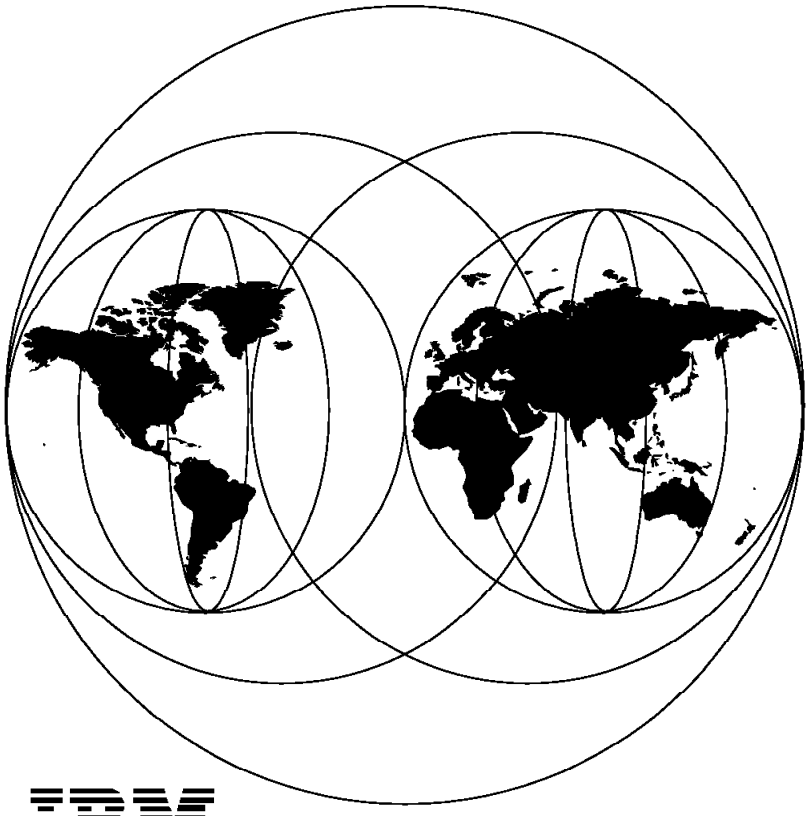
SG24-4892-00

DFSMS/MVS V1R4 Technical Guide

June 1997



IBM

International Technical Support Organization

San Jose Center

International Technical Support Organization

SG24-4892-00

DFSMS/MVS V1R4 Technical Guide

June 1997

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special Notices” on page 143.

First Edition (June 1997)

This edition applies to Version 1, Release 4 of DFSMS/MVS, Program Number 5695-DF1 for use with the MVS/ESA platform and OS/390 operating system.

Warning

This book is based on a pre-GA version of a product and may not apply when the product becomes generally available. It is recommended that, when the product becomes generally available, you destroy all copies of this version of the book that you have in your possession.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. QXXE Building 80-E2
650 Harry Road
San Jose, California 95120-6099

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Preface

This redbook introduces the latest version of DFSMS/MVS. It provides a technical overview of the functions and enhancements of DFSMS/MVS V1R4.

DFSMS/MVS V1R4 improves system and data availability, increases productivity, offers performance enhancements, and fulfills several important customer requirements.

The first chapter provides an overview, by functional component, of the new functions and enhancements in DFSMS/MVS V1R4. Each subsequent chapter is dedicated to a component of DFSMS/MVS. Installation, migration, and coexistence considerations with releases of DFSMS/MVS previous to V1R4 are provided.

Some of the new features of DFSMS/MVS V1R4 covered in this redbook include:

- Reducing space-related outages
- DFSMShsm Duplex Tape, ABARS, and VSAM RLS CDS processing enhancements
- DFSMSrmm trial-run capability
- System-managed buffering and load enhancements of VSAM Extended Format KSDSs
- SAM compression improvements
- New OAM SMF records
- Improved access to enterprise data through the DFM/MVS data agent
- Catalog Search Interface
- Batch ACS testing improvements

This redbook is written for experienced storage administrators, systems programmers, and other technical professionals who require a technical update of the new features and enhancements in this release of DFSMS/MVS.

The Team That Wrote This Redbook

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 Software Engineer at the International Technical Support Organization, San Jose Center. Before joining the ITSO, Mary worked in the Education Support Center in Poughkeepsie, New York. She has more than 18 years of experience with IBM in large systems and storage product education, system engineering and consultancy, marketing support, and system programming.

Peter Zerbini is an Adviser in Germany. He has worked at IBM for 24 years and has 12 years of experience in storage software and storage management. His areas of expertise include DFSMS/MVS, DFSMS Optimizer, DASD, tape, and optical. Peter is currently the Marketing Support and Last Level specialist for DFSMS/MVS, DFSMS Optimizer, and optical in Germany.

Geoff Littlewood is a Senior Systems Specialist in the United Kingdom. He has 18 years of experience in storage and storage management. Geoff holds an MSC in Mathematics and Computing Science from Sheffield University in the United Kingdom. His areas of expertise include DASD, tape, and SMS. Geoff is currently the Marketing and Technical Support specialist for DFSMS/MVS and storage management in the United Kingdom.

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

Ed Baker
Storage Systems Division - Tucson

Charlie Burger
Advanced Technical Support Organization - San Jose

Cecilia Carranza Lewis
Storage Systems Division - San Jose

Jerry Codde
Storage Systems Division - San Jose

Ed Daray
Storage Systems Division - San Jose

Scott Drummond
DFSMS Brand Manager - San Jose

Tina Dunton
Storage Systems Division - San Jose

Nadine Hart
Storage Systems Division - San Jose

Bob Kern
Storage Systems Division - Tucson

Ron Kern
Storage Systems Division - Tucson

Larry Law
Storage Systems Division - San Jose

William Nettles
Storage Systems Division - San Jose

Deborah Norberg
Storage Systems Division - San Jose

Tony Pearson
Storage Systems Division - Tucson

Jerry Pence
Storage Systems Division - Tucson

Savur Rao
Storage Systems Division - San Jose

Dave Thompson
Storage Systems Division - San Jose

Carmen Yep
Storage Systems Division - San Jose

Joy Nakamura
Storage Systems Division - San Jose

Ron Ward
Storage Systems Division - Tucson

Taylor Winship
Storage Systems Division - San Jose

Mike Wood
Sortware Services - IBM UK

Thanks are also due to the many other collaborators and reviewers who contributed to this book. I especially want to acknowledge our technical editor, Maggie Cutler, whose work substantially improved the quality of this book.

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in ″ITSO Redbook Evaluation″ on page 159 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Home Pages at the following URLs:

For Internet users	http://www.redbooks.ibm.com
For IBM Intranet users	http://w3.itso.ibm.com/redbooks

- Send us a note at the following address:

  redbook@vnet.ibm.com

Chapter 1. Introduction to DFSMS/MVS Version 1 Release 4

DFSMS/MVS provides and enhances functions formerly provided by MVS/DFP, Data Facility Data Set Services (DFDSS), and the Data Facility Hierarchical Storage Manager (DFHSM). DFSMS/MVS is easier to install and order than those earlier offerings, as it eliminates the need to install multiple products. DFSMS/MVS improves on earlier offerings by adding support for new functions.

DFSMS/MVS is a licensed program that delivers four MVS storage management functional components as a single, integrated software package:

- DFSMSdfp - Provides storage, data, program, and device management functions
- DFSMSdss - Provides data movement, copy, backup, and space management functions
- DFSMShsm - Provides backup, recovery, migration, and space management functions
- DFSMSrmm - Provides management functions for removable media such as tape cartridges and 3420 reels

DFSMS/MVS V1R4 expands on the DFSMS/MVS product by providing new enhancements and function. The subsequent chapters in this book describe the functions of and enhancements to DFSMS/MVS provided in this release, with each chapter being devoted to a functional area. Thus those readers with an interest in DFSMShsm or in VSAM can go directly to a particular chapter to see what has been included or enhanced for that specific functional area. This book is intended for those who are familiar with DFSMS/MVS and its components.

