modules being managed by VLF can be seen. Note the term LIBRARY as used in the command output does not refer to only PDSE data sets. PDSERES.LOADED.PDS is a user data set that was added to the LLA after IPL.

```
D LLA,STATS
CSV620I 12.03.37 LLA STATS DISPLAY 819
TOTAL DASD FETCHES: 845  TOTAL VLF RETRIEVES: 5212
94 LIBRARY ENTRIES FOLLOW
LIBRARY: EQQ.SEQQLMD0
  MEMBERS:                619
  MEMBERS FETCHED:          0  MEMBERS IN VLF:          0
  DASD FETCHES:             0  VLF RETRIEVES:           0
LIBRARY: TCPIP.SEZALNK2
  MEMBERS:                  2
  MEMBERS FETCHED:          0  MEMBERS IN VLF:          0
  DASD FETCHES:             0  VLF RETRIEVES:           0
LIBRARY: FPE.V2R1M0.SFPELINK
  MEMBERS:                  5
  MEMBERS FETCHED:          0  MEMBERS IN VLF:          0
  DASD FETCHES:             0  VLF RETRIEVES:          0
.
.
LIBRARY: PDSERES.LOADED.PDS
  MEMBERS:                  1
  MEMBERS FETCHED:          1  MEMBERS IN VLF:          1
  DASD FETCHES:             0  VLF RETRIEVES:       1975
LIBRARY: SCRIPT.R40.DCFLOAD
  MEMBERS:                 45
  MEMBERS FETCHED:          0  MEMBERS IN VLF:          0
  DASD FETCHES:             0  VLF RETRIEVES:          0
```

Figure 9-10 Example of D LLA,STATS command with parts of the response

An example of using the extended DISPLAY LLA,STATS command in controlled form for specific information about the member IKJLDI00 of SYS1.LINKLIB (a PDS data set but the format is identical for PDSE data sets), with response shown in Figure 9-11. The command output shows that the member is managed by VLF and that 12 retrieves have been from VLF.

```
D LLA,STATS,LIBRARY=SYS1.LINKLIB,MEMBER=IKJLDI00,FETCHED
CSV630I 13.09.30 LLA STATS DISPLAY 975
LIBRARY: SYS1.LINKLIB
MEMBERS: 1  MEMBERS FETCHED: 1  MEMBERS IN VLF: 1
TOTAL DASD FETCHES: 0  TOTAL VLF RETRIEVES: 12
MEMBER: IKJLDI00
  DASD FETCHES:          0  VLF RETRIEVES:         12
  AVERAGE:          7.2500
```

Figure 9-11 Example of extended D LLA,STATS co*mmand with response*

9.7 Flow chart to run performance issue analysis

This section includes two flow charts. Figure 9-12 on page 201 may be used for analysis of performance issues *copying* PDSE data sets. Figure 9-13 on page 201 may be used for analysis of performance issues *accessing* a PDSE.

These provide an overview of the information contained in this book, after you have read through this entire chapter.

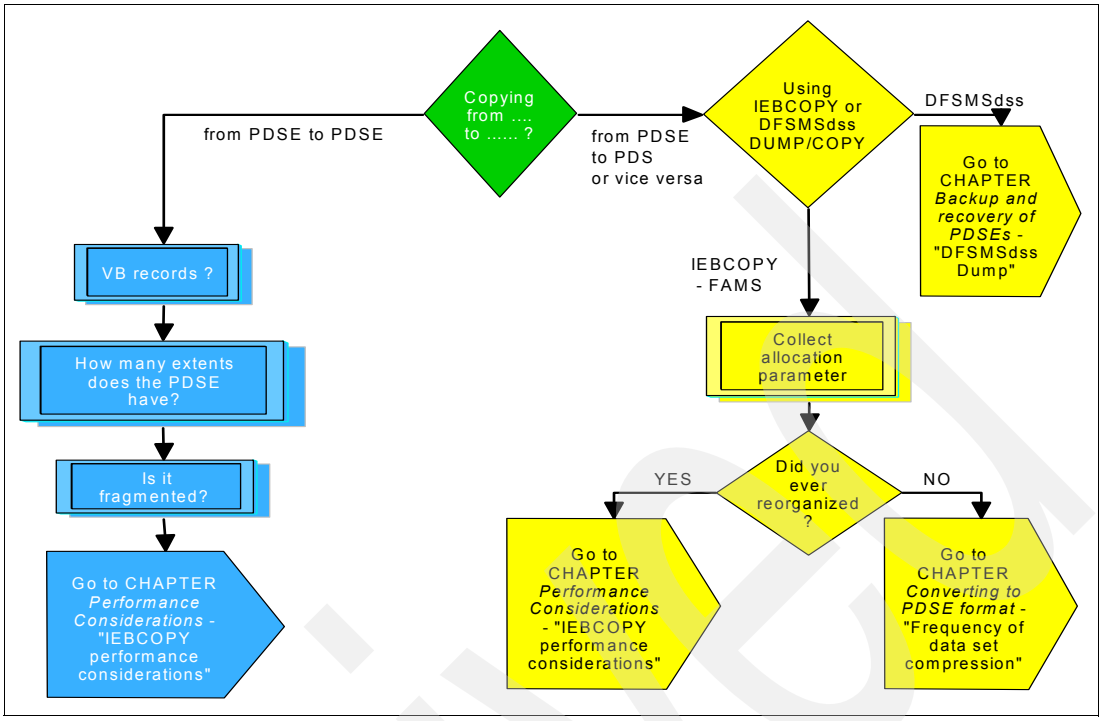


Figure 9-12 Flow-chart for performance issue COPYING PDSEs

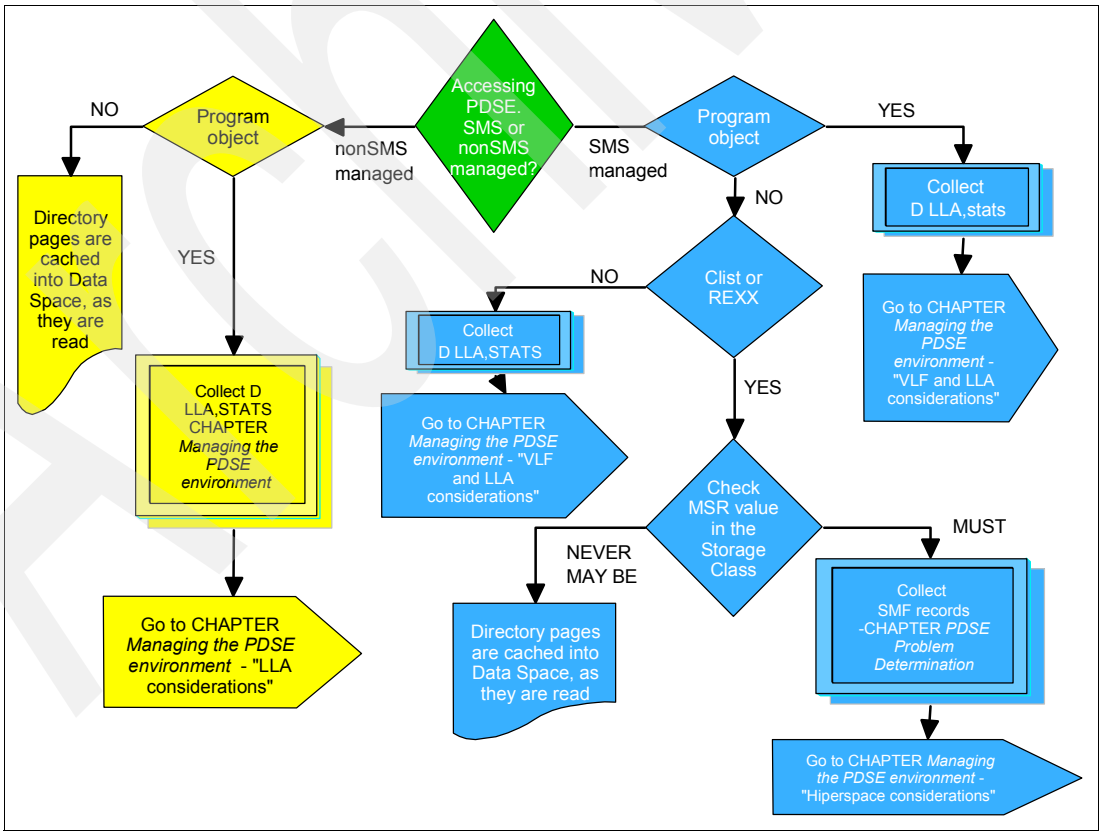


Figure 9-13 Flow-chart for performance issues ACCESSING PDSEs

9.8 PDSE buffer management statistics

SMF records data that may be useful in determining how the PDSE buffer management function (BMF) is performing, and may enable values to be tuned to improve performance or reduce overcommitment of real storage. BMF is responsible for managing access to and from the PDSE Hiperspace, and other than through SMF there is no way to determine whether the Hiperspace is being used.

SMF record type 42, subtype 1 provides overall buffer use totals for SMSPDSE and, if implemented, separately for SMSPDSE1, and totals for each storage class being used by the SMSPDSE and SMSPDSE1 address spaces and their Hiperspaces.

9.8.1 BMF data capture preparation

In order to have SMF capture BMF data, the following items must be addressed:

- PDSE_BMFTIME and, if SMSPDSE1 is in use, PDSE1_BMFTIME must be set to appropriate values. The default setting of 3600 seconds (1 hour) may be too long for meaningful analysis. See PDSE_BMFTIME in 3.1.3, “PARMLIB optional IGDSMSxx changes for SMSPDSE support” on page 35 and PDSE1_BMFTIME in 3.1.4, “PARMLIB IGDSMSxx parameters for SMSPDSE1 support” on page 38 respectively. Any change to PDSE_BMFTIME or PDSE1_BMFTIME by use of the SETSMS command takes place at the end of the previously set interval.
- If no PDSE eligible activity occurs, no BMF data is captured because Hiperspace cache is not used. After eligible PDSE data sets are accessed, the BMF data recording function starts, and continues even after the triggering PDSE is no longer in use.
- If APAR OA09162 (see “OA09162” on page 331) is installed, then BMF will not be using Hiperspace to cache program objects unless the PDSE is associated with a storage class that contains a low MSR value. Most LINKLIST managed data sets are not SMS managed and so are not associated with an eligible storage class. If you wish to see the BMF activity associated with the LINKLIST, then you must add an SMS-managed PDSE to the LINKLIST, ensuring that it is associated with a storage class that is eligible for Hiperspace cache use. However, note that there will be no data recorded for the directory activity for the non-managed PDSEs.
- The SMF recording process may not initialize correctly on all systems. This situation will be indicated if no SMF type 42 records are being recorded even when eligible PDSE data sets are in use and the BMFTIME values are appropriately set. If this condition is occurring, install the correcting fix when available. (See “OA10091” on page 333.)
- There is no eye catcher in the data to distinguish BMF data associated with SMSPDSE from that associated with SMSPDSE1. However, as each address space has its own LRUCYCLES definition, making these different provided a way of inferring which is which. The default setting for SMSPDSE is 15 and for SMSPDSE1 is 50. Note that LRUCYCLES is one of the variables that may have to be adjusted to limit CPU or real storage use, so if they are changed, be sure to maintain some difference between the two.

9.8.2 BMF analysis preparation

This section provides a sample program that formats the information recorded by SMF. The program must be assembled and linked into an appropriate data set.

In order to relate the program output to the SMF manuals and other sources of information, the field names as defined in the SMF macros are used.

Figure 9-14 on page 203 shows the SMF records that are collected from the BMF totals.

```
SMF42O1A DSECT ,
SMF42BMF DS    0CL0036   Description of BMF totals section
SMF42TNA DS    1FL4      Total number of storage classes
SMF42TMT DS    1FL4      Interval length (total time of measurement
                         period)
SMF42TRT DS    1FL4      Total number of member data page reads
SMF42TRH DS    1FL4      Total number of member data page read hits
                         (found in BMF)
SMF42TDT DS    1FL4      Total number of directory data page reads
SMF42TDH DS    1FL4      Total number of directory data page read hits
                         (found in BMF)
SMF42BUF DS    1FL4      Total number of active BMF 4K buffers   @18A
SMF42BMX DS    1FL4      High water mark of BMF buffers          @18A
SMF42LRU DS    1FL2      BMF LRU interval time                   @18A
SMF42UIC DS    1FL2      BMF LRU cycles before buffer cast out   @18A
```

Figure 9-14 Extract of the BMF SMF data fields used

Figure 9-15 shows the data collected for the storage classes for the same SMF record.

```
SMF42O1B DSECT ,
SMF42SC  DS    0CL0048   Description of Stor Class Sum section @OW00573
SMF42PNA DS    0CL0032   Storage class name
SMF42PNL DS    1FL2      Storage class name length
SMF42PNN DS    1CL0030   Storage class name
SMF42SRT DS    1FL4      Total number of member data page reads
SMF42SRH DS    1FL4      Total number of member data page read hits    X
                         (found in BMF)
SMF42SDT DS    1FL4      Total number of directory data page reads
SMF42SDH DS    1FL4      Total number of directory data page read hits X
                         (found in BMF)
```

Figure 9-15 Extract of the BMF storage class fields used

Depending on the structure of the system, there may be one or more BMF totals sections, which are shown in the program output as BMF TOTALS SET #: *nnnnnnn* where nnnnnnn is a numeric value. There may be one or more storage class sections, identified by their names as defined to SMS. Normally, there is one set of SMF records for SMSPDSE and one for SMSPDSE1.

The extract program assumes that the data it is processing is an extract from the SMF data prepared by the SMF-supplied extraction program IFASMFDP. The SMFPRMxx member of SYS1.PARMLIB should be checked to ensure that SMF records type 42 is being selected or that it is not being suppressed.

The extract may be taken from the running copy of the SYS1.MANx VSAM data set or from a previously extracted sequential data set. IFASMFDP manages the correct handing of the input data set and regardless of the form of the input produces a sequential data set. In order to limit the amount of data extracted, the recommended control statement input to IFASMFDP limits the output to select only records of type 42 subtype 1. Any records that are not type 42 subtype 1 are discarded by the SMF42T1 program.

Listings of the program source and examples of JCL to assemble and link the program are shown in “SMF record type 42 subtype 1 data” on page 318. Assembly is required once for the initial implementation at a given level of the operating system, and again if maintenance is applied to the IGWSMF macro.

Figure 9-16 shows an example of the job required to extract the SMF records and run the formatting program. This assumes that the formatting program is called SMF42T1 and that it has previously been assembled and stored in the data set PDSERES.SMF42T1J.PDS. The IFASMFDP selection specification is part of the OUTDD statement OUTDD(OUTDD,TYPE(42(1))).

```
//MHLRES1S JOB (999,POK),MSGLEVEL=1,NOTIFY=MHLRES1
//EXTRACT EXEC PGM=IFASMFDP
//SYSPRINT DD SYSOUT=A
//ADUPRINT DD SYSOUT=A
//*DUMPIN DD DISP=SHR,DSN=SMFDATA.ALLRECS
//DUMPIN DD DISP=SHR,DSN=SYS1.SC64.MAN1
//DUMPOUT DD DUMMY
//OUTDD   DD DSN=&T1,SPACE=(CYL,(10,5)),RECFM=VB,LRECL=5096,
// DISP=(,CATLG,DELETE),
// UNIT=SYSDA
//SYSIN DD  *
          INDD(DUMPIN,OPTIONS(DUMP))
          OUTDD(OUTDD,TYPE(42(1)))
/*
//FORMAT  EXEC PGM=SMF42T1
//STEPLIB DD DISP=SHR,DSN=PDSERES.SMF42T1J.PDS
//SYSUDUMP DD SYSOUT=A
//SMFIN   DD DISP=SHR,DSN=&T1
//PRINT   DD SYSOUT=A,RECFM=UA
```

Figure 9-16 Example of JCL to format SMF record type 42 subtype 1 records

Figure 9-17 shows a portion of the output from the IFASMFDP program. The full output shows all records found. The right-most column shows the number of records selected. In this partial output example, no records were selected for SMF record types 41 and 43. There were 41 records found for the SMF type 42 record, which is consistent with the selection criteria specified on the OUTDD control statement.

```
IFA010I SMF DUMP PARAMETERS
IFA010I END(2400) -- DEFAULT
IFA010I START(0000) -- DEFAULT
IFA010I DATE(1900000,2099366) -- DEFAULT
IFA010I OUTDD(OUTDD,TYPE(42(1))) -- SYSIN
IFA010I INDD(DUMPIN,OPTIONS(DUMP)) -- SYSIN
IFA020I OUTDD    -- SYS04329.T172111.RA000.MHLRES1S.T1.H02
IFA020I DUMPIN   -- SMFDATA.ALLRECS.G7719V00

.
.
41           438           .25 %          286.13           146           332             0
42        21,931         12.67 %          529.70           180        32,628            41
43             2           .00 %           32.00            32            32             0 .
.
```

Figure 9-17 Example of part of the output from IFASMFDP showing selection of type 42 records

9.8.3 SMF statistics interpretation

There are no absolute values that are right or wrong for the tuning parameters. The SMF data is useful to determine what is happening and after a change to verify that the change had some effect.

Figure 9-18 shows an example of output from the SMF type 42 record formatting program. Note that the value for SMF42TNA of 0000001 corresponds to the one SCLASS entry that follows it. By checking the output from the D SMS,OPTIONS command, we infer from the value of 0001500 for SMF42LRU that this is data from SMSPDSE.

```
BMF TOTALS SET #:  0000001
HH:MM:SS YYYYDDD SMF42TNA SMF42TMT SMF42TRT SMF42TRH SMF42TDT SMF42TDH SMF42BUF SMF42BMX SMF42LRU SMF42UIC
15:12:36 2004338 0000001  0000899  0000000  0000000  0000002  0000000  0000013  0000122  0001500  0000240

SMF42PNN (SCLASS): SCTEST
HH:MM:SS YYYYDDD SMF42SRT SMF42SRH SMF42SDT SMF42SDH
15:12:36 2004338 0000000  0000000  0000002  0000000
```

Figure 9-18 Example 1 of output from program formatting SMF type 42 subtype 1 records

Figure 9-19 shows a further example of output from the SMF type 42 record formatting program. The value for SMF42TNA of 0000001 corresponds to the one SCLASS entry that follows it. By checking the output from the D SMS,OPTIONS command, we infer from the value of 0005000 for SMF42LRU that this is data from SMSPDSE1.

```
BMF TOTALS SET #:  0000001
HH:MM:SS YYYYDDD SMF42TNA SMF42TMT SMF42TRT SMF42TRH SMF42TDT SMF42TDH SMF42BUF SMF42BMX SMF42LRU SMF42UIC
15:17:36 2004338 0000001  0000599  0000000  0000000  0000055  0000044  0000685  0001731  0005000  0000200

SMF42PNN (SCLASS): SCTEST
HH:MM:SS YYYYDDD SMF42SRT SMF42SRH SMF42SDT SMF42SDH
15:17:36 2004338 0000000  0000000  0000055  0000044
```

Figure 9-19 Example 2 of output from program formatting SMF type 42 subtype 1 records

For reference, some of the reported data shows what the current settings are, and there would not be any change in this unless some specific action was taken, but the various counts can be expected to change.

For each of the PDSE data spaces in use there will be data for the BMF (buffer management processing) and for the processing as it relates to the storage classes involved.

The data that is recorded by SMF, and ultimately returned in the SMF type 42 subtype 1 records, is shown in the examples in Figure 9-14 on page 203 and Figure 9-15 on page 203.

Buffer Management Function (BMF) Totals data

The following comments are based on the examples shown in Figure 9-18 and Figure 9-19.

There is no specific data in the BMF Totals data section to indicate which data space it applies to, SMSPDSE or SMSPDSE1. In the SMF42T1 program output, the sections are identified in the output in the program header as:

```
BMF TOTALS SET #:  000000n
```

n will be replaced by the relative number of the BMF totals section found.

The default system settings for LRUCYCLES and BMFTIME are different for the SMSPDSE and SMSPDSE1 data sets, so as long as a difference is retained in one or the other of these, you can determine which one is for SMSPDSE and which is for SMSPDSE1. In the sample report in Figure 9-19 on page 205, you can see the values 0005000 and 000200 under the SMF42LRU and SMF42UIC columns respectively.

From the output of the D SMS,OPTIONS command output in Figure 9-20, you can see both the LRUCYCLES and BMFTIME values and from that determine which set of BMF totals is which. Refer to the SMF42LRU explanation below for comment on the apparent multiplication factor applied over what was specified in the IGDSMSxx parameters.

```
YYYYDDD SMF42TNA SMF42TMT SMF42TRT SMF42TRH SMF42TDT SMF42TDH SMF42BUF SMF42BMX SMF42LRU SMF42UIC
2004327 0000002  0003599  0000000  0000000  0000010  0000008  0000011  0001080  0005000  0000200
```

Figure 9-20 D SMS,OPTIONS command output

Note: The number assigned to the BMF totals heading represents the relative order that the data appears in the SMF type 42 subtype 1 records. For the life of a given IPL, it is likely that the order will remain the same, if there is more than one, but it is possible that after a subsequent IPL, the order will have changed.

- SMF42TNA - Total number of storage classes

  Number of storage classes defined in the system. This is useful to correlate the number of storage classes reported on in the Storage Class data report.

- SMF42TMT - Interval length for data gathering

  This is the value defined in the BMF_TIME fields and may not be what is being used because this value is documented as overridden by the SMF INTVAL if the SMS OPTION SMS_TIME is set to YES. This override is not in effect in z/OS V1.6.

- SMF42TRT - Total number of member data page reads

  This is the overall number of member data page reads, including those found in BMF.

- SMF42TRH - Total number of member data page reads found in BMF

  This is the number of member data page reads satisfied from BMF caching. When BMF caching is functioning in support of a PDSE, we would expect to see that SMF42TRH is not zero, and that it is less than SMF42TRT. Whether member data is cached by BMF depends on the MSR value specified in the storage class associated with a particular data set being low.

- SMF42TDT - Total number of directory data page reads

  This is the overall number of directory data page reads, including those found in BMF caching.

- SMF42TDH - Total number of directory data page reads found in BMF caching.

  This is the number of directory data page reads satisfied from BMF caching. When BMF caching is functioning in support of a PDSE, we would expect to see that SMF42TDH is not zero, and that it is less than SMF42TDT. Directory data page reads should be managed by BMF whether or not the data set is associated with a storage class.

- SMF42BUF - Total number of active BMF buffers

  This number should correspond to the number of 4K pages currently being used.

- SMF42BMX - High-water mark of BMF buffers

  This number represents the largest number of BMF buffers used from the total number of available BMF buffers. If the number presented here is consistently significantly lower that the number of available buffers, this may be an indication that the HSP_SIZE can safely be reduced. Likewise, if the number presented here is consistently close to the total number of available buffers, it may be necessary to consider increasing the HSP_SIZE. However, the number of buffers in use is influenced by the values used for LRUCYCLES and LRUTIME, and it may be that adjusting these should be done before considering changing HSP_SIZE as that requires a system IPL to implement.

- SMF42LRU - BMF LRU interval time

  This value corresponds to the PDSE[1]_LRUTIME value as specified or as defaulted in the IGMSMSxx options member, or as subsequently altered by the SETSMS command. Note that the value as presented is shown multiplied by 100.

- SMF42UIC - BMF LRU cycles before buffer cast out

  This value corresponds to the PDSE[1]_LRUCYCLES value as specified in the SYS1.PARMLIB IGSSMSxx options member, or as subsequently altered by SETSMS command. LRUTIME is not an actual time value, but an indication of how many times a buffer may remain in the buffer but not allocated when buffer management cycle runs, which is controlled by LRUCYCLES.

  Attention: The higher LRUCYCLES is set, the more CPU time is used running the BMF process; the lower LRUTIME is set, the more real storage remains allocated. Tuning LRUCYCLES and LRUTIME for each of the SMSPDSE and SMSPDSE1 address spaces can be carried out without a system IPL. Any changes should be carefully monitored in conjunction with measurements of the committed CPU and real storage frames. Any sudden increase in CPU or REAL storage commitment may reflect the introduction of a PDSE into the system.

Storage Class summary data

There is one of these sections for each storage class in the system related to each of the SMSPDSE or SMSPDSE1 address spaces. For ease of interpretation, Storage Classes intended to manage the PDSE buffer management processes should be named according to a convention that indicates that they are intended for PDSE data set management, and indicate whether the management intended was to have the PDSE cached in the PDSE Hiperspace (or Hiperspaces).

- SMF42PNL - Storage class data section name length (reference information)
- SMF42PNN - Storage class name (reference information)
- SMF42SRT - Total number of member data page reads

  This is the overall number of member pages read.

- SMF42SRH - Total number of member data pages read from BMFcache.

  This is the number of member directory page reads supplied from the BMF cache.

  When a PDSE associated with this storage class is being serviced by BMF cache, the SMF42SRH value should not be zero, but it will be less than the total number of member data reads. This will show zero if the storage class does not have a low MSR value.

- SMF42SDT - Total number of directory data page reads

  This is the total number of directory pages read using this storage class.

- SMF42SDH - Total number of directory data page reads serviced by BMF

  This is the total number of directory data pages read that were found in the BMF buffers. Directory pages are eligible for BMF processing whether or not the PDSE is associated with an eligible storage class.

  The value in SMF42SDH should not be zero and will likely be less than the value in SMF42SDT if the BMF function is managing the data to and from Hiperspace.

Recommendation for further analysis

If the SMF42SRH results are significantly lower than SMF42SRT, it is likely to be an indication that for some reason few PDSE data sets are being cached to Hiperspace. If the PDSE data sets in use are being used by single or very few users at a time, then there may be no concern. However, if a PDSE is intended to be shared, then the benefits of using Hiperspace should be investigated.

One comparatively simple way to measure whether a particular data set comes into that category is to assign it its own storage class for test purposes. That will require the creation of the storage class if not already available, and probably require an adjustment to the ACS routines to ensure that the data set is assigned, by itself, to the test storage class. When implemented, the SMF type 42 subtype 1 records will automatically pick up the additional storage class, and the effect of changes to the MSR can be monitored.

When the data set is being managed by BMF caching, any specific changes made to improve the BMF member data hit ratio can then be fitted back into the original storage class.

10

PDSE problem determination

In this chapter, we give an overview of the tools and techniques that you can use to diagnose problems with PDSEs.

Attention: If you are searching IBM product support databases for information, you should use the format ABENDxxx with no space between ABEND and the numeric code. A space is used in this chapter to aid readability.

The following sections are contained in this chapter:

10.1 Overview diagnosis and restarting

Using PDSE Restartable Address Space, the customer can resolve certain system hang conditions without performing another IPL on the system.

The following events indicate a PDSE problem that might require a PDSE restart:

- PDSE is not responding.
- Operator suspects a PDSE serialization problem (for example, latch or lock).
  - PDSE Analyze does not help.
  - CANCEL does not help.
  - PDSE Free Latch command does not help.

"General approach to restarting a PDSE address space" on page 214 provides a road map to restart a PDSE restartable address space, and "PDSE1 restart example" on page 218 provides a sample restart scenario.

In the past, it was necessary to re-IPL a system or systems if the partitioned data set extended (PDSE) address space caused a hang condition, a deadlock condition, or an out-of-storage condition.

With z/OS V1R6, DFSMSdfp has two PDSE address spaces available:

- SMSPDSE
- SMSPDSE1

The new SMSPDSE1 address space is restartable. With the PDSE address space restart feature, you may be able to:

- Recover from a PDSE problem without having to re-IPL a system or systems.
- Determine which systems require a re-IPL or repair to eliminate PDSE hangs and failures.

The existing PDSE address space, SMSPDSE, continues as a non-restartable address space for PDSEs that are in the LNKLST concatenation. It cannot be restarted because an address space cannot be created without using LLA global connections and therefore SMSPDSE. SMSPDSE is the only PDSE address space for the z/OS systems when the following conditions exist:

- The IGDSMSxx initialization parameter, PDSESHARING, is set to NORMAL.
- The IGDSMSxx initialization parameters in a sysplex coupled systems environment are set as follows:
  - PDSESHARING(EXTENDED)
  - PDSE_RESTARTABLE_AS(NO)

If these conditions do not apply, the z/OS system has both the existing PDSE address space, SMSPDSE, and the new PDSE address space, SMSPDSE1. The new SMSPDSE1 address space provides connections to and processes requests for those PDSEs that are not accessed through the LNKLST concatenation.

To ensure that the new PDSE address space, SMSPDSE1, is created during IPL in a sysplex coupled systems environment, set the IGDSMSxx initialization parameters as follows:

- PDSESHARING(EXTENDED)
- PDSE_RESTARTABLE_AS(YES)

See "PARMLIB requirements" on page 34 for more information about PDSE related IGDSMSxx parmlib parameters.

10.2 Overview of diagnosis tools

Several tools are available to help you to diagnose problems within the PDSE environment.

The following commands and utilities can help you in verifying locking and latching and the physical status of your PDSE environment:

- The VARY SMS,PDSE operator command has several options to help you determine whether there is a sharing problem, and which PDSE or PDSEs are involved. The command also includes tools to recover from certain problems.
  - The RESTART command recycles the restartable PDSE address space (SMSPDSE1). See “RESTART” on page 224 for more information.
  - The ACTIVATE command brings up the restartable PDSE address space. See “ACTIVATE” on page 227 for more information.
  - The ANALYSIS command detects a number of the most common latch and lock contention problems. See “ANALYSIS” on page 228 for more information.
  - The FREELATCH command releases any latch that the ANALYSIS command indicates is held. See “FREELATCH” on page 232 for more information.
  - The MONITOR command modifies the processing for the PDSE monitor. See “MONITOR” on page 233 for more information.
- The DISPLAY SMS,PDSE operator command has two options to display a PDSE latch and to get further information about modules loaded to the SMSPDSE or SMSPDSE1 address space.
  - The LATCH command displays the status of a PDSE latch for the SMSPDSE or SMSPDSE1 address space. See “LATCH” on page 237 for more information.
  - The MODULE command displays the address and maintenance level of module in the SMSPDSE or SMSPDSE1 address space. See “MODULE” on page 238 for more information.
- For sharing problems, the D GRS operator command can provide useful information about which PDSEs have outstanding enqueues. See “Interpreting output of D GRS commands” on page 239 for more information.
- D SMS,OPTIONS show the PDSE sharing mode that is currently in use. See “PDSE sharing status” on page 244 for more information.
- The IEBCOPY utility can be used in these ways:
  - To verify the integrity of a PDSE by copying the PDSE and all members into another PDSE.

    The copy process reads all members and the associated directory structures. Your PDSE should be good if the copy process completes with a condition code of zero.
  - To recover any index damage by copying the affected PDSE into a newly allocated PDSE.

    By doing this, you will re-create all PDSE index structures from scratch. However, depending on the kind of index damage, any access to the input PDSE might be impossible. In this case, you must recover the PDSE from your backup repository using a program such as DFSMShsm.
  - To defragment a PDSE by copying the PDSE into a newly allocated PDSE.

    PDSEs can become fragmented depending on the access pattern. This does not normally occur when the adding and deleting of members is balanced, but might occur when members are deleted and new members are not added to reclaim the space. To

reclaim the space and reorganize the PDSE, copy it to a new PDSE using IEBCOPY or DFSMSdss COPY.

See “IEBCOPY” on page 131 for more information about using IEBCOPY utility.

- The PDSE HFS Analysis Tool (*PHATOOL*; also know as PDSE inspection tool *PIT*)) is a service aid that operates as an application program. It can be used to check the integrity of the directory structure and to display the allocation of data set pages within a PDSE. See “PDSE and HFS Analysis Tool (PHATOOL)” on page 306 for more information.
- The **D XCF** operator command helps determine the status of systems currently participating in the sysplex, the XCF groups, and members of a particular group. Proper XCF communication is required to use extended sharing protocol. Otherwise, PDSE sharing is forced into normal mode rather than extended mode. See “XCF status” on page 242 for more information.
- The IEHLIST LISTVTOC utility formats the VTOC and the Format-1 DSCBs for a DASD volume or a single data set. See “IEHLIST utility” on page 313 for more information.

10.3 Diagnosis guidelines

Although PDSE errors are not common, when errors do occur in the code for PDSEs, the results often extend beyond the job or user who encountered the error. This is a result of the breadth of the PDSE function.

For example, PDSE-related processes are using control blocks in common storage, in the requester’s or user’s address spaces, the PDSE address, and PDSE data space. The execution of the majority of its code is under control of the user’s tasks (user address space <=> home address space). PDSE services are exploiters of Cross memory mode services. This means that most activities, behind a user request, are transferred to the SMSPDSE and SMSPDSE1 address spaces for execution (SMSPDSE(1) address space <=> primary address space). See *z/OS: MVS Programming: Extended Addressability Guide*, SA22-7614 and *z/Architecture: Principles of Operation*, SA22-7832 for information about *cross-memory mode* processing.

While we cannot give you a general rule about how to handle all potential situations that might occur in a PDSE environment, we can address a few common concerns.

10.3.1 Research IBM product support databases

We recommend that you search IBM product support databases for known solutions (APARs and PTFs) or contact the local IBM software support to address your concern whenever you encounter an unexpected situation within your PDSE environment.

Review the current information APARs for recommended PDSE maintenance.

APAR numbers for currently supported z/OS DFSMS releases:

II13336	z/OS DFSMS 1.3 *HDZ11G0* CURRENT PDSE MAINTENANCE
II13875	z/OS DFSMS 1.5 *HDZ11H0* CURRENT PDSE MAINTENANCE
II13967	z/OS DFSMS 1.6 *HDZ11J0* CURRENT PDSE MAINTENANCE

10.3.2 Common errors

Most PDSE errors are encapsulated and reported by a system ABEND 0F4, along with return and reason codes that explain the specific problem. PDSE services request an SDUMP whenever the S0F4 is encountered.

However, many users have S0F4 dump processing suppressed via the Dump Analysis And Elimination (DAE) facility. If you are not getting dumps on ABEND0F4 problems in PDSE, it is likely that you are suppressing them.

Table 10-1 shows some common PDSE errors and where to go for additional information.

Table 10-1 PDSE errors and possible resolution

PDSE condition	Diagnosis actions	Data gathering
System abend such as ABEND 0F4 or ABEND 0C4	"System abend considerations" on page 217	collection_A from every affected system
Hang due to latch or lock contention	"General approach to restarting a PDSE address space" on page 214	collection_A from every affected system
PDSE corruption	"Hard-error considerations" on page 217	collection_B
Performance degradation	"Performance considerations" on page 218	collection_C

- collection_A
  - System abend dump or dynamic/console dump (or both) of:
    - SMSPDSE or SMPDSE1 address space and the associated SYSBMFDS data space,
    - "Master" address space, and
    - Affected user address spaces
  - System log data (SYSLOG; prior to the situation)
  - LOGREC data (prior to situation)
  - SYSOUT output for the job (joblog)

  See "Data collection" on page 249 for more about the individual items listed above.
- collection_B
  - System abend dump or dynamic/console dump (or both) of:
    - SMSPDSE or SMPDSE1 address space and the associated SYSBMFDS data space,
    - "Master" address space, and
    - Affected user address spaces
  - DFSMSdss physical dump of affected PDSE or PHATOOL 'DUMPT ALL'
  - IEHLIST LISTVTOC report
  - System log (SYSLOG) data
  - LOGREC data
  - SYSOUT output for the job (joblog)

  See "Data collection" on page 249 for more about the individual items listed above.
- collection_C
  - Dump of SMSPDSE or SMPDSE1 address space and the associated SYSBMFDS data space
  - SMF record type 42 subtypes 1,2,6

- System log (SYSLOG) data
- LOGREC data
- SYSOUT output for the job (joblog)
- PDSE component trace (CTRACE) on request from IBM service representative

Similar diagnostic data from a better performing job or process will be helpful to analyze the performance issue. See “Data collection” on page 249 for more information about the individual items listed above.

10.4 Diagnosis actions

The next sections provide an overview of how to diagnose an unexpected situation in a PDSE environment.

10.4.1 General approach to restarting a PDSE address space

Before you decide whether to restart the SMSPDSE1 address space, review the “Considerations for restarting the SMSPDSE1 address space” on page 226.

Ensure that PDSE_RESTARTABLE_AS(YES) and PDSESHARING(EXTENDED) are specified in the IGDSMSxx PARMLIB member.

Figure 10-1 provides an overview of the steps that must be performed to restart a PDSE restartable address space.

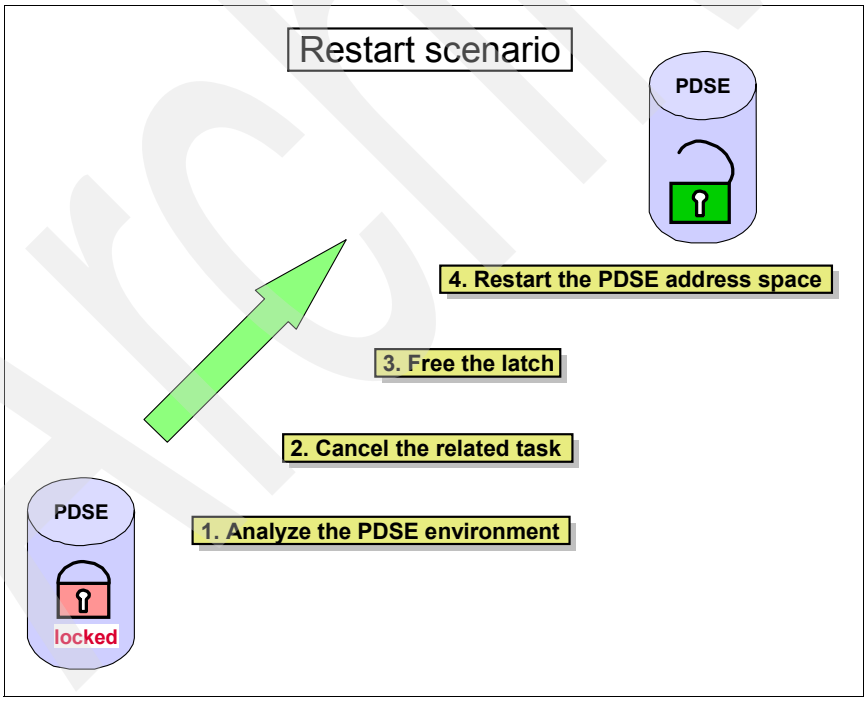


Figure 10-1 Restart scenario

Note: A restart of the PDSE address space may get you past the problem, but it does not solve it. Before restarting the PDSE address space, you should gather documentation about the problem such as dumps, syslog, and logrec.

Perform the following steps to restart the SMSPDSE1 address space:

1. Issue this operator command to determine which PDSE is causing the hang and which latch is being used:

   ```
   VARY SMS,[PDSE,PDSE1],ANALYSIS
   ```

   This command detects common problems that cause PDSE breakage.

   The example in Figure 10-2 analyzes all of the PDSEs in the SMSPDSE1 address space.

```
VARY SMS,PDSE1,ANALYSIS
```

Figure 10-2 Example VARY SMS,PDSE1,ANALYSIS

   If there is a message outstanding to another system, issue the ANALYSIS command on that system and resolve the problem on that system.

   See “ANALYSIS” on page 228 for more information about the command. “Recommended use of the ANALYSIS command” on page 234 provides some guidelines about resolving the individual conditions detected by the ANALYSIS command.

   If the ANALYSIS does not show any problems but you still encounter problems accessing a specific PDSE, then we suggest that you review the current SYSDSN and SYSZIGW0 ENQs for an overview of the current users of your PDSE. It is possible that your PDSE is affected by a long-running process.

2. Cancel the related task to attempt to free the latch or lock that is held.

   Note: To resolve a lock contention, you might want to cancel the job that holds the lock either on the system where you detect the contention or on another system in the sysplex.

   Dumps should be captured prior to any CANCEL, FREELATCH, or restart activity on the SMSPDSE1 address space.

   The CANCEL command in Figure 10-3 cancels job MHLRES2A associated with ASID x’001A’.