1.1 DFSMS/MVS Release Focus

DFSMS/MVS provides storage management, data access, device support, program management, and distributed data access for the MVS/ESA platform. DFSMS/MVS V1R4 improves system and data availability, performance, and productivity and fulfills important customer requirements.

What has become apparent since DFSMS/MVS was first delivered is that many of the most significant functional enhancements can be realized only if the data is system-managed storage (SMS)-managed. That is true of this release of DFSMS/MVS as well, as you will discover as you read this book.

Follow-on releases of DFSMS/MVS are likely to continue this trend of delivering new function and enhancements for SMS-managed data.

Most of the significant benefits of this release of DFSMS/MVS will be available only to customers who have invested some time and effort in implementing SMS. For those customers who have not yet made this step, the DFSMSdfp NaviQuest component functionality should make it easier than ever before to migrate to SMS, particularly if the DFSMS Fast Implementation Techniques methodology is used.

1.1.1 Functional Enhancements

DFSMS/MVS V1R4 includes the following new functions and enhancements:

- DFSMSdfp
  - Space allocation failure reduction

    This enhancement reduces out-of-space conditions for SMS-managed data sets for new volume processing only. DADSM may use more than five extents to allocate the requested space quantity. Two new DATA CLASS definition parameters can be used to influence the allocation and extension of data sets (to new volumes) in such a manner that allocations that might have failed for lack of space may succeed.
  - Sequential Access Method tailored compression

    SAM tailored compression is an enhancement to the DFSMS compression feature that was introduced in DFSMS/MVS 1.2.0. It creates dictionaries tailored for the data set that are embedded in the data set. The primary purpose of tailored compression is to significantly improve the compression ratio for SAM (physical sequential) data set organizations. This technique improves the compression ratios over the generic dictionary building block (DBB) compression techniques.
  - Open/close/end of volume (O/C/EOV) serviceability improvements

    Many serviceability improvements have been added in this release of DFSMS/MVS.
  - Enhanced protection of checkpointed data sets

    Changes have been made to DFSMSdfp so that an MVS physical sequential (PS) checkpointed data set can be distinguished from an IMS GSAM checkpointed data set. DFSMSdss operations that can make PS checkpointed data sets unusable for restart are performed only when specifically requested.
  - Object Access Method (OAM) SMF recording

    A new SMF record type 85 is provided for OAM object access. The SMF record type 85 supports subtypes, and there is a subtype for almost every activity performed by OAM.
  - Program Management 3 (PM3)

    This release of PM will include the following enhancements:

    - A conversion function added to the Binder to process extended object (XOBJ) structures built by the C/C++ compilers
    - The Binder and Loader will be enhanced to support OS/390 dynamic linking and dynamic load libraries (DLLs)
    - Support for the two above enhancements in OpenEdition and the C89 shell command
    - Loader enhancements to support DLLs and deferred loading, as well as the OS/390 R4 Dynamic LPA functions.
- Virtual Storage Access Method (VSAM)
  - System-managed buffering

    VSAM system-managed buffering allows VSAM to optimize batch programs by determining the number of data and index buffers as well

as the type of buffer management—sequential or direct optimized, or sequential or direct weighted.

- VSAM load enhancements

  When system-managed buffering is allowed, the performance of loading an extended format key-sequenced data set (KSDS) can be improved by reducing the number of I/Os required to write the data.

- Update last reference date at close

  The last reference date will now be updated in the VTOC on the first volume for the base data component when the data set is closed if the date at close is different from that at open. Updating the last reference date at close ensures that data sets are not automatically migrated based on the last reference date at open in the DFSMShsm space management cycle after a subsystem is brought down after many days.

- Data class constructs

  All VSAM data set attributes are available from the data class using Interactive Storage Management Facility (ISMF). The enhancements to data class for VSAM support the BWO, LOG, LOGSTREAMID, SPANNED, and NONSPANNED attributes. Data sets defined with JCL can use these attributes if they are assigned a data class having these attributes.

- VSAM record level sharing (RLS) extended addressability

  This enhancement allows RLS access to VSAM KSDSs defined with extended addressability beyond 4GB.

- DCOLLECT enhancements

  DCOLLECT is enhanced to collect more information as well as to have more granularity to obtain information about specific volumes.

- DFSMShsm
  - Many enhancements to ABARS have been made in this release.
  - Access Method Services for ICF (IDCAMS) allows altering the storage and management class without recalling a migrated data set.
  - DFSMShsm provides an option for its control data sets (CDSs) to be accessed in RLS mode. The ability to use VSAM RLS to access the DFSMShsm CDSs will eliminate data set contention against the CDSs when three or more hosts are processing in a shared environment.
  - DFSMShsm uses dump analysis elimination (DAE) to suppress duplicate dumps.
  - DFSMShsm provides an alternative to TAPECOPY processing for backup and migration cartridge tapes. This option allows DFSMShsm to create two tapes concurrently. It is intended that the original tape be kept onsite and the alternate taken offsite or written to a remote tape library.
  - DFSMSrmm no longer uses the DFSMShsm installationwide exit, ARCTVEXT. It provides a new general-use programming interface, EDGTVEXT, for releasing tape volumes. Thus ARCTVEXT is freed from DFSMSrmm usage, allowing it to be used by other tape management systems.
  - Many serviceability items have been added to this release of DFSMShsm.

- DFSMSdss
  - DFSMSdss Multivolume Selection and Filtering

    A new data set selection keyword, SELECTMULTI, acts in conjunction with the user-supplied input volume list to control the selection of data sets to be copied, dumped, or converted to or from SMS management.
- DFSMSrmm
  - New parameters have been added to the EDGRMMxx member of PARMLIB that allow an installation to specify a percentage-full threshold and the name of an automatic backup procedure.
  - DFSMSrmm will no longer provide its own version of the ARCTVEXT exit.
  - The Problem Determination Aid (PDA) trace is added to DFSMSrmm so that diagnostic information can be gathered at entry and exit points in the DFSMSrmm modules.
  - DFSMSrmm now allows an installation to use concurrent copy to back up the control data set and journal.
  - Improved support is provided for DFSMShsm alternate tape processing.
  - DFSMSrmm can now be instructed to manage a tape volume but not to track all of the data sets that are written to that volume. With this enhancement DFSMSrmm and other tape management systems will no longer maintain duplicate information.
  - DFSMSrmm provides a trial-run capability for inventory management. Thus you can check what DFSMSrmm will do during VRSEL processing without making changes to the control data set or journal.
- Open and distributed enhancements
  - Network File System (NFS) client support

    A new NFS client is provided by the DFSMS/MVS V1R3 NFS Feature for the OS/390 and MVS platforms. Thus one OS/390 (or MVS) system can exchange data files with another, using the NFS client/server technology. The DFSMS/MVS NFS Client provides the full capability for OpenEdition applications to read or write MVS conventional data sets and OpenEdition Hierarchical File System (HFS) files on another OS/390 system.
  - Distributed File Manager (DFM) DataAgent

    The DFM DataAgent is an extension to the DFM and the VSAM/x component of SmartData Utilities (SdU) on Windows, OS/2, and RS/6000. The DFM DataAgent allows users of SdU to issue TSO commands and access otherwise unsupported MVS data sets or databases. The DFM DataAgent provides flexibility in allowing client applications access to the data itself rather than limiting access to predefined remote transactions.

1.1.2 Usability Enhancements in DFSMS/MVS V1R4

- NaviQuest

  The DFSMSdfp NaviQuest component enables you to perform SMS migration functions and many routine storage management tasks, including testing and reporting, in batch.
- Catalog Search Interface (CSI)

CSI provides an application programming interface (API) from assembler and high-level languages to read information from Integrated Catalog Facility (ICF) catalogs. It is an alternative to LISTCAT that allows tailoring of the output and includes some information that LISTCAT does not provide.

- DFSMS Optimizer

  The DFSMS Optimizer product has been totally refreshed. Even though it is not a DFSMS/MVS V1R4 line item, an overview of the product and its function is included in this book. See Chapter 8, “DFSMS/MVS Optimizer” on page 115.

Chapter 2. DFSMSdfp

The DFSMSdfp component in DFSMS/MVS V1R4 provides usability and performance enhancements, greater reporting capabilities, and enhanced protection of checkpointed data sets and satisfies many customer requirements. In this chapter we cover the following DFSMSdfp enhancements:

- Space allocation failure reduction
- SAM tailored compression
- O/C/EOV serviceability enhancements
- Enhanced protection of checkpointed data sets
- Optical Access Method SMF recording
- Program Management 3 (PM3)

2.1 Space Allocation Failure Reduction

Sometimes a data set allocation or extension fails because there is not enough space on a volume to satisfy the space allocation request using current algorithms. DFSMS/MVS solves this problem to some extent by performing volume selection and trying all candidate volumes before failing an allocation .

The space allocation failure reduction further reduces the incidence of these allocation and extension failures. It applies to SMS-managed data sets only and is limited to new data set allocations and extends on new volumes. The space allocation failure reduction enhancement does not provide relief for data set extensions on the current volume.

The following enhancements may make having separate storage groups for large, medium, and small data sets less of a requirement:

- The five-extent limit in DADSM for new data set allocation and for extends to new volumes has been removed.
- Data sets can grow to either 59 volumes (which is the limit imposed for multivolume data sets by the task input/output table (TIOT)) or the number of volumes in the storage group (as long as a volume in the storage group has space), whichever is smaller.
- Maximum number of extents per component of VSAM data sets is now 255 instead of 123 for multivolume data sets.
- Data sets that currently have a 16-extent or 123-extent per volume limit will continue to have the same limit.

Solving the allocation and extent problem results in the added benefit of not having to run DEFRAG as often, especially in a sysplex environment. DEFRAG can move extents of a data set only when it is not in use anywhere in the sysplex and requires GRS ring/star or equivalent product support to ensure data integrity.

2.1.1 Allocations Affected

The allocations affected by the space allocation failure reduction are:

- Allocations initiated by *BATCH* (JCL) or dynamic allocation (SVC 99)
- Allocations initiated by *IDCAMS DEFINE* (VSAM only) and *IDCAMS* or *TSO ALLOCATE* commands
- Allocations initiated by DFSMSdss or DFSMShsm.

  DFSMSdss and DFSMShsm only use the enhancement that removes the five-extent limit. The space allocated will not be reduced.
- Data set extensions to new volumes

  Note that the *abend x37 reduction* does not address data set extensions on the current volume.
- VSAM allocations

  VSAM allocations are affected by an increase in the maximum number of extents per VSAM component from 123 to 255. The maximum number of extents per data set per volume remains at 123.

Note: DFSMSdss or DFSMShsm initiates allocations to allocate space for data sets that must be restored, and the allocations cannot tolerate a reduction in the requested space amount. Therefore, the %X value in the data class is ignored for these types of allocations.

2.1.2 Implementation Details

As data set allocations proceed, one of the following may occur:

- The allocation is successful.
- The allocation fails because enough space was not available.
- The allocation fails for a reason other than space.

Of the two failing cases (case 2 and case 3), space outage reduction addresses case 2. It is based on retrying allocation failures by a combination of:

- Spreading the requested space quantity over multiple volumes
- Reducing the requested space quantity by the allowed limit
- Using more than five extents to satisfy the allocation

If a volume selection failure occurs for a reason other than space, the *space outage reduction* solution will not apply.

Note: *Spreading* data over multiple volumes means that SMS allocates the primary space allocation over as few volumes as possible under the constraints imposed by the maximum number of extents that may be allocated on each volume. (For VSAM data sets, the constraints are based on the number of extents allowed for the entire component over all volumes.)

2.1.3 Volume Selection Failure Recovery

Users may, through use of newly defined fields in the *data class*, indicate to SMS that they wish to attempt a retry using one or more of the above methods. These new fields are:

- **SPACE CONSTRAINT RELIEF: Y|N (default is N)**

  If a user specifies *Y* for space constraint relief:

- On initial allocations, more than one volume will be used to allocate the primary quantity if the data set is multivolume eligible.
- SMS may reduce the amount of space to be allocated on the basis of the *REDUCE SPACE UP TO* parameter.
- SMS may remove the five-extent limit.

- **REDUCE SPACE UP TO (%): X(0 to 99, 0 is the default)**

In the paragraphs that follow, we illustrate how SMS processes these new fields.

2.1.4 New Allocations (VSAM and non-VSAM)

If a user specifies *N* for SPACE CONSTRAINT RELIEF, either explicitly or implicitly, SMS will not attempt a retry. Allocation will not be attempted by spreading the data on multiple volumes.

Retry is to allocate the reduced space quantity in as many extents as allowed by the removal of the DADSM five-extent limit.

If a user specifies *Y* for SPACE CONSTRAINT RELIEF and the volume count for the allocation is 1, SMS will redrive the allocation after reducing the requested space quantity (which could be *PRIMARY* or *SECONDARY* space quantity) on the basis of the REDUCE SPACE UP TO parameter. If the reduced space quantity (the requested space multiplied by the *REDUCE SPACE UP TO* parameter) is less than 1, it will be rounded up to 1. Simultaneously SMS will also allow more than five extents to allocate this recomputed space amount.

Note: It is valid for a user to specify 0 space quantity for primary space (when allocating a model data set control block). The space quantity for non-VSAM data sets is the PRIMARY space for new allocations and the SECONDARY space for extensions. The space quantity for EF and non-EF VSAM data sets is the PRIMARY space for new allocations and either the PRIMARY or the SECONDARY space for extensions on new volumes (depending on a parameter in the data class). Such allocations should not fail for space except in the rare circumstance where there is no space in the VTOC for allocating a DSCB. In any event, allocations where the primary space quantity is 0 will not be retried.

If the user specifies a Y and the volume count for the allocation is more than 1, on retry SMS will attempt to allocate the requested space using more than one but as few volumes as possible.

For allocation, the maximum number of volumes is equal to the largest of the unit count, volume count, and volser count. If none of these three values is specified, the volume count in the *data class* definition is used. This volume selection failure recovery method is called *best-fit allocation*. It is applicable only during the initial allocation of a data set. It is not applicable during extensions to new volumes.

Note: For non-VSAM data sets, if the secondary space quantity is zero, the data set is not extended on the current or new volume.

The amount of space allocated during extent processing of VSAM data sets differs from that of non-VSAM data sets. If a VSAM data set is extended on the current volume, the SECONDARY space specified by the user is allocated. If the VSAM data set is extended to a new volume, the PRIMARY or SECONDARY space specified is allocated as follows: the default is PRIMARY space; the

SECONDARY quantity is used if the data class allows it *and* the data set is allocated in extended format.

If the best-fit allocation fails, SMS will retry the allocation as follows:

- If *REDUCE SPACE UP TO (X%)* is specified and *X* is between 0 and 99, SMS will reduce the requested space quantity by X% and redrive the best-fit allocation. A specification of 0 implies that the user wants to use more than five extents to satisfy the allocation without reducing the allocation amount.
- SMS will use as many extents as are allowed.

2.1.5 Extents on New Volumes

Extensions will be tried only if candidate volumes exist for the data set as indicated in the TIOT.

2.1.6 Non-VSAM Data Sets

If the initial attempt to extend the data set fails and the data class indicates that *SPACE CONSTRAINT RELIEF* is applied, SMS will retry the extents as follows:

1. If the user has specified (with the *REDUCE SPACE UP TO*) parameter that a smaller amount of space is acceptable, SMS will reduce the amount of space and will indicate to DADSM that the five-extent limit is to be removed.
2. If the user has indicated that space is not to be reduced, by either specifying 0 for *REDUCE SPACE UP TO* or defaulting to 0, SMS will indicate to DADSM that only the five-extent limit is to be removed.

2.1.7 VSAM Data Sets

If the initial attempt to extend the data set fails and the data class indicates that *SPACE CONSTRAINT RELIEF* is to be applied, SMS will retry the extent using the smaller of up to 123 extents per volume or 255 extents for the VSAM component.