   See the “CANCEL Command” section of *z/OS: MVS System Commands*, SA22-7627, for more information.

```
CANCEL MHLRES2A,A=001A
```

Figure 10-3 Sample CANCEL jobname

   The CANCEL command in Figure 10-4 cancels user MHLRES6 associated with ASID x’0060’.

```
CANCEL U=MHLRES6,A=0060
```

Figure 10-4 Example CANCEL USER

3. If the system is still hung and your situation is related to a latch contention (and is not related to a lock contention), issue the VARY SMS,PDSE1,FREELATCH command to force the release of the latch.

Note: Only PDSE *latches* can be freed by using a FREELATCH command. PDSE locks cannot be freed by using FREELATCH.

To resolve a lock contention problem, you might want to cancel the job that holds the lock either on the system where you detect the contention or on another system in the sysplex.

The VARY SMS command in Figure 10-5 frees the latch at the specified latch address.

```
VARY SMS,PDSE1,FREELATCH(latchaddr,asid,tcb)
```

Figure 10-5 Example VARY SMS,PDSE1,FREELATCH(latchaddr)

Important: If this command is used to release a latch held by a process (on the same system or on another system in a multisystem environment) that is still running, it could result in breaking the PDSE and it could cause system abends.

If you suspect that a PDSE is failing, you should issue the VARY SMS,PDSE1,ANALYSIS command from each system where the PDSE has been reported as failing. Under no circumstances should you issue the VARY SMS,PDSE1,FREELATCH command without preceding it with a VARY SMS,PDSE1,ANALYSIS command.

See “FREELATCH” on page 232 for more information.

4. If the system is still hung, restart the SMSPDSE1 address space. This command is valid if the system was IPLed with the restartable SMSPDSE1 address space.

   The VARY SMS command in Figure 10-6 terminates and restarts the SMSPDSE1 address space and reconnects previously connected PDSEs.

```
VARY SMS,PDSE1,RESTART
```

Figure 10-6 Sample VARY SMS,PDSE,RESTART

The VARY SMS command in Figure 10-7 has the system wait 30 seconds to quiesce current operations before shutting down the current SMSPDSE1 address space. During this quiesce interval, new requests are held until the SMSPDSE1 address space is restarted. The restarted SMSPDSE1 address space reuses the storage cell pools in the extended common storage area (ECSA).

```
VARY SMS,PDSE1,RESTART,QUIESCE(30),COMMONPOOLS(REUSE)
```

Figure 10-7 Sample VARY SMS,PDSE,RESTART,QUIESCE,COMMONPOOLS

See “RESTART” on page 224 for more information.

5. If a restart is not possible or if the PDSE is still hung after performing a PDSE restart, you should consider IPLing your system.

10.4.2 System abend considerations

First, distinguish between an error that is related to an unexpected condition in the PDSE environment on the z/OS system (including user and PDSE address spaces) and between an unexpected condition related to the physical structure of a PDSE on a DASD volume.

We will refer to the errors related to the PDSE structures on a z/OS system as *soft-errors* and all problems related to PDSE structures on a DASD volume as *hard-errors*.

Soft-error considerations

If the data set can be accessed successfully from another system, the failure is probably a soft-error. There is a good chance that soft-errors are bypassed by following the instructions in “General approach to restarting a PDSE address space” on page 214 if the error situation is related to the restartable address space. If not and your problem is related to a single PDSE, then it might be possible to recover the PDSE with a *new name* from your backup repository and to connect the new PDSE to your applications. If none of these actions provide any relief, then an IPL should be scheduled for a convenient time.

Hard-error considerations

Hard-errors cannot be bypassed by restarting a PDSE address space. These kinds of errors are related to the physical structure of a specific PDSE. The error situation should be reproducible every time you access the PDSE. You might run into a hard-error situation when accessing the PDSE the first time or when accessing a specific member. This depends on the kind of error that exists.

Note: If you are not getting any dumps on subsequent access to a PDSE, then it is most likely that the dumps are suppressed by the DAE facility.

Note that any dump is historical in nature. The real challenge is to identify what caused the error. Therefore it is important to get as much information as possible about the history of the affected PDSE. For example:

- When was the PDSE last successfully accessed?
- What actions were performed afterwards?

An error could have occurred seconds before or days before, since the affected area may not have been accessed immediately after it was originally damaged. A long period of time could pass before the error condition is detected. These facts make it extremely difficult to determine the root cause for a hard-error such as a PDSE index problem.

Improper serialization is a reason for a hard-error. The following situations might cause an index corruption:

- Simultaneously access a PDSE without proper serialization
- Sharing outside of a GRSplex
- DFSMSdss DUMP processing (and subsequent RESTORE) without proper serialization
- Cancel during DFSMSdss RESTORE processing
- An internal error occurring in PDSE processing

Note that you can never repair a damaged PDSE by installing maintenance. A PTF may prevent having the same situation again. The installation of a PTF would never modify existing PDSE structures that might be invalid.

You should probably go to a good backup copy of the PDSE to bypass the error (for example, an index corruption) condition, or if possible use IEBCOPY to copy all members into a newly allocated PDSE.

10.4.3 Performance considerations

Due to the complexity of the PDSE implementation we do not have a common guideline to bypass or resolve your performance concerns, but a good starting point is to compare the affected process with a similar process that has better performance. The following questions should help you determine the cause of the performance degradation:

- What are the differences between the [job | task | application] in question compared to a similar process that performs or performed reasonably?
- How did the same or similar [task | job] perform previously?
- What kind of modifications were made? Did they have a direct or indirect effect on the system or PDSE environment?

Depending on the cause of the performance degradation you might consider restarting the PDSE address space or recovering the PDSE by copying the affected PDSE into a newly allocated PDSE.

- Restarting re-creates the internal structures in the PDSE address space. By doing this, you eliminate unnecessary structures that might have a performance impact.
- Copying a PDSE into a newly allocated PDSE re-creates the internal indexes within a PDSE from scratch. Therefore, it optimizes the index structures.

  Another real benefit is that the PDSE will be defragmented. Member data will be in sequential order on a DASD volume after the copy process has finished. A defragmentation could led to a remarkable performance improvement in some situations.

See Chapter 9, “Performance considerations” on page 179 for more information.

10.4.4 PDSE1 restart example

This section provides an example of resolving a latch and lock contention.

Note: The latch and lock contentions shown in this book were created artificially to show the restart processing in general. The situation that you might run into, as well as the resolution, could be different from the scenario shown in this section.

Message IGW038A indicates possible problems in SMSPDSE1 when the PDSE MONITOR is active. If the MONITOR is switched OFF then you do not get any notification about possible PDSE problems such as latch and lock contentions. (See Figure 10-8.)

```
*IGW038A Possible PDSE Problem(s) (SMSPDSE1)
 Recommend issuing V SMS,PDSE1,ANALYSIS
```

Figure 10-8 Restart example: IGW038A message

As expected, an ANALYSIS against SMSPDSE does not show any exceptional conditions, because the detected problem is related to the SMSPDSE1 address space (Figure 10-9 on page 219).

```
V SMS,PDSE,ANALYSIS
IGW031I PDSE ANALYSIS  Start of Report(SMSPDSE )
++ no exceptional data set conditions detected
PDSE ANALYSIS  End of Report(SMSPDSE )
```

Figure 10-9 Restart example: V SMS,PDSE,ANALYSIS command

Message IGW038A refers to the related PDSE address space SMSPDSE1. Figure 10-10 shows the result for a V SMS,PDSE1,ANALYSIS command.

```
V SMS,PDSE1,ANALYSIS
IGW031I PDSE ANALYSIS  Start of Report(SMSPDSE1) 936
---------data set name---------------------- -----vsgt-------
PDSERES.SMS.PDSE.LATCH1                       01-MHLV03-002D12
++ Unable to latch DIB:7F7004E0
   Latch:7FF6BC18 Holder(7888:00666660)
   Unable to determine Job or started task name for holder
   CallingSequence:IGWLHJIN IGWDACND IGWDBPAR IGWFARR3
PDSERES.SMS.PDSE.LOCK2                        01-MHLV03-002D15
++ Lock GLOBAL DIRECTORY EXCLUSIVE
   held for at least 550 seconds
   Hl1b:7FF6C320 HOLDER(001B:007DEC48)
   Holding Job:MHLRES2B
++ Lock GLOBAL FORMATWRITE EXCLUSIVE
   held for at least 550 seconds
   Hl1b:7FF6C320 HOLDER(001B:007DEC48)
   Holding Job:MHLRES2B
PDSERES.SMS.PDSE.LATCH2                       01-SBOX25-002E1D
++ Unable to latch DIB:7F7010A0
   Latch:7FF6CCB8 Holder(7888:00666660)
   Unable to determine Job or started task name for holder
   CallingSequence:IGWLHJIN IGWDACND IGWDBPAR IGWFARR3
PDSE ANALYSIS  End of Report(SMSPDSE1)
```

Figure 10-10 Restart example: V SMS,PDSE1,ANALYSIS command

A job name is not associated with the DIB latch for PDSERES.SMS.PDSE.LATCH1. (VSGT is 01-MHLV03-002D12.)

The D GRS,RES=(SYSDSN,*) command shows the current users (job names) of the PDSE (Figure 10-11).

```
D GRS,RES=(SYSDSN,PDSERES.SMS.PDSE.LATCH1)
ISG343I 12.52.09 GRS STATUS
S=SYSTEMS SYSDSN   PDSERES.SMS.PDSE.LATCH1
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC64       MHLRES2A           001A       007FD098   SHARE      OWN
```

Figure 10-11 Restart example: D GRS,RES=(SYSDSN,)*

First, we try to cancel job MHLRES2A to release the latch that is associated with PDSE PDSERES.SMS.PDSE.LATCH1 (Figure 10-12 on page 220).

```
CANCEL MHLRES2A,A=001A
IEA989I SLIP TRAP ID=X222 MATCHED.  JOBNAME=MHLRES2A, ASID=001A.
IEE301I MHLRES2A          CANCEL COMMAND ACCEPTED
```

Figure 10-12 Restart example: CANCEL command

As shown by a V SMS,PDSE1,ANALYSIS command, the cancel has not released the DIB latch for data set PDSERES.SMS.PDSE.LATCH1, as shown in Figure 10-13.

```
V SMS,PDSE1,ANALYSIS
IGW031I PDSE ANALYSIS  Start of Report(SMSPDSE1) 984
---------data set name---------------------- -----vsgt-------
PDSERES.SMS.PDSE.LATCH1                      01-MHLV03-002D12
++ Unable to latch DIB:7F7004E0
   Latch:7FF6BC18 Holder(7888:00666660)
   Unable to determine Job or started task name for holder
   CallingSequence:IGWLHJIN IGWDACND IGWDBPAR IGWFARR3
...
```

Figure 10-13 Restart example: ANALYSIS

Next we try to free the latch, referenced by message IGW031I, by issuing this command: V SMS,PDSE1,FREELATCH(latch_addr,asid#,tcb_addr). All required information, such as latch_addr(→ Latch:7FF6BC18), asid#, and tcb_addr (→Holder(7888:00666660)) are provided by the IGW031I message (Figure 10-14).

```
VARY SMS,PDSE1,FREELATCH(7FF6BC18,7888,00666660)
IGW032I PDSE FREELATCH  Start of Report 002
++ latch:7FF6BC18 released
PDSE FREELATCH  End of Report
```

Figure 10-14 Restart example: FREELATCH command

An additional V SMS,PDSE1,ANALYSIS proves that the latch related to 'PDSERES.SMS.PDSE.LATCH1' was freed (see Figure 10-15 on page 221).

```
V SMS,PDSE1,ANALYSIS
IGW031I PDSE ANALYSIS  Start of Report(SMSPDSE1) 020
---------data set name---------------------- -----vsgt-------
PDSERES.SMS.PDSE.LOCK2                       01-MHLV03-002D15
++ Lock GLOBAL DIRECTORY EXCLUSIVE
   held for at least 3324 seconds
   Hl1b:7FF6C320 HOLDER(001B:007DEC48)
   Holding Job:MHLRES2B
++ Lock GLOBAL FORMATWRITE EXCLUSIVE
   held for at least 3324 seconds
   Hl1b:7FF6C320 HOLDER(001B:007DEC48)
   Holding Job:MHLRES2B
PDSERES.SMS.PDSE.LATCH2                      01-SBOX25-002E1D
++ Unable to latch DIB:7F7010A0
   Latch:7FF6CCB8 Holder(7888:00666660)
   Unable to determine Job or started task name for holder
   CallingSequence:IGWLHJIN IGWDACND IGWDBPAR IGWFARR3
PDSE ANALYSIS  End of Report(SMSPDSE1)
IGWLHA10:1FD0D168
```

Figure 10-15 Restart example: ANALYSIS

The lock related to PDSERES.SMS.PDSE.**LOCK2** and latch to PDSE to PDSERES.SMS.PDSE.**LATCH2** are still held.

Note: We could bypass the latch and lock by freeing the associated DIB latch. However to show an example of the restart process we want to restart the SMSPDSE1 address space instead of freeing the latch.

Restart the PDSE1 (SMSPDSE1) address space to release the remaining lock and latch contentions (Figure 10-16 on page 222).

```
 V SMS,PDSE1,RESTART
 IGW036I VARY SMS,PDSE1,RESTART COMMAND ACCEPTED.
 IGW057I WAITING FOR SMSPDSE1 SHUTDOWN.
 IGW055I SMSPDSE1 SHUTDOWN IN PROGRESS.
 IGW999I XQUIESCE Started
 IGW062I SMSPDSE1 IS QUIESCING.
 IGW064I SMSPDSE1 IGNORING IN-PROGRESS TASK 001B:MHLRES2B, TCB@=007DEC48.
*169 IGW074D SMSPDSE1 QUIESCE FAILED. RETRY? (Y/N)
 R 169,N
 IEE600I REPLY TO 169 IS;N
 IGW065I SMSPDSE1 QUIESCE COMPLETE.
 IGW058I SMSPDSE1 SHUTDOWN COMPLETE.
 IGW059I SMSPDSE1 IS BEING ACTIVATED.
 IGW040I PDSE IGWLGEDC Connected
 IGW040I PDSE Connecting to XCF for Signaling
 IGW040I PDSE Connected to XCF for Signaling
 IGW040I PDSE Posting initialization
 IGW043I PDSE MONITOR IS ACTIVE 040
 ++ INVOCATION INTERVAL:60 SECONDS
 ++ SAMPLE DURATION:15 SECONDS
 IGW061I SMSPDSE1 INITIALIZATION COMPLETE.
 IGW066I SMSPDSE1 IS RECONNECTING ALL USERS.
 IGW066I SMSPDSE1 IS RECONNECTING ALL USERS.

 IGW069I SMSPDSE1 RECONNECT PHASE COMPLETE.
 IGW070I SMSPDSE1 WILL RESUME ALL USER TASKS.
 IGW999I XQUIESCE Stopping
 IGW999I Reconnect Completed Normally
```

Figure 10-16 Restart example RESTART command

IGW999I messages indicate internal processes. The message will be removed via an APAR/PTF in the future.

IGW064I and IGW074D messages indicate that a user task did not return within the quiesce phase. Specifying NO stops user tasks immediately. As referred to in the IGW064I message text, the user task might experience unpredictable results when the SMSPDSE1 address space is subsequently reactivated and the system attempts to resume the task and complete any in-flight PDSE requests.

- IGW064I SMSPDSE1 IGNORING IN-PROGRESS TASK ASID:JOBNAME, TCB=nnnnnnnnX

  Explanation: The operator entered the VARY SMS,PDSE1,RESTART command. The SMSPDSE1 address space was in the quiesce phase of shutdown processing. The system determined that a user task was running in the SMSPDSE1 address space but did not return to the user address space within the time limit specified in the quiesce parameter of the VARY SMS,PDSE1,RESTART command, or the default quiesce value if the quiesce parameter was not specified.

  System Action: The system ignores the user task in order to shut down SMSPDSE1. The task might experience unpredictable results when the SMSPDSE1 address space is subsequently reactivated and the system attempts to resume the task and to complete any in-flight PDSE request.

- IGW074D SMSPDSE1 QUIESCE FAILED, RETRY? (Y/N)

  Explanation: The operator entered the VARY SMS,PDSE1,RESTART command. One or more messages IGW063S, IGW064I, IGW075S, or

IGW076I were issued, identifying user tasks that were running in the SMSPDSE1 address space at the end of a quiesce time interval. The operator can choose one of the following responses:

Y Instructs SMSPDSE1 to retry the quiesce process.

N Instructs SMSPDSE1 to stop immediately and ignore the user tasks.

System Action: If the operator response is N, the system immediately abandons all further attempts to quiesce SMSPDSE1 and terminates it. User tasks identified in IGW063S messages are likely to fail due to their inability to continue processing PDSEs. Those identified in IGW064I messages are likely to be resumed successfully when a new SMSPDSE1 address space becomes available. If the operator response is **Y**, the system continues its attempts to quiesce SMSPDSE1 for another quiesce time interval. If quiesce processing is still not successful, messages IGW063S, IGW064I, IGW075S, and IGW076I reappear.

Operator Response: Determine whether the user tasks identified in IGW063S messages, and any job identified in message IGW076I, are critical to the successful operation of the system. If so, reply **Y** to instruct SMSPDSE1 to continue attempting to quiesce them. If your installation can tolerate failures of those user tasks, reply **N** to avoid any further delay in terminating SMSPDSE1. Note ASIDs and jobnames for subsequent problem determination.

A final V SMS,PDSE1,ANALYSIS command proves that all latches and locks are released now (Figure 10-17).

```
V SMS,PDSE1,ANALYSIS
IGW031I PDSE ANALYSIS  Start of Report(SMSPDSE1) 068
++ no exceptional data set conditions detected
PDSE ANALYSIS  End of Report(SMSPDSE1)
IGWLHA10:1FD0D168
```

Figure 10-17 Restart example ANALYSIS command

10.5 Operating the PDSE environment

This section describes the commands to operate your PDSE environment when you encounter an unexpected situation.

10.5.1 Definitions

You should be familiar with the following terminology to use these commands:

Latch A serialization object that PDSE processing uses to maintain the integrity of its control block structures. These latches are not GRS latches. Other than the V SMS,PDSE commands, there are no mechanisms that enables you to directly determine or modify the status of the PDSE latches.

DIB Control block used by PDSE processing to represent a data set. It is the root from which all open DCB information is stored.

HL1B	Control block used by PDSE processing to represent the serialization for the PDSE.
XCM	The subcomponent of PDSE processing that is responsible for communicating between systems using the XCF facilities.
Hash table	A structure that can be used to allow for the lookup of various PDSE control blocks, including DIBs and HL1Bs.
Lock	The internal mechanism used to serialize access to the PDSE internal structures.
Directory lock	A lock that is used to serialize the operations associated with extending the PDSE and formatting space within the PDSE. When this lock is held, it will be impossible to perform new member create functions or open the PDSE if it is not already open on the same system.
Attribute directory	The attribute directory (AD) is an internal hierarchical structure, implemented as a B-tree (a balanced tree, where balancing minimizes the tree depth and speeds up searching), which contains attribute information about individual members as well as attributes of the PDSE itself. The AD index is based on FSN and not by member names.
Name directory	The name directory (ND) pages represent the external members of a PDSE. The ND is also implemented in a B-tree structure. The ND provides the connection between the individual member names and the file sequence number (FSN). The ND index is based on member names and MLTs.
Index	The AD and ND are index structured.
FSN	PDSE members are tracked internally by a file sequence number (FSN).
VSGT	A token that is an internal representation of a PDSE. The VSGT includes the volser for the PDSE and the TTR for the associated Format-1 DSCB.

10.5.2 VARY SMS,[PDSE,PDSE1]

This section describes VARY SMS,PDSE command parameters.

RESTART

This command recycles the restartable PDSE address space (SMSPDSE1). It is valid if the system was IPLed with a restartable PDSE address space.

Restriction: PDSERESTART(YES) and PDSESHARING(EXTENDED) must be up and running to use this command.

Command format

```
V SMS,PDSE1,RESTART
   [  , QUIESCE ( duration | 3 )
   COMMONPOOLS ( NEW | REUSE ) ]
```

Operands

PDSE1	Enables you to select the restartable SMSPDSE1 address space.

RESTART Use this operand to terminate the SMSPDSE1 address space and immediately activate a new instance of the SMSPDSE1 address space.

QUIESCE Specifies the maximum time interval, in seconds, that existing in-flight operations quiesce before the current instance of the SMSPDSE1 address space is terminated.

Important: If the chosen interval is too long and SMSPDSE1 address requests do not complete, the address space restart is delayed. This delay affects processing on this system, and can also affect PDSE processing on other systems in the sysplex.

COMMONPOOLS(NEW)

SMSPDSE1 abandons the old common storage cell pools and creates a new set of cell pools in ECSA.

COMMONPOOLS(REUSE)

This is the default. SMSPDSE1 reuses the existing common storage cell pools that were created by the previous instance of SMSPDSE1.

Important: When you restart SMSPDSE1, the old storage cell pools are not deleted. Reuse the existing ECSA cell pools to avoid ECSA depletion. If you restart SMSPDSE1 because of a problem with the existing ECSA cell pools, create new cell pools by specifying COMMONPOOLS(NEW).

Example

You could initiate the PDSE restart processing by using V SMS,PDSE1,RESTART as shown in Figure 10-18. Figure 10-19 on page 226 provides another example with QUIESCE(60) and COMMONPOOLS(NEW) operands.

```
V SMS,PDSE1,RESTART
IGW036I VARY SMS,PDSE1,RESTART COMMAND ACCEPTED.
IGW057I WAITING FOR SMSPDSE1 SHUTDOWN.
IGW055I SMSPDSE1 SHUTDOWN IN PROGRESS.
IGW999I XQUIESCE Started
IGW062I SMSPDSE1 IS QUIESCING.
...                                  /* 3 second delay (deafult) */
IGW065I SMSPDSE1 QUIESCE COMPLETE.
IGW058I SMSPDSE1 SHUTDOWN COMPLETE.
IGW059I SMSPDSE1 IS BEING ACTIVATED.
IGW040I PDSE IGWLGEDC Connected
IGW040I PDSE Connecting to XCF for Signaling
IGW040I PDSE Connected to XCF for Signaling
IGW040I PDSE Posting initialization
IGW043I PDSE MONITOR IS ACTIVE 531
++ INVOCATION INTERVAL:60 SECONDS
++ SAMPLE DURATION:15 SECONDS
IGW061I SMSPDSE1 INITIALIZATION COMPLETE.
IGW066I SMSPDSE1 IS RECONNECTING ALL USERS.
IGW069I SMSPDSE1 RECONNECT PHASE COMPLETE.
IGW070I SMSPDSE1 WILL RESUME ALL USER TASKS.
IGW999I XQUIESCE Stopping
IGW999I Reconnect Completed Normally
```

Figure 10-18 Example RESTART command

```
VARY SMS,PDSE1,RESTART,QUIESCE(60),COMMONPOOLS(NEW)
IGW036I VARY SMS,PDSE1,RESTART COMMAND ACCEPTED.
IGW057I WAITING FOR SMSPDSE1 SHUTDOWN.
IGW055I SMSPDSE1 SHUTDOWN IN PROGRESS.
IGW999I XQUIESCE Started
IGW062I SMSPDSE1 IS QUIESCING.
...                                     /* 90 second delay */
IGW065I SMSPDSE1 QUIESCE COMPLETE.
IGW058I SMSPDSE1 SHUTDOWN COMPLETE.
IGW059I SMSPDSE1 IS BEING ACTIVATED.
IGW040I PDSE IGWLGEDC Connected
IGW040I PDSE Connecting to XCF for Signaling
IGW040I PDSE Connected to XCF for Signaling
IGW040I PDSE Posting initialization
IGW043I PDSE MONITOR IS ACTIVE 118
++ INVOCATION INTERVAL:60 SECONDS
++ SAMPLE DURATION:15 SECONDS
IGW061I SMSPDSE1 INITIALIZATION COMPLETE.
IGW066I SMSPDSE1 IS RECONNECTING ALL USERS.
IGW069I SMSPDSE1 RECONNECT PHASE COMPLETE.
IGW070I SMSPDSE1 WILL RESUME ALL USER TASKS.
IGW999I XQUIESCE Stopping
IGW999I Reconnect Completed Normally
```

Figure 10-19 Sample RESTART command (QUIESCE and COMMONPOOLS)

Message IGW999I identifies internal processes. The message will be removed via an APAR/PTF fix in the future.

Considerations for restarting the SMSPDSE1 address space

You might decide to restart the SMSPDSE1 address space to try to correct system problems caused by a problem in PDSE processing. However, before doing so, be aware that this action could result in failures of currently running jobs and TSO sessions that are accessing PDSEs. In addition, you should note the following restrictions and limitations:

1. The SMSPDSE1 address space restart might not work correctly if more than one SMSPDSE1 address space restart is attempted concurrently because of the xQUIESCE time interval that is initiated by the restarting SMSPDSE1. If an SMSPDSE1 restart is performed on more than one system at the same time, these restarting systems will not get the benefit of the other systems' xQUIESCE time duration interval.
2. If the SMSPDSE1 address space terminates as a result of a hard failure or because of the FORCE command, then the results of an SMSPDSE1 address space activate command are unpredictable. Undesirable effects that could be experienced because of a FORCE command include:
   a. If a member is deleted while SMSPDSE1 is down, then a reconnection to that member will fail.
   b. If a user on another system opens for update when the restartable connection had a data set open for read/write, the restart connection will fail.
3. If all systems in the PDSE extended sharing sysplex do not have support for the restartable PDSE address space or the toleration PTFs installed, then restart results will be unpredictable. Support for the PDSE restartable address space should be installed on all systems in the sysplex before activating the SMSPDSE1 address space on any system.
4. If the user address space fails to reconnect after the SMSPDSE1 address space restarts, then the user address space should be forced off of the system.

5. The restart of the SMSPDSE1 address space could result either in a loss of some SMF I/O counts or in duplicate counts.
6. If you used the VARY SMS,PDSE1,MONITOR command to change the PDSE MONITOR values and the SMSPDSE1 address space is restarted, then the values provided either by system defaults or by the MONITOR values specified in the IGDSMSxx member of SYS1.PARMLIB will be used to restart the PDSE1 MONITOR.
7. If callers that do not use the standard PDSE interface are connected, then PDSE will not know that the connection is active (and not behaving as expected). At this time there are no known callers of PDSE that fit this criteria.

ACTIVATE

The ACTIVATE command brings up the restartable PDSE address space. If the SMSPDSE1 address space is stopped, you can use the VARY SMS,PDSE1,ACTIVATE command to activate the SMSPDSE1 address space. This command is valid only after the operator forces down the SMSPDSE1 address space.

Important: IBM does not recommend forcing down the SMSPDSE1 address space because the results are unpredictable.

Command format

```
V SMS,PDSE1,ACTIVATE
   [  ,  COMMONPOOLS( NEW | REUSE ) ]
```

Operands

`PDSE1` Enables you to select the restartable SMSPDSE1 address space.

`COMMONPOOLS(NEW)`

SMSPDSE1 abandons the old common storage cell pools and creates a new set of cell pools in ECSA.

`COMMONPOOLS(REUSE)`

This is the default. SMSPDSE1 reuses the existing common storage cell pools that were created by the previous instance of SMSPDSE1.

Restriction: When you restart SMSPDSE1, the old storage cell pools are not deleted. Reuse the existing ECSA cell pools to avoid ECSA depletion. If you restart SMSPDSE1 because of a problem with the existing ECSA cell pools, create new cell pools by specifying COMMONPOOLS(NEW).

Example

See Figure 10-20 on page 228 for an example of reactivating the SMSPDSE1 address space.

In this example, the SMSPDSE1 address space is activated and the common storage cell pools in ECSA are reused. (REUSE is the default.) SMSPDSE1 startup processing creates storage cell pools in the ECSA. When SMSPDSE1 restarts, the old storage cell pools are not deleted. Normally, it is preferable to reuse the existing ECSA cell pools to avoid ECSA depletion. If the SMSPDSE1 restart is related to a problem with the existing ECSA cell pools, then you can create new ones by specifying COMMONPOOLS(NEW).

```
V SMS,PDSE1,ACTIVATE
IGW038I VARY SMS,PDSE1,ACTIVATE COMMAND ACCEPTED.
IGW059I SMSPDSE1 IS BEING ACTIVATED.
IGW040I PDSE IGWLGEDC Connected
IGW040I PDSE Connecting to XCF for Signaling
IGW040I PDSE Connected to XCF for Signaling
IGW040I PDSE Posting initialization
IGW043I PDSE MONITOR IS ACTIVE 605
++ INVOCATION INTERVAL:60 SECONDS
++ SAMPLE DURATION:15 SECONDS
IGW061I SMSPDSE1 INITIALIZATION COMPLETE.
IGW066I SMSPDSE1 IS RECONNECTING ALL USERS.
IEA989I SLIP TRAP ID=X058 MATCHED.  JOBNAME=MHLRES2A, ASID=001A.
IEA989I SLIP TRAP ID=X058 MATCHED.  JOBNAME=MHLRES2A, ASID=001A.
IEA848I NO DUMP WAS PRODUCED FOR THIS ABEND, DUE TO SYSTEM OR
INSTALLATION REQUEST
IGW069I SMSPDSE1 RECONNECT PHASE COMPLETE.
IGW070I SMSPDSE1 WILL RESUME ALL USER TASKS.
IGW999I XQUIESCE Stopping
IGW999I Reconnect Completed Normally
```

Figure 10-20 Example ACTIVATE command

ANALYSIS

Issue the ANALYSIS command only when you suspect one or more users or jobs are having problems accessing a PDSE. This command and the FREELATCH command use a sampling algorithm that interrogates the state of the PDSE every hundredth of a second for the number of retries that the user specifies. Errors will be reported only if the state of the PDSE determined from its control blocks is stable and does not change as the PDSE is used.

The analysis is performed for a single system. If information for more than one system is required, the ROUTE command should be used.

Refer to “Recommended use of the ANALYSIS command” on page 234 for more information about the condition detected by the ANALYSIS command. The section will also provide some guidelines for resolving the condition.

The following errors can be detected:

CONTROL BLOCK BROKEN

The system encountered an invalid control block in running this command.

H1LB Hash table latch held

Prevents the opening and closing of a subset of the PDSEs or PDSE1s on this system.

DIB Hash Table latch held

Prevents the opening and closing of a subset of the PDSEs or PDSE1s on this system.

HL1B latch held

Prevents locks for this data set from either being gotten or released. It also blocks lock negotiation with other systems.

HL1BPLCH latch held

Prevents locks for this data set from either being gotten or released. It also blocks lock negotiation with other systems.

DIB latch held

Prevents the selected PDSE or PDSE1 from being opened or closed on this system.

Contention message to another system is outstanding

Indicates that another system is failing to perform its required role in the sharing of PDSEs or PDSE1s across the sysplex. Whenever this error occurs, the problem source usually will be the system that has not responded to the message.

XCM latch held

When the XCM latch is not released, the system is unable to negotiate with other systems. Over time this prevents all other systems from accessing any PDSEs or PDSE1s that this system has locked exclusively. The latch is acquired only by tasks that run in the SMXC address space.

LATCH NULLIFIED

The PDSE or PDSE1 code has made the latch unavailable. The code usually does this when it closes the PDSE or PDSE1 for the last time on a system. This is an unexpected situation.

LATCH BROKEN

The data structures for a latch have been overlaid.

DIRECTORY OR FORMAT WRITE LOCK HELD OR WAITING

Detects that a job has held or that a job waited for a Directory or Format Write lock for an excessive amount of time.

Command format

```
VARY SMS,[PDSE,PDSE1], ANALYSIS
   [  , DSNAME ( dsname ) [ , VOLSER( volser ) ] ]
   [  , RETRIES( retries | 1500 ) ]
```

Operands

`dsname` When specified, causes the analysis to be performed for a particular PDSE or PDSE1. If the volser is omitted, the data set is found by using the default system catalog.

`volser` Enables you to specify an uncataloged PDSE or PDSE1.

`retries` Enables you to control the amount of time for which the particular PDSE or PDSE1 situation must remain static. The PDSE or PDSE1 control blocks are examined every hundredth of a second for the number of specified retries. By default, the data set is examined 1500 times or for approximately 15 seconds before reporting exceptional conditions. If no exceptional conditions are found, the command returns immediately after the first examination of the control blocks.

Example

Figure 10-21 on page 230 shows the output for a ANALYSIS against SMSPDSE address space where no exception is detected.

```
V SMS,PDSE,ANALYSIS
IGW031I PDSE ANALYSIS  Start of Report(SMSPDSE )
++ no exceptional data set conditions detected
PDSE ANALYSIS  End of Report(SMSPDSE )
IGWLHA10:1FD0D168
```

Figure 10-21 Example ANALYSIS command (no contention detected)

Figure 10-22 shows the output for a ANALYSIS against SMSPDSE1:

- A DIB latch contention is detected for PDSERES.SMS.PDSE.LATCH1. (VSGT is 01-MHLV03-002D12.)
- The affected DIB latch is located at virtual address 7F6FF4E0 and is held by ASID 7888 and TCB 00666660.

```
V SMS,PDSE1,ANALYSIS
IGW031I PDSE ANALYSIS  Start of Report(SMSPDSE1) 581
---------data set name---------------------- -----vsgt-------
PDSERES.SMS.PDSE.LATCH1                      01-MHLV03-002D12
++ Unable to latch DIB:7F6FF4E0
   Latch:7FF56C18 Holder(7888:00666660)
   Unable to determine Job or started task name for holder
   CallingSequence:IGWLHJIN IGWDACND IGWDBPAR IGWFARR3
PDSE ANALYSIS  End of Report(SMSPDSE1)
IGWLHA10:1FD0D168
```

Figure 10-22 Example ANALYSIS command (contention detected)

Issue the ANALYSIS,DSNAME(*pdse.ds.name*) command when you want to verify a specific PDSE only.

Some of the information displayed by ANALYSIS,DSNAME(*pdse.ds.name*) identifies internal processes.

Attention: Message text `++ no exceptional data set conditions detected` indicates that the PDSE data set is connected to the PDSE address space that the command was issued against.

In Figure 10-23 on page 231, PDSERES.SMS.PDSE is connected to the SMSPDSE1 address space.

```
V SMS,PDSE1,ANALYSIS,DSNAME(PDSERES.SMS.PDSE)
getting VOLSER
got VOLSER:SBOXB0
Allocationing volume
IEF196I IEF237I 35D1 ALLOCATED TO IGWVOL00
Allocated volume
Locating UCB
Located UCB
Getting TTR
Got TTR:002D09
Got TTR:002D09
Invoking transport
IGW031I PDSE ANALYSIS  Start of Report(SMSPDSE1) 188
++ no exceptional data set conditions detected
PDSE ANALYSIS  End of Report(SMSPDSE1)
Back from Transport
Invoking Free volume 1
IEF196I IEF285I   SYS04334.T025013.RA000.MSTRJCL.R0200256      KEPT
IEF196I IEF285I   VOL SER NOS= SBOXB0.
```

Figure 10-23 V SMS,PDSE1,ANALYSIS,DSNAME(pdse.dsname) output (1)

Attention: The message text ++ no PDSEs connected displays when no connection exists between the PDSE data set and the PDSE address space you queried (Figure 10-24).

```
V SMS,PDSE,ANALYSIS,DSNAME(PDSERES.SMS.PDSE)
getting VOLSER
got VOLSER:SBOXB0
Allocationing volume
IEF196I IEF237I 35D1 ALLOCATED TO IGWVOL00
Allocated volume
Locating UCB
Located UCB
Getting TTR
Got TTR:002D09
Got TTR:002D09
Invoking transport
IGW031I PDSE ANALYSIS  Start of Report(SMSPDSE ) 205
++ no PDSEs connected
PDSE ANALYSIS  End of Report(SMSPDSE )
Back from Transport
Invoking Free volume 1
```

Figure 10-24 V SMS,PDSE1,ANALYSIS,DSNAME(pdse.dsname) output (2)

The VOLSER and TTR of the Format-1 DSCB, which are part of the Virtual Storage Group Token (VSGT), are displayed in both cases:

```
got VOLSER:SBOXB0
Got TTR:002D09
```

The VSGT is 01-SBOXB0-002D09.

Note: A VSGT is a token that is used internally in PDSE services to represent a PDSE. The VSGT includes the volser for the PDSE and the TTR for the associated Format-1 DSCB.

The VSGT is also used for SYSZIGW0 ENQs. See “ENQ on SYSZIGW0” on page 172 for more information about SYSZIGW0 ENQ.

FREELATCH

The VARY SMS,[PDSE, PDSE1], FREELATCH command releases any latch that the ANALYSIS command indicates is held.

Attention: If this command is used to release a latch held by a process that was still running, it could result in breaking the PDSE or cause different types of system ABENDs.

The latch is not released unless it is held by the ASID, tcbaddr, indicated in the command. The latch is released only if it is held by the same user for each of the retries.

Command format

```
VARY SMS,[PDSE, PDSE1], FREELATCH( latchaddr , ASID ,tcbaddr)
   [ , Retries( retries | 1500 ) ]
```

Operands

`latchaddr` Address of the latch that is to be released.

`ASID` ASID of the holder of the latch.

`tcbaddr` Address of TCB that holds the latch.

When an SRB holds the latch, the address for the TCB actually points to a control block that represents the active SRB. The value will be above the 16 M line.

`retries` Enables you to control the amount of time for which the particular PDSE or PDSE1 situation must remain static. The PDSE or PDSE1 control blocks are examined every hundredth of a second. By default the data set is examined 1500 times or for approximately 15 seconds before the latch will be released.

Example

Figure 10-25 on page 233 shows an example of an ANALYSIS and the related FREELATCH against SMSPDSE1. We free the latch, referenced by IGW031I message, by issuing a VARY SMS,PDSE1,FREELATCH(7FF53A38,7888,00666660). All required information, such as latch_addr(Latch:7FF53A38), asid#, and tcb_addr (Holder(7888:00666660)) are provided by IGW031I message.

```
V SMS,PDSE1,ANALYSIS
IGW031I PDSE ANALYSIS  Start of Report(SMSPDSE1) 067
---------data set name---------------------- -----vsgt-------
PDSERES.SMS.PDSE.LATCH1                      01-MHLV03-002D12
++ Unable to latch DIB:7F6F1D60
   Latch:7FF53A38 Holder(7888:00666660)
   Unable to determine Job or started task name for holder
   CallingSequence:IGWLHJIN IGWDACND IGWDBPAR IGWFARR3
PDSE ANALYSIS  End of Report(SMSPDSE1)
IGWLHA10:1FD0D168

VARY SMS,PDSE1,FREELATCH(7FF53A38,7888,00666660)
IGW032I PDSE FREELATCH  Start of Report 079
++ latch:7FF53A38 released
PDSE FREELATCH  End of Report
```

Figure 10-25 Example FREELATCH command

MONITOR

The MONITOR command should be used to modify the processing for the PDSE monitor and to display the current status of the PDSE monitor. See message IGW043I for details about the information displayed. See “PARMLIB requirements” on page 34 for more information about interval and duration operands.

Command format

```
V SMS,[PDSE,PDSE1],MONITOR,ON[,interval[,duration]] ¦
                   OFF ¦
                   RESTART
```

Operands

ON | OFF	Turns monitor processing on or off.
RESTART	Resets the state at which the monitor is shut down due to an unexpected error in its processing.
interval	The number of seconds between successive scans of the monitor.
duration	The number of seconds a possible error condition must exist before it is treated as an error.

Note: The default values for interval and duration are not necessarily the optimum values for your system. Observation and adjustment will probably be necessary to attain the right values. Over time, even your chosen values may have to be re-adjusted**.**

Example

The current MONITOR settings are displayed if you omit any operand. See Figure 10-26 on page 234.

```
V SMS,PDSE,MONITOR
IGW043I PDSE MONITOR IS ACTIVE 152
++ INVOCATION INTERVAL:60 SECONDS
++ SAMPLE DURATION:15 SECONDS

V SMS,PDSE1,MONITOR
IGW043I PDSE MONITOR IS ACTIVE 150
++ INVOCATION INTERVAL:60 SECONDS
++ SAMPLE DURATION:15 SECONDS
```

Figure 10-26 Example MONITOR command (display)

The example in Figure 10-27 overrides the monitor settings:

- Interval = 120 seconds
- Duration = 30 seconds

```
V SMS,PDSE1,MONITOR,ON,120,30
IGW043I PDSE MONITOR IS ACTIVE 186
++ INVOCATION INTERVAL:120 SECONDS
++ SAMPLE DURATION:30 SECONDS
```

Figure 10-27 Example MONITOR command (modify)

Recommended use of the ANALYSIS command

If you suspect that a PDSE is failing, issue the ANALYSIS command from each system where the PDSE has been reported as failing.