2.1.8 Exceptions to Using Volume Selection Failure Recovery

The following types of data set allocations cannot use volume selection failure recovery:

- All allocations where the storage class specifies a nonzero sustained data rate (SDR), implying a multistriped data set, and the data class specifies *REQUIRED* for striping. With this exception, all VSAM extended format data sets (which are always *single* striped) *will* be candidates for space constraint relief. Furthermore, all non-VSAM extended format data sets that specify a *0* SDR (which results in *single* striping) *will* be candidates for space constraint relief. All non-VSAM extended format data sets with SDR other than *0* will not be candidates for space constraint relief.
- All multivolume guaranteed space allocations (VSAM and non-VSAM) will be retried with the five-extent limit removed, but the space quantity will not be reduced.
- VSAM key-range data sets will not be candidates for space constraint relief.
- If IMBED is specified (VSAM data sets only), SMS will not use the best-fit allocation method during the redrive because it is not supported in conjunction with IMBED. SMS will still attempt the redrive with greater than five extents and a smaller space quantity (if specified by the user).

2.1.9 ISMF Changes

The data class Application Selection panels will be enhanced for the following options:

- Data class Define/Alter
- Data class Display
- Data class List Display
- Data class List Sort
- Data class List View
- Data class List Print

The data class Define/Alter page 3 panel, DGTDCDC5, and the data class Display page 3 panel, DGTICDC3, have been modified to accommodate the new *SPACE CONSTRAINT RELIEF* and *REDUCE SPACE UP TO* fields. The fields have been added to the last page, at the end, after all existing fields.

Data class List Display panel DGTLGP21, List Sort page 2 panel DGTDCDC4, List View page 2 panel DGTDVW42, and List Print page 2 panel DGTDPR42 have been modified to accommodate the *SPACE CONSTRAINT RELIEF* and *REDUCE SPACE UP TO* new columns. They are to be added to the end or existing entries are to be moved down or across columns to accommodate the new entries in the appropriate place.

2.1.10 ISMF Message Changes

ISMF has been enhanced with the following new messages on the data class Define/Alter panels:

```
DGTDC060   INVALID VALUE
SPACE CONSTRAINT RELIEF must be Y or N


DGTDC061   INVALID VALUE
REDUCE SPACE UP TO must be 0 to 99 when SPACE CONSTRAINT RELIEF is Y


DGTDC062   FIELD RESET TO DEFAULT
REDUCE SPACE UP TO is reset to 0, since SPACE CONSTRAINT RELIEF is Y


DGTDC063   FIELD RESET TO DEFAULT
REDUCE SPACE UP TO is reset to Blank, since SPACE CONSTRAINT RELIEF is N
```

2.1.11 Migration and Coexistence

The following areas might be impacted by the space outage reduction changes:

- Storage administrators

  Storage administrators may not have to schedule DEFRAG as often, and they may not have to have separate storage groups based on data set size to avoid space failures.

  Storage administrators must determine whether or not to modify existing data classes and/or define new data classes with the new *SPACE CONSTRAINT RELIEF* and *REDUCE SPACE UP TO* attributes. Generally

speaking, it is beneficial to specify *Y* for *SPACE CONSTRAINT RELIEF* and a nonzero value for *REDUCE SPACE UP TO* unless the user application cannot tolerate a reduction in the amount of space requested. Data class changes apply to both new and existing data sets.

- Application programmer

  Application programmers must be aware of the new messages that they may encounter. If they expect the requested space to be satisfied within five extents and on a single volume, the changes introduced by space outage reduction could affect applications. If the new data class attribute parameters are not used, applications will not be affected.

PTFs will be shipped on supported releases of MVS/DFP and DFSMS/MVS to support 255 extents per VSAM component.

With these PTFs applied, lower-level systems can access all 255 extents of a VSAM component that was created on a DFSMS/MVS V1R4 system. It will not be possible, however, to extend VSAM components on lower-level systems if they occupy 123 or more extents.

If customers do not want to use the new parameters in the data class, they need take no action, and the system will continue to work as before. If customers want to use the new parameters, they have to define new data classes or modify existing definitions.

Customers specifying *SPACE CONSTRAINT RELIEF* in one or more data classes should note the following points:

- Very large allocations that would have failed may succeed if a sufficiently large volume count is specified in the data class or JCL.
- Existing data sets may end up with less space than requested on extents just by changing a data class and activating a configuration.
- The space allocated for new data sets may be less than requested, again depending on the use of new attributes in the data class.
- The number of extents used during initial allocation may result in fewer extensions to new volumes or current volumes.
- There should be fewer X37 abends.

2.2 SAM Tailored Compression

This enhancement significantly improves the compression ratios that can be achieved for sequential data sets accessed by BSAM or QSAM. Tailored compression introduces a new form of compression for sequential extended format data sets. With tailored compression, the system attempts to derive a compression dictionary that is tailored specifically to the initial data written to a data set. Once a tailored dictionary is derived, it is imbedded in the compressed data set. This technique is expected to provide improved compression ratios, thereby reducing DASD usage and channel traffic. Because the dictionary is tailored to the user data, significantly more data is sampled than was required by generic compression. The process of sampling the data and building the dictionary during creation of a new data set takes more CPU cycles and is, therefore, most noticeable when compressing small data sets. For larger data sets, the cost of sampling is amortized significantly. Reuse of a tailored compressed data set saves the cost of sampling and dictionary creation.

An installation has the option of either using the new tailored compression or continuing to use generic DBB-based compression first introduced with DFSMS/MVS V1R2.

Tailored compression allows for more types of data to be compressed, for example, where non-English languages are used. Because the original generic dictionary was developed with standard American-English scripts, files containing sequences of characters that do not appear in the American scripts do not compress very well. The tailored dictionary avoids this problem as it is generated dynamically from the data itself, for each data set. Dynamically generating a tailored dictionary from the data itself should make tailored compression more useful to the non-English-speaking world.

Tailored compression support does not apply to VSAM KSDSs, which can continue to be compressed with generic DBB dictionary compression (as illustrated in Figure 1).