Important: Do not issue the FREELATCH command without preceding it with an ANALYSIS command.

These variables indicate the kinds of message information:

nnnnnnnn	Number of seconds a lock or latch is held.
aaaa	ASID (address space identifier) of the latch holder.
ttttttt	TCB address of the latch holder. If an SRB holds the latch, the address is greater than 01000000.
llllllll	Address of the latch that is held.
ssssssss	System to which a message is directed.
hhhhhhhh	Address of the HL1B that is the control block that contains the locking information for a PDSE.
dddddddd	Address of the DIB that contains the data structure information for a PDSE.
jjjjjjjj	Job name for the apparent hold of a latch or lock.
stststst	Started task name for the apparent holder of a latch or lock. The SMXC is the locking address space for the PDSE. You should never force or cancel the SMXC.
nummsgs	Number of outstanding messages from this system.
n holders	Number of holders for a lock.
n writers	Number of writers for a lock.

You should see one of the following results:

```
++ No PDSEs connected
```

```
++ No exceptional data set conditions detected
```

This indicates that the problem is not one of the class of problems that this tool is designed to detect or fix. Obtain an SVC dump and contact your IBM service representative.

```
++ Message to ssssssss pending for nnnnnnnn seconds
```

This system has been waiting for another system to respond to it. Try running the ANALYSIS command on the other system. The problem is probably on the other system. This command is sometimes displayed with the DIB latch message. When the other system fails to respond during an OPEN, both messages will appear, but the problem is still on the other system and no attempt should be made to issue the FREELATCH command to free the latch on this system.

```
++ Unable to latch DIB:dddddddd
LATCH:llllllll Holder (aaaa: tttttttt)
Holding JOB:jjjjjjjj | Holding Started Task (stststst |
Unable to determine job or started task name for holder
```

This latch is held only during OPEN and should not be held for very long, unless the system is waiting for a response from another system. The latch blocks new OPENs and CLOSEs for the data set in question. You should first attempt to cancel and force the job or user that has the latch. If that does not work, issue a FREELATCH command for this data set. If that does not work, either re-IPL or stop using the data set.

```
++ Unable to latch HL1B:hhhhhhhh
LATCH: Holder (aaaa: tttttttt)
Holding JOB:jjjjjjjj | Holding Started Task (stststst |
Unable to determine job or started task name for holder
```

This latch is held only during an attempt to serialize updates to the PDSE. The latch should never be held for very long because it blocks most updates and reads of the data set in question. You should first attempt to cancel and force the job or user that has the latch. If that does not work, issue a FREELATCH command for this data set. If that does not work, either re-IPL or stop using the data set.

```
++ Unable to latch HL1BPLCH:hhhhhhhh
LATCH:llllllll Holder (aaaa: tttttttt)
Holding JOB:jjjjjjjj | Holding Started Task (stststst |
Unable to determine job or started task name for holder
```

This latch is held only during an attempt to send messages to another system. The latch should not be held for very long unless a message is being sent to another system. The latch blocks most updates and reads of the data set in question. If any messages to another system are outstanding, you should perform analysis and repair on that system. If no messages to other systems are outstanding, you should first attempt to cancel and force the job or user that has the latch. If that does not work, issue a FREELATCH command for this data set. If that does not work, either re-IPL or stop using the data set.

++ Unable to latch DIB:llllllll Holders changing

++ Unable to latch HL1B:llllllll Holders changing

++ Unable to latch HL1BPLCH:llllllll Holders changing

This is an informational message. It indicates that although the latch was held every time you tried to access the structures, the holder was dynamic. This indicates high activity on the data set but does not indicate any error. The occurrence of any of these messages is unlikely. They are here only for information.

++ Unable to obtain latch for DIB | HL1B HashTable:llllllll Holder(aaaa:tttttttt)

This message indicates that a global latch that is used to locate the structures for the PDSE is frozen. You should attempt to cancel the job or user that is associated with the ASID. If the system still holds the latch, force the initiator. If the system still holds the latch, you should attempt the FREELATCH command for this latch.

++ Unable to obtain latch for DIB | HL1B HashTable:llllllll Holders changing

This message indicates that a global latch used to locate the structures for the PDSE is very busy, preventing successful completion of the analysis. Reissue the FREELATCH command until the analysis is successful.

++ Unable to latch XCM Holder(aaaa:tttttttt)

The SMXC address space obtains this latch only when XCF messages are sent to another system or when messages are received from another system. This indicates that a global latch that is not associated with a particular data set is frozen. You should attempt to issue the FREELATCH command for this latch.

++ Unable to latch XCM Holders changing

This message indicates that a global latch used to serialize messaging between systems is very busy, preventing successful completion of some analysis. You should reissue the FREELATCH command until the analysis is successful.

++ Latch DIB | HL1B | HL1BPLCH Nullified

++ Latch DIB | HL1B | HL1BPLCH Broken

This message indicates that the specified latch either is broken or has been nullified. Both cases indicate that the data structures for the PDSE are compromised. You should attempt to have all users unallocate the PDSE; this might clean up the data structures.

++ Lock GLOBAL | LOCAL DIRECTORY | FORMATWRITE SHARED | EXCLUSIVE Held for at least nnnnnnnn seconds HL1B:hhhhhhhh HOLDER(aaaa:tttttttt) Holding JOB:jjjjjjjj | Holding Started Task:stststst | Unable to determine job or started task name for holder

This message indicates that the system has held a directory or format write lock for a long time. The indicated amount of time begins with the first ANALYSIS command that detected the waiting lock request.

If the lock is GLOBAL, it has been granted for the sysplex. If there is contention on this resource, you might want to cancel the job that holds the lock.

If the lock is held locally, this system is waiting for another system to release the lock, so that it can be granted. If there is no message to another system, you might need to cancel this job. If there is a message to another system,

issue the ANALYSIS command on that system and resolve the problem on that system.

If the command cannot detect the lock holder, you might be able to fix the problem by canceling all users of the data set.

```
++ n holders Additional holders of DIRECTORY |FORMAT WRITE Lock
```

```
++ n writers waiting for DIRECTORY | FORMAT WRITE LOCK Shared | Exclusive
```

These messages give an indication of the amount of activity against a data set when a lock has been held for an excessive amount of time.

```
++ nummsgs unresponded messages to ssssssss
```

This is a summary of messages sent to other systems that have been outstanding for too long. If you do not find any other errors, run the ANALYSIS command on the other system.

```
++ Broken PDSE structure --
```

```
Bad HL3B | LRE encountered
```

Internal structures of the PDSE are broken. Attempt to quiesce the use of the PDSE. This frees all existing PDSE structures and might resolve the PDSE problem.

10.5.3 DISPLAY SMS,[PDSE,PDSE1]

This section describes the various DISPLAY SMS,[PDSE,PDSE1] parameters.

LATCH

The LATCH command displays the status of a PDSE latch at laddr for the SMSPDSE SMSPDSE1 address space. See message IGW045I for information provided by this command.

Command format

```
D SMS,[PDSE,PDSE1],LATCH(laddr),DETAILED!SUMMARY
```

Operands

`laddr` Address of the latch that is to be displayed.

`DETAILED | SUMMARY`

Detailed or summary report about the latch. Summary is the default. The detailed report can be used to identify jobs and users that are waiting for the latch.

Example

Figure 10-28 on page 238 provides two examples of displaying a PDSE latch. The detailed report also identifies the current tasks waiting for the latch.

Tasks that are waiting to access the latch are:

- `Asid:005F Tcb:007DD878 StartedName:MHLRES2`
- `Asid:001A Tcb:007DEC48 jobname:MHLRES2A`

```
D SMS,PDSE1,LATCH(7FF53A38)
IGW045I PDSE Latch Display Start of Report(SMSPDSE1)
++ Latch:7FF53A38 held by 7888:00666660 Holder unknown
++ 1 Known requests on the latch wait queue
PDSE Latch Display End of Report(SMSPDSE1)

D SMS,PDSE1,LATCH(7FF53A38),DETAILED
IGW045I PDSE Latch Display Start of Report(SMSPDSE1)
++ Latch:7FF53A38 held by 7888:00666660 Holder unknown
++ Waiters follow:
++   Asid:005F Tcb:007DD878 StartedName:MHLRES2
++   Asid:001A Tcb:007DEC48 jobname:MHLRES2A
PDSE Latch Display End of Report(SMSPDSE1)
```

Figure 10-28 Example LATCH command

MODULE

The MODULE parameter displays the virtual address, maintenance level (PTF or APAR number), and compile date of the requested module in the SMSPDSE or SMSPDSE1 address space.

Note: The PTF or APAR number and the compile date are derived from the header of each module. You might verify this information with the information provided in the header of each module delivered by a PTF (or APARFIX (++APAR)) if you found unexpected information in the IGW046I message.

This command is provided to help users when a PER SLIP is needed to trap a particular module. See message IGW046I for information provided by this command. See “SLIP dump” on page 255 for more information about SLIP.

Command format

```
D SMS,[PDSE,PDSE1],MODULE(modname)
```

Operands

`modname` The name of the module.

Example

Figure 10-29 on page 239 shows three examples of using DISPLAY MODULE commands for PDSE-owned modules.

- PDSE module IGWLHA22 starts at virtual address 1FD076C8 (at SMSPDSE1 address space) and is on maintenance level OA08941 (→ APAR number). APAR numbers are normally used for APARFIXs (++APAR).
- PDSE module IGWISRCH starts at virtual address 047D4008 (at SMSPDSE address space). `None` indicates that no maintenance was applied to the requested module (the module is on base level).
- PDSE module IGWBLMPS starts at virtual address 01E8B378 (at SMSPDSE1 address space). It is on maintenance level UA10649 (→ PTF number).

```
D SMS,PDSE1,MODULE(IGWLHA22)
IGW046I PDSE Module IGWLHA22(1FD076C8) 0A08941  10/28/04

D SMS,PDSE,MODULE(IGWISRCH)
IGW046I PDSE Module IGWISRCH(047D4008)  NONE    03/19/04

D SMS,PDSE1,MODULE(IGWBLMPS)
IGW046I PDSE Module IGWBLMPS(01E8B378) UA10649  05/06/04
```

Figure 10-29 Example MODULE command

10.5.4 Interpreting output of D GRS commands

You can interpret the output from the D GRS command to find out the name of a PDSE that is enqueued upon.

An ENQ with a major name of SYSZIGW0 represents an open to a PDSE. In PDSE terms, the PDSE (directory or member) is connected to a task.

The minor name of a SYSZIGW0 ENQ contains a token plus an additional character that identifies a PDSE.

There is no direct external association to the PDSE user task using the SYSZIGW0 resource. However, for diagnosis purposes, it may be important for you to identify the users that are connected to the PDSE. Or, conversely, you know the PDSE data set name, and you want to find associated SYSZIGW0 ENQs.

Follow these steps to identify the PDSE user tasks from a SYSZIGW0 ENQ:

1. Locate the volume and TTR of the Format-1 DSCB.

   The token (VSGT), which is shown in the hexadecimal format of the D GRS command, contains the VOLSER as well as the TTR of the associated Format-1 DSCB for the PDSE. These two items uniquely identify the PDSE within a multisystem environment.

   The major name for the enqueue is SYSZIGW0, and the minor name is 11 bytes long, made up of the following four components:

   a. One byte identifying the token (always x'01')
   b. Six-character volume serial number of the PDSE
   c. Three-byte TTR of the Format-1 DSCB for the PDSE data set
   d. One character (I or O) to indicate locking for Input or Output

   See Figure 10-30 on page 240 for the structure of the ENQ token.

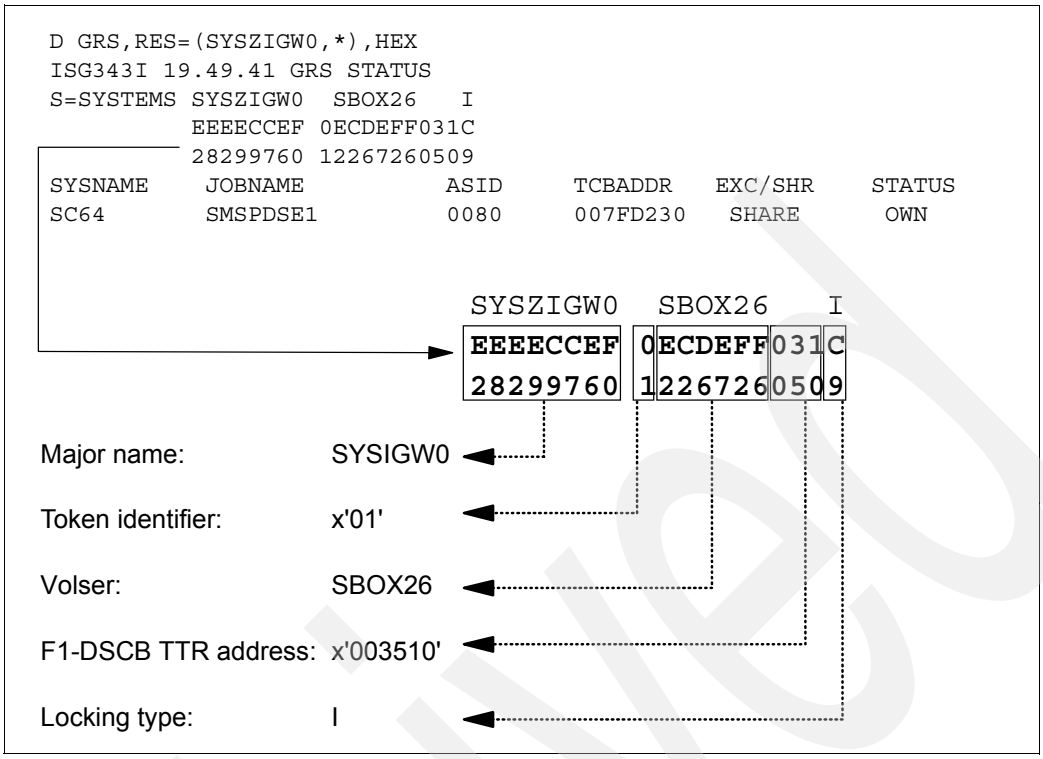


Figure 10-30 SYSZIGW0 ENQ structure

In Figure 10-30:

- The scope is SYSTEMS.
- The major name is SYSZIGW0.
- The 11-byte minor name in this case is `x'01' SBOX26 x'003510' I`. The minor name is often written in this format: 01-volser-ttr-I; in this case, `01-SBOX26-003510-I`.
  - Hex'01' is the one-character identification for the token.
  - SBOX26 is the six-character volser that contains the PDSE data set.
  - Hex'003510' is the three-character TTR (Track-Track-Record) of the Format-1 DSCB of the PDSE data set. In this case it is track x'0035' and record x'10'.
  - The `I` in the minor name indicates that the directory is locked.
- The enqueue is obtained by system SC64.

2. Converting the Format-1 DSCB pointer from TTR (hex) to C-H-R (dec).

   The hexadecimal TTR pointer must be converted to a decimal C-H-R (Cylinder-Head-Record) format since the LISTVTOC reports C-H-R.

   For example, with TTR = x'003510:

   Track x'0035' / x'F' (3390 trks/cyl) = 3 with remainder of 8

   => cylinder 3

   => head 8

   Record x'10' → record 16

   C - H - R = 3 - 8 - 16 (decimal)

3. Identify the data set name.

An IEHLIST LISTVTOC report for the volume referenced in the VSGT lists the entries from the volume table of contents (VTOC), whether indexed or non-indexed. The program lists the contents of selected data set control blocks (DSCBs) in edited or unedited form.

You have two possibilities to request a LISTVTOC report for a volume:

a. Using IEHLIST utility as shown in Figure 10-31 to format the VTOC of volume SBOX26

b. Using ISMF filter functions as shown in “Which PDSE address space is used?” on page 245 to list all PDSE on a specified volume, and by using the ISMF VTOCLIST line command for each listed volume

We show how to use the IEHLIST LISTVTOC utility in this section. The ISMF functions can be used in a similar way.

```
//VTOCL  EXEC PGM=IEHLIST
//SYSPRINT DD SYSOUT=*
//DD1      DD UNIT=3390,VOL=SER=SBOX26,DISP=SHR
//SYSIN    DD *
    LISTVTOC FORMAT,VOL=3390=SBOX26
/*
```

Figure 10-31 IEHLIST LISTVTOC sample

Now perform a find for the specific C-H-R (decimal ‘3 8 16’). It should refer to a DSCB field:

```
DSCB(C-H-R)
  3   8  16
```

The associated entry shows the data set name.

Figure 10-32 shows that Format-1 DSCB at CC-HH-R 3-8-16 is referring to data set name PDSERES.SMS.DIRK.PDSE1.

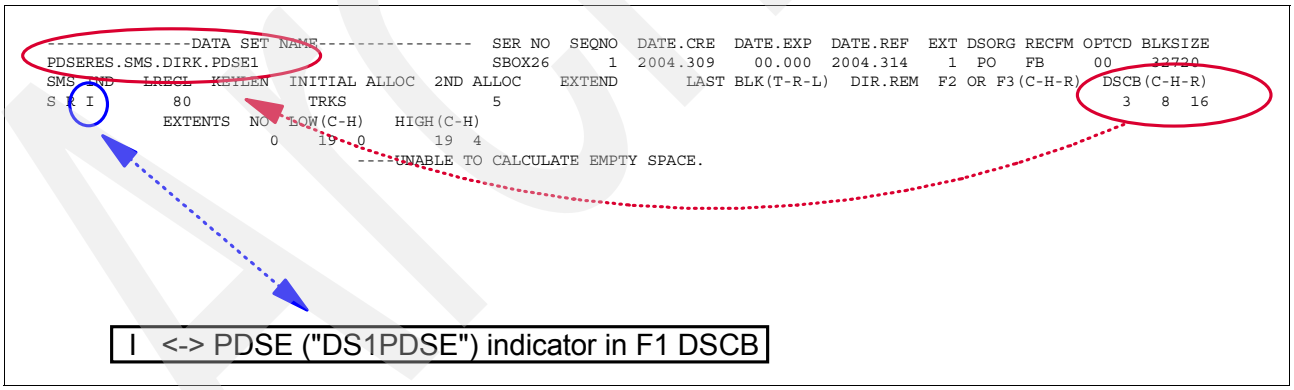


Figure 10-32 IEHLIST LISTVTOC output

Each Format-1 DSCB entry is identified by one or more indicator bits. A few of them are displayed in the formatted output of a LISTVTOC report by assigning a character to an indicator. They are displayed below the field name SMS.IND.

PDSE data sets are referred to an SMS.IND of `I`. `S` indicates that the data set is SMS managed. `R` indicates that system determined block (SDB) size and data set can be reblocked.

In *z/OS: DFSMSdfp Utilities*, SC26-7414, the appendixes “IEHLIST VTOC Listing” and “Explanation of Fields in IEHLIST Formatted VTOC Listing” provide further information.

Further detailed information about the VTOC is available in *z/OS: DFSMSdfp Advanced Services*, SC26-7400, “Using the Volume Table of Contents.”

4. Display the user tasks.

   A display of the SYSDSN enqueues by data set name should list all connected jobs or tasks for the PDSE.

   Figure 10-33 identifies job name MHLRES6 as a user who is currently accessing PDSE PDSERES.SMS.DIRK.PDSE1 (represented by ENQ with major name SYSZIGW0 & minor name 01-SBOX26-003510-I).

```
D GRS,RES=(SYSDSN,PDSERES.SMS.DIRK.PDSE1)
ISG343I 21.04.04 GRS STATUS 867
S=SYSTEMS SYSDSN   PDSERES.SMS.DIRK.PDSE1
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC64      MHLRES6            0079       007FD230   SHARE      OWN
```

Figure 10-33 D GRS output

10.5.5 XCF status

Various DISPLAY XCF commands can be used to verify the XCF status.

D XCF displays a list of all systems currently participating in the sysplex in message IXC334I.

D XCF,S,ALL displays, in message IXC335I, detailed information for all systems in the sysplex including the system status and the last recorded system status monitor time stamp for a system. See Figure 10-34.

```
D XCF
IXC334I  18.02.04  DISPLAY XCF
   SYSPLEX SANDBOX:    SC63             SC64             SC65
                       SC70

D XCF,S,ALL
IXC335I  18.02.13  DISPLAY XCF
SYSTEM   TYPE SERIAL LPAR STATUS TIME         SYSTEM STATUS
SC63     2084 6A3A   0C   11/10/2004 18:02:10 ACTIVE
SC64     2084 6A3A   11   11/10/2004 18:02:12 ACTIVE
SC65     2084 6A3A   12   11/10/2004 18:02:09 ACTIVE
SC70     2084 6A3A   13   11/10/2004 18:02:10 ACTIVE
```

Figure 10-34 D XCF sample

D XCF,GROUP,SYSZIGW0 displays the members of the specified group in message IXC332I (Figure 10-35). You will see one member for every SMSPDSE or SMSPDSE1 address in the sysplex.

```
D XCF,GROUP,SYSIGW00
IXC332I  18.05.02  DISPLAY XCF 732
 GROUP SYSIGW00:  IGWCLMR1SC64      IGWCLMR1SC65      IGWCLMR1SC70
                  IGWCLM01SC63      IGWCLM01SC64      IGWCLM01SC65
                  IGWCLM01SC70
```

Figure 10-35 D XCF,GROUP,SYSIGW00 output

XCF member name format: IGWCLMxxssssssss

In this format:

xx	Identifies the SMSPDSE or SMSPDSE1 address space. 01 → SMSPDSE (non-restartable) address space R1 → SMSPDSE1 (restartable) address space
ssssssss	Associated system name.

D XCF,GROUP,SYSIGW00,membername (Figure 10-36) or D XCF,GROUP,SYSIGW00,* (Figure 10-37 on page 244) display, in message IXC333I, detailed information (the system name, MVS job name, and current status) about all members of a particular group or all groups on one system.

Use the RO *ALL,D XCF,GROUP,SYSIGW00,* system command for detailed information about every XCF member in the sysplex (that belongs to group SYSIGW00).

```
D XCF,GROUP,SYSIGW00,IGWCLMR1SC64
IXC333I  18.09.54  DISPLAY XCF
   INFORMATION FOR GROUP SYSIGW00
   MEMBER NAME:        SYSTEM:     JOB ID:    STATUS:
   IGWCLMR1SC64        SC64        SMSPDSE1   ACTIVE

 INFORMATION FOR GROUP SYSIGW00 MEMBER IGWCLMR1SC64

 MEMTOKEN: 02000133 00080004      ASID: 0078

 SIGNALLING SERVICE
   MSGO ACCEPTED:          84  NOBUFFER:           0
   MSGO XFER CNT:          70   LCL CNT:          14  BUFF LEN:   956

   MSGI RECEIVED:          79  PENDINGQ:           0
   MSGI XFER CNT:          70  XFERTIME:         N/A

 GROUP SERVICE
   EVNT RECEIVED:           1  PENDINGQ:           0

   EXIT 01BF1CC8: 11/10/2004 12:36:46.552582 01 00:00:00.000007
```

Figure 10-36 D XCF,GROUP,SYSIGW00 membername output

```
D XCF,GROUP,SYSIGW00,*
IXC333I  18.14.27  DISPLAY XCF
   INFORMATION FOR GROUP SYSIGW00
   MEMBER NAME:        SYSTEM:     JOB ID:    STATUS:
   IGWCLMR1SC64        SC64        SMSPDSE1   ACTIVE
   IGWCLMR1SC65        SC65        SMSPDSE1   ACTIVE
   IGWCLMR1SC70        SC70        SMSPDSE1   ACTIVE
   IGWCLM01SC63        SC63        SMSPDSE    ACTIVE
   IGWCLM01SC64        SC64        SMSPDSE    ACTIVE
   IGWCLM01SC65        SC65        SMSPDSE    ACTIVE
   IGWCLM01SC70        SC70        SMSPDSE    ACTIVE

 INFORMATION FOR GROUP SYSIGW00 MEMBER IGWCLMR1SC64
...
 INFORMATION FOR GROUP SYSIGW00 MEMBER IGWCLM01SC64
...
```

Figure 10-37 D XCF,GROUP,SYSIGW00, output*

See *z/OS: MVS System Commands*, SA22-7627 and *z/OS: MVS Programming: Sysplex Services Guide*, SA22-7617, for a description of the XCF commands to verify XCF status and to verify the connected (joined) clients (or members) in the XCF group.

Note: Systems in XCF-local mode cannot use the PDSE extended sharing protocol. XCFLOCAL indicates that the system is a single, standalone MVS system that is not a member of a sysplex and cannot use couple data sets. In XCF-local mode, XCF does not provide signaling services between MVS systems.

10.5.6 PDSE sharing status

You can issue the DISPLAY SMS,OPTIONS command to determine the PDSE sharing mode as well as other PDSE-related settings. See Figure 10-38.

```
D SMS,OPTIONS
IGD002I 19:43:43 DISPLAY SMS 773
...
CF_TIME = 1800           PDSE_RESTARTABLE_AS = YES
PDSE_BMFTIME = 3600      PDSE1_BMFTIME = 3600
PDSE_LRUTIME = 15        PDSE1_LRUTIME = 50
PDSE_LRUCYCLES = 240     PDSE1_LRUCYCLES = 200
LOCAL_DEADLOCK = 15      GLOBAL_DEADLOCK = 4
REVERIFY = NO            DSNTYPE = PDS
ACSDEFAULTS = NO         PDSESHARING = EXTENDED
OVRD_EXPDT = NO          SYSTEMS = 8
PDSE_HSP_SIZE = 256MB    PDSE1_HSP_SIZE = 256MB
...
```

Figure 10-38 Display SMS,OTPIONS

See “Optional Keywords for IGDSMSxx” in *z/OS: MVS Initialization and Tuning Reference*, SA22-7592, for a description of the individual parameters.

You should check all systems for the following IPL messages, which indicate that the PDSE sharing mode is to be used on this system:

```
IGW303I NORMAL PDSE SHARING FORCED, INCOMPATIBLE PROTOCOL FOUND

IGW305I EXTENDED PDSE SHARING FORCED, INCOMPATIBLE PROTOCOL FOUND
```

10.5.7 Which PDSE address space is used?

The SYSZIGW0 ENQ is always obtained by a PDSE address space. Therefore, the SYSZIGW0 ENQ can be used to verify whether a PDSE is connected to the SMSPDSE or SMSPDSE1 address space. If you see more than one user of SYSZIGW0 ENQ for the same PDSE from different systems, then the PDSE is shared between these systems.

If you could identify the appropriate SYSZIGW0 ENQ then you can easily determine the PDSE address space by verifying the requester of that ENQ. D GRS,RES=(SYSZIGW0,*) displays the names of the jobs that requested the resource.

The minor name of a SYSZIGW0 ENQ contains a token plus an additional character. The token enables a unique identification of the associated PDSE within the sysplex. This token is called the Virtual Storage Group Token (VSGT). The VSGT, which is shown in the hexadecimal format of the D GRS command contains the VOLSER as well as the TTR of the associated Format-1 DSCB for the PDSE. These two items uniquely identify the PDSE within a multisystem environment.

You have two ways to identify the owning PDSE address space and the VSGT for a PDSE:

- Using PDSE analysis command
- Using ISMF functions

Using VARY SMS,[PDSE,PDSE1],ANALYSIS command

The V SMS,PDSE1,ANALYSIS,DSNAME(pdse.data.set.name) command can be used to verify whether a PDSE is currently connected to a PDSE address space.

In Figure 10-39 on page 246, message text `++ no exceptional data set conditions detected` indicates that the PDSE data set is currently connected to the PDSE address space that the command was issued against.

In this example, `PDSERES.SMS.DIRK.PDSE1` is currently connected to the SMSPDSE1 address space.

```
SMSPDSE1 : → non global connection  → non linklist  → restartable
```

If the VARY command is issued against a PDSE address space that does not have a connection to the PDSE, the message text `++ no PDSEs connected` will be displayed (as in Figure 10-40 on page 246).

The VOLSER and TTR of the Format-1 DSCB are displayed in both cases:

```
got VOLSER:SBOX26

Got TTR:003510
```

The VSGT is 01-SBOX26-003510.

Note: A VSGT is a token that is used by PDSE services to represent a PDSE. The VSGT includes the VOLSER for the PDSE and the TTR for the associated Format-1 DSCB.

The VSGT is used for SYSZIGW0 ENQs as well. See “ENQ on SYSZIGW0” on page 172 for more information about SYSZIGW0 ENQ.

```
V SMS,PDSE1,ANALYSIS,DSNAME(PDSERES.SMS.DIRK.PDSE1)
getting VOLSER
got VOLSER:SBOX26
Allocationing volume
IEF196I IEF237I 6012 ALLOCATED TO IGWVOL00
Allocated volume
Locating UCB
Located UCB
Getting TTR
Got TTR:003510
Got TTR:003510
Invoking transport
IGW031I PDSE ANALYSIS  Start of Report(SMSPDSE1) 275
++ no exceptional data set conditions detected
PDSE ANALYSIS  End of Report(SMSPDSE1)
```

Figure 10-39 V SMS,PDSE1,ANALYSIS,DSNAME(pdse.dsname) - connected

```
V SMS,PDSE,ANALYSIS,DSNAME(PDSERES.SMS.PDSE)
getting VOLSER
got VOLSER:SBOXB0
Allocationing volume
IEF196I IEF237I 35D1 ALLOCATED TO IGWVOL00
Allocated volume
Locating UCB
Located UCB
Getting TTR
Got TTR:002D09
Got TTR:002D09
Invoking transport
IGW031I PDSE ANALYSIS  Start of Report(SMSPDSE ) 205
++ no PDSEs connected
PDSE ANALYSIS  End of Report(SMSPDSE )
Back from Transport
Invoking Free volume 1
```

Figure 10-40 V SMS,PDSE1,ANALYSIS,DSNAME(pdse.dsname) - not connected

Using ISMF data set function

ISMF data set filtering functions can be used to identify the related TTR (Track-Track-Record) of the Format-1 DSCB for any data set. The TTR is part of the minor name for a PDSE (SYSZIGW0) ENQ. See “Interpreting output of D GRS commands” on page 239 and Figure 10-30 on page 240 for more detailed information about SYSZIGW0 ENQ.

From the ISMF Primary Option Menu, select option `1 Data Set` (Figure 10-41 on page 247).

```
 ISMF PRIMARY OPTION MENU - z/OS DFSMS V1 R6
Enter Selection or Command ===> 1

Select one of the following options and press Enter:

0  ISMF Profile            - Change ISMF User Profile
1  Data Set                - Perform Functions Against Data Sets
2  Volume                  - Perform Functions Against Volumes
3  Management Class        - Specify Data Set Backup and Migration Criteria
4  Data Class              - Specify Data Set Allocation Parameters
5  Storage Class           - Specify Data Set Performance and Availability
9  Aggregate Group         - Specify Data Set Recovery Parameters
L  List                    - Perform Functions Against Saved ISMF Lists
R  Removable Media Manager - Perform Functions Against Removable Media
X  Exit                    - Terminate ISMF









Use HELP Command for Help; Use END Command to Exit.
```

Figure 10-41 ISMF primary option menu

You can select your PDSE by requesting 2 - Catalog as a source for your requests and specifying the data set name (fully or partially specified) on the first ISMF data set selection entry panel. Partial data set name PDSERES.SMS.DIRK.** is specified in Figure 10-42.

```
Command ===>

For a Data Set List, Select Source of Generated List  . . 2  (1 or 2)

  1  Generate from a Saved List         Query Name To
       List Name  . .                   Save or Retrieve

  2  Generate a new list from criteria below
       Data Set Name . . . 'PDSERES.SMS.DIRK.**'
       Enter "/" to select option      Generate Exclusive list
       Specify Source of the new list  . . 2    (1 - VTOC, 2 - Catalog)
       1 Generate list from VTOC
          Volume Serial Number . . .            (fully or partially specified)
       2 Generate list from Catalog
          Catalog Name . . .
          Volume Serial Number . . .            (fully or partially specified)
          Acquire Data from Volume . . . . . . . Y  (Y or N)
          Acquire Data if DFSMShsm Migrated  . . N  (Y or N)

Use ENTER to Perform Selection; Use DOWN Command to View next Selection Panel;
Use HELP Command for Help; Use END Command to Exit.
```

Figure 10-42 ISMF data set select entry panel (page 1)

Your requested PDSE data set name should be listed now. The ISMF line command VTOCLIST displays the Format-1 DSCB for the listed data set name (Figure 10-43).

```
 DATA SET LIST
Command ===>

Enter Line Operators below:

     LINE                                  ALLOC    ALLOC   % NOT  COMPRESSED
     OPERATOR          DATA SET NAME       SPACE    USED    USED   FORMAT
    ---(1)----  ------------(2)------------ --(3)--  --(4)--  -(5)-  ---(6)----
    VTOCLIST    PDSERES.SMS.DIRK.PDSE1        277      111     59  ---
    ----------  ------  --------  --------- -------------  BOTTOM  OF  DATA  ---
```

Figure 10-43 ISMF data set list

The ISMF command VTOCLIST requests an IEHLIST LISTVTOC report for the requested data set. In Figure 10-44, the address of the Format-1 DSCB is referred to field DSCB(C-H-R).

```
 SYSTEMS SUPPORT UTILITIES---IEHLIST                              PAGE    1
DATE: 2004.320  TIME: 18.35.53
              CONTENTS OF VTOC ON VOL SBOX26  <THIS IS AN SMS MANAGED VOLUME>
---------------DATA SET NAME----------------  SER NO  SEQNO  DATE.CRE  DATE.EXP  DATE.REF  EXT DSORG RECFM OPTCD BLKSIZ
PDSERES.SMS.DIRK.PDSE1                       SBOX26      1  2004.309    00.000  2004.315    1  PO   FB     00    32720
SMS.IND   LRECL  KEYLEN  INITIAL ALLOC  2ND ALLOC    EXTEND       LAST BLK(T-R-L)  DIR.REM  F2 OR F3(C-H-R)  DSCB(C-H-R
S R I        80           TRKS                5                                                              3   8  1
           EXTENTS  NO  LOW(C-H)   HIGH(C-H)
                     0    19  0       19  4
                          ----UNABLE TO CALCULATE EMPTY SPACE.
```

Figure 10-44 ISMF VTOCLIST (IEHLIST LISTVTOC) report

The C-H-R 3-8-16 (cylinder-head-record (in decimal)) must be converted into TTR (track-track-head (in hexadecimal)) format:

1. Convert decimal record value into a 1-byte hexadecimal value:

   Record: 16 → x'10'

2. Calculate Tracks (15 tracks/cylinder):

   Tracks = "# of cylinder" * 15 + "# of heads"

   = 3 * 15 + 8

   Tracks = 53

3. Convert decimal amount of tracks into a two-byte hexadecimal value:

   Tracks: 53 → x'0035'

4. Combine track and record:

   TTR → x'003510'

The associated volume serial number SBOX26 is listed in the LISTVTOC report.

The associated VSGT is 01-SBOX26-003510.

The PDSE named PDSERES.SMS.DIRK.PDSE1 is represented by VSGT 01-SBOX26-003510.

The VSGT is part of the minor name for SYSZIGW0 ENQ. An ENQ with a major name of SYSIGW0 represents an open/connection to a PDSE. A D GRS,RES=(SYSZIGW0,*),HEX command displays all SYSZIGW0 ENQs, in a hexadecimal view, that currently exist.

Identify the related SYSZIGW0 ENQ by comparing all minor names with the VSGT (VOLSER and TTR information) just derived. The job name in ISG343I represents the associated PDSE address space.

```
D GRS,RES=(SYSZIGW0,*),HEX
ISG343I 19.46.36 GRS STATUS
...
S=SYSTEMS SYSZIGW0  SB0X26   I
          EEEECCEF 0ECDEFF031C
          28299760 1226726050 9
SYSNAME        JOBNAME          ASID     TCBADDR    EXC/SHR    STATUS
SC64       SMSPDSE1            0078       007FD230    SHARE       OWN
```

Figure 10-45 Display GRS for resource SYSZIGW0

The ISG343I message identifies SMSPDSE1 (the job name) as the owning PDSE address space for the PDSE named PDSERES.SMS.DIRK.PDSE1.

SMSPDSE1: → non global connection → non linklist → restartable

10.6 Data collection

The PDSE implementation information must be collected to document an unexpected situation in the PDSE environment.

PDSE processes use control blocks in common storage, in the requesters/users address space, the PDSE address space, and PDSE data space. The execution of the majority of its code is under control of the user's tasks (user address space or home address space). PDSE services are exploiters of cross-memory mode services on your z/OS system. This means that most activities, behind a user request, are transferred to the SMSPDSE and SMSPDSE1 address spaces for execution (SMSPDSE(1) or the primary address space). See *z/OS: MVS Programming: Extended Addressability Guide*, SA22-7614 and *z/Architecture: Principles of Operation*, SA22-7832 for information about cross-memory mode processing.

Figure 10-46 on page 250 provides an overview of the most common data that must be collected to analyze a PDSE problem.

- System ABEND dump or dynamic dump (or both) of:
  - [SMSPDSE,SMPDSE1] address space and the associated data space SYSBMFDS
  - "Master" address space
  - Affected user address spaces
- LOGREC data (prior to situation) for problem-history data.
- SYSOUT output for the job (JOBLOG).
- System log data (SYSLOG) providing supplementing data for problem diagnosis.
- IEHLIST, a system utility used to list entries in the directory of one or more partitioned data sets or PDSEs, or entries (such as Format 1 DSCBs) in an indexed or non-indexed volume table of contents (VTOC).
- Depending on the error situation, a physical copy (DFSMSdss DUMP) of the PDSE itself or a PHATOOL report is required to verify the status and index structure of a PDSE.

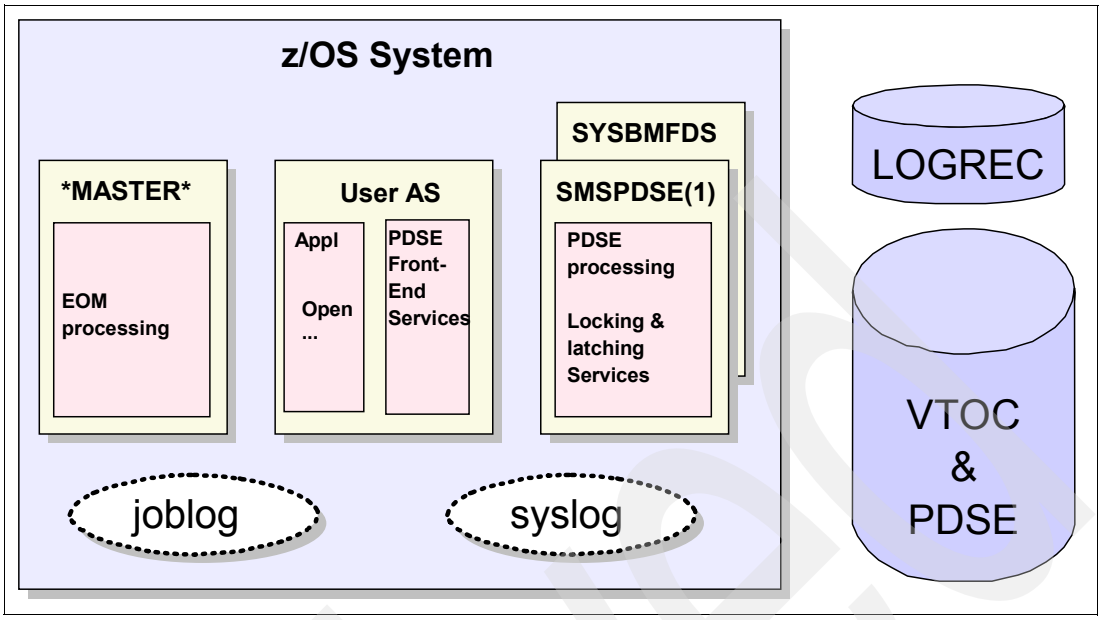


Figure 10-46 Overview PDSE data collection

10.6.1 System ABENDs

If PDSE services detects an internal problem, they normally issue an abend with a system completion code of x'0F4' (ABEND 0F4). The PDSE recovery routines also normally take a system dump (SDUMP) associated with the abend. In addition to the dump, a software LOGREC entry is written to the SYS1.LOGREC data set for every internal problem. The detail edit report for a system abend also indicates whether a dump was successfully started. See "EREP and LOGREC recording" on page 283 for more information about LOGREC.

To improve the serviceability of z/OS DFSMS PDSE service, a unique reason code is associated with an ABEND 0F4 occurrence. The reason code identifies the subcomponent (cc), the module (mm), and the defined reason for an ABEND 0F4.

As shown in Figure 10-47, all PDSE abend reason codes are one word (4 bytes) in length.

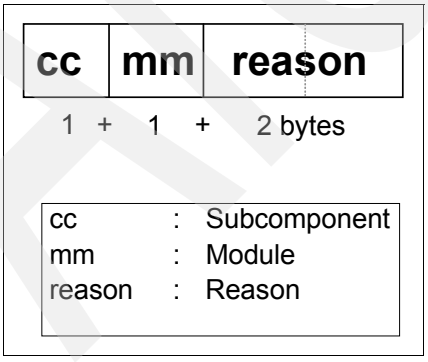


Figure 10-47 PDSE reason code format

Note: Add the keyword prefix RSN before the data for a reason code that contains nonzero data in all four bytes when you search problem-reporting databases.

This is an example of an ABEND0F4 dump title:

```
Title=COMPID=DF115,CSECT=IGWFVGTM+0504,DATE=10/18/98,
MAINTID=NONE,ABND=0F4,RC=00000014,RSN=2504C002
```

The dump title can be used to identify:

- The module (CSECT) and offset within the module where the ABEND SVC (SVC 13) was issued (IGWFVGTM in this example):

  ```
  CSECT=IGWFVGTM+0504
  ```

  The offset usually is not important because the reason code identifies the reason uniquely.

- The maintenance level of the module:

  ```
  MAINTID=NONE
  ```

  No PTF is applied to this module

- The system completion code (mostly identifying an ABEND 0F4 or ABEND 0C4):

  ```
  ABND=0F4
  ```

- The return code (RC), RC14:

  ```
  RC=00000014
  ```

- The reason code (RSN), RSN2504C002:

  ```
  RSN=2504C002
  ```

  - The first byte is the subcomponent id = x'25'.
  - The second byte is the module id = x'04'.
  - The third and fourth bytes are the module defined reason = x'C002'.

  The subcomponent and the module mapping are not published. However, a few reason codes are described in *z/OS: DFSMSdfp Diagnosis Reference*, GY27-7618, in the "PDSE Return and Reason Codes" chapter.

  The common reason code, x'C002' in our example, is described in the Diagnosis Reference book:

  ```
  X'C002' --> Exhausted ECSA storage
  ```

  In this example, PDSE services may be a victim. Some other task may have grabbed most of the ECSA storage. Therefore, the next step in diagnosing this problem is to verify the ECSA storage consumption.

Important: Use module names, abend codes, reason codes, and return codes to search in IBM product support databases.

General Purpose Register (GPR) usage information for ABEND 0F4

General Purpose Registers 0, 1, and 15 can be used to identify the system completion code (=system abend code), the PDSE reason code, and return code. See Figure 10-48 on page 252.

GPR 0 Contains the 4-byte DFSMSdfp (PDSE) reason code:

```
RSN150A001E
```

GPR 1 Contains the system completion code (_ _ c c c_ _ _):

```
ABEND 0F4
```

GPR 15 Contains the 1-byte abend reason code (RC20). The field is also referred to as a return code in PDSE terms and dump titles.

```
 CPU STATUS:
PSW=075C0000 84661106
    (Running in PRIMARY, key 5, AMODE 31, DAT ON)
    DISABLED FOR PER
   ASID(X'0009') 04661106. IGWBBMF1+04A106 IN EXTENDED PLPA
  ASCB26 at F9FA80, JOB(MHLRES2A), for the home ASID
  ASXB26 at 7FDD00 and TCB26E at 7DDC48 for the home ASID
  HOME ASID: 001A PRIMARY ASID: 0009 SECONDARY ASID: 001A

  General purpose register values
    Left halves of all registers contain zeros
     0-3  150A001E  440F4000  00000000  0281B000
     4-7  00000020  00FD6438  00FDBB40  04661C1C
     8-11 150A001E  04662390  04661391  00FDBBB0
    12-15 84660392  7F4E2898  846610FC  00000020
```

Figure 10-48 IP STATUS REGS

FAMS translation for reason codes

File and Attribute Management Services (FAMS) provides interfaces for managing data set attributes, and for data copying and conversion involving a PDSE. PDSEs may be converted to sequential data sets for use by backup and migration products, or re-created from sequential data sets previously produced by FAMS.

Note that FAMS processing is not an externally documented interface. FAMS (or IGWFAMS) is called by IEBCOPY, DFSMSdss (for logical data set processing only), and other ISV software products to process PDSEs. These products may return a FAMS-related reason code to the user application interface.

This section describes how to convert FAMS return codes and reason codes to FAMS messages so you can refer to z/OS: MVS Messages publications for diagnostic assistance. The message explanation gives the text form that refers to a specific problem or reason code.

Every return and reason code issued by FAMS (RSN28xxxxxx) can be translated into an IGW message. These are listed in *z/OS: MVS System Messages Volume 9 (IGF - IWM)*, SA22-7639.

The format of a FAMS reason code, x'28xxxxxx', can be converted to FAMS Message format, IGW01xxxz.

For example, for FAMS Return Code 12 (RC12) and FAMS Reason Code 28040015 (RSN28040015):

1. Convert the last two bytes of this reason code from hexadecimal value, 0015 to decimal value 21. This results in IGW01021.
2. Convert the FAMS Return Code using the following rules to represent the z in FAMS Message format IGW01xxxz:

   FAMS RC4 z converts to I, meaning Informational

   FAMS RC8 z converts to E, meaning Error

   FAMS RC12 z converts to T, meaning Terminate

   FAMS RC16 z converts to S, meaning Severe

Using the FAMS return code of 12, add a `T` at the end of the FAMS Message. For this example, this would result in FAMS Message IGW01021T.

After converting the FAMS Reason Code to a FAMS Message, refer to *z/OS: MVS System Messages Volume 9 (IGF - IWM)*, SA22-7639 for the IGW01021T message explanation:

```
IGW01021T THE SOURCE DATA SET dsname COULD NOT BE ACCESSED

Explanation:  An attempt was made to locate the source data set for the request (COPY,
GETATTR, FLOC, or ALTER) but the system indicated it was unable to access the data set.
...
```

10.6.2 Dumps

A dump provides a representation of the virtual storage for the system when an error occurs. Typically, a system component requests the dump from a recovery routine when an unexpected error occurs.

To analyze problems in the PDSE environment, the following resources should be included in any kind of SVC dump:

- SMSPDSE or SMPDSE1 address space and the associated SYSBMFDS data space
- *Master* address space
- Affected user address spaces

Figure 10-49 shows the address spaces that should be included in a system dump of a PDSE environment.

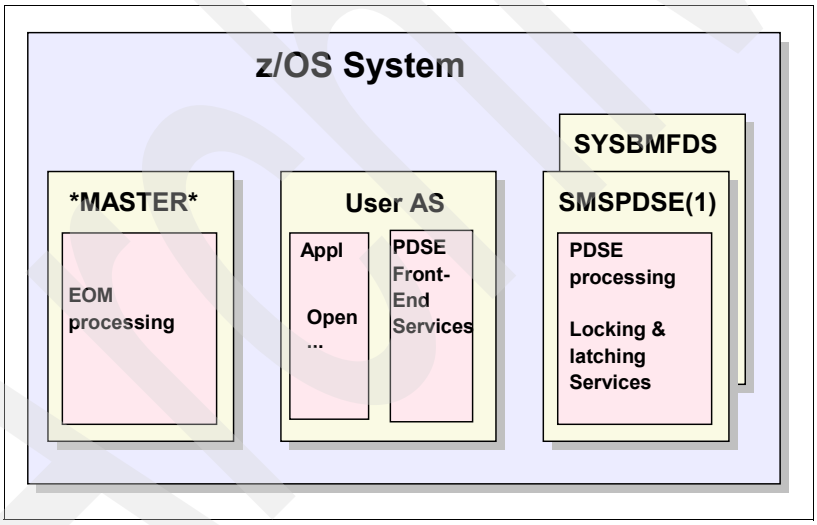


Figure 10-49 PDSE-related resources (single system)

Depending on whether a global (linklist) or non-global (non-linklist) connected PDSE is affected, you should include either SMSPDSE or SMSPDSE1 address space and the associated data space in your dump request. SMSPDSE and SMSPDSE1 provide all required services to process a request in a PDSE environment by their own. You do not have to dump both PDSE address spaces (SMSPDSE and SMSPDSE1) because they operate independently of each other. The related SYSZIGW0 ENQ can be used to identify the associated PDSE address space. “Which PDSE address space is used?” on page 245 provides an example of how to identify the related PDSE address space for a PDSE that is currently opened/connected.

If more than one system is affected in the situation that requires a dump within the PDSE environment, the appropriate SMSPDSE or SMPDSE1 address spaces and data spaces on all affected systems should be included in the dump process. Figure 10-50 provides an overview about the required resources that should be included in the dump request.

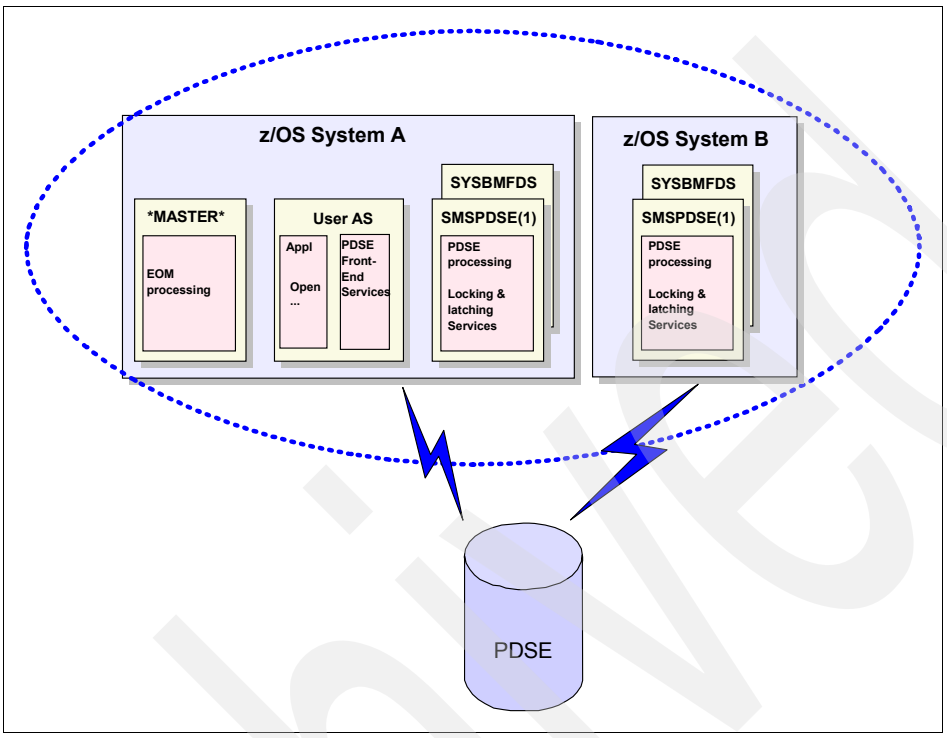


Figure 10-50 PDSE-related resources (multi-system)

Note: Message IEA611I indicates the end of the dump processing. Refer to message IEA611I for complete or partial dump information. As indicated by the name, a partial dump does not contain all requested resources. A complete dump is always preferred for problem determination.

```
IEA611I COMPLETE DUMP ON DUMP.D1118.H20.SC64.MHLRES2A.S00006
DUMPID=006 REQUESTED BY JOB (MHLRES2A)
FOR ASIDS(001A,0009,0001)
INCIDENT TOKEN: SANDBOX  SC64     11/18/2004 20:32:43

IEA611I PARTIAL DUMP ON SYS1.MCEVSF.DUMP.S00007
DUMPID=007 REQUESTED BY JOB (SMSPDSE1)
FOR ASIDS(005C,0001,0092,0089)
INCIDENT TOKEN: VSFPLEX  MCEVSF   11/17/2004 02:00:17
SDRSN = 80000000 00000000 00000000 00000000
SOME STORAGE COULD NOT BE DUMPED RC=4
```

See SDRSN Programming Interface information in *z/OS: MVS Data Areas, Volume 4 (RD - SRRA)*, GA22-7584 for an interpretation of SDRSN settings.

SDUMPs

PDSE routines include a *dump service* to collect information from all subcomponents that are involved in a specific error situation. The PDSE dump service subcomponent usually provides all of the necessary information to the MVS SDUMP service to produce the desired dump, including PDSE address and data space as well as the user address space.

In nearly all instances, such a system-adapted dump is sufficient to analyze the cause of a system abend. That is the main reason why a system dump is preferred over an SLIP or dynamic requested dump via using the DUMP command.

Important: Most PDSE errors are reported via an ABEND 0F4 system abend, along with return and reason codes that explain the specific problem. PDSE services are mostly requesting an SDUMP whenever the S0F4 is encountered.

Dump analysis and elimination (DAE) suppresses dumps that match a dump you already have. Each time DAE suppresses a duplicate dump, the system does not collect data for the duplicate or write the duplicate to a data set.

If you are not getting dumps on ABEND0F4 problems in PDSE, it is most likely that the dump was suppressed by the DAE facility. The detail edit report for a system abend indicates whether a dump was suppressed by DAE; see “EREP and LOGREC recording” on page 283 for more information about LOGREC.

See “General Purpose Register (GPR) usage information for ABEND 0F4” on page 251 for details about GPR usage if the 4-byte PDSE reason code is not listed in an ABEND 0F4 dump title.

SLIP dump

The SLIP command controls the serviceability level indication processing (SLIP), a diagnostic aid that intercepts or traps certain system events and specifies what action to take. Using the SLIP command, you can set, modify, and delete SLIP traps.

SLIP traps in systems in a sysplex

For a sysplex containing similar systems, certain problems might require identical SLIP traps on those similar systems. To set up these traps:

1. Assign similar names to identical jobs on different systems. The names should form a pattern, such as JOB1, JOB2, JOB3, and so on.
2. Create one IEASLPxx member containing the trap you need for the problem.

   Use a REMOTE parameter in the SLIP command so that the first time a trap matches on a system, the action will also be taken on other systems in the sysplex. For example, the SLIP command could request a dump on its system and, through REMOTE, on all the similar systems.

   Use an IDGROUP parameter so that, after the match, the identical traps on the other systems will be disabled.

   Use wildcards in parameters so that the command will process in all systems in the sysplex. For example, `JOB?` would indicate JOB1, JOB2, JOB3, and so on.
3. Place the member in the shared parmlib data set or in the parmlib data set for each of the similar systems.
4. In systems using JES2 or JES3, activate the member or members by entering this command on one of the systems:

   ```
   ROUTE *ALL,SET SLIP=xx
   ```

   If only some systems in the sysplex are similar, use a ROUTE command specifying a named subset of systems; see *z/OS: MVS System Commands*; SA22-7627 for details.

When a SLIP trap results in SVC dumps from multiple systems, each dump contains the same incident token. You can use the incident token to correlate the multiple dumps to one problem.

Structure of a SLIP SET command

In SLIP SET traps, you can indicate what kinds of events you want trapped and the system conditions for the trap, then specify what action the system is to take when the event occurs during the specified conditions.

- *Error event.* This is also called a *non-PER* event. The trap is set by this command:

  ```
  SLIP SET[,options],END
  ```

  The error events include, but are not limited to:

  - An ABEND macro issued by a task
  - Program check interruption

  Note: We only handle error events in this section.

- *Program event recording (PER) event*. The PER events are:
  - Instruction fetch. The trap is set by the command:

    ```
    SLIP SET,IF[,options],END
    ```

  - Successful branch. The trap is set by the command:

    ```
    SLIP SET,SBT[,options],END
    ```

  - Storage alteration. The trap is set by one of these commands:

    ```
    SET SET,SA[,options],END
    SET SET,SAS[,options],END
    ```

  Note: You cannot set a SLIP trap for the storage alteration of a Hiperspace.

  We do not describe PER events in this section because PER traps require detailed internal information about the modules that should be trapped. Authorized IBM service representatives can provide the necessary information if a PER trap is required for problem determination.

Set a error event SLIP

Use a SLIP command only at the direction of the system programmer. You can enter a SLIP command:

- On a console with MVS master authority.
- On a TSO terminal in OPERATOR mode.
- In a TSO CLIST.
- In the CLIST, use the line continuation character at the end of each line and the END parameter at the end of the last line.
- In an IEACMD00, COMMNDxx, or IEASLPxx parmlib member.

You can enter a SLIP command in any of these members, but we recommend placing your SLIP commands in IEASLPxx and enter a SET SLIP=xx command to activate the member.

The following list addresses some common parameters related to an error event SLIP. The parameters are presented alphabetically. Review the SLIP command in *z/OS: MVS System Commands*; SA22-7627 for more details and a list of all available parameters.

ACTION=value Specifies what you want the system to do when the trap matches system conditions. If you omit the ACTION parameter, the default is

ACTION=SVCD. If you specify more than one value, enclose the values in parentheses and separate them by commas.

Abbreviation: A

ACTION=SVCD

When the trap matches for an error or PER event, this schedules an SVC dump for the current or failing address space. For more information about SVC dumps, see the SVC dump chapter of *z/OS MVS Diagnosis: Tools and Service Aids.*

The SVCD value overrides DAE suppression but does not override suppression specified in a CHNGDUMP NODUMP operator command. If the dump cannot be written, perhaps because another SVC dump is in progress, SLIP issues message IEA412I, continues processing, and does not reschedule the dump.

The ASIDLST, DSPNAME, JOBLIST, LIST, SDATA, and SUMLIST parameters specify the data to be included in the dump. The SVC dump scheduled by the system includes the registers and PSW for the current or failing task.

Note that ACTION=SYNCSVCD is not valid for an error event trap.

ASID=asid

ASID=(asid[,asid]...)

For an error event or PER trap, this specifies the address space identifier (ASID) for the address space that must be in control when the error event or PER interruption occurs.

Each ASID is one to four hexadecimal digits. You can specify one to 16 ASIDs. If you specify one ASID, you can omit the parentheses. If you specify both ASID and JOBNAME, one of the specified address spaces must be the one in which the job is running or the trap will not match.

Abbreviation: AS

ASIDLST=asid

ASIDLST=(asid[,asid]...)

As an option of an ACTION or REMOTE parameter, specifies the address space or spaces to dump.

The ASID is one to four hexadecimal digits or a symbolic ASID. You can specify one to 15 ASIDs. If you specify only one ASID, you can omit the parentheses. The symbolic values are:

CURRENT or CU	Current address space.
HASID or H	Home address space.
I	Address space where the instruction ran.
PASID or P	Primary address space.
SASID or S	Secondary address space.

Note that zero indicates the current address space.

Abbreviation: AL

COMP=code

For an error event trap, specifies a system or user completion code that is associated with the error. For a system completion code, the form is hhh, three hexadecimal digits. You can indicate a set of codes by substituting x for one or more of the digits. For example, x23 means 123, 223, 323, 423, and so forth. You can use an x in any position.

Avoid setting a general trap, such as:

```
SLIP SET,COMP=0C4,ACTION=SVCD,END
```

The system normally has many expected program interrupts, each resulting in a 0C4 completion code.

Abbreviation: C

DISABLE For a SLIP SET trap, indicates that the trap set is to be inactive initially (ineligible for checking). If DISABLE is omitted, ENABLE is the default.

Abbreviation: D

DSPNAME=asid.name

DSPNAME='jobname'.name

DSPNAME=(asid.name | 'jobname'.name[, asid.name | ,'jobname'.name]...)

As an option of an ACTION or REMOTE parameter, specifies the data space or spaces to be included in an SVC dump.

Specify from 1 to 15 data space names in the parameter. When you specify more than one name, enclose the data space names in parentheses and separate them by commas. When you specify only one name, you can omit the parentheses. You can specify a data space name in two forms: `asid.name` or `'jobname'.name`.

jobname

The name of the job associated with the data space. The jobname is 1 to 8 alphanumeric and national ($, #, @) characters and must be enclosed in single quotes. You can specify jobname in a DSPNAME parameter:

- With ACTION=SVCD

- On the REMOTE parameter for an ACTION=SVCD trap

You can specify wildcards in the jobname.

name

Specifies the 1 to 8 character name associated with the data space at its creation. You can specify wildcards in the name on the DSPNAME option.

The name must be specified, unless the trap event is SA. For an SA trap event, the data space of the storage being altered is dumped.

When the interrupted unit of work holds a lock higher than the RSM lock, the system cannot determine the specific data spaces. In this case, no data spaces are included in the dump.

Abbreviation: DN

ENABLE For a SLIP SET trap, indicates that the defined trap is to be active initially (eligible for checking). If DISABLE is omitted, ENABLE is the default.

Abbreviation: EN

END For a SLIP SET trap, marks the end of the SLIP SET command. If you omit this parameter, the system prompts you for additional parameters.

Abbreviation: E

ID=trapid

For a SLIP SET trap, specifies a trap identifier. The trapid is 1 to 4 alphanumeric or national characters. If ID is not indicated in a SLIP SET command, the system assigns a unique ID.

JOBLIST=jobname

JOBLIST=(jobname[,jobname]...)

As an option of an ACTION or REMOTE parameter, identifies the names of jobs whose address spaces are to be dumped when the action is SVCD or SYNCSVCD in:

- The system in a sysplex that consists of one system
- The local system in a sysplex
- Another system in a sysplex, if REMOTE is specified

Specify from one to 15 job names. When you specify more than one name, enclose the names in parentheses and separate them by commas. When you specify only one name, you can omit the parentheses.

A jobname is 1 to 8 alphanumeric and national ($, #, and @) characters. You can specify wildcards in a jobname.

Abbreviation: JL

JOBNAME=userid

JOBNAME=jobname

For an error event or PER trap, specifies the user ID of a TSO/E user or the job name of the job or started task to be monitored.

The userid is 1 to 7 characters and the jobname is 1 to 8 characters. You can specify wildcards in the user ID or job name with the following exception: an asterisk (*) must be a suffix and cannot appear alone.

For error event traps, the specified job name must be for the home (dispatched) address space.

Abbreviation: J

MATCHLIM=m

For an error event or PER trap, specifies that the SLIP trap is to be disabled after *m* matches, where m is an integer from 1 to 65535.

If you omit MATCHLIM but specify ACTION=SVCD or ACTION=SYNCSVCD, the trap is disabled after one match.

Use a DISPLAY operator command to display the number of times that the conditions for a SLIP trap are met since the last time the trap was enabled.

Abbreviation: ML

MSGID

Causes control to be passed to the SLIP action processor under the unit of work issuing the WTO when the MSGID of the WTO matches the message ID specified on the MSGID parameter.

The message ID consists of the characters up to the first blank in the WTO but is never more than 10 characters. SLIP does not get control for messages that are reissued (for example, messages that are issued on one system and appear on another).

For minor lines associated with a branch entry WTO.

REMOTE=...

As an option of an ACTION parameter, enables SLIP to specify actions to be taken within the sysplex, on systems other than the

system on which the trap matches. A SLIP trap on one system can initiate an SVC dump or load a wait state on another system. The REMOTE parameter values specify other systems in the sysplex, actions for those systems, and options for dumps on those systems.

The REMOTE parameter can be specified only when the ACTION for the local system is SVCD, SYNCSVCD, or WAIT.

The parameters within the REMOTE parameter are UNCOND, COND, SYSLIST, ACTION, ASIDLIST, DSPNAME, JOBLIST, LIST, SDATA, and STRLIST:

UNCOND

On a REMOTE parameter, indicates whether remote actions should be performed conditionally or unconditionally when the trap matches. COND or UNCOND must be the first value specified on the REMOTE parameter. If you omit UNCOND, you do not have to code a comma in its place.

You can specify COND only on a PER trap when the action for the local system is ACTION=WAIT.

SYSLIST=([sysname,group.member,group.*,...])

On a REMOTE parameter, identifies systems in the sysplex on which the actions specified in remote will be performed. You can specify any combination of system names and member specifications. When group.* is specified, all systems in which any member of the group is running are affected.

If you omit SYSLIST, the default is all systems.

When a system is identified more than once, implicitly or explicitly, the first occurrence is used, and the others are ignored.

ACTION=SVCD

On a REMOTE parameter, identifies the action to be taken by the systems identified in SYSLIST. Specify ACTION=SVCD to initiate an SVC dump.

The dump options are ASIDLST, JOBLIST, DSPNAME, LIST, and SDATA. Dump options are processed only when the action is specified as SVCD or is the default.

When ACTION is not specified within the REMOTE parameter: If the local action is SVCD or SYNCSVCD, the default action is SVCD. All systems identified in SYSLIST use the default SLIP SVCD parameters as their default dump options.

ASIDLST, DSPNAME, JOBLIST, LIST, SDATA, and STRLIST

On a REMOTE parameter, the syntax is identical to the parameters for the dump option on the local system. When specified without an equal sign (=) and value, the options specified for the local system are used for the systems identified in SYSLIST.

Note that the only symbolic ASIDs accepted for the ASIDLIST parameter are PRIMARY and CURRENT.

Allowable abbreviations:

RM for REMOTE

SY for SYSLIST

SET Specifies that the SLIP command sets a trap. This parameter is positional; it must appear following a blank after SLIP.

SDATA=area

SDATA=(area[,area]...)

As an option of an ACTION or REMOTE parameter, specifies the kind of system areas to dump. You can specify any combination of the areas, enclosed in parentheses and separated by commas.

If you specify only one area, you can omit the parentheses.

SDATA options override the installation-defined defaults set by the CHNGDUMP command for any parameters that can be specified. Also, although it cannot be specified via the SLIP command, SDATA=SERVERS is always used for SLIP.

Allowable abbreviations: SD for SDATA

Examples

Figure 10-51 shows an example error event SLIP on system completion code 0F4 provided in a IEASLPxx PARMLIB member.

Note: uasn1 must be replaced by a job or user name before setting the SLIP.

```
EDIT       SYS1.PARMLIB(IEASLPxx)
****************************** Top of Data ******************
SLIP SET,ACTION=SVCD,COMP=0F4,ID=PDSE,MATCHLIM=1,ENABLE,
     ASIDLST=(H,P,S,1),
     JOBLIST=(SMSPDSE1,uasn1),
     DSPNAME=('SMSPDSE1'.*),
     SDATA=(CSA,PSA,SQA,LSQA,LPA,TRT,SUM,SWA,RGN,GRSQ),
     END
*************************** Bottom of Data ****************
```

Figure 10-51 SLIP on COMP

Figure 10-52 on page 262 shows another example for a error event SLIP. The SLIP will trap on a message ID (IGW038A). A REMOTE statement is included to capture a dump on another system at the same time when the SLIP will match.

Note: uasn1 and sysname must be replaced by a job/user and system name before setting the SLIP.

The SLIP is disabled (inactive) and must be enabled to be active.

```
EDIT       SYS1.PARMLIB(IEASLPMH)
***************************** Top of Data ******************************
SLIP SET,ACTION=SVCD,MSGID=IGW038A,ID=PDSE,MATCHLIM=1,DISABLE,
     ASIDLST=(H,P,S,1),
     JOBLIST=(SMSPDSE1,uasn1),
     DSPNAME=('SMSPDSE1'.*),
     SDATA=(CSA,PSA,SQA,LSQA,LPA,TRT,SUM,SWA,RGN,GRSQ),
     REMOTE=(UNCOND,SYSLIST=(sysname),ACTION=SVCD,
     JOBLIST=(*MASTER*,SMSPDSE1),SDATA),
     END
***************************** Bottom of Data ***************************
```

Figure 10-52 SLIP on MSGID with REMOTE operand

Set a SLIP from PARMLIB

Use the SET SLIP command to select the IEASLPxx member of the logical PARMLIB that contains the SLIP commands:

```
SET SLIP=xx
```

In this example, xx is two alphanumeric characters indicating the IEASLPxx member of the logical PARMLIB that contains the commands that SLIP processing is to use.

Modify a SLIP

Use the SLIP MOD command to modify an existing SLIP trap:

```
SLIP MOD{,ENABLE | ,DISABLE},ID=trapid
```

In this example:

`ENABLE`	The specified SLIP trap is to be made active. Abbreviation: EN
`DISABLE`	The specified SLIP trap is to be made inactive. Abbreviation: D
`ID=trapid`	Only the SLIP trap with the identifier trapid is to be modified.

Delete a SLIP

Use the SLIP DEL command to delete a SLIP trap:

```
SLIP DEL,ID=trapid
```

In this example:

`DEL`	Specifies that the SLIP command remove one SLIP traps from the system. This positional parameter must appear following a blank after SLIP.
`ID=trapid`	Only the SLIP trap with the identifier trapid is to be deleted.

Display a SLIP

Use the DISPLAY SLIP command to display information about SLIP traps.

```
D SLIP=trapid
```

In this example:

`SLIP`	Indicates that the system is to display summary information about SLIP traps or detailed information about one SLIP trap (message IEE735I).
`trapid`	The system is to display detailed information about the SLIP trap associated with the identifier `trapid`. If you do not specify trapid, the system lists all SLIP traps in the system and tells whether each trap is enabled or disabled.

Examples

We recommend using the IEASLPxx parmlib member to set up a SLIP. Figure 10-53 shows an example to get an SVC dump:

- On system completion code 0F4 (COMP=0F4).
- ID is PDSE.
- The SLIP is enabled.
- Match limit is 1.
- ASIDLIST identifies home, primary, secondary, and *MASTER* address space to be dumped.
- JOBLIST identifies SMSPDSE1 and user address space MHLRES2A.
- DSPNAME keyword asks to dump all data spaces associated with SMSPDSE1 address space.
- SDATA specifies the kind of system areas to be dumped.

```
EDIT       SYS1.PARMLIB(IEASLPMH)
***************************** Top of Data ****************
SLIP SET,ACTION=SVCD,COMP=0F4,ID=PDSE,MATCHLIM=1,ENABLE,
     ASIDLST=(H,P,S,1),
     JOBLIST=(SMSPDSE1,MHLRES2A),
     DSPNAME=('SMSPDSE1'.*),
     SDATA=(CSA,PSA,SQA,LSQA,LPA,TRT,SUM,SWA,RGN,GRSQ),
     END
***************************** Bottom of Data **************
```

Figure 10-53 Example SLIP on COMP=0F4

SET SLIP=MH selects the SYS1.PARMLIB(IEASLPMH) PARMLIB member (Figure 10-54).

```
SET SLIP=MH
IEE252I MEMBER IEASLPMH FOUND IN SYS1.PARMLIB
IEE727I SLIP TRAP ID=PDSE SET
IEE536I SLIP     VALUE MH NOW IN EFFECT

D SLIP=PDSE
IEE735I 14.55.49 SLIP DISPLAY 821
ID=PDSE,NONPER,ENABLED
ACTION=SVCD,SET BY CONS MHLRES6,RBLEVEL=ERROR,MATCHLIM=1,O
COMP=0F4,ASIDLST=HASID,PASID,SASID,0001,DSPNAME='SMSPDSE1'.*
SDATA=PSA,SQA,LSQA,RGN,LPA,TRT,CSA,SWA,SUMDUMP,GRSQ
JOBLIST=SMSPDSE1,MHLRES2A
```

Figure 10-54 Example DELETE, SET, and DISPLAY SLIP

Figure 10-55 shows common messages when the SLIP trap matches.

```
IEA992I SLIP TRAP ID=PDSE MATCHED.  JOBNAME=MHLRES2A, ASID=001A.
IEA411I SLIP TRAP ID=PDSE DISABLED FOR MATCHLIM
...
IEA794I SVC DUMP HAS CAPTURED:
DUMPID=006 REQUESTED BY JOB (MHLRES2A)
DUMP TITLE=SLIP DUMP ID=PDSE
...
IEA611I COMPLETE DUMP ON DUMP.D1118.H20.SC64.MHLRES2A.S00006
DUMPID=006 REQUESTED BY JOB (MHLRES2A)
FOR ASIDS(001A,0009,0001)
INCIDENT TOKEN: SANDBOX  SC64     11/18/2004 20:32:43
```

Figure 10-55 Example SLIP trap matched

The dump refers back to the job names displayed in the IEA611I message:

```
Dump Title: SLIP DUMP ID=PDSE
   ASID JOBNAME
   ---- --------
   0001 *MASTER*
   0009 SMSPDSE1
   001A MHLRES2A
```

See “IPCS basics” on page 276 for more about using IPCS. Figure 10-56 shows an example of deleting a SLIP.

```
SLIP DEL,ID=PDSE
IEE727I SLIP TRAP ID=PDSE DELETED
```

Figure 10-56 Example of SLIP DEL

Figure 10-57 on page 265 shows a sample IEASLPxx PARMLIB member for generating an SVC dump. In the IGW038I message:

- ID is PDSE.
- The SLIP is disabled.
- Match limit is 1.
- ASIDLIST identifies home, primary, secondary, and *MASTER* address space to be dumped.
- JOBLIST identifies SMSPDSE1 and user address space MHLRES2A.
- DSPNAME keyword requests to dump all data spaces associated with SMSPDSE1 address space.
- SDATA specifies the kind of system areas to be dumped.
- REMOTE keyword requests an SVC dump of *MASTER* and SMSPDSE1 address space on the SC70 system. SDATA is the same as used for the system the SLIP is activated for.

```
EDIT       SYS1.PARMLIB(IEASLPMH)
***************************** Top of Data ******************************
SLIP SET,ACTION=SVCD,MSGID=IGW038A,ID=PDSE,MATCHLIM=1,DISABLE,
     ASIDLST=(H,P,S,1),
     JOBLIST=(SMSPDSE1,MHLRES2A),
     DSPNAME=('SMSPDSE1'.*),
     SDATA=(CSA,PSA,SQA,LSQA,LPA,TRT,SUM,SWA,RGN,GRSQ),
     REMOTE=(UNCOND,SYSLIST=(SC70),ACTION=SVCD,
     JOBLIST=(*MASTER*,SMSPDSE1),SDATA),
     END
***************************** Bottom of Data ***************************
```

Figure 10-57 Example of SLIP on MSGID with REMOTE keyword

SET SLIP=MH selects the SYS1.PARMLIB(IEASLPMH) PARMLIB member (Figure 10-58).

```
SET SLIP=MH
IAT7450 JOB SMFCLEAR (JOB12996) PURGED
IEE252I MEMBER IEASLPMH FOUND IN SYS1.PARMLIB
IEE727I SLIP TRAP ID=PDSE SET
IEE536I SLIP     VALUE MH NOW IN EFFECT
IEA631I  OPERATOR BZZOSC65 NOW INACTIVE, SYSTEM=SC65    , LU=OPTSO

D SLIP=PDSE
IEE735I 16.37.19 SLIP DISPLAY
ID=PDSE,NONPER,DISABLED
ACTION=SVCD,SET BY CONS MHLRES6,RBLEVEL=ERROR,MATCHLIM=1,0
MSGID=IGW038A,ASIDLST=HASID,PASID,SASID,0001
DSPNAME='SMSPDSE1'.*
SDATA=PSA,SQA,LSQA,RGN,LPA,TRT,CSA,SWA,SUMDUMP,GRSQ
REMOTE=(UNCOND
SYSLIST=SC70,ACTION=SVCD,JOBLIST=*MASTER*,SMSPDSE1,SDATA
       )
JOBLIST=SMSPDSE1,MHLRES2A
```

Figure 10-58 Example SET and DISPLAY SLIP

The SLIP must be enabled to be active (Figure 10-59).

```
SLIP MOD,ENABLE,ID=PDSE
IEE727I SLIP TRAP ID=PDSE ENABLED
```

Figure 10-59 Example of SLIP MODIFY (issued on SC64 system)

Figure 10-60 on page 266 shows the messages that are issued on the SC64 system where the SLIP was set. Figure 10-61 on page 266 shows the related message on the SC70 system.

```
 IEA992I SLIP TRAP ID=PDSE MATCHED.  JOBNAME=SMSPDSE1, ASID=0009.
 IEA411I SLIP TRAP ID=PDSE DISABLED FOR MATCHLIM
 IEA041I SDUMP SCHEDULED FOR REMOTE SLIP TRAP ID=PDSE FROM SYSTEM SC64
*IGW038A Possible PDSE Problem(s) (SMSPDSE1) 280
 Recommend issuing V SMS,PDSE1,ANALYSIS
...
IEA794I SVC DUMP HAS CAPTURED: 283
 DUMPID=007 REQUESTED BY JOB (SMSPDSE1)
 DUMP TITLE=SLIP DUMP ID=PDSE
 IEF196I IGD101I SMS ALLOCATED TO DDNAME (SYS00015)
 IEF196I         DSN (DUMP.D1118.H21.SC64.SMSPDSE1.S00007        )
 IEF196I         STORCLAS (SCDUMP) MGMTCLAS (MCDB22) DATACLAS (
 )
 IEF196I         VOL SER NOS= SBOXAE
 IEF196I IGD104I DUMP.D1118.H21.SC64.SMSPDSE1.S00007         RETAINED,
 IEF196I DDNAME=SYS00015

 IEA611I COMPLETE DUMP ON DUMP.D1118.H21.SC64.SMSPDSE1.S00007 290
 DUMPID=007 REQUESTED BY JOB (SMSPDSE1)
 FOR ASIDS(0009,0001,001A)
 INCIDENT TOKEN: SANDBOX  SC64     11/18/2004 21:40:16
```

Figure 10-60 Syslog on SC64 system

The dump refers back to the job names displayed in the IEA611I message:

```
Dump Title: SLIP DUMP ID=PDSE
   ASID JOBNAME
   ---- --------
   0001 *MASTER*
   0009 SMSPDSE1
   001A MHLRES2A
```

```
 IEA794I SVC DUMP HAS CAPTURED: 110
 DUMPID=008 REQUESTED BY JOB (XCFAS   )
 DUMP TITLE=SLIP DUMP REMOTE ID=PDSE FROM SYSTEM=SC64
 IGD17380I STORAGE GROUP (SGDUMP) IS ESTIMATED AT 86% OF CAPACITY, 111
 WHICH EXCEEDS ITS HIGH ALLOCATION THRESHOLD OF 85%
 IEF196I IGD101I SMS ALLOCATED TO DDNAME (SYS00023)
 IEF196I         DSN (DUMP.D1118.H21.SC70.XCFAS.S00008           )
 IEF196I         STORCLAS (SCDUMP) MGMTCLAS (MCDB22) DATACLAS (
 )
...
 IEF196I IGD104I DUMP.D1118.H21.SC70.XCFAS.S00008             RETAINED,
 IEF196I DDNAME=SYS00023
 IEA611I COMPLETE DUMP ON DUMP.D1118.H21.SC70.XCFAS.S00008 118
 DUMPID=008 REQUESTED BY JOB (XCFAS   )
 FOR ASIDS(0006,0001,0009)
 INCIDENT TOKEN: SANDBOX  SC64     11/18/2004 21:40:16
```

Figure 10-61 Syslog on SC70 system

The dump refers back to the job names displayed in the IEA611I message:

```
Dump Title: SLIP DUMP REMOTE ID=PDSE FROM SYSTEM=SC64
   ASID JOBNAME
   ---- --------
   0001 *MASTER*
   0006 XCFAS
   0009 SMSPDSE1
```

See “IPCS basics” on page 276 for more information about using IPCS.

Dynamic/console dump

The DUMP command requests a system dump (SVC dump) of virtual storage of one or more address spaces and data spaces on your z/OS system. The output data set may be either a pre-allocated dump data set named SYS1.DUMPxx, or an automatically allocated dump data set named according to an installation-specified pattern.

You should request only one dump at a time on one system. Otherwise, you might have trouble determining the dump request that causes a particular IEE094D message. Also, a system writes only one SVC dump at a time, so it does not save anything to make several requests at once.