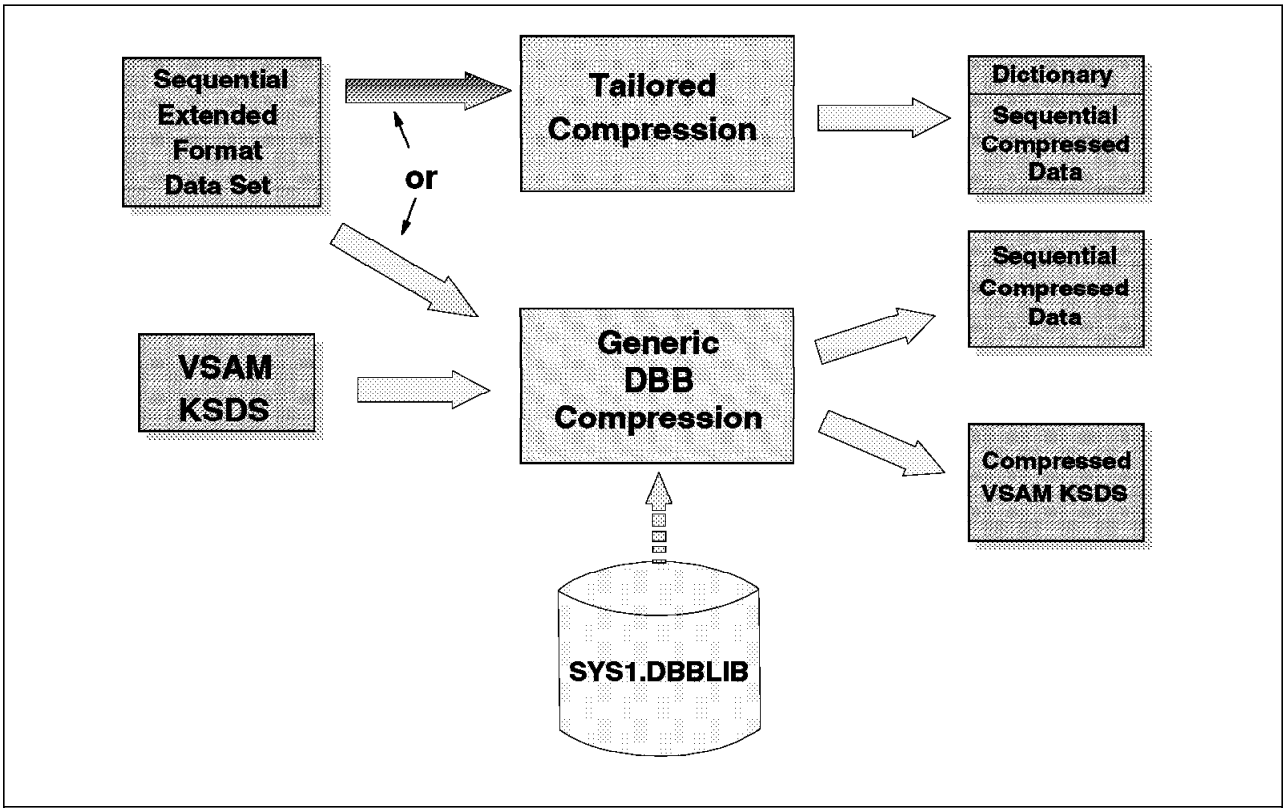


Figure 1. Tailored Compression. The IGDSMxx PARMLIB member specifies whether tailored compression or generic DBB compression will be used. VSAM KSDSs can only use generic compression.

2.2.1 Tailored Compression and Generic Compression

The existing generic compression algorithm was used for both VSAM KSDSs and SAM data sets. Standard DBBs continue to be shipped with the product in a system library called SYS1.DBBLIB and will be used whenever:

- A VSAM data set is compressed.
- The installation specifies `generic` as the default compression algorithm.
- A data set was previously compressed using the `GENERIC` option or was compressed on a pre-DFSMS/MVS V1R4 system.

Standard DBBs consist of a fixed set of padding run blocks, numeric blocks, and English text strings commonly found in the English language. As such, standard generic compression algorithms do not work well for non-English languages, or any text data that involves repetitive strings other than common English words; for example, any data set made up of information not found in the available DBBs, such as the repeating occurrence of a person′s name.

In contrast to generic, tailored dictionaries are built from data strings extracted from the data to be compressed, assembled into a compression dictionary, and stored in the first few tracks of a SAM data set; that is, tailored compression is completely insensitive to the text language. As with generic, tailored compression samples the first portion of data as it is written, choosing the optimal data strings to use. These data strings are then assembled into a compression dictionary and stored into the data set. This compression dictionary is read and activated whenever the data set is opened.

Tailored compression′s design requires it to sample significantly more data than was required by the generic algorithms (400K on average verses 64K). This additional sampling uses more CPU cycles and is therefore most noticeable when compressing small data sets. Because the cost of tailored compression sampling is significantly higher than generic sampling, generic is the default system option.

2.2.2 Sampling the Data

For larger data sets, the cost of sampling is amortized significantly. If a small data set is opened for output a second time, the sampling process is not repeated. During subsequent opens of a data set, the already created compression dictionary is read from disk and is reused. Reuse of the compression dictionary saves CPU time, but it also means that if the nature of the new data is very different, it is essential that the data set be scratched and reallocated in order to force the system to re-create the compression dictionary.

Evidence shows that for large data sets, tailored compression significantly improves the data compression ratio while it reduces the CPU time needed to compress a data set (when compared to generic), although this CPU time is still much higher than not using compression at all. Therefore, customers must still make a trade-off between CPU time costs, channel traffic, and DASD space savings, but tailored compression significantly reduces DASD space usage for small as well as large data sets, even when taking into consideration the cost of storing the compression dictionary. Also, because less DASD space is used, the elapsed time to read or write the data set is reduced.

When evaluating whether or not to use compression for all data sets, it is important to take into consideration factors other than the size of the data sets, such as how long the data set is expected to be kept. When a data set is reused (either read or rewritten), the CPU cost is much less. Therefore, it may be desirable to restructure applications so that sampling takes place less frequently.

Compression is never used for temporary data sets. The above discussion applies only to permanent data sets.

2.2.3 Enablement of Tailored Compression

A new parameter, COMPRESS(TAILORED|GENERIC) is available for the IGDSMSxx member of SYS1.PARMLIB. It enables you to specify the form of compression the system is to use for newly created physical sequential extended format data sets.

COMPRESS(GENERIC) refers to DFSMS/MVS V1R2 generic DBB-based compression. This is the default parameter to maintain consistency with previous DFSMS/MVS function. However, IBM generally recommends that customers change the option to tailored compression. When all the systems in the sysplex and in the disaster recovery site are not at the DFSMS/MVS V1R4 level, specify COMPRESS(GENERIC).

COMPRESS(TAILORED) refers to tailored compression where the system attempts to derive a dictionary tailored specifically to the initial data written to the data set.

Once the COMPRESS(TAILORED) parameter is activated (through the SETSMS=xx command or an IPL), the system uses tailored compression for newly created sequential compressed format data sets.

Customers can choose at the data set level through a DATACLAS parameter whether or not a data set should be compressed.

2.2.4 Wellness Logic

A new feature of DFSMS/MVS V1R4 compression is called *wellness logic* (see Figure 2). If DFSMS recognizes that the data set was not compressing to a desired level, it stops the compression process for a set period of calls. After this skip duration, data compression resumes, and the wellness logic is again activated. If a data set consistently compresses poorly, over the entire length of the data set DFSMS samples 20% of the time and 20% of the data is compressed.

For example, suppose that in DFSMS/MVS V1R3 a data set was compressed by only 8% and the CPU time cost was 1 sec. In DFSMS/MVS V1R4, the data set would be compressed by 1.6%, but the CPU time for sampling would be reduced to 0.2 sec (as illustrated in Figure 2).

```
      -----DS Size (MB)-------   Compression CPU seconds
           ---GENERIC--  TCOM    --GENERIC---   TCOM
Data  Non-  DFSNS  DFSNS          DFSMS  SFSMS
Set   Comp. 1.3.0  1.4.0          1.3.0  1.4.0  DISP=NEW
 6     9.4   8.6    9.2    3.3     3.8    0.8     3.1
```

Figure 2. Wellness Logic

Wellness logic applies to both generic and tailored compression. However, tailored compression almost always exceeds the wellness criteria, so wellness logic will very rarely come into play for tailored compression.

2.2.5 Measurements

All measurements were done using a 9672-RX73 processor and IEBGENER to copy an existing data set. The CPU cost of compression was derived by first measuring the CPU time when only the output data set is compressed and then subtracting the CPU time when both the input and output data sets are noncompressed. Likewise, the CPU cost of decompression was derived by first measuring the CPU time when only the input data set is compressed and then subtracting the CPU time when both the input and output data sets are noncompressed.

In the case of tailored compression, two variations were measured. The first time IEBGENER was run is called DISP=NEW, which includes the cost of sampling. The second time IEBGENER was run is called DISP=OLD, which avoids the cost of sampling. Generic compression was not measured with DISP=OLD, but it is expected that the cost of generic sampling is relatively small compared to the cost of tailored compression sampling.

In Figure 3, data sets 2 through 5, generic compression achieved a compression ratio around 1.8. In contrast, for these same data sets, tailored compression achieved a compression ratio between 2.7 and 4.2. For data set 1 where generic compression achieved only a 13% size reduction, tailored compression achieved a compression ratio of 2.7.

	---DS Size (MB)----			-------Compression-------			Decompression	
Data	Non-			-------CPU seconds-------			CPU seconds	
Set	Comp.	GENERIC	TCOM	GENERIC	------ TCOM -----		GENERIC	TCOM
					DISP=NEW	DISP=OLD		
1	420	367	155	177.0	148.6	143.1	14.6	12.8
2	28.5	15.5	6.7	10.6	16.1	9.6	0.9	0.6
3	16.1	8.7	4.9	6.4	13.3	5.7	0.5	0.6
4	13.5	7.3	4.9	5.2	12.7	4.8	0.5	0.4
5	6.4	3.8	1.9	3.1	7.2	2.2	0.4	0.2

Figure 3. Generic and Tailored Compression Measurements

Note that although most of these data sets might be considered small or medium size data sets, the same compression ratios would occur if the data were replicated to create larger data sets.