You can request a system dump by entering the DUMP command either:

- On an MVS console (the direct method)
- In an IEACMD00 parmlib member (indirect method)

We recommend that you place your DUMP commands in IEADMCxx and enter a DUMP PARMLIB=xx command to start the dump process.

Note: Hiperspace information is not included in SVC dumps.

Direct method by using DUMP command

The DUMP command captures an SVCDUMP of the requested job-, users-, and system address spaces.

Specify TSONAME=(tsoname) if the PDSE exploiter that should be dumped is a TSO user; otherwise specify the name of the job on the JOBNAME keyword (JOBNAME=(jobname).

A sample DUMP command to dump SMSPDSE address space and all required resources on one system only:

```
DUMP COMM=(PDSE DUMP)
R #,JOBNAME=(*MASTER*,SMSPDSE,jobname),CONT
R #,TSONAME=(tsouser),CONT
R #,DSPNAME=('SMSPDSE'.*),CONT
R #,SDATA=(PSA,CSA,SQA,GRSQ,LPA,LSQA,RGN,SUM,SWA,TRT),END
```

A sample DUMP command to dump SMSPDSE1 address space and all required resources on every system in a sysplex:

```
DUMP COMM=(PDSE DUMP)
R #,JOBNAME=(*MASTER*,SMSPDSE1,jobname),CONT
R #,TSONAME=(tsouser),CONT
R #,DSPNAME=('SMSPDSE1'.*),CONT
R #,SDATA=(PSA,CSA,SQA,GRSQ,LPA,LSQA,RGN,SUM,SWA,TRT),CONT
R #,REMOTE=(SYSLIST=(*('*MASTER*','SMSPDSE1'),SDATA,DSPNAME)),END
```

Note: The command should be issued on the system where the latch/lock is held. Replace either `jobname` or `tsouser` with the name identified by V SMS,[PDSE,PDSE1],ANALYSIS output.

Example

Figure 10-62 provides an example of requesting a dump by using system commands.

```
 DUMP COMM=(PDSE DUMP)
*150 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
 R 150,JOBNAME=(*MASTER*,SMSPDSE),CONT
 IEE600I REPLY TO 150 IS;JOBNAME=(*MASTER*,SMSPDSE),CONT
*151 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
 R 151,TSONAME=(MHLRES6),CONT
 IEE600I REPLY TO 151 IS;TSONAME=(MHLRES6),CONT
*152 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
 R 152,DSPNAME=('SMSPDSE'.*),CONT
 IEE600I REPLY TO 152 IS;DSPNAME=('SMSPDSE'.*),CONT
*153 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
 R 153,SDATA=(PSA,CSA,SQA,GRSQ,LPA,LSQA,RGN,SUM,SWA,TRT),END
 IEE600I REPLY TO 153 IS;SDATA=(PSA,CSA,SQA,GRSQ,LPA,LSQA,RGN,SUM,SW
 IEA794I SVC DUMP HAS CAPTURED: 237
 DUMPID=018 REQUESTED BY JOB (*MASTER*)
 DUMP TITLE=PDSE DUMP
 IGD17380I STORAGE GROUP (SGDUMP) IS ESTIMATED AT 89% OF CAPACITY, 238
 WHICH EXCEEDS ITS HIGH ALLOCATION THRESHOLD OF 85%
 IEF196I IGD101I SMS ALLOCATED TO DDNAME (SYS00027)
 IEF196I         DSN (DUMP.D1112.H05.SC64.#MASTER#.S00018        )
 IEF196I         STORCLAS (SCDUMP) MGMTCLAS (MCDB22) DATACLAS (
 )
 IEF196I         VOL SER NOS= SBOXB1
...
 IEF196I IGD104I DUMP.D1112.H05.SC64.#MASTER#.S00018          RETAINED,
 IEF196I DDNAME=SYS00027
 IEA611I COMPLETE DUMP ON DUMP.D1112.H05.SC64.#MASTER#.S00018 263
 DUMPID=018 REQUESTED BY JOB (*MASTER*)
 FOR ASIDS(0001,0008,0076)
 INCIDENT TOKEN: SANDBOX  SC64     11/12/2004 05:49:47
```

Figure 10-62 Sample DUMP command

Indirect method by using IEADMCxx PARMLIB member

The IEADMCxx PARMLIB member provides the possibility of requesting an SVCDUMP by preparing the dump information in an IEADMCxx PARMLIB member.

1. Create PARMLIB member IEADMCxx (Figure 10-63 on page 269).

```
EDIT       SYS1.PARMLIB(IEADMCMH)
***************************** Top of Data ***********************
COMM=(PDSE DUMP)
JOBNAME=(*MASTER*,SMSPDSE1,jobname),
TSONAME=(tsouser),
DSPNAME=('SMSPDSE1'.*),
SDATA=(PSA,CSA,SQA,GRSQ,LPA,LSQA,RGN,SUM,SWA,TRT),
REMOTE=(SYSLIST=(*('*MASTER*','SMSPDSE1'),SDATA,DSPNAME)),END
***************************** Top of Data ***********************
```

Figure 10-63 Example IEADMCxx PARMLIB member

Note: Replace either `jobname` or `tsouser` with the name identified by V SMS,[PDSE,PDSE1],ANALYSIS output.

2. Specify `DUMP PARMLIB=xx` to start the dump process.

Note: The command should be issued on the system where the latch/lock is held.

Example

Figure 10-64 shows a sample IEASLPxx PARMLIB member to get an SVC dump:

- Dump title is `PDSE DUMP`.
- Jobs `*MASTER*` and `SMSPDSE1` and TSO user `MHLRES6` are selected.
- DSPNAME keyword requests to dump all data spaces associated with the SMSPDSE1 address space.
- SDATA specifies the kind of system areas to be dumped.
- REMOTE keyword requests an SVC dump of *MASTER* and SMSPDSE1 address space on the SC63 system. SDATA and DSPNAME are the same as used for the system for which the SLIP is activated.

```
EDIT       SYS1.PARMLIB(IEADMCMH)
***************************** Top of Data ********************
COMM=(PDSE DUMP)
JOBNAME=(*MASTER*,SMSPDSE1),
TSONAME=(MHLRES6),
DSPNAME=('SMSPDSE1'.*),
SDATA=(PSA,CSA,SQA,GRSQ,LPA,LSQA,RGN,SUM,SWA,TRT),
REMOTE=(SYSLIST=(SC63('*MASTER*','SMSPDSE1'),SDATA,DSPNAME)),END
***************************** Bottom of Data ******************
```

Figure 10-64 IEADMCMH parmlib member (single system)

DUMP PARMLIB=MH selects the SYS1.PARMLIB(IEADMCMH) PARMLIB member. Figure 10-65 on page 270 shows messages issued to the SC64 system, and Figure 10-66 on page 270 shows messages issued to the SC63 system.

```
DUMP PARMLIB=MH
IEE252I MEMBER IEADMCMH FOUND IN SYS1.PARMLIB
...
IEA794I SVC DUMP HAS CAPTURED: 213
DUMPID=017 REQUESTED BY JOB (*MASTER*)
DUMP TITLE=PDSE DUMP
IEF196I IGD101I SMS ALLOCATED TO DDNAME (SYS00026)
IEF196I         DSN (DUMP.D1112.H05.SC64.#MASTER#.S00017         )
IEF196I         STORCLAS (SCDUMP) MGMTCLAS (MCDB22) DATACLAS (
)
IEF196I         VOL SER NOS= SBOXAF
...
IEF196I IGD104I DUMP.D1112.H05.SC64.#MASTER#.S00017          RETAINED,
IEF196I DDNAME=SYS00026
IEA611I COMPLETE DUMP ON DUMP.D1112.H05.SC64.#MASTER#.S00017 220
DUMPID=017 REQUESTED BY JOB (*MASTER*)
FOR ASIDS(0001,0078,0076)
REMOTE DUMPS REQUESTED
INCIDENT TOKEN: SANDBOX  SC64     11/12/2004 05:40:44
```

Figure 10-65 Syslog system SC64

```
IEA794I SVC DUMP HAS CAPTURED: 842
DUMPID=006 REQUESTED BY JOB (DUMPSRV )
DUMP TITLE=PDSE DUMP
IEF196I IGD101I SMS ALLOCATED TO DDNAME (SYS00146)
IEF196I         DSN (DUMP.D1112.H05.SC63.DUMPSRV.S00139          )
IEF196I         STORCLAS (SCDUMP) MGMTCLAS (MCDB22) DATACLAS (
)
IEF196I         VOL SER NOS= SBOXB1
...
IEF196I IGD104I DUMP.D1112.H05.SC63.DUMPSRV.S00139           RETAINED,
IEF196I DDNAME=SYS00146
IEA611I COMPLETE DUMP ON DUMP.D1112.H05.SC63.DUMPSRV.S00139 849
DUMPID=006 REQUESTED BY JOB (DUMPSRV )
FOR ASIDS(0005,0001)
REMOTE DUMP FOR SYSNAME: SC64
```

Figure 10-66 Syslog system SC63

Dump suppression (DAE)

Dump analysis and elimination (DAE) suppresses dumps that match a dump you already have. Each time DAE suppresses a duplicate dump, the system does not collect data for the duplicate or write the duplicate to a data set. In this way, DAE can improve dump management by only dumping unique situations and by minimizing the number of dumps.

To perform dump suppression, DAE builds a symptom string if the data for it is available. If the symptom string contains the minimum problem data, DAE uses the symptom string to recognize a duplicate SVC dump or SYSMDUMP dump requested for a software error. When installation parameters request suppression, DAE suppresses the duplicate dump.

The following list summarizes DAE processing:

1. DAE obtains problem data. DAE receives the data in the system diagnostic work area (SDWA) or from values in a SYMREC parameter on the SDUMP or SDUMPX macro that requested the dump.
2. DAE forms a symptom string. DAE adds a descriptive keyword to each field of problem data to form a symptom. DAE forms MVS symptoms, rather than RETAIN symptoms. DAE combines the symptoms for a requested dump into a symptom string.
3. DAE tries to match the symptom string from the dump to a symptom string for a previous dump of the same type. When DAE finds a match, DAE considers the dump to be a duplicate.
4. DAE updates the symptom strings in storage and (when the dump is written to a dump data set) in the DAE data set, if updating is requested.
5. DAE suppresses a duplicate dump, if DAE is enabled for dump suppression.

ADYSETxx PARMLIB members

DAE processing can be modified by activating ADYSETxx PARMLIB members. IBM supplies three ADYSETxx members:

ADYSET00 Starts DAE and keeps 400 symptom strings in virtual storage. The IBM-supplied ADYSET00 member contains:

```
DAE=START,RECORDS(400),
SVCDUMP(MATCH,SUPPRESSALL,UPDATE,NOTIFY(3,30)),
SYSMDUMP(MATCH,UPDATE)
```

ADYSET00 does not suppress SYSMDUMP dumps because installation-provided programs deliberately request them. If desired, change the ADYSETxx member being used to suppress SYSMDUMP dumps.

ADYSET01 Stops DAE processing. The IBM-supplied ADYSET01 member contains:

```
DAE=STOP
```

When using the DAE Display facility's TAKEDUMP (T) action in a sysplex where DAE is active, you must change the contents of ADYSET01 to:

```
DAE=STOP,GLOBALSTOP
```

ADYSET02 or ADYSET03

Contains the same parameters as ADYSET00.

Stopping, starting, and changing DAE

If an ADYSET00 PARMLIB member is used and the DAE data set is allocated, DAE starts during IPL. Normally, DAE should run the whole time that the system is running.

An operator can stop and start DAE with the following steps. One reason to use these steps is to change to a different ADYSETxx PARMLIB member with different parameters.

The operator can *stop* DAE with a SET DAE command that specifies the ADYSET01 PARMLIB member, which contains a DAE=STOP statement: SET DAE=01.

The operator can *start* DAE with a SET DAE command that specifies an ADYSETxx PARMLIB member that contains the DAE=START parameter, such as an installation-provided ADYSET03 PARMLIB member: SET DAE=03. See Figure 10-67 on page 272.

```
SET DAE=03
ADY012I THE FOLLOWING DAE OPTIONS ARE IN EFFECT: 949
   START
   SVCDUMP  = NOTIFY(3,30)  MATCH  UPDATE  SUPPRESSALL
   SYSMDUMP = MATCH  UPDATE  SUPPRESSALL
   RECORDS  = 400
   DSN      = SYS1.DAE.TEST
   SHARE    = DSN  OPTIONS
   GLOBAL   = DSN  OPTIONS
```

Figure 10-67 Start DAE

Stopping DAE is shown in Figure 10-68.

```
SET DAE=01
IEE252I MEMBER ADYSET01 FOUND IN SYS1.PARMLIB
IEF196I IEF285I   SYS1.DAE.TEST
IEF196I IEF285I   VOL SER NOS= SBOX01.
ADY015I DAE STOP PROCESSING IS COMPLETE 517
```

Figure 10-68 Stop DAE

Invoking the IPCS DAE Display Panel

The DAE Display Facility provides a formatted display of any DAE data set. The first display is sorted by the Last Date (most recent incident). Entries can be sorted by placing the cursor on a column heading and pressing Enter, or from the View menu, or by issuing SORT column-heading, for example, Sort Events. (Use the last word of each column heading.)

For ways to invoke the panel, see IPCS option 3.5 in the *z/OS: MVS Interactive Problem Control System (IPCS) User's Guide*, SA22-7596. On the panel, you can:

- View the symptom strings that the data set contains by entering:
  - The date of the dump
  - The last date for the string
  - The number of times the dump has been requested
  - The last system that requested the dump
- Search the Entry list for symptoms, system names, dates, and such.
- Navigate through the sysplex dump directory (or whatever dump directory is active) for the symptom string.
- View the dump title for a symptom string.

Select option `3 UTILITY` from the IPCS PRIMARY OPTION MENU (Figure 10-69 on page 273).

```
------------------- z/OS 01.06.00 IPCS PRIMARY OPTION MENU
OPTION  ===> 3

   0  DEFAULTS    - Specify default dump and options
   1  BROWSE      - Browse dump data set
   2  ANALYSIS    - Analyze dump contents
   3  UTILITY     - Perform utility functions
   4  INVENTORY   - Inventory of problem data
   5  SUBMIT      - Submit problem analysis job to batch
   6  COMMAND     - Enter subcommand, CLIST or REXX exec
   T  TUTORIAL    - Learn how to use the IPCS dialog
   X  EXIT        - Terminate using log and list defaults

Enter END command to terminate IPCS dialog
```

Figure 10-69 IPCS Primary Option Menu

Select option `3 DAE` from the IPCS UTILITY MENU as shown in Figure 10-70.

```
--------------------------------- IPCS UTILITY MENU  --
OPTION  ===> 5

   1  COPYDDIR    - Copy dump directory data
   2  COPYDUMP    - Copy a dump data set
   3  COPYTRC     - Copy trace data
   4  DSLIST      - Process list of data set names
   5  DAE         - Process DAE data

Enter END command to terminate
```

Figure 10-70 IPCS Utility Menu

Enter the DAE data set name and press Enter (Figure 10-71).

```
 DAE Display Facility Dataset Name Input Panel

Enter the DAE Dataset name  . . .  'SYS1.DAE'
        VOLSER (Optional)  . . .
```

Figure 10-71 DAE Display Facility Dataset Name Input Panel

Issue `F string` to search for data contained in the entries. PDSE-related modules start with `IGW`. To find the first entry related to PDSE modules, enter `F IGW` on the command line.

Attention: When PDSE services detects an internal problem, they normally issue an abend with a system completion code of x'0F4' (ABEND 0F4).

DAE Action Codes

Place the cursor on an Action Code and press Enter.

S	Show entry details
T	Take next dump
V	View dump index entry
W	Leave this panel

Select action command **S** to choose a DAE entry, as shown in Figure 10-72.

```
-------------------------------- DAE Display --------------------------------
                                                                 Row 1 to 3 of 3
Command ===>                                                   Scroll ===> CSR
  Enter an Action Code next to an entry.
  Enter / next to an entry to choose from a list of Action Codes.

  Dataset: 'SYS1.DAE.TEST'
  Dumps  since last DAE Display: -119           Total Dumps suppressed: 21
  Events since last DAE Display: -368           Suppression rate:       87%

A  Last      Last      Total   Date of   Symptom String information:
C  Date      System    Events  Dump      Abend  Reason    Module    CSECT
S  11/18/04  SC64           8  11/18/04  SOOF4  OOOOOO2O  IGWBBMF1  IGWBITX1
   11/18/04  SC64           8  11/18/04  SOOF4  OOOOOO24  IGWDCCSO  IGWDSAAC
   11/18/04  SC64           8  11/18/04  SOOF4  OOOOOO24  IGWDLACO  IGWDLCLS
****************************** Bottom of data ******************************
```

Figure 10-72 DAE Display

Figure 10-73 shows the DAE Entry Details for the selected DAE entry.

```
------------------------------ DAE Entry Details ------------ Row 1 to 9 of 9
Command ===>                                                   Scroll ===> CSR

 Total Events: 8         Type: SVC Dump


                            Date      Time       System Name
Last (most recent) Event:  11/18/04  15:31:06   SC64
Dump Taken:                11/18/04  15:28:43   SC64

Symptoms used for Dump Suppression:
MVS     RETAIN
Key     Key     Symptom Data              Explanation
MOD/    RIDS/   IGWBBMF1                  LOAD MODULE NAME
CSECT/  RIDS/   IGWBITX1                  ASSEMBLY MODULE CSECT NAME
PIDS/   PIDS/   5695DF115                 PRODUCT/COMPONENT IDENTIFIER
AB/S    AB/S    OOF4                      ABEND CODE-SYSTEM
REXN/   RIDS/   IGWBSGLR                  RECOVERY ROUTINE CSECT NAME
FI/     VALU/H  71C818F40A0D4130D2845860  FAILING INSTRUCTION AREA
REGS/   REGS/   OEOOA                     REG/PSW DIFFERENCE
REGS/   REGS/   OCD74                     REG/PSW DIFFERENCE
HRC1/   PRCS/   OOOOOO2O                  ABEND REASON CODE
****************************** Bottom of data ******************************
```

Figure 10-73 DAE Entry Details

Note: The abend reason code corresponds to the PDSE return code and not to the PDSE reason code.

Figure 10-74 on page 275 shows a sample IPCS inventory when you specify the action command **V**.

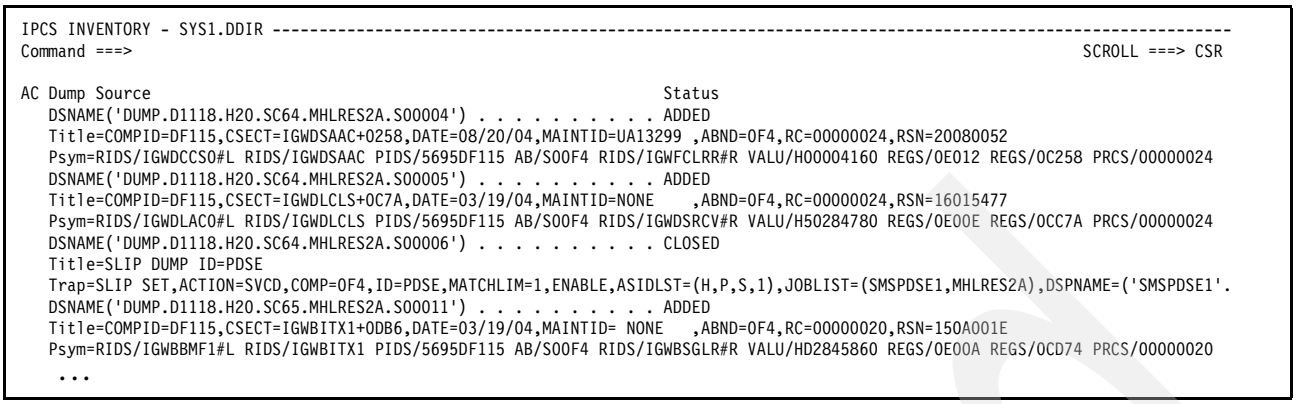


Figure 10-74 IPCS Inventory

These line commands may be typed in the space preceding the name of the data source:

BR Activates the BROWSE option of the IPCS dialog for that source.

CL CLOSEs the source and releases resources obtained by OPEN processing.

DD Deletes description of the source and, optionally, the source data set.

DT Deletes translation results for the source.

LA Lists dump description with storage attributes.

LB Lists dump description with record locations.

LD Lists dump description with dumped storage summary.

LT Lists dump description with translation results.

LZ Lists dump description with storage attributes, record locations, dumped storage summary, and translation results.

OP OPENs the source.

SD Establishes the source as both the local and global IPCS default.

See *z/OS: MVS Interactive Problem Control System (IPCS) User's Guide*, SA22-7596 for more details on using IPCS.

How can I request an SDUMP on next occurrence?

Attention: PDSE-related modules start with *IGW*.

When PDSE services detects an internal problem, they normally issue an abend with a system completion code of x'0F4' (ABEND 0F4)

There are two ways to request an SDUMP on next occurrence if a dump is currently suppressed by DAE:

- By clearing the DAE data set:
  a. Stop DAE by issuing a SET DAE=01 command.
  b. Clear the DAE data set of `IGW` and `0F4` entries by using ISPF edit functions.
  c. Start DAE by issuing either the SET DAE=01 or the SET DAE=03 command.
- By specifying the action command **T** (Take Next Dump) on the DAE Display menu. See Figure 10-72 on page 274.

IPCS basics

The interactive problem control system (IPCS) is provided in z/OS to aid in diagnosing software failures. IPCS provides formatting and analysis support for dumps and traces produced by MVS, other program products, and applications that run on a z/OS system.

Note: We do not explain how to diagnose MVS or PDSE dumps by using IPCS tools, but we provide some helpful IPCS diagnosis actions to verify the status of a dump and collect helpful information from a dump that can be used to search IBM product support databases.

Selecting dump

You must specify a default dump data set name before you can diagnose a dump. Select IPCS option **0** (Figure 10-75).

```
------------------- z/OS 01.06.00 IPCS PRIMARY OPTION MENU
OPTION  ===> 0

   0  DEFAULTS    - Specify default dump and options
   1  BROWSE      - Browse dump data set
   2  ANALYSIS    - Analyze dump contents
   3  UTILITY     - Perform utility functions
   4  INVENTORY   - Inventory of problem data
   5  SUBMIT      - Submit problem analysis job to batch
   6  COMMAND     - Enter subcommand, CLIST or REXX exec
   T  TUTORIAL    - Learn how to use the IPCS dialog
   X  EXIT        - Terminate using log and list defaults

Enter END command to terminate IPCS dialog
```

Figure 10-75 IPCS primary option menu

Specify the dump data set name for your IPCS analysis as shown in Figure 10-76. Press Enter to save your entered data and PF3 to return to the IPCS primary option menu.

```
------------------------ IPCS Default Values --------------------------------
Command ===>

  You may change any of the defaults listed below.  The defaults shown before
  any changes are LOCAL.  Change scope to GLOBAL to display global defaults.

  Scope   ==> LOCAL   (LOCAL, GLOBAL, or BOTH)

  If you change the Source default, IPCS will display the current default
  Address Space for the new source and will ignore any data entered in
  the Address Space field.

  Source  ==> DSNAME('DUMP.D1118.H20.SC64.MHLRES2A.S00006')
  Address Space   ==>
  Message Routing ==> NOPRINT TERMINAL
  Message Control ==> CONFIRM VERIFY FLAG(WARNING)
  Display Content ==> NOMACHINE REMARK REQUEST NOSTORAGE SYMBOL

Press ENTER to update defaults.

Use the END command to exit without an update.
```

Figure 10-76 IPCS Default Values

IPCS LIST TITLE and SLIPTRAP

Use the IP LIST TITLE command for an overview of what the dump might represent. System generated dumps typically have a COMPID= and other generated information, depending on the recovery routine that takes the dump. The component identification keyword (COMPID) for PDSE components is 5695DF115. The short form is COMPID=DF115 as shown in the dump title in Figure 10-80 on page 278.

- Whatever the user puts in COMM= as the dump title will be the console dump title.
- Dumps taken as a result of a SLIP trap have SLIPTRAP as the title, as in Figure 10-78.
- Any program can issue an SDUMP macro and generate a title of its choosing. For IBM products, see in *z/OS: MVS Diagnosis: Reference*, GA22-7588 for a dump title directory.

```
****************************************************** TOP OF DATA *********

TITLE
LIST 00. LITERAL LENGTH(X'11') CHARACTER
00000000 | SLIP DUMP ID=PDSE                                              |
****************************************************** END OF DATA *********
```

Figure 10-77 IP LIST TITLE screen

Use IP LIST SLIPTRAP if you believe that a SLIP trap was used to produce the dump and you want to know what was set. If a SLIP trap was used, output similar to Figure 10-78 appears.

If a SLIP trap was not used, you will see the message `Symbol SLIPTRAP not found`.

```
****************************************************** TOP OF DATA **********

SLIPTRAP
LIST 00. LITERAL LENGTH(X'AE') CHARACTER
00000000 | SLIP SET,ACTION=SVCD,COMP=0F4,ID=PDSE,MATCHLIM=1,ENABLE,ASIDLST= |
00000040 | (H,P,S,1),JOBLIST=(SMSPDSE1,MHLRES2A),DSPNAME=('SMSPDSE1'.*),SDA |
00000080 | TA=(CSA,PSA,SQA,LSQA,LPA,TRT,SUM,SWA,RGN,GRSQ)                   |
****************************************************** END OF DATA **********
```

Figure 10-78 IP LIST SLIPTRAP screen

IPCS STATUS

Use the STATUS subcommand to display data usually examined during the initial part of the problem determination process. STATUS produces different diagnostic information depending on the report type parameter or parameters entered.

IPCS STATUS WORKSHEET

IPCS STATUS WORKSHEET displays useful information. The example in Figure 10-79 on page 278 is based on a SLIP trap on a completion code 0F4:

- Dump title

  ```
  Dump Title: SLIP DUMP ID=PDSE
  ```

- Date and time dump was taken

  ```
  Date: 11/18/2004     Time: 15:32:43.805802 Local
  ```

- Original dump data set name (for example, can be useful for reference with syslogs)

  ```
  Original dump dataset: DUMP.D1118.H20.SC64.MHLRES2A.S00006
  ```

- Dump ID (can be useful for reference with syslogs)

  ```
  Dump ID: 006
  ```

See Figure 10-55 on page 264 as a reference for messages when the SLIP matches. IEA794I and IEA611I messages refer to the dump ID as well.

```
                                MVS Diagnostic Worksheet

Dump Title: SLIP DUMP ID=PDSE

CPU Model 2084 Version 00 Serial no. 116A3A Address 01
Date: 11/18/2004     Time: 15:32:43.805802 Local

Original dump dataset: DUMP.D1118.H20.SC64.MHLRES2A.S00006

Information at time of entry to SVCDUMP:

HASID 001A  PASID 0009  SASID 001A  PSW 075C0000 84661106

CML ASCB address 00000000  Trace Table Control Header address 7F6FA000

Dump ID: 006
Error ID: N/A

SDWA address N/A
```

Figure 10-79 IP status worksheet: SLIP on completion code D(ump ID=003)

The worksheet in Figure 10-80 is based on system abend 0F4 dump. It shows the Error ID, which can be useful for reference with syslogs; for example:

```
Error ID: Seq 00579  CPU 0000  ASID X'001A'  Time 15:28:43.8
```

```
******************************************************* TOP OF DATA ***********************************

                                MVS Diagnostic Worksheet

Dump Title: COMPID=DF115,CSECT=IGWBITX1+0DB6,DATE=03/19/04,MAINTID= NONE   ,ABND=0F4,RC=00000020,RSN=150A001E

CPU Model 2084 Version 00 Serial no. 116A3A Address 01
Date: 11/18/2004     Time: 15:28:44.297630 Local

Original dump dataset: DUMP.D1118.H20.SC64.MHLRES2A.S00003

Information at time of entry to SVCDUMP:

HASID 001A  PASID 0009  SASID 001A  PSW 070C0000 81ED8FF2

CML ASCB address 00000000  Trace Table Control Header address 7F678000

Dump ID: 003
Error ID: Seq 00579  CPU 0000  ASID X'001A'  Time 15:28:43.8

SDWA address 7F2177C8
```

Figure 10-80 IP status worksheet: ABEND0F4 SDUMP (dump ID=006)

IPCS STATUS SYSTEM

Figure 10-81 on page 279 shows an IP STATUS SYSTEM output that is based on a SLIP trap on a completion code 0F4 (dump ID=003).

- Displays a TOD clock value placed in the dump to indicate when the dump was produced. The TOD clock value is in hexadecimal and in a date and time of day for local time and

Greenwich mean time (GMT). To determine local time, the system uses field CVTTZ in the CVT.

```
TIME OF DAY CLOCK: BC23A62B 6086A0E8  11/18/2004 15:32:43.805802 local
TIME OF DAY CLOCK: BC23E939 83C6A0E8  11/18/2004 20:32:43.805802 GMT
```

- Identifies the programs requesting and producing the dump:

```
Program Producing Dump: SLIPDUMP
```

```
SYSTEM STATUS:
  Nucleus member name: IEANUC01
   I/O configuration data:
     IODF data set name: SYS6.IODF61
     IODF configuration ID: L06RMVS1
     EDT ID: 01
  Sysplex name: SANDBOX
  TIME OF DAY CLOCK: BC23A62B 6086A0E8  11/18/2004 15:32:43.805802 local
  TIME OF DAY CLOCK: BC23E939 83C6A0E8  11/18/2004 20:32:43.805802 GMT
  Program Producing Dump: SLIPDUMP
  Program Requesting Dump: IEAVTSDT
```

Figure 10-81 IP system status (SLIP on completion code)

The IP STATUS SYSTEM output in Figure 10-82 is based on a ABEND 0F4 SDUMP (dump ID=006).

- Normally the system generates an incident token when an SVC dump is requested. The incident token refers to sysplex and system names.

```
Incident token: SANDBOX  SC64     11/18/2004 20:28:43.823465 GMT
```

```
SYSTEM STATUS:
  Nucleus member name: IEANUC01
  I/O configuration data:
    IODF data set name: SYS6.IODF61
    IODF configuration ID: L06RMVS1
    EDT ID: 01
  Sysplex name: SANDBOX
  TIME OF DAY CLOCK: BC23A546 F6D9ECAC  11/18/2004 15:28:44.297630 local
  TIME OF DAY CLOCK: BC23E855 1A19ECAC  11/18/2004 20:28:44.297630 GMT
  Program Producing Dump: SVCDUMP
  Program Requesting Dump: IEAVTSDT


  Incident token: SANDBOX  SC64     11/18/2004 20:28:43.823465 GMT
```

Figure 10-82 IP STATUS SYSTEM (SDUMP)

IPCS STATUS REGS

The IP STATUS REGS command tells what the registers were at time of the dump:

- For slip dump, REGS at time SLIP matched
- For console dump, typically all zeros
- For abend dump, theoretically the REGS at time of abend

See “General Purpose Register (GPR) usage information for ABEND 0F4” on page 251 for more information.

Figure 10-83 shows an example output for a IP STATUS REGS command for an ABEND 0F4 (dump ID=003 and dump ID=006).

- Displays the program status word (PSW) followed by a description of what the PSW indicates.

```
PSW=075C0000 84661106
     (Running in PRIMARY, key 5, AMODE 31, DAT ON)
```

- Displays the current ASCB, ASXB, and/or TCB.

  The output might also display the processor status. `HOME ASID: hhhh PRIMARY ASID: pppp SECONDARY ASID: ssss` IPCS identifies the applicable address spaces (in hexadecimal) relevant to the unit of work running on the CPU at the time of the dump:

  – hhhh is the home address space identifier.
  – pppp is the primary address space identifier.
  – ssss is the secondary address space identifier.

```
ASCB26 at F9FA80, JOB(MHLRES2A), for the home ASID
TCB26E at 7DDC48 for the home ASID
   HOME ASID: 001A PRIMARY ASID: 0009 SECONDARY ASID: 001A
```

- Displays the general purpose register (GPR) contents in hexadecimal:

```
General purpose register values
     Left halves of all registers contain zeros
GPR 0:  150A001E  →  RSN150A001E
GPR 1:  440F4000  →  ABEND 0F4
GPR 15: 00000020  →  RC20
```

```
CPU STATUS:
PSW=075C0000 84661106
    (Running in PRIMARY, key 5, AMODE 31, DAT ON)
    DISABLED FOR PER
   ASID(X'0009') 04661106. IGWBBMF1+04A106 IN EXTENDED PLPA
  ASCB26 at F9FA80, JOB(MHLRES2A), for the home ASID
  ASXB26 at 7FDD00 and TCB26E at 7DDC48 for the home ASID
  HOME ASID: 001A PRIMARY ASID: 0009 SECONDARY ASID: 001A

  General purpose register values
    Left halves of all registers contain zeros
     0-3  150A001E  440F4000  00000000  0281B000
     4-7  00000020  00FD6438  00FDBB40  04661C1C
     8-11 150A001E  04662390  04661391  00FDBBB0
    12-15 84660392  7F4E2898  846610FC  00000020

  Access register values
      0-3  00000000  00000000  00000000  00000002
      4-7  00000002  00000000  00000000  00000000
     8-11 00000000  00000000  00000000  00000000
    12-15 00000000  00000000  00000000  00000000
...
```

Figure 10-83 IP status regs: SLIP dump (dump ID=003)

Important: Use module names, system abend code, reason codes, and return codes to search in IBM product support databases.

The RTCT control block: getting information about what was dumped

The RTCT (Recovery/Termination Control Table) contains information about what can be expected to be found in the dump. The command IP CBF RTCT shows what ASIDs were requested under the SDAS heading (Figure 10-84).

The dumped address spaces are identified by their ASIDs in the SDAS column. All values greater than zero identify an address space. Dumped ASIDs: 0001, 0009, and 001A.

```
RTCT: 00F4EB20
  ...
  ...
          ASTB

              SDAS  SDF4  SDF5
              ----  ----  ----
          001 001A  A0    00
          002 0009  80    00
          003 0001  F8    00
          004 0000  00    00
```

Figure 10-84 IP CBF RTCT (dump ID=003)

IPCS SELECT ALL

The IPCS SELECT subcommand generates Address Space storage map entries. Use this subcommand to produce a report that displays the ASID and the associated job name.

Figure 10-85 shows the IP SELECT ALL output for dump ID=003. In reference to Figure 10-84:

```
ASID JOBNAME
0001 *MASTER*
0009 SMSPDSE1
001A MHLRES2A
```

```
 ASID JOBNAME  ASCBADDR  SELECTION CRITERIA
 ---- -------- --------  ------------------
 0001 *MASTER* 00FD4F00  ALL
 0002 PCAUTH   00F95880  ALL
 0003 RASP     00F95700  ALL
 0004 TRACE    00FB8B00  ALL
 0005 DUMPSRV  00FB8980  ALL
 0006 XCFAS    00FB8800  ALL
 0007 GRS      00FB8680  ALL
 0008 SMSPDSE  00F95E80  ALL
 0009 SMSPDSE1 00F95D00  ALL
 ...
 001A MHLRES2A 00F9FA80  ALL
 ...
```

Figure 10-85 IP SELECT ALL (dump ID=003)

IPCS VERBX LOGDATA

Specify the LOGDATA verb name on the VERBEXIT subcommand to format the logrec buffer records that were in storage when the dump was generated. LOGDATA locates the logrec records in the logrec recording buffer and invokes the EREP program to format and print the logrec records. The records are formatted as an EREP detail edit report. See “EREP and LOGREC recording” on page 283 for more details about EREP(LOGDATA) reports.

Use the LOGDATA report to examine the system errors that occurred just before the error that caused the dump to be requested.

The IP VERBEXIT LOGDATA subcommand has no additional parameters.

IPCS VERBX SMSXDATA

Use the VERBEXIT subcommand to run an installation-supplied or IBM-supplied verb exit routine. IPCS VERBEXIT SMSXDATA supports dump formatting of multiple DFSMS components. Control blocks are formatted as hexadecimal data.

An IPCS verb exit can be invoked during an IPCS session or from a batch job.

The VERBX SMSXDATA output summarizes:

- PDSE control blocks
- Entry points of CSECTs of multiple DFSMS components

  Each line in the list of CSECTs/modules provides information about a CSECT or an external reference such as:

  IGWBLMPS 800D 0000 **026CF378 00002CB0** 0000 09 0B 026A7720 **UA10649 05/06/04**

  – Name of entry point (CSECT/module)

    IGWBLMPS

  – Entry point address

    026CF378

  – Length (if CSECT)

    00002CB0

  – Maintenance levels and

    UA10649

  – Compile date

    05/06/04

  To locate a corresponding entry, perform **FIND *modulename*** for the module name. For example, issue F IGWBLMPS to locate the module name IGWBLMPS (Figure 10-86).

```
********************************************************* TOP OF DATA ***********************************************************



DFSMS verbexit processing
...
IGWCAT: 026A6C20
   +0000  C9C7E6C3  C1E34040  000153E0  01000000  01000000  00000714  00000000  00000000  | IGWCAT  ...\..................... |

 RY_NM    G    D    RESS     GTH      D          T_PTR    NT       E
 -------- ---- ---- -------- -------- ---- -- -- -------- -------- --------
 IGGADMAT 8000 0000 00D36000 000000A8 0000 03 63 026A6C70 NONE     03/19/04
 IGG019V7 800F 0000 00D360A8 000012D0 0000 03 0F 00000000 NONE     03/19/04
 ...
 IGWDQCRS 8016 0000 026CE5A0 00000DD8 0000 0B 16 026A76F0 UA13353  08/23/04
 IGWBLMPS 800D 0000 026CF378 00002CB0 0000 09 0B 026A7720 UA10649  05/06/04
 IGWBIPNR 8014 0000 026D2028 00000238 0000 15 14 026A7750 \.{.....   2004.0
 IGWBLMPR 8018 0000 026D2260 000014D0 0000 09 0F 026A7780 NONE     03/19/04
 IGWBPDX0 8003 0000 026D3730 00000FE0 0000 1B 0A 026A77B0 NONE     03/19/04
```

Figure 10-86 IP VERBX SMSXDATA

10.6.3 EREP and LOGREC recording

When an error occurs, the system records information about the error in the logrec data set or the logrec log stream (LOGREC). The information provides a history of all hardware failures, selected software errors, and selected system conditions. Use the Environmental Record, Editing, and Printing program (EREP):

- To print reports about the system records
- To determine the history of the system
- To learn about a particular error

Use the records in the logrec data set or the logrec log stream as additional information when a dump is produced. The information in the records points you in the right direction and supplies symptom data about the failure.

You can set up your system to record errors on either a *logrec data set* or in a *logrec log stream*.

The system creates records for every hardware or software failure and system condition that occurs. It stores these records in the logrec data set or the logrec log stream. The records can contain two types of data that document failures and system conditions:

- Error statistics, which include the number of times that channels, machine models, and I/O devices have failed
- Environmental data, which include time and circumstances for each failure or system condition

A programmer can also build symptom records using the SYMRBLD macro and have those records written into the logrec data set or the logrec log stream using the SYMREC macro.

In addition to producing a system dump, PDSE routines write a software record or symptom record to the logrec data set or to the logrec log stream if they detect an internal failure in PDSE processing. EREP reads these records and processes them to produce the requested report.

See *EREP User's Guide*, GC35-0151, and *EREP Reference,* GC35-0152, for more information about using EREP.

Software-related recording

Software records and *symptom records* are written to the LOGREC if an unexpected situation is detected by the software. A software record based on a system abend (ABEND 0F4 as well as ABEND 0C4) includes information similar to a dump title as shown in “System ABENDs” on page 250. The PDSE symptom records can also be used to identify incidents associated with PDSE ABENDs.

When PDSE symptom records occur without an ABEND 0F4, use the symptom strings to search for matching problems in the problem reporting database. If no fix exists, contact the IBM Support Center.

We recommend always to check LOGREC for previously issued abends that might have been suppressed, because Dump Analysis and Elimination (DAE) suppresses dumps that match an existing dump. The detail edit report for an abend provides information about whether a dump was started. See “Dump suppression (DAE)” on page 270 for more information about DAE if a dump that is required for further analysis was suppressed.

You can use EREP to print reports from software and symptom records in logrec data sets or logrec log streams.

Important: It is important to identify which error occurred first by scanning scan all entries (software and symptom records) in LOGREC. The sequence number (SEQ) indicates the order of the errors, but the records might not be listed in order.

The *PDSE reason code* is one of the most helpful information for identifying a problem in PDSE services. We recommend using the reason code as a search argument to search in IBM product support databases to verify that a known problem has occurred. Use the keyword prefix *RSN* before the reason code.

See “General Purpose Register (GPR) usage information for ABEND 0F4” on page 251 for more information about GPR usage for ABEND 0F4 system abends.

Using EREP

EREP presents information from the logrec software error records in different reports.

Tip: We recommend that you gather two reports when sending LOGREC data to an IBM support center for further analysis:

1. Event history report (EVENT=Y)
2. Detail edit report (PRINT=AL)

Detail edit report for an abend

The detail edit report for a software record shows the complete contents of an error record for an abnormal end, including the system diagnostic work area (SDWA). The report is produced by EREP and, through the VERBEXIT LOGDATA subcommand, under IPCS.

The system obtains most of the information for an abend logrec error record from the system diagnostic work area (SDWA). The report contents are:

- Record header: report type (SOFTWARE RECORD), system, job name, error identifier (ERRORID), date, and time
- Search argument abstract
- Serviceability information
- Time of error information
- Status information from the request block
- Recovery environment
- Recovery routine action
- Hexadecimal dump of the SDWA, including the variable recording area (VRA)

Use the detail edit report for a software record to determine the cause of an abend and any recovery action that the system or application has taken. This report enables you to locate where an error occurred, similar to the analysis of an SVC dump. After you locate the error, you can develop a search argument to obtain a fix for the problem.

See *EREP User's Guide*, GC35-0151, for information about producing a detail edit report for an SDWA-type record. See *z/OS MVS IPCS Commands*, SA22-7594 for information about the VERBEXIT LOGDATA subcommand.

Figure 10-87 on page 285 shows EREP parameters for generating a detail edit report.

Figure 10-93 on page 289 and Figure 10-94 on page 290 show a sample *software record* software edit report.

Detail edit report for a symptom record

The system obtains most of the information for a non-abend logrec error record from the symptom record identified in the SYMREC macro. The report is produced by EREP and, through the VERBEXIT LOGDATA subcommand, under IPCS. The report contents are:

- Record reader: report type (SYMPTOM RECORD), system, date, and time
- Search argument abstract
- System environment
- Component information
- Primary and secondary symptom strings
- Free-format component information
- Hexadecimal dump of the symptom record

Figure 10-87 shows the EREP parameters to generate a detail edit report. The detail software records are listed first, then the detail symptom records after all software records. You could locate the first detail symptom string with a FIND command such as F 'SYMPTOM RECORD'.

```
//SYSIN    DD *
   HIST
   ACC=N
   EVENT=N
   TYPE=S
   PRINT=PT
/*
```

Figure 10-87 Detail edit report

Figure 10-96 on page 291 shows an example of a *symptom record* software edit report.

Event history report

The report shows the error history: frequency, order, and pattern of errors. Its contents are:

- Record header: report type (EVENT HISTORY)
- Abstracts for abend and non-abend logrec error records in chronological order
- Totals of the types of logrec error records for the system and for each processor

Figure 10-88 and Figure 10-98 on page 292 show the EREP parameters to generate an event history report. Figure 10-99 on page 293 shows an example of an event history report.

```
//SYSIN    DD *
   HIST
   ACC=N
   EVENT=Y
   TYPE=S
   PRINT=AL
/*
```

Figure 10-88 Event history report

Detail summary report

This report summarizes information about data in logrec error records. Its contents are:

- Record header: report type being summarized
- Summary information and counts

Figure 10-89 on page 286 shows the EREP parameters to generate a detail summary report.

```
//SYSIN    DD *
   HIST
   ACC=N
   EVENT=N
   TYPE=S
   PRINT=SU
/*
```

Figure 10-89 Detail summary report

Figure 10-95 on page 290 shows an example of a *software record* software summary report. Figure 10-97 on page 292 shows an example of a *symptom record* software summary report.

Related EREP parameters

TYPE Record type (selection parameter)

Select only the specified types of records.

Syntax: TYPE=code[code]...

S - Software (SFT): system abends and other software events

EVENT Event history (report parameter)

Produce an event history report (one-line abstracts of selected records in chronological order). EVENT=N generates a detail edit report.

Syntax: EVENT[=Y] | **=N**

HIST History input (processing parameter)

Use the records in a history file, instead of those in the ERDS, for the requested report.

Syntax: HIST[=Y] | **=N**

DATE Date range (selection parameter)

Select records created during the specified date range.

Syntax: DATE=({yyddd[,yyddd] | yyddd[-yyddd]})

yyddd is the year yy and the Julian day ddd. The first yyddd is the year and day when the date range begins; the second yyddd is the ending year and day. The second date is optional; you can select records from a single date as well as from a range of dates. To select a single date, code only one yyddd.

When you code a date range, the second yyddd must be greater than or equal to the first. If it is not, EREP issues a syntax-error message.

TIME Time range (selection parameter)

Select only those records created during the specified time period.

Syntax: TIME=({hhmm,hhmm | hhmm-hhmm})

hhmm is a valid time period, hours and minutes.

Defaults: EREP selects records regardless of when they were created.

Coding:

You must always code DATE when you code TIME.

Code hhmm using a 24-hour clock (for example: 1400 for 2 p.m.).

PRINT Print reports (report parameter)

Produce the PRINT reports specified (or produce no report output).

Syntax: PRINT={AL | DR | NO | PS | PT | SD | SU}

AL requests all detail (PRINT) reports: detail edits of the records, detail summaries, and, if applicable, data reduction reports.

DR requests only data reduction reports.

NO requests that no reports be generated at all.

PS requests both detail edit and detail summary reports.

PT requests only detail edit reports.

SD requests detail summaries and data reduction reports.

SU requests only detail summary reports.

Using EREP to obtain records from the Logrec Log Stream

You can use EREP to access the records in the logrec log stream for each system. The log stream subsystem enables existing programs to access error records from a log stream in the same way records were accessed from a logrec data set.

Use the SUBSYS parameter to invoke the LOGR subsystem to access log stream data:

```
//ddname   DD  DSNAME=log.stream.name,
//           SUBSYS=(LOGR[,exit_routine_name][,'SUBSYS-options1'][,'SUBSYS-options2'])
```

SUBSYS-options1:

- [FROM={({[yyyy/ddd][,hh:mm[:ss]]}) | OLDEST}]
- [TO={({[yyyy/ddd][,hh:mm[:ss]]}) | YOUNGEST}]
- [,DURATION=(nnnn,HOURS)]
- [,VIEW={ACTIVE|ALL|INACTIVE}]
- [,GMT|LOCAL]

SUBSYS-options2: defined by the log stream owner

See “Chapter 14. Recording Logrec Error Records” in *z/OS: MVS Diagnosis: Tools and Service Aids*, GA22-7589 for more details.

Examples

A DISPLAY LOGREC lists the LOGSTREAM name (=SYSPLEX.LOGREC.ALLRECS) that is used in the EREP job (Figure 10-90).

```
D LOGREC
IFB090I   15.09.26  LOGREC DISPLAY
 CURRENT MEDIUM = LOGSTREAM
    MEDIUM NAME = SYSPLEX.LOGREC.ALLRECS
         STATUS = CONNECTED
```

Figure 10-90 DISPLAY LOGREC

Figure 10-91 shows a sample job to request a software edit report (TYPE=S) including detail edits of the records and detail summaries (PRINT=AL).

```
//*****************************************************************/
//* PRODUCING AN EREP REPORT FROM LOGSTREAM
//*****************************************************************/
/*JOBPARM L=999,SYSAFF=SC64
//EREPDALY EXEC  PGM=IFCEREP1,PARM=('CARD')
//ACCIN      DD DSN=SYSPLEX.LOGREC.ALLRECS,DISP=SHR,
//           DCB=(RECFM=VB,BLKSIZE=4000),
//           SUBSYS=(LOGR,IFBSEXIT,
//           'FROM=(2004/323,16:00),TO=(2004/323,22:00)',)
//DIRECTWK DD UNIT=VIO,SPACE=(CYL,15,,CONTIG)
//TOURIST  DD SYSOUT=*,DCB=BLKSIZE=133
//EREPPT   DD SYSOUT=*,DCB=BLKSIZE=133
//SYSIN    DD *
   HIST
   ACC=N
   EVENT=N
   TYPE=S
   PRINT=AL
/*
```

Figure 10-91 Example EREP job (TYPE=S, EVENT=N, and PRINT=AL)

This EREP job generates the following reports:

- PARAMETER OPTIONS VALID FOR THIS EXECUTION (Figure 10-92).

```
PARAMETER OPTIONS VALID FOR THIS EXECUTION
   RECORD TYPES(SOFT),MODE ALL,PRINT(EDIT,SUMMARY,DATA REDUCTION),HISTORY INPUT
   DATE/TIME RANGE - ALL
   TABLE SIZE - 0024K,LINE COUNT - 050
   LINE LENGTH - 132

IFC120I      95    RECORDS THAT PASSED FILTERING
...
```

Figure 10-92 EREP detail summary report (PRINT=AL) part 1

- SOFTWARE RECORD type:
  - Software edit report (Figure 10-93 and Figure 10-94 on page 290)
  - Software summary report (Figure 10-95 on page 290)

```
TYPE:  SOFTWARE RECORD      REPORT:  SOFTWARE EDIT REPORT          DAY.YEAR
       (SVC 13)                                        REPORT DATE: 328.04
FORMATTED BY: IEAVTFDE  HBB7703                         ERROR DATE: 323.04
                          MODEL:   2084                             HH:MM:SS.TH
                          SERIAL:  116A3A                  TIME: 14:59:39.10
JOBNAME: MHLRES2A   SYSTEM NAME: SC64
ERRORID: SEQ=00309  CPU=0000  ASID=001A  TIME=14:59:39.0

SEARCH ARGUMENT ABSTRACT
  PIDS/5695DF115 RIDS/IGWBBMF1#L RIDS/IGWBITX1 AB/S00F4 PRCS/00000020 REGS/0E00A
  REGS/0CD74 RIDS/IGWBSGLR#R

  SYMPTOM             DESCRIPTION
  -------             -----------
  PIDS/5695DF115      PROGRAM ID: 5695DF115
  RIDS/IGWBBMF1#L     LOAD MODULE NAME: IGWBBMF1
  RIDS/IGWBITX1       CSECT NAME: IGWBITX1
  AB/S00F4            SYSTEM ABEND CODE: 00F4
  PRCS/00000020       ABEND REASON CODE: 00000020
  REGS/0E00A          REGISTER/PSW DIFFERENCE FOR R0E: 00A
  REGS/0CD74          REGISTER/PSW DIFFERENCE FOR R0C: D74
  RIDS/IGWBSGLR#R     RECOVERY ROUTINE CSECT NAME: IGWBSGLR
OTHER SERVICEABILITY INFORMATION
  RECOVERY ROUTINE LABEL:  IGWFRCSD
  DATE ASSEMBLED:          03/19/04
  MODULE LEVEL:             NONE

SERVICEABILITY INFORMATION NOT PROVIDED BY THE RECOVERY ROUTINE
  SUBFUNCTION
```

Figure 10-93 EREP detail summary report (PRINT=AL) part 2

Important: A detailed software record based on a system abend indicates in the RECOVERY ROUTINE ACTION section whether a dump was started.

Information referred to in RECOVERY ROUTINE ACTION could include:

- AN SVC DUMP WAS NOT REQUESTED.
- THE REQUESTED SVC DUMP WAS SUCCESSFULLY STARTED.
- THE REQUESTED SVC DUMP WAS NOT TAKEN. ANOTHER DUMP WAS IN PROGRESS.
- THE REQUESTED SVC DUMP WAS NOT TAKEN. **THE DUMP WAS SUPPRESSED BY DAE.**

```
TIME OF ERROR INFORMATION
  PSW: 075C0000 84661106   INSTRUCTION LENGTH: 02   INTERRUPT CODE: 000D
  FAILING INSTRUCTION TEXT: 71C818F4 0A0D4130 D2845860

  REGISTERS 0-7
  GR: 150A001E 440F4000 00000000 0281B000  00000020 00FD6438 00FDBB40 04661C1C
  AR: 00000000 00000000 00000000 00000002  00000002 00000000 00000000 00000000
  REGISTERS 8-15
  GR: 150A001E 04662390 04661391 00FDBBB0  84660392 7F4E2898 846610FC 00000020
  AR: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000

  HOME ASID: 001A    PRIMARY ASID: 0009    SECONDARY ASID: 001A
  PKM: 84C0          AX: 0001              EAX: 0000

NO LOCKS WERE REQUESTED TO BE FREED.
  THE SDWA WAS REQUESTED TO BE FREED BEFORE RETRY.
...
RTM WAS ENTERED BECAUSE AN SVC WAS ISSUED IN AN IMPROPER MODE.
  THE ERROR OCCURRED WHILE AN ENABLED RB WAS IN CONTROL.
  NO LOCKS WERE HELD.
  NO SUPER BITS WERE SET.

STATUS FROM THE LINKAGE STACK ENTRY
  PSW: 075C2000 84785FB2   INSTRUCTION LENGTH: 02   INTERRUPT CODE: 000D

RECOVERY ENVIRONMENT
...
RECOVERY ROUTINE ACTION
  THE RECOVERY ROUTINE RETRIED TO ADDRESS 04660840.
  THE REQUESTED SVC DUMP WAS NOT TAKEN. ANOTHER DUMP WAS IN PROGRESS.
...
VARIABLE RECORDING AREA (SDWAVRA)
...
```

Figure 10-94 EREP detail summary report (PRINT=AL) part 3

```
TYPE:  SOFTWARE RECORD        REPORT:  SOFTWARE SUMMARY                DAY.YEAR
                                                          REPORT DATE: 328.04
FORMATTED BY: IEAVTFDS  HBB7706     MODEL:   N/A          PERIOD FROM: 323.04
                                    SERIAL:  N/A                   TO: 323.04
    COUNT OF SOFTWARE ERROR RECORDS PROCESSED:   0000000064
    COUNT OF UNIQUE SOFTWARE ERROR RECORDS:      0000000014
 PIDS/5695DF107 RIDS/IGC0013I#L RIDS/ICVDSD03 AB/S018B PRCS/0000AA28
              COUNT: 0000000001    FIRST: 04.323 14:05:39    LAST:  04.323 14:05:39
...
 PIDS/5695DF115 RIDS/IGWAFMS0#L RIDS/######## AB/S0213 PRCS/00000050
              COUNT: 0000000013    FIRST: 04.323 14:59:39    LAST:  04.323 15:32:44
 PIDS/5695DF115 RIDS/IGWBBMF1#L RIDS/IGWBITX1 AB/S00F4 PRCS/00000020
              COUNT: 0000000013    FIRST: 04.323 14:59:39    LAST:  04.323 15:32:44
 PIDS/5695DF115 RIDS/IGWDCCS0#L RIDS/IGWDSAAC AB/S00F4 PRCS/00000024
              COUNT: 0000000013    FIRST: 04.323 14:59:39    LAST:  04.323 15:32:44
 PIDS/5695DF115 RIDS/IGWDLAC0#L RIDS/IGWDLCLS AB/S00F4 PRCS/00000024
              COUNT: 0000000013    FIRST: 04.323 14:59:39    LAST:  04.323 15:32:44
...
```

Figure 10-95 EREP detail summary report (PRINT=AL) part 4

- SYMPTOM RECORD type:
  - Software edit report (Figure 10-96)
  - Software summary report (Figure 10-97 on page 292)

```
TYPE:  SYMPTOM RECORD       REPORT:  SOFTWARE EDIT REPORT          DAY YEAR
                                                    REPORT DATE: 328  04
SCP:   VS 2 REL 3                                    ERROR DATE: 323  04
                            MODEL:  2084                          HH MM SS.TH
                            SERIAL: 116A3A                 TIME: 14:59:39.17
SEARCH ARGUMENT ABSTRACT:
    PIDS/5694DF115 LVLS/DFSMS RIDS/IGWBDSL2 PRCS/07100050
    PRCS/00000008

SYSTEM ENVIRONMENT:
    CPU MODEL:  2084               DATE:  323  04
    CPU SERIAL: 116A3A             TIME:  14:59:39.17
    SYSTEM:     SC64               BCP:   MVS
    RELEASE LEVEL OF SERVICE ROUTINE:      HBB7709
    SYSTEM DATA AT ARCHITECTURE LEVEL:     10
    COMPONENT DATA AT ARCHITECTURE LEVEL:  10
    SYSTEM DATA:  00000000 00000000    |........|
COMPONENT INFORMATION:
    COMPONENT ID:             5694DF115
    COMPONENT RELEASE LEVEL:  1H0
    PID NUMBER:               5694A01
    PID RELEASE LEVEL:         1.6
    SERVICE RELEASE LEVEL:    UA13299
DESCRIPTION OF FUNCTION:  PDSE CACHING SUPPORT
    PROBLEM ID:               IGW00003
    SUBSYSTEM ID:             SMS

PRIMARY SYMPTOM STRING:
    PIDS/5694DF115  LVLS/DFSMS 1#6     RIDS/IGWBDSL2   PRCS/07100050
    PRCS/00000008   #

    THE PRIMARY SYMPTOM STRING CONTAINS INVALID SYMPTOMS.
    SYMPTOM           SYMPTOM DATA     EXPLANATION
    ---------------   -------------    -----------
    PIDS/5694DF115    5694DF115        COMPONENT IDENTIFIER
    LVLS/DFSMS        DFSMS            PROGRAM PRODUCT RELEASE LEVEL
    RIDS/IGWBDSL2     IGWBDSL2         ROUTINE IDENTIFIER
    PRCS/07100050     07100050         RETURN CODE
    PRCS/00000008     00000008         RETURN CODE

THE SYMPTOM RECORD DOES NOT CONTAIN A SECONDARY SYMPTOM STRING.
FREE FORMAT COMPONENT INFORMATION:
...
```

Figure 10-96 EREP detail summary report part 5

```
TYPE:  SYMPTOM RECORD       REPORT:  SOFTWARE SUMMARY              DAY YEAR
                                                      REPORT DATE: 328  04
SCP:   VS 2 REL 3           MODEL:   N/A              PERIOD FROM: 323  04
                            SERIAL:  N/A                       TO: 323  04
  COUNT OF SYMPTOM RECORDS PROCESSED:   0000000031
  COUNT OF UNIQUE SYMPTOM STRINGS:      0000000006
 PIDS/5694DF115  LVLS/DFSMS 1#6     RIDS/IGWBDSL2   PRCS/07100050
                      COUNT: 0000000002 FIRST: 04.323 14:59:39 LAST: 04.323 15:01:02
 PIDS/5694DF115  LVLS/DFSMS 1#6     RIDS/IGWBITX1   PRCS/00000020
                      COUNT: 0000000013 FIRST: 04.323 14:59:38 LAST: 04.323 15:32:43
 PIDS/5694DF115  LVLS/DFSMS 1#6     RIDS/IGWBSRPR   PRCS/00000024
                      COUNT: 0000000013 FIRST: 04.323 14:59:39 LAST: 04.323 15:32:44
 PIDS/5752SCXCF RIDS/IXCA3SGO RIDS/IXCI2PVT#L
                      COUNT: 0000000001 FIRST: 04.323 11:01:34 LAST: 04.323 11:01:34
 PIDS/5752SCXCF RIDS/IXCSPTSK RIDS/IEANUC01#L  FLDS/PARTSYSTEM VALU/CSC64
                      COUNT: 0000000001 FIRST: 04.323 11:01:37 LAST: 04.323 11:01:37
 PIDS/5752SCXCF RIDS/IXCS2WTO RIDS/IEANUC01#L
                      COUNT: 0000000001 FIRST: 04.323 11:01:33 LAST: 04.323 11:01:33
```

Figure 10-97 EREP detail summary report (PRINT=AL) part 6

Figure 10-98 shows a sample job for requesting an event history report (EVENT=Y).

```
//*****************************************************************/
//* PRODUCING AN EREP REPORT FROM LOGSTREAM
//*****************************************************************/
/*JOBPARM L=999,SYSAFF=SC64
//EREPDALY EXEC  PGM=IFCEREP1,PARM=('CARD')
//ACCIN      DD DSN=SYSPLEX.LOGREC.ALLRECS,DISP=SHR,
//           DCB=(RECFM=VB,BLKSIZE=4000),
//           SUBSYS=(LOGR,IFBSEXIT,
//           'FROM=(2004/323,16:00),TO=(2004/323,22:00)',)
//DIRECTWK DD UNIT=VIO,SPACE=(CYL,15,,CONTIG)
//TOURIST  DD SYSOUT=*,DCB=BLKSIZE=133
//EREPPT   DD SYSOUT=*,DCB=BLKSIZE=133
//SYSIN    DD *
   HIST
   ACC=N
   EVENT=Y
   TYPE=S
   PRINT=AL
/*
```

Figure 10-98 Example EREP job (TYPE=S and EVENT=Y)

An event history report (Figure 10-99 on page 293) provides an overview of the software events recorded in LOGREC with each event represented on a single line. The event history is a helpful report to start the analysis.

The event history report consists of one-line abstracts of selected information from each record. It shows errors in a time sequence that enable you to see how often and in what order errors occur. It also enables you to establish a pattern and diagnose problems. The report gives an overview about:

- Timing (for example, when a problem-related situation started)
- Which kind of records were written to document the problem situation

```
EVENT HISTORY (S/370XA)                                                                                    REPORT DATE  328 04
                                                                                                           PERIOD FROM  323 04
                                                                                                           PERIOD TO    323 04
                                                SPID                    SNID
                                  SSYS ID  REASON   PSW-MCH /PROG-EC  RCYRYXIT   COMP/MOD  CSECTID  ERROR-ID
   TIME     JOBNAME  RECTYP CP CUA  DEVT     CMD CSW  SENSE    04  06   08  10   12  14   16  18   20  22   VOLUME    SEEK    SD CT
                           *  DNO  CRW  CHP     SCSW                     ESW      STOR ADDR

DATE  323 04
11 01 33 60   N/A    SFTSYM 00 PIDS/5752SCXCF RIDS/IXCS2WTO RIDS/IEANUC01#L
11 01 34 85   N/A    SFTSYM 00 PIDS/5752SCXCF RIDS/IXCA3SG0 RIDS/IXCI2PVT#L
11 01 37 21   N/A    SFTSYM 03 PIDS/5752SCXCF RIDS/IXCSPTSK RIDS/IEANUC01#L  FLDS/PARTSYSTEM VALU/CSC64
14 01 34 90 DB2GDBM1 SFTABN 03 PIDS/5740XYR00 RIDS/DSNPAMS#L RIDS/DSNPAMS2 AB/S004E REGS/09E6E REGS/C0191 RIDS/DSNTFRCV#R
14 05 28 03 PAOLOR2  SFTABN 03 PIDS/5695DF107 RIDS/IGC0013I#L RIDS/ICVDSD03 AB/S018B PRCS/0000A828 REGS/0A5B9 REGS/80755
14 05 30 78 PAOLOR2  SFTABN 03 PIDS/5695DF107 RIDS/IGC0013I#L RIDS/ICVDSD03 AB/S018B PRCS/0000A928 REGS/0A5B9 REGS/80755
...
14 59 38 27   N/A    SFTSYM 02 PIDS/5694DF115  LVLS/DFSMS 1#6     RIDS/IGWBITX1   PRCS/00000020   PRCS/150A001E
14 59 39 10 MHLRES2A SFTABN 02 PIDS/5695DF115 RIDS/IGWBBMF1#L RIDS/IGWBITX1 AB/S00F4 PRCS/00000020 REGS/0E00A REGS/0CD74
14 59 39 11   N/A    SFTSYM 02 PIDS/5694DF115  LVLS/DFSMS 1#6     RIDS/IGWBSRPR   PRCS/00000024   PRCS/20080052
14 59 39 12 MHLRES2A SFTABN 02 PIDS/5695DF115 RIDS/IGWDCCS0#L RIDS/IGWDSAAC AB/S00F4 PRCS/00000024 REGS/0E012 REGS/0C258
14 59 39 13 MHLRES2A SFTABN 02 PIDS/5695DF115 RIDS/IGWDLAC0#L RIDS/IGWDLCLS AB/S00F4 PRCS/00000024 REGS/0E00E REGS/0CC7A
14 59 39 16 MHLRES2A SFTABN 02 PIDS/5695DF115 RIDS/IGWAFMS0#L RIDS/######## AB/S0213 PRCS/00000050 REGS/0E764 REGS/0389C
14 59 39 17   N/A    SFTSYM 02 PIDS/5694DF115  LVLS/DFSMS 1#6     RIDS/IGWBDSL2   PRCS/07100050   PRCS/00000008
15 00 50 84   N/A    SFTSYM 01 PIDS/5694DF115  LVLS/DFSMS 1#6     RIDS/IGWBITX1   PRCS/00000020   PRCS/150A001E
...
```

Figure 10-99 Event history report

Interpreting software records

Two types of software records are recorded in the logrec data set or the logrec log stream:

Software record The system generates these records, providing information from the system diagnostic work area (SDWA) that describes problems detected because of an abend or a program check.

Symptom record Either a user's application program or the system can issue the SYMREC macro to request the creation of a symptom record. Generally, the symptom record describes problems not accompanied by an abend, but there are exceptions.

To analyze an unexpected situation, follow this sequence of reports:

- Generate event history report

  With each event represented on a single line, this report gives an overview about:

  - Timing: to identify the point in time where a problem situation might have started.
  - Type of events: the kind of records that were written to document the problem situation and which components were involved.

- Detail summary report (optional)

  The report summarizes information about data in logrec error records. Its contents are:

  - Record header: report type being summarized
  - Summary information and counts

- Detail edit report for an abend

  The detail edit report for a software record shows the complete contents of an error record for an abnormal end, including the system diagnostic work area (SDWA). The software record can be used to identify the detecting CSECT, the ABEND 0F4 reason and return code. The detail record also identifies whether a dump was started successfully.

  The size of the report can be limited by selecting only those records created during the specified time period. Use an event history report to identify the most pertinent period of time. The system obtains most of the information for a non-abend logrec error record from the symptom record identified in the SYMREC macro.

Notes: PDSE services write both kind of records (software records and symptom records) to LOGREC.

The component ID for PDSE components is 5695DF115.

CSECTs related to PDSE services starts with *IGW*.

PDSE services issue a system completion code of 0F4 (ABEND 0F4) if they detect any inconsistency in their processing.

Using report Information

Use the information that you obtain from the detail edit reports for either a software record or a symptom record to search for a known problem. If you do not conduct the search yourself, contact the IBM Support Center. The result of the search will be one of these:

- The PTF that corrects the problem.
- The APAR, and possibly the related APAR, that describes the problem. In some cases, a temporary fix or a procedure might circumvent the problem. Apply the temporary fix if it is available; otherwise, follow the circumvention procedure.

Detail edit report for a software record

The detail edit report for a software record shows the complete contents of an error record for an abnormal end, including the system diagnostic work area (SDWA). The report is produced by EREP and, through the VERBEXIT LOGDATA subcommand, under IPCS.

Use the detail edit report for a software record to determine the cause of an abend and any recovery action that the system or application has taken. This report enables you to locate where an error occurred, similar to the analysis of an SVC dump. After you locate the error, you can develop a search argument to obtain a fix for the problem.

TYPE: SOFTWARE RECORD

Indicates that the detail edit report is for an SDWA-type record.

REPORT DATE Indicates the date on which the EREP report was created.

ERROR DATE Indicates the date on which the error occurred.

TIME Indicates the time, as local, at which the error occurred.

JOBNAME If the job name is NONE-FRR, the error being recorded occurred in system or subsystem code covered by a functional recovery routine (FRR).

ERRORID Enables you to coordinate diagnostic information from logrec, the console log (SYSLOG), and system dumps. The ERRORID is a concatenation of the following information:

SEQ A unique number assigned to each error. The sequence number indicates the order of the errors, but the records might not be listed in order. It is important to scan all entries and examine the sequence numbers to understand which error occurred first.

You might find the same sequence number used in more than one entry when several recovery routines, as a result of percolation, get control and request recording for the same error; however, the error time stamp will be different.

CPU The internal identification number of the central processor that the failing process was running on when the error occurred. Use

information from the system trace table about this CPU to learn more about the error.

ASID The address space identifier (ASID) of the current, or home, address space at the time the error occurred.

TIME Indicates the time of the error.

PROGRAM ID The component identification keyword for PDSE components is 5695DF115.

CSECT NAME Supplied by the recovery routine that obtained control for the error. See the PSW for more information.

CSECTs related to PDSE services start with IGW.

S**YSTEM ABEND CODE**

Indicates what system or user completion code was issued by the system, application, or component. See *z/OS: MVS System Codes*, SA22-7626 for information about system abend codes. See the appropriate product documentation for user abend codes.

A system completion code of 0F4 (ABEND 0F4) indicates that an error occurred in DFSMSdfp support. The ABEND reason code (in register 15) is the DFSMSdfp return code. The DFSMSdfp reason code is in register 0.

Note that other z/OS DFSMS components such as VSAM RLS and HFS are issuing ABEND 0F4s.

REGISTERS Indicates the reason code, when available, associated with a system or user abend code.

GPR 0: Contains the 4-byte DFSMSdfp *PDSE reason code* (RSN).

GPR 1: Contains the *system completion code* (_ _ c c c_ _ _) such as ABEND 0F4.

GPR 15: Contains the 1-byte *PDSE return code* (RC). The field is also referred to as an abend reason code in MVS terms.

RECOVERY ROUTINE ACTION

Describes the recovery action performed or requested to be performed by the recovery routine. In the preceding example, an SVC dump was not requested. However, sometimes the recovery routine will request an SVC dump. If `SVC DUMP SUCCESSFULLY STARTED` appears in this section, the error identifier (ERROR ID) appears in the SVC dump and in message IEA911E as it appears in the logrec error record.

If `THE REQUESTED  SVC DUMP WAS NOT TAKEN. THE DUMP WAS SUPPRESSED BY DAE` appears, see “Dump suppression (DAE)” on page 270 for further information about DAE and how to request a dump for the next event.

Detail edit report for a symptom record

The SYMREC macro updates a symptom record with system environment information and then logs the symptom record in the logrec data set or logrec log stream. The system or application, using the SYMREC macro, creates a symptom record.

As an application or a system component detects errors during processing, it stores diagnostic information into the symptom record and issues the SYMREC macro to log the

record. The diagnostic information consists of a description of a programming failure and a description of the environment in which the failure occurred.

TYPE: SYMPTOM RECORD

Indicates that the detail edit report is for a symptom record.

SEARCH ARGUMENT ABSTRACT

Provides information that you can use to create a search argument. If enough information exists in this field, you can search the IBM product support database to see whether there is a PTF to correct the error.

The information that follows the search argument abstract in a symptom record depends on the options specified on the SYMREC macro either by a user program or by a system component. The information contained in a symptom record is variable. To obtain an interpretation, contact the IBM Support Center for the product or for the component that built the record.

References

See the chapter about recording logrec error records in *z/OS: MVS Diagnosis: Tools and Service Aids*, GA22-7589 and *EREP User's Guide*, GC35-0151 for more information about a detail edit report for an SDWA-type record.

See *z/OS: MVS Interactive Problem Control System (IPCS) Commands*, SA22-7594 for information about the VERBEXIT LOGDATA subcommand.

10.6.4 PDSE component trace (CTRACE)

DFSMSdfp provides a trace service for use with all functions that manipulate PDSEs. A special trace table is created in each address space in which PDSEs are used and tracing is selected.

PDSE trace services provides a single interface for trace data management for PDSE. This trace data management is based on MVS Component Trace Facility (referred to as CTRACE). The primary method of communicating with the trace is by issuing the TRACE CT system command for Component Trace Facility. During execution of modules using this trace function, callers request that trace services create entries in trace tables.

PDSE tracing services determines whether to create the entry in a table for that request. This determination is made based on whether tracing for the component SYSSMS has been turned on with the MVS operator TRACE CT command and, if so, with what options and that all required information is being passed. The jobname, ASID, type of entry requested, and subcomponent are checked against the options.

Trace entries can be captured on an ongoing basis by an external writer or as a snapshot in a dump. When a dump is used to capture trace entries, if the table size is too small, the table may have wrapped and overwritten some entries. Capturing trace buffers by an external writer increases diagnostic capability and decreases the possibility of losing trace data.

The default SYSSMS trace table size is 72 KB. Complete syntax and usage information for TRACE can be found in *z/OS MVS System Commands*, GC28-1781.

```
TRACE ,WTRSTART=membername [ WRAP|NOWRAP ] ] ] CT [,WTRSTOP=jobname ] ]
   [CT [,ON ] { {COMP=SYSSMS } [,SUB=(sub[sub],...)]
   [,PARM=CTISMSxx] } ]
   [CT [,nnnK ]{ } ]
   [CT [,nnnnM] { } ]
   [CT [,OFF] { } ]
```

PARM=CTISMSxx identifies the SYS1.PARMLIB member that contains the tracing options. The nnnK and nnnnM keywords can be used to specify trace table sizes of 16 KB to 999 KB or 1 MB to 2047 MB, respectively.

You are then prompted to specify the trace options to be in effect. You must respond with the REPLY command, using this syntax:

```
REPLY id
   [,JOBNAME=(jobnamelist)]
   ASID[,=(asidlist.)]
   [,OPTIONS=(namename...)]
   [,WTR=procname|DISCONNECT]
   [,END]
```

Notes:

1. The id value is the identification number, 0 to 99, specified on the prompting message.
2. You may list up to 16 jobs or ASIDs to be used as filters in tracing.
3. The REPLY may be continued on multiple lines; the END must be given as the final parameter to identify the end of the REPLY.

z/OS MVS System Commands, GC28-1781, details the TRACE and REPLY commands.

To *turn off* the trace facility, use:

```
TRACE CT,OFF,COMP=SYSSMS
```

PDSE trace options

Table 10-2 shows valid options for use with this trace facility.

Table 10-2 SYSSMS trace options

Trace event	Description
ENTRY	All module entries.
EXIT	Module exits only with non-zero return codes.
EXITA	All module exists.
CALL	Equivalent to specifying both ENTRY and EXIT.
RRTN	Entries and exits of recovery routines.
CB	Control block changes.
SPECIAL	Entries and exits of commonly shared functions.
COMP=(component- list)	Specifies major functional areas of DFSMSdfp used to manipulate PDSEs.
SUBCOMP=(subcomponent- list)	Specifies minor functional areas in the specified major areas.

List of PDSE components:

- Buffer Management Facility (BMF)
- Common Data Manager (CDM)
- Common Lock Manager (CLM)
- Index Management Facility (IMF)
- System Services Facility (SSF)
- File and Attribute Management Services (FAMS)

Table 10-3 lists valid PDSE components and subcomponents that can be specified for a CTRACE.

Table 10-3 PDSE components and subcomponents

COMP	SUBCOMP
BMF	BDS LMC CDS SRS IOC VBM
CDM	ADC LAC DSC RAC DCC
CLM	BRL HLS BTR LIC CLS LRS ENS SLS GLS XCM
IMF	
SSF	ARM RCS CHF SAC DMP VSM ERS

Tips:

- You may specify ALL instead of a list for both the COMP or SUBCOMP keywords.
- The SUBCOMP keyword can be specified without specifying the COMP keyword.
- Any of the options listed above except for ALL, COMP, and SUBCOMP can be turned off individually by prefixing them with NO (for example, NOSSF, NOIMF).

PDSE CTRACE writer options

You can trace without or with a CTRACE writer.

Tracing without a CTRACE writer data set

When the trace buffer is filled, it wraps and starts overwriting the oldest trace records with the newest trace records. Therefore, only the most recent records are available, and the trace buffer size is the limiting factor. Whether you get an abend dump or take a console dump, the dump contains the trace buffers with the most current data, and all of the data in the trace buffer can be formatted by the CTRACE formatter.

If you are experiencing problems in a PDSE environment, you can start a PDSE CTRACE to analyze the problem:

1. If needed, set the required SLIP or prepare the DUMP command by placing the command in the IEADMCxx parmlib member (Figure 10-100).

```
EDIT       SYS1.PARMLIB(IEADMCDG)
******************************** Top of Data ********************************
COMM=(PDSE DUMP)
TSONAME=(GUTS),
JOBNAME=(SMSPDSE1),
DSPNAME=('SMSPDSE1'.*),
SDATA=(PSA,CSA,SQA,GRSQ,LPA,LSQA,RGN,SUM,SWA,TRT),
END
****************************** Bottom of Data ******************************
```

Figure 10-100 SYS1.PARMLIB(IEADMCDG) parmlib member

2. Start PDSE CTRACE before re-creating the problem (Figure 10-101):

```
TRACE CT,4M,COMP=SYSSMS
R xx,OPTIONS=(ENTRY,EXIT,EXITA,SPECIAL,COMP=(xxxx)),END
```

```
 TRACE CT,4M,COMP=SYSSMS
*27 ITT006A SPECIFY OPERAND(S) FOR TRACE CT COMMAND.
 R 27,OPTIONS=(ENTRY,EXIT,EXITA,CB,SPECIAL,COMP=(CDM)),END
 IEE600I REPLY TO 27 IS;OPTIONS=(ENTRY,EXIT,EXITA,CB,SPECIAL,COMP=(
 ITT038I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND WERE
 SUCCESSFULLY EXECUTED.
 IEE839I ST=(ON,1000K,01000K) AS=ON  BR=OFF EX=ON  MT=(ON,024K)
         ISSUE DISPLAY TRACE CMD FOR SYSTEM AND COMPONENT TRACE STATUS
         ISSUE DISPLAY TRACE,TT CMD FOR TRANSACTION TRACE STATUS

 D TRACE,COMP=SYSSMS
 IEE843I 08.13.31  TRACE DISPLAY
         SYSTEM STATUS INFORMATION
  ST=(ON,1000K,01000K) AS=ON  BR=OFF EX=ON  MT=(ON,024K)
   COMPONENT     MODE BUFFER HEAD SUBS
  -------------------------------------------------------------
   SYSSMS        ON   0004M
      ASIDS      *NONE*
      JOBNAMES   *NONE*
      OPTIONS    ENTRY,EXIT,EXITA,CB,SPECIAL,COMP=(CDM)
      WRITER     *NONE*
```

Figure 10-101 TRACE CT,4M,COMP=SYSSMS output

3. Run the job or process that should be traced.
4. If an abend does not occur, take a PDSE environment dump prior to turning off CTRACE (Figure 10-102). See “Dynamic/console dump” on page 267 for more about taking a dump of your PDSE environment.

```
DUMP PARMLIB=xx
```

```
DUMP PARMLIB=DG
IEE252I MEMBER IEADMCDG FOUND IN SYS1.PARMLIB
IEA794I SVC DUMP HAS CAPTURED:
DUMPID=029 REQUESTED BY JOB (*MASTER*)
DUMP TITLE=PDSE DUMP
IEF196I IGD100I 732F ALLOCATED TO DDNAME SYS00029 DATACLAS (        )
IEF196I IEF285I   SYS1.MCEVSF.DUMP.S00042                     CATALOGED
IEF196I IEF285I   VOL SER NOS= VSF6C1.
IEA611I COMPLETE DUMP ON SYS1.MCEVSF.DUMP.S00042
DUMPID=029 REQUESTED BY JOB (*MASTER*)
FOR ASIDS(0066,005F)
INCIDENT TOKEN: VSFPLEX  MCEVSF   11/29/2004 07:13:52
```

Figure 10-102 DUMP PARMLIB=DG output

5. After the abend or SLIP hit, turn off CTRACE (Figure 10-103).

```
TRACE CT,OFF,COMP=SYSSMS
```

```
TRACE CT,OFF,COMP=SYSSMS
ITT038I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND WERE
SUCCESSFULLY EXECUTED.
IEE839I ST=(ON,1000K,01000K) AS=ON  BR=OFF EX=ON  MT=(ON,024K)
        ISSUE DISPLAY TRACE CMD FOR SYSTEM AND COMPONENT TRACE STATUS
        ISSUE DISPLAY TRACE,TT CMD FOR TRANSACTION TRACE STATUS
```

Figure 10-103 TRACE CT,OFF,COMP=SYSSMS output

6. See “Formatting the PDSE component trace” on page 305 regarding how to verify that trace data were captured.

   The IPCS command is IPCS CTRACE COMP(SYSSMS) FULL.