For tailored compression with DISP=OLD, the normalized CPU time per kilobyte for each of these data sets is in a narrow range between 330 and 355 sec/KB. (This range is sensitive to the CPU model.) Although generic DISP=OLD measurements were not done, it appears likely that such a metric would land in the same range as with tailored compression, except for data set 1, which generic compression does not compress well. In those cases where data does not compress well, the CPU time per KB is higher.

Data set 6 in Figure 2 on page 15 demonstrates the wellness logic. Generic compression achieved less than 10% size reduction. In DFSMS/MVS V1R3 where wellness logic did not exist, DFSMS still tried to compress the data set and the CPU time cost was very high. In DFSMS/MVS V1R4, the wellness logic with generic compression reduced the CPU cost significantly, with only a slight change in the data set size. On the other hand, tailored compression compressed the data set significantly, with a corresponding cost in CPU time.

2.2.6 Functional Characteristics

All sequential data sets that are eligible are compressed with the compression type specified in the IGDSMxx member. This is true even if the data set is allocated with LIKE and modeled after a data set with a different compression type.

If it is determined that the data set is eligible for tailored compression, the new function:

- Builds a tailored dictionary using the initial data written to the data set
- Validates the useability of the resulting dictionary depending on whether a preset compression criterion is met.
- Compresses the data set (including the initial sampled data) or rejects the compression for the data set if a preset compression criterion is not met.

2.2.6.1 Tailored Compression Identification

Note: The following information is not an intended programming interface. It is provided for diagnostic purposes only.

A dictionary token consists of 32 bytes and is stored in the catalog. Today, the dictionary token may be found in output displayed by functions such as LISTCAT and DCOLLECT, and in certain SMF records. The dictionary token is modified to contain tailored compressed data set indicators. The first byte of the dictionary token indicates the type of compression used for the data set (see Figure 4 on page 18):

- x'100. ....' indicates compression has been rejected for the data set, no data is compressed
- x'010. ....' indicates generic DBB compression is used
- x'011. ....' indicates tailored compression is used

The second four bits in the first byte of the token contain bits to denote the number of blocks containing system data from the beginning of the data set until the user data starts:

- x'.... .xxx' indicates the number of blocks containing system data

The start of user data will begin in the physical block following the blocks containing system data. A compressed data set with a tailored token or a rejection token may contain a number of these system data blocks.

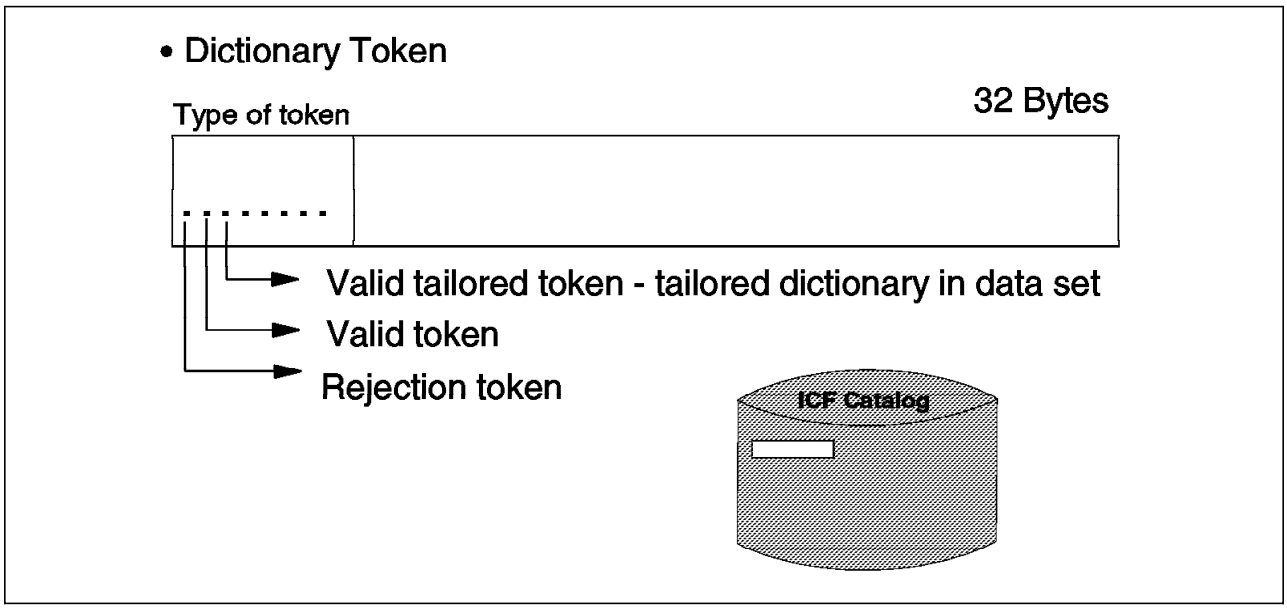


Figure 4. Dictionary Token. Tailored compression is indicated from the first four bits of the first byte.

For tailored compressed data sets, the number of system blocks is always nonzero, as this is where the tailored dictionary is stored. When a compressed format data set contains a rejection token, the number of system blocks may still be nonzero.

When a compressed data set has system blocks, these blocks are reflected in the DS1LSTAR field in the data set's Format 1 DSCB. On reusing an existing data set, it is possible for a compressed data set to contain no user data but still have a nonzero DS1LSTAR.

There may be a significant increase in the amount of data DFSMS compression buffers in storage for new data sets while it derives the dictionary. This increase allows DFSMS compression to optimize dictionary building and compress sampled data. This compares to using the generic DBB-based dictionary where up to the first 64KB of user data may be stored in uncompressed form.

2.2.6.2 Storage Requirements

Because new internal compression control blocks are being used, together with the increased buffering of data during the dictionary build phase, a minimum region size of 3MB is required. This storage will be getmained from above 16MB. Once a dictionary is derived, the ongoing storage requirements are reduced to less than 200KB.

The REGION size parameters of the JOB and EXEC JCL statements do not have to be modified to use tailored compression, even though 3MB more storage above 16MB is used.

2.2.7 Migration and Coexistence Considerations

A data set compressed with a tailored dictionary cannot be processed on any previous to DFSMS/MVS V1R4 system. Compatibility PTFs are required on pre-DFSMS/MVS V1R4 systems to fail attempts to access these data sets on those systems. On a system with tailored compression active, it will not be possible to convert, in place, existing data sets with or without generic

dictionaries to a tailored dictionary data set. You must copy the data into another data set, using a utility like IEBGENER or an IDCAMS REPRO.

Any attempt to access a data set using a tailored dictionary on a pre-DFSMS/MVS V1R4 system through DFSMSdss or DFSMShsm will fail. Error message ADR910E will be issued to describe the reason for the failure.

Tailored compression does not apply to VSAM data sets at this time. Even though the system option might specify tailored compression, VSAM will use only standard generic dictionaries.

2.3 O/C/EOV Serviceability Enhancements

Customers, service, and development personnel have documented a number of O/C/EOV serviceability suggestions. DFSMSdfp V1R4 will be enhanced to satisfy several of these serviceability suggestions, such as more information when an x37 abend occurs by providing a new SMF subtype record. A trace will be also available as well as a new message in cases of O/C/EOV problems.

O/C/EOV provides serviceability enhancements to help customers diagnose the problems by applying information in the SMF record 42 and trace and format the O/C/EOV work area.