7. Send the dump to an IBM software support center.

See “Formatting the PDSE component trace” on page 305 to view and verify the trace under IPCS.

Tracing with a CTRACE writer data set

Trace entries can be captured on an ongoing basis by an external writer.

To produce a CTRACE:

1. Create a member in SYS1.PROCLIB to allocate CTRACE writer data sets. Run the following job (Figure 10-104) to create a member called CTPDSE, which allocates one data set for the CTRACE writer to write to. Change the data set name as appropriate, and use more space if it is available.

```
 EDIT       SYS1.PROCLIB(CTPDSE)
******************************** Top of Data ********************************
//CTPDSE PROC
//IEFPROC  EXEC  PGM=ITTTRCWR,REGION=32M
//TRCOUT01 DD DSNAME=GUTS.CTRACE.SYSSMS.PDSE01,
//*           UNIT=SYSDA,
//*           VOL=SER=ANYDSD,
//            SPACE=(CYL,10),DISP=(NEW,CATLG),DSORG=PS
//TRCOUT02 DD DSNAME=GUTS.CTRACE.SYSSMS.PDSE02,
//*           UNIT=SYSDA,
//*           VOL=SER=ANYDSD,
//            SPACE=(CYL,10),DISP=(NEW,CATLG),DSORG=PS
// ****************************** Bottom of Data ******************************
```

Figure 10-104 SYS1.PROCLIB(CTPDSE) PROCLIB member

Attention: The SPACE attributes (in Figure 10-104) are provided only as an example. The primary quantity values must be adjusted before activating the CTRACE.

2. Create a parmlib member to specify your trace and dump options. Figure 10-105 shows a sample TRACE parmlib member.

```
EDIT       SYS1.PARMLIB(CTISMS01)
******************************** Top of Data ********************************
TRACEOPTS ON
          WTR(CTPDSE)
          OPTIONS('ENTRY','EXIT','EXITA','SPECIAL','CB',
          'COMP=(CDM)','SUBCOMP=(LMC,PMI)')
******************************** Bottom of Data *****************************
```

Figure 10-105 SYS1.PARMLIB(CTISMS01) PARMLIB member

Note: Do not specify an END parameter when using a parmlib member to specify the trace options.

Figure 10-106 shows a sample DUMP parmlib member. Include the associated SMSPDSE or SMSPDSE1 address and data spaces as well as the user task (either in JOBNAME or TSONAME parameter) in your DUMP processing.

```
EDIT       SYS1.PARMLIB(IEADMCDG)
******************************** Top of Data ********************************
COMM=(PDSE DUMP)
TSONAME=(GUTS),
JOBNAME=(SMSPDSE1),
DSPNAME=('SMSPDSE1'.*),
SDATA=(PSA,CSA,SQA,GRSQ,LPA,LSQA,RGN,SUM,SWA,TRT),
END
******************************** Bottom of Data *****************************
```

Figure 10-106 SYS1.PARMLIB(IEADMCDG) PARMLIB member

3. Start the trace writer, which allocates the trace data sets so that the trace buffers are written, rather than wrapped:

```
TRACE CT,WTRSTART=CTPDSE
```

Figure 10-107 on page 302 shows the response.

```
TRACE CT,WTRSTART=CTPDSE
ITT038I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND
        WERE SUCCESSFULLY EXECUTED.
IEE839I ST=(ON,1000K,01000K) AS=ON  BR=OFF EX=ON  MT=(ON,024K) 557
        ISSUE DISPLAY TRACE CMD FOR SYSTEM AND COMPONENT TRACE STATUS
        ISSUE DISPLAY TRACE,TT CMD FOR TRANSACTION TRACE STATUS
IRR812I PROFILE ** (G) IN THE STARTED CLASS WAS USED 558
        TO START CTPDSE WITH JOBNAME CTPDSE.
IEF196I         1 //CTPDSE   JOB MSGLEVEL=1
IEF196I         2 //STARTING EXEC CTPDSE
IEF196I  STMT NO. MESSAGE
IEF196I         2 IEFC001I PROCEDURE CTPDSE WAS EXPANDED USING SYSTEM
IEF196I LIBRARY SYS1.PROCLIB
IEF196I         3 XXCTPDSE PROC
IEF196I                   00010001
IEF196I         4 XXIEFPROC  EXEC  PGM=ITTTRCWR,REGION=32M
IEF196I                   00020000
IEF196I         5 XXTRCOUT01 DD DSNAME=GUTS.CTRACE.SYSSMS.PDSE01,
IEF196I                   00030003
IEF196I           XX*            UNIT=SYSDA,
IEF196I                   00031001
IEF196I           XX*            VOL=SER=ANYDSD,
IEF196I                   00040000
IEF196I           XX             SPACE=(CYL,10),DISP=(NEW,CATLG),
IEF196I DSORG=PS                 00050000
IEF196I         6 XXTRCOUT02 DD DSNAME=GUTS.CTRACE.SYSSMS.PDSE02,
IEF196I                   00051003
IEF196I           XX*            UNIT=SYSDA,
IEF196I                   00052003
IEF196I           XX*            VOL=SER=ANYDSD,
IEF196I                   00053003
IEF196I           XX             SPACE=(CYL,10),DISP=(NEW,CATLG),
IEF196I DSORG=PS                 00054003
IEF196I         7 XX
IEF196I                   00060000
IRR812I PROFILE ** (G) IN THE STARTED CLASS WAS USED 586
        TO START CTPDSE WITH JOBNAME CTPDSE.
IEF403I CTPDSE - STARTED - TIME=02.30.58
IEF196I IEF236I ALLOC. FOR CTPDSE CTPDSE
IEF196I IGD100I 7341 ALLOCATED TO DDNAME TRCOUT01 DATACLAS (        )
IEF196I IGD100I 7341 ALLOCATED TO DDNAME TRCOUT02 DATACLAS (        )
IEF196I AHL906I THE OUTPUT BLOCK SIZE OF    27998 WILL BE USED FOR
IEF196I OUTPUT
IEF196I         DATA SETS:
IEF196I           GUTS.CTRACE.SYSSMS.PDSE01
IEF196I           GUTS.CTRACE.SYSSMS.PDSE02
AHL906I THE OUTPUT BLOCK SIZE OF    27998 WILL BE USED FOR OUTPUT 591
        DATA SETS:
          GUTS.CTRACE.SYSSMS.PDSE01
          GUTS.CTRACE.SYSSMS.PDSE02
ITT110I INITIALIZATION OF CTRACE WRITER CTPDSE COMPLETE.
```

Figure 10-107 TRACE CT,WTRSTART=CTPDSE

4. Activate trace with a parmlib member to specify options:

   ```
   TRACE CT,500K,COMP=SYSSMS,PARM=CTISMS01
   ```

   Here, CTISMS01 has all the options you intend to trace, and nnnK is the trace buffer size.

   – 500K is the recommended trace table size if the CTRACE writer is used.
   – Use a larger size if the CTRACE writer is not used; for example, 4M.
   – Buffer size range is from 72K to 64M.
   – Smaller buffer size causes frequent spooling to the CTRACE writer queue.
   – Larger buffer size causes more paging.

   See Figure 10-108.

```
TRACE CT,500K,COMP=SYSSMS,PARM=CTISMS01
IEE252I MEMBER CTISMS01 FOUND IN SYS1.PARMLIB
ITT038I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND
        WERE SUCCESSFULLY EXECUTED.
IEE839I ST=(ON,1000K,01000K) AS=ON  BR=OFF EX=ON  MT=(ON,024K) 601
        ISSUE DISPLAY TRACE CMD FOR SYSTEM AND COMPONENT TRACE STATUS
        ISSUE DISPLAY TRACE,TT CMD FOR TRANSACTION TRACE STATUS
```

Figure 10-108 TRACE CT,500K,COMP=SYSSMS,PARM=CTISMS01

To display what you have just set up, use:

```
DISPLAY  TRACE,COMP=SYSSMS
```

You should see output similar to Figure 10-109.

```
D TRACE,COMP=SYSSMS
IEE843I 02.53.02  TRACE DISPLAY 641
        SYSTEM STATUS INFORMATION
 ST=(ON,1000K,01000K) AS=ON  BR=OFF EX=ON  MT=(ON,024K)
  COMPONENT     MODE BUFFER HEAD SUBS
 ------------------------------------------------------------
  SYSSMS        ON   0500K
     ASIDS      *NONE*
     JOBNAMES   *NONE*
     OPTIONS    ENTRY,EXIT,EXITA,SPECIAL,CB,COMP=(CDM),
                SUBCOMP=(LMC,PMI)
     WRITER     CTPDSE
```

Figure 10-109 DISPLAY TRACE,COMP=SYSSMS

5. Run your test cases.

6. Take a console dump, as shown in Figure 10-110, to include the trace buffers that have not yet been written to the trace data sets.

   ```
   DUMP PARMLIB=DG
   ```

   If a dump results as part of the re-create, this step is not necessary.

```
DUMP PARMLIB=DG
IEE252I MEMBER IEADMCDG FOUND IN SYS1.PARMLIB
IEA794I SVC DUMP HAS CAPTURED: 604
DUMPID=031 REQUESTED BY JOB (*MASTER*)
DUMP TITLE=PDSE DUMP
IEF196I IGD100I 732F ALLOCATED TO DDNAME SYS00031 DATACLAS (       )
IEF196I IEF285I   SYS1.MCEVSF.DUMP.S00044                      CATALOGED
IEF196I IEF285I   VOL SER NOS= VSF6C1.

IEA611I COMPLETE DUMP ON SYS1.MCEVSF.DUMP.S00044 608
DUMPID=031 REQUESTED BY JOB (*MASTER*)
FOR ASIDS(0066,0061)
INCIDENT TOKEN: VSFPLEX  MCEVSF   12/01/2004 01:33:25
```

Figure 10-110 DUMP PARMLIB=DG

7. Terminate or disconnect the trace.

   Disconnecting the trace retains all of your current trace options. Terminating the trace requires that you respecify the trace options if you restart the trace. Disconnect if you would like to redirect trace data to a new CTRACE writer data set. See Figure 10-111.

   To terminate the trace:

   ```
   TRACE CT,OFF,COMP=SYSSMS
   ```

```
TRACE CT,OFF,COMP=SYSSMS
ITT038I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND
        WERE SUCCESSFULLY EXECUTED.
IEE839I ST=(ON,1000K,01000K) AS=ON  BR=OFF EX=ON  MT=(ON,024K) 611
        ISSUE DISPLAY TRACE CMD FOR SYSTEM AND COMPONENT TRACE STATUS
        ISSUE DISPLAY TRACE,TT CMD FOR TRANSACTION TRACE STATUS
```

Figure 10-111 TRACE CT,OFF,COMP=SYSSMS

8. Terminate the trace writer:

```
TRACE CT,WTRSTOP=CTPDSE,FLUSH
```

FLUSH writes the spooled trace data. Note that trace records not already spooled to the CTRACE writer queue are not written to the CTRACE writer data set. See Figure 10-112.

```
TRACE CT,WTRSTOP=CTPDSE,FLUSH
ITT038I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND
        WERE SUCCESSFULLY EXECUTED.
IEE839I ST=(ON,1000K,01000K) AS=ON  BR=OFF EX=ON  MT=(ON,024K)
        ISSUE DISPLAY TRACE CMD FOR SYSTEM AND COMPONENT TRACE STATUS
        ISSUE DISPLAY TRACE,TT CMD FOR TRANSACTION TRACE STATUS
IEF196I AHL904I THE FOLLOWING TRACE DATASETS CONTAIN TRACE DATA :
IEF196I           GUTS.CTRACE.SYSSMS.PDSE01
IEF196I           GUTS.CTRACE.SYSSMS.PDSE02
AHL904I THE FOLLOWING TRACE DATASETS CONTAIN TRACE DATA :
          GUTS.CTRACE.SYSSMS.PDSE01
          GUTS.CTRACE.SYSSMS.PDSE02
ITT111I CTRACE WRITER CTPDSE TERMINATED BECAUSE OF A WTRSTOP REQUEST.
IEF404I CTPDSE - ENDED - TIME=02.34.08
IEF196I IEF142I CTPDSE CTPDSE - STEP WAS EXECUTED - COND CODE 0000
IEF196I IEF285I   GUTS.CTRACE.SYSSMS.PDSE01                    CATALOGED
IEF196I IEF285I   VOL SER NOS= VSF6ST.
IEF196I IEF285I   GUTS.CTRACE.SYSSMS.PDSE02                    CATALOGED
IEF196I IEF285I   VOL SER NOS= VSF6ST.
```

Figure 10-112 TRACE CT,WTRSTOP=CTPDSE,FLUSH

9. Send both the dump and the trace data sets to an IBM support center for further analysis.

A dump is necessary either because the re-create took a dump or you took one from the console to get the trace buffers that were not flushed to DASD.

The next section shows how to view the trace under IPCS.

Formatting the PDSE component trace

The IPCS CTRACE command is used to format and view the information in the component trace table. This might be of interest to verify whether any trace data were captured.

- Place the first *trace data set name* or a *dump data set name* on the IPCS option 0 screen, then go to option 6 commands.
  - For a single trace data set or dump, enter:

```
CTRACE COMP(SYSSMS) FULL
```

Figure 10-113 shows the IPCS messages that are issued during a single IPCS trace data set initialization process.

```
 TIME-08:01:58 AM. CPU-00:00:02 SERVICE-87817 SESSION-01:52:01 NOVEMBER 27,2004
 Initialization in progress for DSNAME('GUTS.CTRACE.SYSSMS.PDSE1')
 Treat input only as trace data?  Enter Y for yes, N for full initialization
y
 TIME-08:02:00 AM. CPU-00:00:02 SERVICE-87976 SESSION-01:52:03 NOVEMBER 27,2004
 ***
```

Figure 10-113 IPCS trace data set initialization output

– For multiple output writer data sets, enter `IP MERGE` to start the merge process. Figure 10-114 shows the IPCS messages that are issued when merging several trace data sets during IPCS initialization process.

```
 Enter an invocation or MERGEEND
CTRACE COMP(SYSSMS) DSN('GUTS.CTRACE.SYSSMS.PDSE01') full
 Enter an invocation or MERGEEND
CTRACE COMP(SYSSMS) DSN('GUTS.CTRACE.SYSSMS.PDSE02') full
 Enter an invocation or MERGEEND
mergeend
 TIME-02:39:40 AM. CPU-00:00:01 SERVICE-94936 SESSION-00:34:17 DECEMBER 1,2004
 Initialization in progress for DSNAME('GUTS.CTRACE.SYSSMS.PDSE01')
 Treat input only as trace data?  Enter Y for yes, N for full initialization
y
 TIME-02:39:56 AM. CPU-00:00:01 SERVICE-95160 SESSION-00:34:33 DECEMBER 1,2004
 TIME-02:39:56 AM. CPU-00:00:01 SERVICE-95867 SESSION-00:34:33 DECEMBER 1,2004
 Initialization in progress for DSNAME('GUTS.CTRACE.SYSSMS.PDSE02')
 Treat input only as trace data?  Enter Y for yes, N for full initialization
y
 TIME-02:40:03 AM. CPU-00:00:01 SERVICE-96030 SESSION-00:34:40 DECEMBER 1,2004
 ***
```

Figure 10-114 IPCS trace data set initialization output (merged)

Figure 10-115 shows a PDSE CTRACE entry recorded during module entry processing:

```
IPCS CTRACE COMP(SYSSMS) FULL
```

```
******************************************************** TOP OF DATA *************


 COMPONENT TRACE FULL FORMAT
 SYSNAME(MCEVSF)
 COMP(SYSSMS)
 **** 11/27/2004

 SYSNAME   MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
 -------   --------  --------  ---------------  -------------

 MCEVSF    ENTRY     00000000  05:55:52.515011  Module Entry
 ------------------------------------------------------------------------------
 ASID: 0066  COMP: BMF     (DF115)  SCMP: LMC  (009)  FNID: 013  MDID: 011
 MDNM: IGWBLMPS  TCBA: 007FBCF0  RETN: 826CF8FE DINM: 0000   HASID:  0066
 ------------------------------------------------------------------------------
```

Figure 10-115 IPCS CTRACE COMP(SYSSMS) FULL output

10.6.5 PDSE and HFS Analysis Tool (PHATOOL)

The PHATOOL is a service aid that operates as a application program. Its purpose is to check and analyze the directory structures (such as AD and ND) and display the allocation of data set pages. Note that:

- The PHATOOL is *not* a PDSE verification tool.
- The PHATOOL does *not* perform a consistency check between the attribute and name directory structures.
- The PHATOOL does *not* check member data.

The intention of the PHATOOL is to provide an analysis tool to verify the internal structures within a PDSE.

Restriction: The PDSE Tool was developed at the San Jose Programming Lab. This document does not explain the structure of HFS or PDSEs and the contents of their directory pages. That information is documented in other publications that are available to authorized IBM service representatives, who will provide the PHATOOL to you.

There are no warranties of any kind, and there is no service or technical support available for the PHATOOL from IBM.

Operational considerations

Consider the following points when you run the PDSE Tool:

- The PDSE Tool runs in supervisor state and must reside in an authorized library.
- The PDSE Tool is serially reusable, but not reentrant.
- The PDSE Tool issues a RACROUTE macro to check the user's authority to access a dataset. A return code greater than 4 from SAF will cause an abend.
- The HFS data sets must be unmounted before running this tool.

JCL statements

The PDSE Tool is executed using the following job control statements. The minimum region size needed is 200K.

JOB Marks the beginning of the job.

EXEC Invokes the PDSE Tool using PGM=IGWPIT. There are no parameters to specify.

SYSPRINT DD Defines a sequential output message data set that can be written on a system printer, a magnetic tape volume, or a direct access volume. This statement is required for each execution of the PDSE Tool.

SYSPDSE DD Defines the PDSE or HFS that will be accessed by the PHATOOL when performing the operations specified on the control statements. The DSNAME parameter and DISP=OLD are required.

SYSIN DD Defines the input stream data set that contains PDSE Tool control statements.

PHATOOL control statements (SYSIN DD)

PDSE Tool control statements entered through the SYSIN data stream define the processing functions to be performed during a particular execution of the PDSE Tool.

Coding rules for PHATOOL control statements:

- All parameters must be separated by at least one blank space.
- A record beginning with an asterisk and a blank (*)is considered to be a comment.

These are detailed descriptions of the PDSE Tool control statements:

EXTMAP Causes a map showing the allocation of pages within the PDSE to be printed. Pages that are allocated to a data file (member) are identified with this member name if the data set is a PDSE. The EXTMAP output can be used to verify the fragmentation of a PDSE.

CHECK
CHECK ERROR Verifies that the physical structure of the data set directory is correct. This verify is not exhaustive. It does enough checking to verify that the directory structure can be read and that attributes for all members can be obtained. ERROR is an optional parameter that produces only messages associated with errors. If ERROR is not specified, informational messages will not be issued, and the CHECK will pass back a return code 0 or 36 on completion of its processing.

Important: CHECK does not verify member data.

DUMPT ALL Produces a hexadecimal format dump of all internal directories. A DUMPT ALL report (with a corresponding system abend dump) is mostly requested by IBM service representatives to analyze an inconsistency in a PDSE.

Note: Further control statements are available. Authorized IBM service representatives can provide the required information based on the individual situation.

Figure 10-116 provides an example of obtaining an EXTMAP report.

```
//PHATOOL  EXEC  PGM=IGWPIT,REGION=0M
//SYSPRINT DD    SYSOUT=*
//SYSPDSE  DD DISP=OLD,DSN=pdse.data.set.name
//SYSIN DD *
  EXTMAP
/*
```

Figure 10-116 PHATOOL example

Here are the meanings of format words in the EXTMAP:

<VDF> Page is allocated to VDF of the directory.

<BMF> Page is allocated to BMF page of the directory.

<AD> Page is allocated to Attribute Directory.

<ND> Page is allocated to root Name Directory.

number Page is allocated to a data file with this number FSN (file sequence number). Because names can be longer than eight characters, the tool uses the FSN to format an HFS.

name Page is allocated to a data file with this member name if the data set is a PDSE.

<FREE> Page is available for allocation.

<NOTFMT> Page is not formatted.

<LOST> Page is not accounted for in any internal space maps.

<BAD> Page is allocated to more than one object. See the Check Detail messages to learn what the page is allocated to.

Pend Page is pending delete (as shown in Figure 10-118 on page 309).

Example

The job in Figure 10-117 creates an EXTMAP report for PDSE data set PDSERES.SMS.PDSE. The PHATOOL itself (PGM=IGWPIT) is located in authorized library PDSERES.REG.LOAD.

```
//PITS     EXEC  PGM=IGWPIT,REGION=320M
//STEPLIB  DD DISP=SHR,DSN=PDSERES.REG.LOAD
//SYSPRINT DD    SYSOUT=*
//SYSPDSE  DD DISP=OLD,
//            DSN=PDSERES.SMS.PDSE
//SYSIN DD *
EXTMAP
/*
```

Figure 10-117 PHATOOL JCL example

Figure 10-118 shows the EXTMAP report for PDSE data set PDSERES.SMS.PDSE.

Each line in the output represents a track on the DASD volume MHLVOL3 (3390). One track on a 3390 contains 12* 4 KB blocks (pages). The first 4 KB block (page) starts on cylinder-head x'0054'-x'0000'. This information corresponds to the information provided by an IEHLIST LISTVTOC report in Figure 10-128 on page 314. (Cylinder x'0054' is equal to 84 (decimal).)

```
IGWTO01 PIT VERSION UDZ11HS 08.01 ** VOLSER = MHLVO3  **  DSN = PDSERES.SMS.PDSE
 EXTMAP                                                        00060001
Extent Associations:
Cyl  Head   Records:
0054 0000   < VDF >  < BMF >  < AD >   n0000003 n0000003 MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2
0054 0001   MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2
0054 0002   MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2
0054 0003   MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER2
0054 0004   MEMBER2  MEMBER2  MEMBER2  MEMBER2  MEMBER3  MEMBER3  MEMBER3  MEMBER3  MEMBER3  MEMBER3  MEMBER3  MEMBER1
0054 0005   MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  < Pend > < Pend > < Pend > < Pend > < Pend > < Pend >
0054 0006   < Pend > < Pend > < Pend > < Pend > < Pend > < Pend > < Pend > < Pend > < Pend > < Pend > < Pend > < Pend >
0054 0007   < Pend > < Pend > < Pend > < Pend > < Pend > < Pend > < Pend > < Pend > < Pend > < Pend > < Pend > < Pend >
0054 0008   < Pend > < Pend > < Pend > < Pend > < Pend > < Pend > < Pend > < Pend > < Pend > < Pend > < Pend > < Pend >
0054 0009   < Pend > < Pend > < Pend > < Pend > < Pend > MEMBER4  MEMBER4  MEMBER4  MEMBER4  MEMBER4  MEMBER4  MEMBER4
0054 000A   MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1
0054 000B   MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1
0054 000C   MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1  MEMBER1
0054 000D * MEMBER1  MEMBER1  MEMBER1  MEMBER1  < Free > < Free > < Free > < Free > < Free > < Free > < Free > < Free >
0054 000E   <NOTFMT> <NOTFMT> <NOTFMT> <NOTFMT> <NOTFMT> <NOTFMT> <NOTFMT> <NOTFMT> <NOTFMT> <NOTFMT> <NOTFMT> <NOTFMT>
0055 0000   <NOTFMT> <NOTFMT> <NOTFMT> <NOTFMT> <NOTFMT> <NOTFMT> <NOTFMT> <NOTFMT> <NOTFMT> <NOTFMT> <NOTFMT> <NOTFMT>
...
```

Figure 10-118 PHATOOL EXTMAP report

Figure 10-119 shows an ISPF display of the member contained in PDSERES.SMS.PDSE.

```
BROWSE           PDSERES.SMS.PDSE                        Row 00001 of 00004
Command ===>                                               Scroll ===> CSR
          Name     Prompt       Size   Created          Changed          ID
_________ MEMBER1                2223  2004/11/18  2004/11/19 21:14:21  MHLRES2
_________ MEMBER2                2223  2004/11/18  2004/11/18 13:10:56  MHLRES2
_________ MEMBER3                 329  2004/11/18  2004/11/18 13:11:15  MHLRES2
_________ MEMBER4                 329  2004/11/18  2004/11/19 21:14:10  MHLRES2
          **End**
```

Figure 10-119 ISPF BROWSE

The job in Figure 10-120 runs the CHECK function for PDSE data set PDSERES.SMS.PDSE.

```
//PITS     EXEC  PGM=IGWPIT,REGION=320M
//STEPLIB  DD DISP=SHR,DSN=PDSERES.REG.LOAD
//SYSPRINT DD    SYSOUT=*
//SYSPDSE  DD DISP=OLD,
//            DSN=PDSERES.SMS.PDSE
//SYSIN DD *
 CHECK
/*
```

Figure 10-120 PHATOOL CHECK

Figure 10-121 shows the output related to the job shown in Figure 10-120.

```
IGWT001 PIT VERSION UDZ11HS 08.01 ** VOLSER = MHLV03  **  DSN = PDSERES.SMS.PDSE
 CHECK
======= RETURN CODE IS 0. Data set is OK.

IGWT004 PAGE INSPECTION TOOL...ENDED WITHOUT TERMINATING ERRORS
```

Figure 10-121 PHATOOL CHECK output

List of virtual storage related user abends

User abends 1536, 1540, 1541, and 1542 address virtual storage constraints. Specify more region space to bypass the virtual storage shortage.

10.6.6 DFSMSdss physical DUMP and RESTORE

DFSMSdss provides four kinds of backup processing:

- Logical data set dumps
- Physical data set dumps
- Logical volume dumps
- Physical volume dumps

DFSMSdss can perform either logical or physical processing. If you dump a data set logically, DFSMSdss restores it logically; if you dump it physically, DFSMSdss restores it physically.

Logical processing operates against data sets independently of physical device format. The data sets are located by searching either the catalog or VTOC. If input volumes are specified through the LOGINDD, LOGINDYNAM, or STORGRP keywords, then data sets are located by searching VTOCs. Otherwise, data sets are located by searching the catalog.

Physical processing moves data at the track-image level and operates against volumes, tracks, and data sets. The data sets are located by searching the VTOC.

The processing method is determined by the keywords specified on the command. For example, LOGINDYNAM indicates a logical dump and INDYNAM a physical dump. Each type of processing offers different capabilities and advantages.

Physical processing is required to process a PDSE that has been corrupted. Such a condition could be indicated when accessing a PDSE results in an ABEND 0F4. The physical dump processing should complete successfully, compared to logical dump processing, since DFSMSdss is not using PDSE services to access the PDSE. Instead, the physical tracks

related to the PDSE are dumped as an image. An IBM service representative may ask you to capture a DFSMSdss physical dump of a PDSE to preserve the PDSE for later analysis.

You can select data sets for DFSMSdss processing by filtering on specified criteria. DFSMSdss can filter on fully qualified or partially qualified data set names (by using the INCLUDE or EXCLUDE keyword) and on various data set characteristics (by using the BY keyword; BY(DSORG,EQ,PDSE), for example).

Figure 10-122 shows a sample job for a DFSMSdss physical DUMP.

- DUMP command specifies DUMP processing.
- INDYNAM(SBOXB) indicates physical processing for volume SBOXB0.
- DFSMSdss filtering is done for a fully qualified data set name: PDSERES.SMS.PDSE.

```
//DUMP     EXEC PGM=ADRDSSU
//OUT1     DD DISP=(NEW,CATLG),
//            SPACE=(CYL,(40,40)),DCB=BLKSIZE=32760,
//            STORCLAS=SCDUMP,
//            DSN=PDSERES.PDSE.DUMPP4
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DUMP -
      INDYNAM(SBOXB0) -
      DATASET(INCLUDE(PDSERES.SMS.PDSE)) -
      OUTDDNAME(OUT1)
/*
```

Figure 10-122 DFSMSdss physical DUMP

Figure 10-123 shows the DFSMSdss messages related to Figure 10-122.

```
PAGE 0001   5695-DF175  DFSMSDSS V1R06.0 DATA SET SERVICES    2004.329 17:01
 DUMP -                                                        00090000
     INDYNAM(SBOXB0) -                                         00091002
     DATASET(INCLUDE(PDSERES.SMS.PDSE)) -                       00100000
     OUTDDNAME(OUT1)                                           00110002
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'DUMP '
ADR109I (R/I)-RI01 (01), 2004.329 17:01:40 INITIAL SCAN OF USER CONTROL STATEMENTS
COMPLETED.
ADR016I (001)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR006I (001)-STEND(01), 2004.329 17:01:40 EXECUTION BEGINS
ADR378I (001)-DTDS (01), THE FOLLOWING DATA SETS WERE SUCCESSFULLY PROCESSED
                         FROM VOLUME SBOXB0
                          PDSERES.SMS.PDSE
ADR006I (001)-STEND(02), 2004.329 17:01:43 EXECUTION ENDS
ADR013I (001)-CLTSK(01), 2004.329 17:01:43 TASK COMPLETED WITH RETURN CODE 0000
ADR012I (SCH)-DSSU (01), 2004.329 17:01:43 DFSMSDSS PROCESSING COMPLETE.
                         HIGHEST RETURN CODE IS 0000
```

Figure 10-123 DFSMSdss physical DUMP joblog

Figure 10-124 shows an example of a job for a DFSMSdss RESTORE based on Figure 10-122 on page 311.

- DATASET specifies a data set restore operation, using filtering.
- OUTDDNAME or OUTDYNAM is required for a physical restore, even for SMS-managed data. They are optional for a logical restore operation.
- RENAMEUNCONDITIONAL specifies that the data set should be restored with the new name PDSERES.SMS.PDSE.NEWNAME.

```
//RESTOR1  EXEC PGM=ADRDSSU
//IN1      DD DSN=PDSERES.PDSE.DUMPP4,
//          DISP=OLD
//OUT1     DD UNIT=3390,
//             VOL=SER=SBOXB0,
//             DISP=OLD
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 RESTORE DATASET(INCLUDE(PDSERES.SMS.PDSE)) -
            INDDNAME(IN1) -
            OUTDDNAME(OUT1) -
            RENAMEU(PDSERES.SMS.PDSE,PDSERES.SMS.PDSE.NEWNAME)
/*
```

Figure 10-124 DFSMSdss RESTORE

Figure 10-125 shows the DFSMSdss messages related to Figure 10-124.

```
PAGE 0001   5695-DF175 DFSMSDSS V1R06.0 DATA SET SERVICES   2004.329 17:24
 RESTORE DATASET(INCLUDE(PDSERES.SMS.PDSE)) -                 00080002
         INDDNAME(IN1) -                                      00090002
         OUTDDNAME(OUT1) -                                    00100002
       RENAMEU(PDSERES.SMS.PDSE,PDSERES.SMS.PDSE.NEWNAME)     00101002
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'RESTORE '
ADR109I (R/I)-RI01 (01), 2004.329 17:24:57 INITIAL SCAN OF USER CONTROL STATEMENTS
                         COMPLETED.
ADR016I (001)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR006I (001)-STEND(01), 2004.329 17:24:57 EXECUTION BEGINS
ADR780I (001)-TDDS (01), THE INPUT DUMP DATA SET BEING PROCESSED IS IN
                         PHYSICAL DATA SET FORMAT AND WAS CREATED BY
                         DFSMSDSS 1 RELEASE 6 MODIFICATION LEVEL 0
ADR395I (001)-TDPNV(02), DATA SET PDSERES.SMS.PDSE ALLOCATED WITH NEWNAME
                         PDSERES.SMS.PDSE.NEWNAME, ON VOLUME(S): SBOXB0
ADR378I (001)-TDDS (01), THE FOLLOWING DATA SETS WERE SUCCESSFULLY PROCESSED
                         FROM VOLUME SBOXB0
                         PDSERES.SMS.PDSE             RESTORED ON SBOXB0
ADR006I (001)-STEND(02), 2004.329 17:24:58 EXECUTION ENDS
ADR013I (001)-CLTSK(01), 2004.329 17:24:58 TASK COMPLETED WITH RETURN CODE 0000
ADR012I (SCH)-DSSU (01), 2004.329 17:24:58 DFSMSDSS PROCESSING COMPLETE.
                         HIGHEST RETURN CODE IS 0000
```

Figure 10-125 DFSMSdss RESTORE joblog

10.6.7 IEHLIST utility

IEHLIST is a system utility used to list entries in the directory of one or more partitioned data sets or PDSEs, or entries in an indexed or non-indexed volume table of contents. Any number of listings can be requested in a single execution of the program.

LISTVTOC

IEHLIST LISTVTOC can be used to list, partially or completely, entries in a specified volume table of contents (VTOC), whether indexed or non-indexed. The program lists the contents of selected data set control blocks (DSCBs) in edited or unedited form.

The *edited* format is a comprehensive listing of the DSCBs in the VTOC. It provides the status and attributes of the volume, and describes in depth the data sets residing on the volume.

Each Format-1 DSCB entry is identified by one or more indicator bits. A few of them are displayed in the formatted output of a LISTVTOC report by assigning a character to an indicator. They are displayed below the field name SMS.IND.

PDSE data sets are referred to an SMS.IND of *I*. *S* indicates that the data set is SMS managed. *R* indicates system determined block (SDB) size and data set can be reblocked.

The formatted report can also be used to identify the location of the Format-1 DSCB on a volume. The address of the Format-1 DSCB is used in a PDSE-owned token called VSGT.

The VSGT includes the volser for the PDSE and the TTR (track-track-record (hexadecimal)) for the associated Format-1 DSCB. The formatted LISTVTOC listing displays the address of the Format-1 DSCB in format C-H-R (cylinder-head-record (decimal)).

In some situations you might convert the decimal C-H-R to a hexadecimal TTR format (see “Interpreting output of D GRS commands” on page 239), or TTR to C-H-R (see “Which PDSE address space is used?” on page 245).

The unedited (dump) format produces a complete hexadecimal listing of the DSCBs in the VTOC. The listing is in an unedited dump form, requiring you to know the various formats of applicable DSCBs. For more information about (indexed) VTOCs, see *z/OS DFSMSdfp Advanced Services*, SC26-7400.

See *z/OS: DFSMSdfp Utilities*, SC26-7414 for more information about IEHLIST usage.

Examples

Figure 10-126 shows an example of displaying the VTOC on volume `MHLV03` in edited format.

```
//VTOCL  EXEC PGM=IEHLIST
//SYSPRINT DD SYSOUT=*
//DD1      DD UNIT=3390,VOL=SER=MHLV03,DISP=SHR
//SYSIN    DD *
    LISTVTOC FORMAT,VOL=3390=MHLV03
/*
```

Figure 10-126 IEHLIST LISTVTOC (volume)

Figure 10-127 shows an example of displaying a VTOC entry (Format-1 DSCB) for data set PDSERES.SMS.PDSE on volume MHLV03 in an edited format.

```
//VTOCL  EXEC PGM=IEHLIST
//SYSPRINT DD SYSOUT=*
//DD1      DD UNIT=3390,VOL=SER=MHLV03,DISP=SHR
//SYSIN    DD *
    LISTVTOC FORMAT,VOL=3390=MHLV03,DSN=PDSERES.SMS.PDSE
/*
```

Figure 10-127 IEHLIST LISTVTOC (single data set)

Figure 10-128 shows the report related to the job shown in Figure 10-127.

- Data set name: PDSERES.SMS.PDSE
- Volume serial number: MHLV03
- DSORG: PO
- SMS.IND: S I
- Start and end C-H for each extent: 84 0 88 14

```
   SYSTEMS SUPPORT UTILITIES---IEHLIST                              PAGE    1
DATE: 2004.324  TIME: 22.19.56
              CONTENTS OF VTOC ON VOL MHLV03  <THIS IS AN SMS MANAGED VOLUME>
---------------DATA SET NAME----------------  SER NO  SEQNO  DATE.CRE  DATE.EXP  DATE.REF  EXT DSORG RECFM OPTCD BLKSIZE
PDSERES.SMS.PDSE                             MHLV03      1  2004.323    00.000  2004.324    1  PO   FB     00    27920
SMS.IND   LRECL  KEYLEN  INITIAL ALLOC  2ND ALLOC    EXTEND       LAST BLK(T-R-L)  DIR.REM  F2 OR F3(C-H-R)  DSCB(C-H-R)
S   I        80            CYLS              5                                                              3   0  26
           EXTENTS  NO  LOW(C-H)   HIGH(C-H)
                     0    84  0       88 14
                          ----UNABLE TO CALCULATE EMPTY SPACE.
```

Figure 10-128 IEHLIST LISTVTOC report

10.6.8 SMF record type 42 (DFSMS statistics and configuration)

System management facilities (SMF) collects and records system and job-related information that your installation can use to analyze your system configuration.

SMF formats the information that it gathers into system-related records (or job-related records). System-related SMF records include information about the configuration, paging activity, and workload. Job-related records include information about CPU time, SYSOUT activity, and data set activity of each job and job step.

SMF record type 42 contains information about DFSMS statistics and configuration.

Note: APAR OA10091 should be applied to all z/OS V1R6 systems to fix a PDSE SMS recording error.

Subtype information for SMF record type 42:

Subtype 1 Created on a timed interval to collect device statistics. The time interval is specified in the IGDSMSxx parmlib member. Subtype 1 summarizes, on a storage-class basis, the buffer manager hits (number of page-read requests handled by the buffer manager). A Buffer Manager Facility (BMF) totals section (64 bytes) enables analysis of overall BMF performance. There is one storage-class summary section (64 bytes) for each storage class.

Subtype 2 Has one section (88 bytes) for each cache control unit (Model 3990-3) having at least one SMS-managed device attached. There is one section (16 bytes) for each SMS-managed volume attached to such a control unit.

Subtype 6 Records DASD data set level I/O statistics. There are two events that cause subtype 6 to be generated:

- Close

- Immediately after the recording of the type 30 interval record.

There is one type 42 subtype 6 record for each type 30 interval record.

The following list should give you an idea about the statistics related to PDSE (based on SMS classes) and data sets. See “SMF” on page 195 about interpreting the PDSE-related statistics and *z/OS: MVS System Management Facilities (SMF)*, SA22-7630, for general information about SMF.

- SMF Type 42 Subtype1
  - BMF Totals Section
    - SMF42TNA - Total number of storage classes.
    - SMF42TMT - Interval length. This is the elapsed time of the measurement period in seconds.
    - SMF42TRT - Total number of member data page reads.
    - SMF42TRH - Total number of member data page read hits handled by BMF.
    - SMF42TDT - Total number of directory data page reads.
    - SMF42TDH - Total number of directory data page read hits handled by BMF.
  - Storage Class Summary Section
    - SMF42PNL - Length of storage class name.
    - SMF42PNN - Storage class name.
    - SMF42SRT - Total number of member data page reads.
    - SMF42SRH - Total number of member data page read hits handled by BMF.
    - SMF42SDT - Total number of directory data page reads.
    - SMF42SDH - Total number of directory data page read hits handled by BMF.
- SMF Type 42 Subtype 6
  - Job Header Section (data set statistics)

    See *z/OS: MVS System Management Facilities (SMF)*, SA22-7630 for information.
  - Data Set Header Section

    See *z/OS: MVS System Management Facilities (SMF)*, SA22-7630 for information.
  - Data Set I/O Statistics Section
    - S42DSIOR- Average response time.
    - S42DSIOC - Average I/O connect time.
    - S42DSIOP - Average I/O pending time.
    - S42DSIOD - Average I/O disconnect time.
    - S42DSIOQ - Average control unit queue time.
    - S42DSION - Total number of I/Os.
    - S42DSCND - Number of cache candidates.

- S42DSSHTS - Number of cache hits.
- S42DSWCN - Number of write candidates.
- S42DSWHI - Number of write hits.
- S42DSSEQ - Number of sequential I/O operations. Operations counted here are not accumulated in S42DSCND and S42DSWCN.
- S42DSRLC - Number of record level cache I/O operations.
- S42DSICL - Number of inhibit cache load I/O operations.
- S42DSDA0 - Average I/O device-active-only time.
- S42DSMXR - Maximum data set I/O response time.
- S42DSMXS - Maximum data set service time.