The following DFSMSdfp enhancements are introduced in DFSMS/MVS V1R4:

- The O/C/EOV problem determination optional work area trace provides module flow and trace information for DFSMSrmm tape exit modules as is now provided for all other O/C/EOV modules. Thus IBM as well as customer services personnel can determine whether or not the DFSMSrmm tape exit modules were executed.
- A LOGREC is written for Automated Tape Library (ATL) devices whenever an abend 613 (an error occurred during processing of an OPEN macro instruction for a data set on magnetic tape) or abend 637 (an error occurred at an end-of-volume for a data set on magnetic tape or an end-of-volume during concatenation) is issued. This record will provide useful first-time data capture information that could help resolve the cause of the abnormal task termination. The information could include the unit control block (UCB), UCB extension, LACS parameter list, and other pertinent ATL data (such as the token and external volser). The message text for IEC214I and IEC026I will be updated to document the LOGREC identifier.
- A new started task enables the O/C/EOV problem determination optional work area trace to be run easily for dynamically allocated data sets. This task is started by the operator and then requests information such as the data set name, job name, and the DD statement name of the data set to be traced. The task can also be used for batch jobs, thereby eliminating the need to add DCB=DIAGNS=TRACE to the DD statement of the data set to be traced. The maximum number of trace entries is 10. Other functions provided are the ability to display the active entries and to delete either one or all of them. O/C/EOV checks the trace table to determine whether the data set currently being processed should be traced just as if DCB=DIAGNS=TRACE were specified on the DD statement. New messages, IEC980A and IEC980I, are used to interface with and provide information to the operator.

- The O/C/EOV abend message reflects the name of the module that detected the error instead of the name of the module that issued the abend. For example, during a B37, D37, or E37 abend, module IFG0554T is documented in the message text, even though module IFG0554P actually requested the abend. The message text now indicates IFG0554P.
- Currently if a user tries to open a PDS for output by specifying DISP=SHR, and the PDS is already open in this condition, the user has failed to serialize access before attempting to open the data set. The user has no information about who owns the PDS for output. The data set might be in the same system or another system that is sharing the volume. O/C/EOV is enhanced to report helpful information to find out who has the PDS for output. If an abend 213 reason code 30 is issued, a new message, IEC813I, is issued documenting which address space, JOB, and task control block (TCB) owns the PDS resource that is preventing the open from being allowed. The information provided by the IEC813I message provides first-time data capture information.
- A new SMF type 42 subtype 9 record is written for B37, D37, and E37 abends, documenting such information as job name, data set name, volser, number of extents on the volume, and secondary allocation amount that system programmers can use to prevent x37 abends in the future.
- An above 16MB O/C/EOV work area extension has been implemented. This extension does not provide new function; it has been implemented because of storage constraints in the main O/C/EOV work area.

2.3.1 New Trace Processing

Trace setup processing now runs as a started task. It is started, ended, and modified by operators.

The new started task module, IFG0OCET, is installed in SYS1.LINKLIB. The started task procedure, IFGOCETR, is installed in SYS1.PROCLIB.

2.3.2 Enabling IFGOCETR

The steps listed below must be followed to activate the O/C/EOV problem determination work area trace started task for dynamically allocated data sets:

1. The application or system programmer must supply the operator with either the data set name, job name, or DD name (or any combination thereof) of the data sets to be traced. As much information as required to uniquely identify the data set to be traced should be supplied. For instance, if only a job name is specified, then all data sets in the jobs matching that job name will be traced. This is probably not what the application or system programmer intended.
2. The operator starts the task by entering:
   - **START IFGOCETR**
3. IFGOCET issues WTOR message IEC980A, requesting the data set name, job name and/or DD name for the data set to be traced. There can be up to 10 entries in the trace table. O/C/EOV checks the active trace table entries to determine whether the data set being processed should be traced. After a trace entry has been added to the table, the active trace entries are automatically displayed. When an entry is added to the table, all data set parameters (data set name, DD name, and/or job name) must be specified on one response to message IEC980A. In other words, if the data set name

is specified in one response and the DD name is specified in the next response, two separate table entries will result. The data set name and DD name must be specified in one response to create only one table entry. The DISPLAY and DELETE parameters can be specified at any time but not in the same response as the data set name, DD name, or job name. When all responses have been issued, the operator must respond “END” to the messages to automatically end the IFGOCETR task. IFGOCETR does not have to be executing in order for data sets to be traced. Once the table has been updated, IFGOCETR can be terminated.

4. The operator starts the generalized trace facility (GTF) as usual (specifying USR=(FFF)). No new GTF parameters are required to use this new function and, once the trace data has been obtained, the GTF started task must be stopped as usual.
5. The operator can display trace entries and delete either one or all entries from the trace table.
6. Message IEC980I is issued for any error or informational conditions.

2.3.3 IFGOCETR Parameters

The IFGOCETR parameters are supplied by the operator in responding to message IEC980A. The parameters are:

- **DSN**=dsn (dsn is the data set name to be traced)
- **DDN**=ddname (ddname is the DD name of the data set to be traced)
- **JN**=jobname (jobname is the job name to be traced)
- **DISPLAY** (displays all active trace entries)
- **DELETE,<ALL|entno>**
  - *All* deletes all active entries
  - Active trace entry *entno* is disabled
- **END** (All trace parameters have been entered. This causes the started task to terminate.)

Note: At a minimum, either DSN, DDN, or JN must be specified, but any combination of those parameters can be specified. In fact, we recommend specifying a combination to limit the number of data sets that will be traced. For example, if only JN is specified, all data sets processed by the job will be traced. (This is probably not what was intended; however, it will work.) So the combination of DSN, DDN, and JN should be used to limit the tracing to one data set.

Parameters DISPLAY and DELETE are mutually exclusive from DSN, DDN, and JN. If DISPLAY or DELETE is specified with DSN, DDN, or JN, it will be ignored. DSN, DDN, and JN can be specified only once in each operator response. If multiple trace entries are to be created, each must be created separately.

If the data set to be traced is a generation data group (GDG), DSN can be either the complete generation data set (GDS) name (including G000v00) or the GDG base name. If the complete GDG name is specified, only that generation data set will be traced. If the GDG base name is specified, all of the generations are eligible for tracing, depending on the other parameters specified.

The started task does not have to be active at the time of the data set tracing. The started task is used only so the operator can enter parameters and cause the trace table to be built or updated. The trace table, once obtained, is not specifically freed. It will be allocated and available for use during the entire duration of IPL. Only by re-IPLing is the table freed.

If the trace table becomes full, trace table entries must be deleted before a new entry can be added.

2.3.4 Migration and Coexistence

This O/C/EOV serviceability enhancements introduce no specific migration considerations for customers. However, OEM tape vendors may be interested in the new work area extension. They should be encouraged to migrate away from hooks in O/C/EOV modules to using the tape installation exits that are now provided. A good reference for details on tape exits is *DFSMS/MVS Installation Exits* (SC26-4908).

2.4 Enhanced Protection of Checkpointed Sequential DASD Data Sets

If a customer uses both MVS and IMS GSAM checkpointed data sets, it is not possible to distinguish them. That is, DFSMSdfp can recognize whether or not there is a checkpointed data set, but it cannot tell whether it is an MVS checkpointed (MVSC) or an IMS GSAM (IMSC) checkpointed data set. It is important to be able to distinguish these data sets because of the way by which MVS and IMS access them. MVS accesses checkpointed data sets by absolute addressing (cylinder, head, record), so the checkpointed data set must not be movable. If the checkpointed data set is moved, restarts will fail. IMS accesses checkpointed data sets by offset addressing (from the beginning of the data set on the volume). Therefore, the data set is movable if the device characteristics and the amount of data per volume does not change.

In MVS/DFP V3R2 an indicator was provided in the format 1 DSCB called *DS1CPOIT*, which would be set in the MVS checkpoint supervisor call instruction (SVC) for physical sequential data sets and in DFP Open for IMS GSAM data sets. Notice that as far as DFP is concerned, GSAM data sets are physical sequential data sets. Customers were expected to use the DFSMShsm automatic migration exit to disallow migration until the data set was no longer required for a restart. They were expected to use the creation date, last reference date, and DS1CPOIT indicators in the format 1 DSCB passed to the exit to arrive at this decision. Command migration does not call this exit, but that should not be a problem as the user is taking an overt action to migrate the data set himself. This assumes that you know that the data set was accessed during a checkpoint.