– Access Method Statistics Section.

See *z/OS: MVS System Management Facilities (SMF)*, SA22-7630 for more information.

A

SMF analysis source code

This appendix contains the source code for the SMF analysis program.

SMF record type 42 with its various subtypes captures data for various parts of the SMS subsystem.

SMF record type 42 subtype 1 data

SMF record type 42 subtype 1 captures the PDSE buffer management function statistics.

We provide sample source code to print out the contents of the records as captured by SMF. The program is provided in assembler source form so that it can be assembled using the current system macros, but otherwise should not require any user modification for immediate use.

Figure A-1 shows an example of the JCL required to assemble and store the formatting program in a load module data set. The assembler source is expected to be in data set PDSERES.SMF42T1.JCL(SMF42T1A), and the result will be in data set PDSERES.SMF42T1J.PDS.

```
//MHLRES1L JOB (1234567,COMMENT),MHLRES1,TIME=10,NOTIFY=MHLRES1
//ASMHCL PROC
//ASM      EXEC PGM=ASMA90,REGION=0M,
//             PARM='OBJECT,NODECK'
//SYSLIN   DD  DSN=&&OBJ,DISP=(NEW,PASS),UNIT=SYSDA,
//             SPACE=(TRK,(10,2)),DCB=BLKSIZE=3120
//SYSLIB   DD  DISP=SHR,DSN=SYS1.MACLIB
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(CYL,(5,5))
//*
//LKED     EXEC PGM=HEWL,REGION=2048K,COND=(8,LE,ASM),
//             PARM='XREF,LIST,LET'
//SYSLIN   DD  DSN=&&OBJ,DISP=(OLD,DELETE)
//         DD  DDNAME=SYSIN
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(CYL,(5,5))
// PEND
//  EXEC ASMHCL
//ASM.SYSIN DD DISP=SHR,DSN=PDSERES.SMF42T1.JCL(SMF42T1A)
/*
//LKED.SYSLMOD  DD DSN=PDSERES.SMF42T1J.PDS,DISP=SHR
//LKED.SYSIN   DD *
 NAME SMF42T1(R)
```

Figure A-1 Example of JCL to assemble and link SMF type 42 formatting program

The source code in Example A-1 should be cut and pasted into a dataset then assembled. The source lines are provided to verify that all lines are cut and pasted, but attempting to capture the line numbers is not recommended. The line numbers run continuously from start to end. Details on the use of the SMF42T1 program can be found at 9.8, “PDSE buffer management statistics” on page 202.

Example: A-1 Source of program to print out contents of SMF type 42 subtype 1 records

```
        MACRO                                                           00010011
&NAME   SEGSTART                                                        00020011
&NAME   STM   14,12,12(13)           SAVE HIS REGS IN HIS SAVE AREA     00030011
R0      EQU   0                                                         00040011
R1      EQU   1                                                         00050011
R2      EQU   2                                                         00060011
R3      EQU   3                                                         00070011
R4      EQU   4                                                         00080011
R5      EQU   5                                                         00090011
R6      EQU   6                                                         00100011
```

```
R7       EQU   7                                                      00110011
R8       EQU   8                                                      00120011
R9       EQU   9                                                      00130011
R10      EQU   10                                                     00140011
R11      EQU   11                                                     00150011
RB       EQU   12                                                     00160011
R13      EQU   13                                                     00170011
R14      EQU   14                                                     00180011
R15      EQU   15                                                     00190011
         BALR  12,0                  SET UP ADDRESSABILITY            00200011
         USING *,12                  USE REG 12 AS BASE REG           00210011
         ST    13,SAVEREGS+4         SAVE @ OF HIS SAVEAREA IN MINE   00220011
         LA    03,SAVEREGS           LOAD @ OF MY SAVE AREA IN REG 3  00230011
         ST    03,8(13)              SAVE @ OF MY SAVE AREA IN HIS    00240011
         LR    13,03                 LOAD @ OF MY SAVE AREA IN REG 13 00250011
         MEND                                                         00260011
         MACRO                                                        00270011
&NAME2   SEGEND                                                       00280011
&NAME2   L     13,SAVEREGS+4         LOAD REG13 WITH @ OF HIS SAVE    00290011
         LM    14,12,12(13)          RESTORE REGS FROM HIS SAVEAREA   00300011
         XR    R15,R15                                                00310011
         BR    14                    RETURN TO CALLING RTN VIA REG 14 00320011
SAVEREGS DC    18F'0'                SET UP SAVE AREA                 00330011
         MEND                                                         00340011
SMFR42T1  SEGSTART                                                    00350011
*  THIS IS A SIMPLE PROGRAM TO DISPLAY THE CONTENTS OF VARIOUS OF     00360011
*  THE SMF TYPE 42 SUBTYPE 1 RECORDS, WHICH ARE THE SMS BMF           00370011
*  RECORDS                                                            00380011
*  THE IFASMFDP PROGRAM MUST HAVE BEEN USED                           00390011
*  TO SELECT RECORDS FROM EITHER THE ACTIVE SMF 'MAN' DATASETS OR     00400011
*  OFF A PREVIOUSLY EXTRACTED COPY OF THE 'MAN' DATASETS.             00410011
*                                                                     00420011
*  THE STANDARD SMF RECORD MAPPING MACROS ARE USED.                   00430011
*  REGISTER EQUATES TO PARTS OF THE SMF TYPE 42 RECORD                00440011
*  R3    START OF WHOLE RECORD                                        00450011
*  R4    START OF SMF42S1 SECTION                                     00460011
*  R5    START OF SMF42O1A BMF TOTALS SECTION                         00470011
*  R6    START OF SMF42O1B BMF BUFFER MANAGEMENT STATISTICS SECTION   00480011
*  OTHER REGISTER USES                                                00490011
*  R12   OVERALL BASE REGISTER                                        00500011
*  R7    RECORD TYPE/SUBTYPE CHECKING/WORKING                         00510011
*  R8    RECORD TYPE/SUBTYPE CHECKING/WORKING                         00520011
*  R9    USED FOR OFFSET LENGTH ON TRIPLETS                           00530011
*  R10   USED FOR BCT                                                 00540011
*  R11   USED FOR TIME CONVERSION                                     00550011
*  R14   USED FOR TIME CONVERSION                                     00560011
*  R15   USED FOR TIME CONVERSION                                     00570011
*                                                                     00580011
         OPEN SMFIN               QSAM GET LOCATE PROCESSING IS USED 00590011
         OPEN (PRINTDCB,(OUTPUT)) QSAM PUT MOVE   PROCESSING IS USED 00600011
         PUT  PRINTDCB,PRINTHDR                                       00610011
READ     GET SMFIN                                                    00620011
         LR    R3,R1              COPY PARAMETER POINTER             00630011
         USING SMF42,R3          -> SMF RECORD                       00640011
*   CHECK IF TYPE 02                                                 00650011
         CLI   SMF42RTY,X'02' (SAME DISPLAEMENT SMF ADMIN RECORD 02)  00660011
         BE    IGNORE                                                 00670011
         CLI   SMF42RTY,X'2A'      *   CHECK IF TYPE 42              00680011
         BNE   IGNORE                                                 00690011
CHKSTYP1 DS    0H                                                     00700011
```

```
         CLI   SMF42STY+1,X'01'    *   CHECK IF SUBTYPE 1              00710011
         BNE   IGNORE                                                  00720011
*  IS TYPE 42 SUBTYPE 1, SO EXTRACT DATA                              00730011
*  FIRST EXTRACT THE RECORD TIME AND DATE AND CONVERT TO HUMAN        00740011
*  THEN  ESTABLISH ADDRESSIBILITY TO THE VARIOUS SECTIONS.            00750011
*  GENERAL PROCESS IS LOAD R8 WITH OFFSET TO THE RELEVANT SECTION     00760011
*  ADD R8 TO R3                                                       00770011
*  THEN THE DSECTS SHOULD ADDRESS THE SECTIONS, HOWEVER MANY THERE ARE  00780011
         MVC   TIMEF,SMF42TME     SAVE THE SMF TIME IN 100THS OF SECS  00790011
         XR    R14,R14              CLEAR HIGH END OF PAIR             00800011
         L     R15,TIMEF           COMPLETE EVEN/ODD PAIR CONTENTS     00810011
         LA    R11,100            START BY DIVIDING BY 100 TO GET SECS 00820011
         DR    R14,R11             DIVIDE                             00830011
*        DC    F'0'  CREATE AN ABEND TO LOOK AT THE RECORDS           00840011
*  REMAINDER NOW IN R14 WHICH WE IGNORE                               00850011
*  QUOTIENT IN R15 - IE SECONDS WHICH WE CONVERT TO MINS + SECS       00860011
         XR    R14,R14              CLEAR HIGH END OF PAIR             00870011
         LA    R11,60             DIVIDE BY 60 TO GET MINS            00880011
         DR    R14,R11          DIVIDE TO GET MINUTES & SECONDS AS REM. 00890011
*  REMAINDER NOW IN R14 WHICH IS SECONDS WHICH WE MUST SAVE           00900011
*  QUOTIENT IN R15 - IE MINUTES FOR MORE PROCESSING                   00910011
         CVD   R14,TIMET           CONVERT TO PACKED DECIMAL          00920011
         OI    TIMET+7,X'0F'      FIX SIGN                            00930011
         UNPK  TIMEX+6(2),TIMET+6(2)  UNPACK SECONDS                   00940011
         MVI   TIMEX+5,C':'                                           00950011
         XR    R14,R14              CLEAR HIGH END OF PAIR            00960011
         LA    R11,60             DIVIDE BY 60 TO GET HOURS           00970011
         DR    R14,R11          DIVIDE TO GET HOURS & MINUTES AS REM.  00980011
*  REMAINDER NOW IN R14 WHICH IS MINUTES WHICH WE MUST SAVE           00990011
*  QUOTIENT IN R15 - IE HOURS WHICH WE MUST SAVE                      01000011
         CVD   R14,TIMET           CONVERT TO PACKED DECIMAL          01010011
         OI    TIMET+7,X'0F'      FIX SIGN                            01020011
         UNPK  TIMEX+3(2),TIMET+6(2)  UNPACK MINUTES                   01030011
         MVI   TIMEX+2,C':'                                           01040011
         CVD   R15,TIMET           CONVERT TO PACKED DECIMAL          01050011
         OI    TIMET+7,X'0F'      FIX SIGN                            01060011
         UNPK  TIMEX+0(2),TIMET+6(2)  UNPACK HOURS                     01070011
         MVC   P42TME1,TIMEX                                          01080011
         MVC   P42TME,TIMEX                                           01090011
         UNPK  P42DTE(7),SMF42DTE(4)                                  01100011
         OI    P42DTE+3,X'F0'                                         01110011
         CLC   P42DTE(2),=CL2'01'                                     01120011
         BNE   *+10                                                   01130011
         MVC   P42DTE(2),=C'20'                                       01140011
         MVC   P42DTE(2),=C'20'                                       01150011
         MVC   P42DTE1,P42DTE                                         01160011
*                                                                     01170011
         LA    R4,SMF42END    END OF HEADER, START OF DATA            01180011
         USING SMF42S1,R4                                             01190011
         L     R8,SMF42BMO    OFFSET TO BMF TOTALS SECTIONS           01200011
         LH    R9,SMF42BML    LENGTH OF BMF TOTALS SECTIONS           01210011
         LH    R10,SMF42BMN   NUMBER OF BMF TOTALS SECTIONS           01220011
BMFTRIP  DS    0H                                                     01230011
         CVD   R10,DWORD                                              01240011
         OI    DWORD+7,X'0F'                                          01250011
         UNPK  P42T#(7),DWORD+4(4)                                    01260011
         PUT   PRINTDCB,PRBMSL1                                       01270011
         LA    R5,0(R3,R8)                                            01280011
         USING SMF4201A,R5                                            01290011
         L     R7,SMF42TNA                                            01300011
```

```
         CVD   R7,DWORD                                                  01310011
         OI    DWORD+7,X'0F'                                            01320011
         UNPK  P42TNA(7),DWORD+4(4)                                     01330011
         L     R7,SMF42TMT                                              01340011
         CVD   R7,DWORD                                                 01350011
         OI    DWORD+7,X'0F'                                            01360011
         UNPK  P42TMT(7),DWORD+4(4)                                     01370011
         L     R7,SMF42TRT                                              01380011
         CVD   R7,DWORD                                                 01390011
         OI    DWORD+7,X'0F'                                            01400011
         UNPK  P42TRT(7),DWORD+4(4)                                     01410011
         L     R7,SMF42TRH                                              01420011
         CVD   R7,DWORD                                                 01430011
         OI    DWORD+7,X'0F'                                            01440011
         UNPK  P42TRH(7),DWORD+4(4)                                     01450011
         L     R7,SMF42TDT                                              01460011
         CVD   R7,DWORD                                                 01470011
         OI    DWORD+7,X'0F'                                            01480011
         UNPK  P42TDT(7),DWORD+4(4)                                     01490011
         L     R7,SMF42TDH                                              01500011
         CVD   R7,DWORD                                                 01510011
         OI    DWORD+7,X'0F'                                            01520011
         UNPK  P42TDH(7),DWORD+4(4)                                     01530011
         L     R7,SMF42BUF                                              01540011
         CVD   R7,DWORD                                                 01550011
         OI    DWORD+7,X'0F'                                            01560011
         UNPK  P42BUF(7),DWORD+4(4)                                     01570011
         L     R7,SMF42BMX                                              01580011
         CVD   R7,DWORD                                                 01590011
         OI    DWORD+7,X'0F'                                            01600011
         UNPK  P42BMX(7),DWORD+4(4)                                     01610011
         LH    R7,SMF42LRU                                              01620011
         CVD   R7,DWORD                                                 01630011
         OI    DWORD+7,X'0F'                                            01640011
         UNPK  P42LRU(7),DWORD+4(4)                                     01650011
         LH    R7,SMF42UIC                                              01660011
         CVD   R7,DWORD                                                 01670011
         OI    DWORD+7,X'0F'                                            01680011
         UNPK  P42UIC(7),DWORD+4(4)                                     01690011
         DROP  R5                                                       01700011
         PUT   PRINTDCB,PRBMSL2                                         01710011
         PUT   PRINTDCB,PRBMSL3                                         01720011
         PUT   PRINTDCB,PRINTBLK                                        01730011
*  LOOP BACK AT THIS POINT IF THERE ARE ANY MORE TRIPLETS               01740011
         LA    R8,0(R8,R9)                                              01750011
         BCT   R10,BMFTRIP                                              01760011
*  WHEN BCT REACHES ZERO GO ON WITH THE SCLASS ENTRIES                  01770011
*  PROCESS THE SC ENTRIES TRIPLET.                                      01780011
*  FIRST FULLWORD IS OFFSET TO WHERE THE TRIPLETS START                 01790011
*  SECOND HW IS THE LENGTH OF EACH TRIPLET                              01800011
*  THIRD HW IS THE NUMBER OF TRIPLETS                                   01810011
         L     R8,SMF42SCO    OFFSET TO THE SCLASS SECTION              01820011
         LH    R9,SMF42SCL    LENGTH OF THE SCLASS SECTIONS             01830011
         LH    R10,SMF42SCN   NUMBER OF SCLASS SECTIONS                 01840011
SCOTRIP  DS    0H                                                       01850011
         LA    R6,0(R3,R8)                                              01860011
         USING SMF4201B,R6                                              01870011
         MVC   P42PNN,SMF42PNN                                          01880011
         PUT   PRINTDCB,PRINTL1                                         01890011
         L     R7,SMF42SRT                                              01900011
```

```
         CVD   R7,DWORD                                                  01910011
         OI    DWORD+7,X'0F'                                            01920011
         UNPK  P42SRT(7),DWORD+4(4)                                     01930011
         L     R7,SMF42SRH                                              01940011
         CVD   R7,DWORD                                                 01950011
         OI    DWORD+7,X'0F'                                            01960011
         UNPK  P42SRH(7),DWORD+4(4)                                     01970011
         L     R7,SMF42SDT                                              01980011
         CVD   R7,DWORD                                                 01990011
         OI    DWORD+7,X'0F'                                            02000011
         UNPK  P42SDT(7),DWORD+4(4)                                     02010011
         L     R7,SMF42SDH                                              02020011
         CVD   R7,DWORD                                                 02030011
         OI    DWORD+7,X'0F'                                            02040011
         UNPK  P42SDH(7),DWORD+4(4)                                     02050011
WRITEIT  DS    0H                                                       02060011
         PUT   PRINTDCB,PRINTL2                                         02070011
         PUT   PRINTDCB,PRINTL3                                         02080011
         PUT   PRINTDCB,PRINTBLK                                        02090011
*  LOOP BACK AT THIS POINT IF THERE ARE ANY MORE TRIPLETS               02100011
*                                                                       02110011
*  WHEN BCT REACHES ZERO GO GET ANOTHER RECORD                          02120011
         LA    R8,0(R8,R9)                                              02130011
         BCT   R10,SCOTRIP                                              02140011
         B     READ                                                     02150011
IGNORE   DS    0H EXIT WITH OUT WRITING IF NOT THE RIGHT RECORDS        02160011
         B     READ                                                     02170011
FINI     DS    0H                                                       02180011
         SEGEND                                                         02190011
SMFIN    DCB   DDNAME=SMFIN,DSORG=PS,MACRF=(GL),EROPT=SKP,EODAD=FINI    02200011
PRINTDCB DCB   DDNAME=PRINT,DSORG=PS,MACRF=(PM),LRECL=133               02210011
DWORD    DS    D                                                        02220011
         ORG   DWORD                                                    02230011
         DC    C'12345678'                                              02240011
TIMET    DS    D                 WORKAREA FOR TIME CONVERSION           02250011
TIMEX    DS    D                 WORKAREA FOR TIME CONVERSION           02260011
TIMEF    DS    F                 WORKAREA FOR TIME CONVERSION           02270011
PRINTBLK DC    CL133' '                                                 02280011
PRINTHDR DC    CL133'1SMF TYPE 42 S/TYPE 1 RECORDS. COLS USE SMF NAMES' 02290011
PRINTL1  DC    CL133' SMF42PNN (SCLASS):'                               02300011
         ORG   PRINTL1+20                                               02310011
P42PNN   DC    CL30' '                                                  02320011
         ORG                                                            02330011
PRBMSL1  DC    CL133' BMF TOTALS SET #:'                                02340011
         ORG   PRBMSL1+20                                               02350011
P42T#    DC    CL8' '                                                   02360011
         ORG                                                            02370011
*                                                                       02380011
PRBMSL2  DC    CL133' '                                                 02390011
         ORG   PRBMSL2+1                                                02400011
P42TMEH1 DC    CL8'HH:MM:SS '                                           02410011
         DC    CL1' '                                                   02420011
P42DTEH1 DC    CL8'YYYYDDD '                                            02430011
P42TNAH  DC    CL9'SMF42TNA'                                            02440011
P42TMTH  DC    CL9'SMF42TMT'                                            02450011
P42TRTH  DC    CL9'SMF42TRT'                                            02460011
P42TRHH  DC    CL9'SMF42TRH'                                            02470011
P42TDTH  DC    CL9'SMF42TDT'                                            02480011
P42TDHH  DC    CL9'SMF42TDH'                                            02490011
P42BUFH  DC    CL9'SMF42BUF'                                            02500011
```

```
P42BMXH  DC    CL9'SMF42BMX'                                          02510011
P42LRUH  DC    CL9'SMF42LRU'                                          02520011
P42UCIH  DC    CL9'SMF42UIC'                                          02530011
         ORG                                                          02540011
*                                                                     02550011
PRBMSL3  DC    CL133' '                                               02560011
         ORG   PRBMSL3+1                                              02570011
P42TME1  DC    CL8' '                                                 02580011
         DC    CL1' '                                                 02590011
P42DTE1  DC    CL8' '                                                 02600011
P42TNA   DC    CL9' '   SMF42TNA                                      02610011
P42TMT   DC    CL9' '   SMF42TMT                                      02620011
P42TRT   DC    CL9' '   SMF42TRT                                      02630011
P42TRH   DC    CL9' '   SMF42TRH                                      02640011
P42TDT   DC    CL9' '   SMF42TDT                                      02650011
P42TDH   DC    CL9' '   SMF42TDH                                      02660011
P42BUF   DC    CL9' '   SMF42BUF                                      02670011
P42BMX   DC    CL9' '   SMF42BMX                                      02680011
P42LRU   DC    CL9' '   SMF42LRU                                      02690011
P42UIC   DC    CL9' '   SMF42UIC                                      02700011
         ORG                                                          02710011
*                                                                     02720011
PRINTL2  DC    CL133' '                                               02730011
         ORG   PRINTL2+1                                              02740011
P42TMEH  DC    CL8'HH:MM:SS '                                         02750011
         DC    CL1' '                                                 02760011
P42DTEH  DC    CL8'YYYYDDD '                                          02770011
P42SRTH  DC    CL9'SMF42SRT'                                          02780011
P42SRHH  DC    CL9'SMF42SRH'                                          02790011
P42SDTH  DC    CL9'SMF42SDT'                                          02800011
P42SDHH  DC    CL9'SMF42SDH'                                          02810011
         ORG                                                          02820011
*                                                                     02830011
PRINTL3  DC    CL133' '                                               02840011
         ORG   PRINTL3+1                                              02850011
P42TME   DC    CL8' '                                                 02860011
         DC    CL1' '                                                 02870011
P42DTE   DC    CL8' '                                                 02880011
P42SRT   DC    CL9' '     SMF42SRT                                    02890011
P42SRH   DC    CL9' '     SMF42SRH                                    02900011
P42SDT   DC    CL9' '     SMF42SDT                                    02910011
P42SDH   DC    CL9' '     SMF42SDH                                    02920011
         ORG                                                          02930011
SMFDSECT DSECT                                                        02940011
*        IFASMFR  (42) THIS DOES NOT EXPAND THE SUBTYPES AS IT SHOULD 02950011
         IGWSMF  SMF42_01=YES                                         02960011
         END                                                          02970011
```

B

PDSE APARs

This appendix provides reference information for APARs that may be described elsewhere in the text. The content was captured at this writing, but the current APAR information should be rechecked if it is possible that it is applicable. If any of the following APARs have been superseded, PTFs for the later versions should be used.

APARs referenced in the book

This appendix contains details of the APARs referenced in the book.

OW40072

PROBLEM SUMMARY:
****
USERS AFFECTED: Using PDSEs in a dynamic linklist.
****

PROBLEM DESCRIPTION:
If a PDSE is added to a dynamic linklist, you can remove it from the linklist but you cannot scratch the PDSE. When trying to delete the PDSE using ISPF you may get a `data set in use` message or msgIEC331I RC042 RSN006 DIAG INFO 043D57D3 from DADSM.
****

RECOMMENDATION:
****

DFSMS does a global connect to PDSE data sets that are part of the linklist concatenation. There is no global disconnect function, because there may be multiple users of a loaded PDSE member. The PDSE function that is invoked by delete or scratch processing requires that there are no outstanding connections, so the user will get a `data set in use` message in response to deleting a PDSE.

PROBLEM CONCLUSION:
This APAR is being closed as a permanent restriction. PDSE data sets in a linklist concatenation are accessed with a global connection. There is no global disconnect function. The globally connected PDSE cannot be deleted until the system is IPLed and the PDSE is no longer connected.

TEMPORARY FIX:

COMMENTS:
This global connection that remains in place after a PDSE is removed from the linklist can also be the reason for an abend213 RC70 (PDSE sharing conflict) when PDSE NORMAL sharing is used. The abend for *this* cause can be avoided if PDSE EXTENDED sharing can be set up. This is done by adding the parm PDSESHARING(EXTENDED) to parmlib member IGDSMSxx and is allowed only when PDSEs are operated on in a sysplex.

OW44229

PROBLEM SUMMARY:
****
USERS AFFECTED: Currently a PDSE must be cataloged when it is specified in LINKLIST, in order to enable the PDSE connect token to extract the Storage Class that is used to determine the caching attributes for the data set. Because unmanaged PDSEs do not have a storage class, it is unnecessary that the catalog be accessed at NIP. When a PDSE is added to the linklist with a volser specified, the PDSE is bypassed. Users would like to be able to specify an unmanaged PDSE without cataloging it in the master catalog of the system. Because of LLA processing, the unmanaged PDSE will still have to be cataloged in a user catalog when LLA starts.
****

PROBLEM DESCRIPTION:
PDSE linklist processing will be modified to accept managed or unmanaged PDSEs in linklist.

When a managed PDSE is not cataloged in the system master catalog, it will use the may cache attribute instead of the attribute specified in its catalog entry storage class. The PDSEs must still be cataloged in a user catalog because LLA will allocate them by name.
****
RECOMMENDATION:
****
See Description

OW53245

PROBLEM SUMMARY:
****
USERS AFFECTED: PDSEs use excessive amounts of ECSA. The amount of ECSA is largely determined by the number of members that are accessed concurrently. The amount used is normally between 1000 and 2000 bytes per member. The fact that PDSEs can have tens of thousands of members can cause the amount of ECSA used to grow dramatically especially when using programs such as IEBCOPY that access all members of a PDSE at one time. This can also happen for user programs that do BLDLs for all members in a PDSE.
****

PROBLEM DESCRIPTION:
See Users Affected.
****
RECOMMENDATION:
****
The majority of PDSE processing that used to occur in the user address spaces is now moved to the new SMSPDSE address space. In addition, all processing that occurred in the SMXC and SYSBMFAS address spaces is also moved to the new SMSPDSE address space. This has enabled the majority of PDSE control blocks that were placed in the ECSA to be moved to Extended User private in the SMSPDSE address space. There are still some control blocks associated with reading, writing, and loading members from PDSEs; these control blocks will continue to be located in ECSA. In addition, End of Memory processing has been enhanced to use the same code that Cancel processing uses. Because the cancel code has been more extensively tested, this will increase the reliability of PDSE processing after a user address space is forced. System programmers will notice that the SMXC and SYSBMAS address spaces have been replaced with a single SMSPDSE address space. When collecting documentation for PDSE-related problems they will be required to dump the SMSPDSE address space.

Changed message:
IGW040I text explanation: In the message text, text is: PDSE CONNECTING TO XCF FOR IPLID PDSE CONNECTED TO XCF FOR IPLID PDSE CONNECTING TO XCF FOR SIGNALING PDSE CONNECTED TO XCF FOR SIGNALING End of task at EOM Failed ASID:aaaa
During PDSE initialization in a sysplex environment, the system initializing PDSEs must establish communication through XCF with the other systems in the sysplex in order to let the other systems know that PDSEs are accessible on the new system. Also, a system being varied offline must establish communication through XCF with the other systems in the sysplex in order to let the other systems know that the PDSEs on the system are being taken offline are no longer accessible. This message is issued to indicate that the system has either started or completed establishing messaging between two systems. Notifies you that EOM (end of memory) processing for ASID aaaa was not successful. There should be a dump in this case also.
System Action: None.
Operator Response: None.

Application Programmer Response: None.
System Programmer Response: If PDSE processing is unavailable, consult the log to see if both IPLID and Signaling have completed connecting. Source: DFSMSdfp Detecting Module: IGWLXFMX

Added messages: IGW041A XCM MESSAGE UNABLE TO RECEIVE ssssssss UNKNOWN SENDER RECEIVING GROUP gggggggggggggg
Explanation: XCM, which is the XCF subcomponent of DFSMS, has received a message from a member ssssssss for group member gggggggggggggg. XCM system has been unable to identify the member ssssssss using the IXCQUERY service of XCF. This is an unexpected situation. ssssssss The XCF connect token for the member that sent the message. gggggggggggggg The member name for the receiving member.
System Action: PDSE or SMSVSAM services may be unavailable.
Operator Response: None.
Application Programmer Response: None.
System Programmer Response: If gggggggggggggg is IGWSYS01---------- or IGWSYS00----------, perform a dump of SMXC or SMSPDSE, XCFAS, and the XCFAS data spaces, and notify IBM Service.
Source: DFSMSdfp
Detecting Module: IGWLXFMX IGW042A PDSE End of Memory Processing Stalled ASID:aaaa
Explanation: PDSE has been unable to complete EOM (end of memory) processing for ASID aaaa. Run the V SMS,PDSE,ANALYSIS command to see whether there are any problems associated with PDSE processing that may have to be cleared up to enable EOM processing to complete.
System Action: None
Operator Response: None.
Application Programmer Response: None.
System Programmer Response: Run V SMS,PDSE,ANALYSIS to see whether the underlying cause of the delayed EOM processing can be determined.

OW57609

PROBLEM SUMMARY:
****
USERS AFFECTED: Currently if a PDSE is in linklist, there is no way the data set can be deleted without removing the PDSE from linklist and re-IPLing all systems that are using the data set. This restriction does not apply to PDSs.
****

PROBLEM DESCRIPTION: See Users affected.
****
RECOMMENDATION:
****
PDSE support will be modified to enable a PDSE that is globally connected by linklist to be disconnected. By disconnecting the data set, the PDSE can be deleted in the same way that PDSs are. Note that even when a linklist is undefined, LLA will continue to be allocated to the PDSEs and PDSs in the prior linklist and you will have to shut down and restart LLA or activate a new LLA parmlib member that includes a delete for the specified PDSE. This is the same process that would be required for a PDS. Note: The fixes provided by OW57609 / OW57671 are applicable to PDSE libraries added to LNKLST at some point after IPL. For PDSE libraries included in LNKLST at IPL time, the permanent restriction described by OW40072 will continue to be applicable.

If fix for OW57609 is applied, the fix for OA09361 must also be applied.

OW57671

PROBLEM SUMMARY:
****
USERS AFFECTED: Currently if a PDSE is in linklist, there is no way the data set can be deleted without removing the PDSE from linklist and re-IPLing all systems that are using the data set. This restriction does not apply to PDSs.
****

PROBLEM DESCRIPTION: See Users affected.
****
RECOMMENDATION:
****
PDSE support will be modified to enable a PDSE that is globally connected by linklist to be disconnected. By disconnecting the data set, the PDSE can be deleted in the same way that PDSs are. It should be noted that even when a linklist is undefined, LLA will continue to be allocated to the PDSEs and PDSs in the prior linklist and you will have to shut down and restart LLA or activate a new LLA parmlib member that includes a delete for the specified PDSE. This is the same process that would be required for a PDS. NOTE: The fixes provided by OW57609 / OW57671 are applicable to PDSE libraries added to LNKLST at some point after IPL. For PDSE libraries included in LNKLST at IPL time, the permanent restriction described by OW40072 will continue to be applicable.

OA08941

PROBLEM SUMMARY:
****
USERS AFFECTED: Currently when a problem is identified with the PDSE analysis and monitor, a system programmer may not be able to decide whether the holder of a latch should be cancelled or forced immediately or whether the resolution of the problem can be delayed. The missing information is which jobs are waiting on a latch that is not being released. In addition there are cases where PDSE service requests a SLIP dump, but there is no easy way to determine the address of a PDSE module.
****

PROBLEM DESCRIPTION: See users affected.
****
RECOMMENDATION:
****
Add two new PDSE commands D SMS,PDSE<1>,LATCH(aaaaaaaa),DETAILED|SUMMARY
D SMS,PDSE<1>,MODULE(mmmmmmmm)

OA06701

PROBLEM SUMMARY:
PROBLEM CONCLUSION:
TEMPORARY FIX:
COMMENTS: In order to improve performance for the PDSE-to-PDS copy, the FAMS component of DFSMS would require a redesign of its code. The size of the redesign would exceed the scope of an APAR fix. This APAR is closed as a suggestion for future improvement. At the present time the most efficient way to copy data members between PDSEs and PDS data sets is to use a two-step process:
1) Use IEBCOPY UNLOAD to copy selected members or the entire PDSE to a sequential file.
2) Use IEBCOPY LOAD to copy these members or data set into a PDSE.

OA06308

PROBLEM SUMMARY:
****
USERS AFFECTED: Users of IEBCOPY and FAMS who selectively copy very few members from very large data sets, where the copy involves a PDSE data set.
****

PROBLEM DESCRIPTION:
The performance of IEBCOPY and FAMS is poor for selective copy involving PDSEs, especially when very few members are copied from very large PDSEs.
****
RECOMMENDATION:
****
The performance of IEBCOPY and FAMS is poor for selective copy involving PDSEs, especially when very few members are copied from very large PDSEs. The original APAR OW49451 was meant to improve the performance. But, it had a small bug, due to which the performance did not improve. This APAR fixes that bug.

PROBLEM CONCLUSION:
FAMS has been corrected

TEMPORARY FIX:
******** HIPER ********

OA09265

PROBLEM SUMMARY:
****
USERS AFFECTED: Currently when a problem is identified with the PDSE analysis and monitor, a system programmer may not be able to decide whether the holder of a latch should be cancelled or forced immediately or whether the resolution of the problem can be delayed. The missing information is which jobs are waiting on a latch that is not being released. In addition, there are cases where PDSE service requests a SLIP dump, but there is no easy way to determine the address of a PDSE module.
****

PROBLEM DESCRIPTION: See Users Affected.
****
RECOMMENDATION:
****
Add two new PDSE commands D SMS,PDSE<1>,LATCH(aaaaaaaa),DETAILED|SUMMARY
D SMS,PDSE<1>,MODULE(mmmmmmmm)

OA06884

The fix provided by this APAR can result in an increase in CPU time, use of more real storage than before, or both. This is because PDSE program objects become eligible for load into Hiperspace with this fix. The problem can be addressed by changes to the Hiperspace management parameters as documented in the PTF for OA08991. The PTF for OA08991 does not change the executable code from what was provided on AO06884. The PTF fix for OA06884 should not be applied, or if it is applied, the HOLDDATA associated with the PTF fixing OA08991 should be reviewed even if the PTF is not applied.

PROBLEM SUMMARY:
****
USERS AFFECTED: Users of PDSEs that are caching PDSE data in Hiperspaces.
****

PROBLEM DESCRIPTION:
When caching PDSE data in Hiperspace, the Hiperspace fills up and no new data is cached. This will cause decreased performance but no other symptoms.
****
RECOMMENDATION:
****
PDSE data caching does not work. PROBLEM CONCLUSION: Code is changed to delete data from Hiperspace when the member is disconnected or via LRU.

TEMPORARY FIX:

COMMENTS: **** PE04/09/09 PTF IN ERROR. See APAR OA08991 for description.

OA08991

ERROR DESCRIPTION:
High CPU utilization in SMSPDSE address space after installation of UA10647. Utilization steadily increases and does not level off. This eventually results in CPU performance degradation.

LOCAL FIX:

PROBLEM SUMMARY:
****
USERS AFFECTED: Users of PDSEs containing program objects.
****
PROBLEM DESCRIPTION: This APAR results in increased PDSE member caching. This can cause increased CPU utilization.
****
RECOMMENDATION:
****
This APAR results in increased PDSE member caching. This can cause increased CPU utilization. PROBLEM CONCLUSION: This APAR results in increased PDSE member caching. This can cause increased CPU utilization.
See the ++hold data for each PTF for information about how to deal with this.

OA09162

PROBLEM SUMMARY:
****
USERS AFFECTED: All users of PDSEs in z/Architecture and users of PDSEs in 390 architecture with PDSE Hiperspaces.
****

PROBLEM DESCRIPTION:
All users of PDSEs in z/Architecture and users of PDSEs in 390 architecture with PDSE Hiperspaces will get unexpected member caching. All program objects will be cached and other members may be cached incorrectly if the PDSEs are opened in a concatenation.
****

RECOMMENDATION:
****

All users of PDSEs in z/Architecture and users of PDSEs in 390 architecture with PDSE Hiperspaces will get unexpected member caching. All program objects will be cached and other members may be cached incorrectly if the PDSEs are opened in a concatenation.
PROBLEM CONCLUSION: Change code so that only members of PDSEs with a storage class that indicates that they should be cached are cached. Members of PDSEs in linklist will be cached after IPL if their storage classes indicate that they should be cached. Members of unmanaged PDSEs will not be cached. PDSEs opened in a concatenation will have their members cached if, and only if, their storage classes indicate that they should be cached.

OA07807

PROBLEM SUMMARY:
****

USERS AFFECTED: All DFSMS users of R13 and higher.
****

PROBLEM DESCRIPTION:
Raise the upper limit of HSP_SIZE parameter (PDSE_HSP_SIZE and PDSE1_HSP_SIZE in release R16) of the sys1.parmlib member IGDSMSxx from 512 MB to 2047 MB.
****

RECOMMENDATION:
****

This enhancement allows up to 2047 MB for PDSE member caching.

OA09361

ERROR DESCRIPTION: ABEND0F4 RC24 RSN070A0021 in IGWBVLP1 +0516 (UA03482). Abend occurs because of mismatched validity check fields in the LPRB+18 and the related LSCB +24 (IGWBVT_INVALID_LSCB). This happens during a connect file for a PDSE in linklist. The related DUB points to an LSCB that is for a different data set.

LOCAL FIX: Removing UA12169 or refraining from altering the linklist may provide relief from this problem.

PROBLEM SUMMARY: Any system with fix for OW57609 installed is affected.

USERS AFFECTED:
When a linklist is updated and the code that enables globally connected PDSEs to be disconnected, the chain of globally connected PDSEs can become corrupted and be cross-chained with a normally connected chain. This can result in the global connects being lost when a job completes if its chain points into the global chain. The most likely result of this will be loss of access to PDSEs in linklist.

OA09955

ERROR DESCRIPTION:
During IEBCOPY PDSE to PDSE, when the output PDSE is VLF-managed, VLF is not notified of a new or changed member.

OA10091

ERROR DESCRIPTION: SMF type 42 records may not be written for PDSEs.

Info APAR numbers for currently supported z/OS DFSMS releases

II12221	INFORMATION ON SUPPORT FOR UNMANAGED HFS AND PDSE
II13335	DFSMS 1.6 *HDZ11F0* CURRENT PDSE MAINTENANCE
II13336	z/OS DFSMS 1.3 *HDZ11G0* CURRENT PDSE MAINTENANCE
II13875	z/OS DFSMS 1.5 *HDZ11H0* CURRENT PDSE MAINTENANCE
II13967	z/OS DFSMS 1.6 *HDZ11J0* CURRENT PDSE MAINTENANCE

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering this book, see “How to get IBM Redbooks” on page 335.

- *z/OS v1R3 and V1R5 DFSNS Technical Guide,* SG24-5694

Other publications

These publications are also relevant as further information sources:

- *EREP Reference,* GC35-0152
- *EREP User's Guide*, GC35-0151
- *z/OS DFSMSdfp Storage Administration Reference,* SC26-7402
- *z/OS DFSMSdfp Utilities*, SC26-7414
- *z/OS: MVS Diagnosis: Tools and Service Aids*, GA22-7589
- *z/OS MVS Initialization and Tuning Reference,* SA22-7592
- *z/OS: MVS Interactive Problem Control System (IPCS) Commands*, SA22-7594
- *z/OS MVS System Commands*, GC28-1781
- *z/OS: MVS System Management Facilities (SMF)*, SA22-7630
- *z/OS V1R3.0-V1R6.0 DFSMS Macro Instructions for Data Sets*, SC26-7408
- *z/OS V1R6.0 DFSMS/MVS Using Data Sets*, SC26-7410

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications, and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at ibm.com®:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

D

E

N

O

P

Q

R

S

T

U

V

W

X

Z

Partitioned Data Set Extended Usage Guide

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

Partitioned Data Set Extended Usage Guide

Redbooks

Learn the latest PDSE functions

Use PDSEs for storing programs

Compare PDS and PDSE capabilities

This IBM Redbook will help you install, manage, and use PDSEs. It is a substantial rewrite of *Partitioned Data Set Extended Usage Guide*, SG24-6106, which was published in March 2001. It brings the originally published material up to date and includes enhancements made to PDSEs since the original book was published.

The book describes how to define and use PDSEs, and shows how PDSEs offer improvements over ordinary partitioned data sets, such as sharing between users and systems with integrity, an automatically extending directory, and reuse of space.

Information for storage administrators and systems programmers includes system-level information about sharing in a sysplex, the use and definition of non-SMS PDSEs, and how to diagnose problems with PDSEs. The advantages of using PDSEs for storing executable code are explained.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6106-01 ISBN 0738490741