A single-bit DS1CPOIT was used to indicate both IMSC and physical sequential MVSC data sets, and the primary intent was to stop migration or any kind of DFSMSdss processing such as DEFRAG. Remember that we are not talking about checkpoint data sets but *checkpointed* data sets. For MVSC, this means physical sequential data sets that were open during a checkpoint SVC. For IMSC, this means physical sequential (GSAM) data sets that, as indicated at open time, would be used during an IMS checkpoint. MVSC and IMSC data sets have different properties. GSAM data sets can be DEFRAGed, migrated and recalled, dumped and restored, and copied as long as the amount of data per

volume and the device geometry (track size) remain the same before and after the operations and the data set is not reblocked.

The handling of checkpointed data sets in this way results in several problems:

- DEFRAG was changed to recognize DS1CPOIT and not move the data set to support the more restrictive MVSC. This resulted in problems for those installations that were not using MVSC but used IMSC extensively and depended on DEFRAGing their tape mount management (TMM) DASD volumes to prevent allocation failures.
- Other DFSMSdss operations such as logical dump, restore, and copy ignore DS1CPOIT altogether and move the data sets, which can result in a failure on restarts.
- DFSMSdss physical operations also ignore DS1CPOIT, but they do not change the amount of data per volume, device geometry (track size), and do not reblock the data set.
- There was no way to satisfy all customers because of the use of a single bit for these different types of data sets.

The DFSMS/MVS enhancement enables differentiation between IMSC and MVSC data sets. It also addresses DEFRAG and all DFSMSdss logical data set operations.

MVSC will continue to set the DS1CPOIT (checkpointed) indicator and now sets the DS1DSGU (unmovable) indicator in the format 1 DSCB of the first and current volume being processed for PS data sets. Open will continue to set DS1CPOIT for ISMC data sets on the first volume of a multivolume data set. The bit may not be (and need not be) propagated on secondary volumes for GSAM data sets.

DS1DSGU is new for SMS-managed data sets. It is called the *SMS unmovable* indicator. It is treated differently from the DS1DSGU indicator on non-SMS-managed volumes. In SMS, DS1DSGU indicates that the data set is unmovable until it is no longer required to be used on a restart. The criterion of when a data set can be moved is based on when it was accessed during a checkpoint. Because this time of access is not retained in the format 1 DSCB, it has to be based on the number of days since last access (last reference date). New unmovable data sets (PSUs) cannot be created on SMS volumes. The checks in SMS will continue to be enforced to prevent creation of unmovable data sets.

The enhanced protection of checkpointed data sets is affected by three DFSMS components: DFSMSdfp, DFSMSdss, and DFSMShsm.

2.4.1.1 DFSMSdfp Enhancements

Currently checkpoint sets DS1CPOIT in the format 1 DSCB of the first volume of an SMS-managed physical sequential data set that is open. Checkpoint now additionally sets DS1DSGU in the format 1 DSCB for these data sets (MVSC data sets). Checkpoint also updates the date last referenced (DS1REFD) if the current volume is also the first volume.

O/C/EOV: If open finds DS1DSGU on in a format 1 DSCB for an SMS-managed data set, it sets DS1DSGU off and issues abend 213-58. Open only issues the abend if *unmovable* is specified in either the DCB or JFCB (JCL). Open does not abnormally terminate if DS1DSGU is on in the DSCB for an SMS-managed data set.

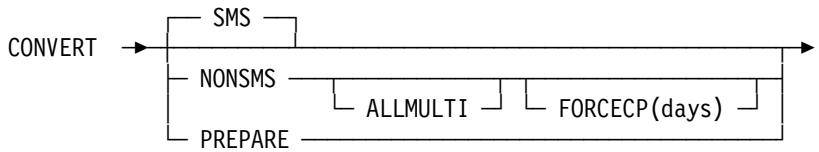
Finding DS1DSGU on the format 1 DSCB of a physical sequential SMS data set is not treated as an error.

2.4.1.2 DFSMSdss Enhancements

When DFSMSdss copies or logically restores a data set, even to a device with the same track geometry, the extent size and volume layout of the original data set may be lost. The loss may occur because DFSMSdss attempts to allocate the copied or restored data set on as few volumes as possible. The IMS Checkpoint/Restart facilities depend on relative displacement from the beginning of the first extent on a volume to perform restart on a checkpointed data set. The MVS Checkpoint/Restart facilities depend on, among other restrictions (like same volser and extent location), the absolute track location of restart point to perform a restart on a checkpointed data set. Neither restart facilities can tolerate a change in the device type or reblocking of the data.

CONVERTV: DFSMSdss allows a data set that has DS1CPOIT set, with or without DS1DSGU set, to be converted to NONSMS only if the FORCECP(*days*) keyword is specified. If a checkpointed data set is encountered during conversion to non-SMS and either FORCECP(*days*) is not specified or FORCECP(*days*) is specified and the minimum *days* have not elapsed, message ADR878E RC102 is issued. If conversion is allowed, DS1CPOIT and DS1DSGU as well as DS1SMSDS are reset for this data set.

The CONVERTV FORCECP syntax is as follows:



The FORCECP(*days*) parameter specifies that the checkpointed data set resident on the SMS-managed volume can be converted to non-SMS-managed volumes. All checkpoint indications are removed from the data set during conversion.

- *days* is a 1- to 3-digit number in the range 0 to 255. It indicates the number of days that must have elapsed since the last date referenced before conversion can take place.

FULL and TRACKS COPY/DUMP: Both full-volume copy/dump and tracks copy/dump are allowed for a data set that has DS1CPOIT set, with or without DS1DSGU set. Neither DS1CPOIT nor DS1DSGU is checked or reset.

Full-volume copy/dump of an SMS volume always copies to a device with like track geometry and always uses the volser of the source volume as the volser of the target volume.

Tracks copy/dump ought to be reserved as a tool to be used for data repair. Misuse of tracks copy/dump can cause significant damage to data sets. Not checking the checkpoint status of a data set has no effect on the potential for data destruction.

COPY DATASET: DFSMSdss allows a data set that has DS1CPOIT set, without or with DS1DSGU set, to be copied only if the FORCECP(*days*) parameter is specified. If a checkpointed data set is encountered during a data set copy and either FORCECP(*days*) is not specified or FORCECP(*days*) is specified and the minimum *days* have not elapsed, message ADR298E is issued. If copy is allowed, DS1CPOIT and DS1DSGU are reset for the target data set. The settings of DS1CPOIT and DS1DSGU are not changed for the source data set. Message ADR297I is issued.

When the target of a data set copy is preallocated and indicated as checkpointed, the same eligibility criteria are applied to the target data set. If the target data set is eligible to be replaced, DS1CPOIT and DS1DSGU are reset for the target, and message ADR297I is issued.

COPY DATASET: The IMS GSAM restart facility can tolerate the relocation of an extent on a volume. MVS Checkpoint/Restart cannot tolerate the relocation of any extent needed for restart.

APAR OW08803 changed the DEFRAG function not to move extents of any data set for which DS1CPOIT was set. DFSMSdss changes the code provided with APAR OW08803 so that FORCECP(*days*) is required to move the extents of a checkpointed data set if both DS1CPOIT and DS1DSGU are set in the format 1 DSCB. When FORCECP(*days*) is not specified or when FORCECP(*days*) is specified and the minimum *days* have not elapsed, existing message ADR211I is issued.

DUMP DATASET LOGICAL: A data set that is dumped logically by DFSMSdss can be restored (in most cases) to a device with different track geometry. Even when restore is to a device with the same track geometry, DFSMSdss attempts to allocate the data set in a minimum number of volumes.

The DFSMSdss logical data set dump function is invoked by DFSMShsm for MIGRATION, BACKUP, and ABACKUP.

An SMS-managed physical sequential data set that has DS1CPOIT set, with or without DS1DSGU set, is not logically dumped by DFSMSdss unless FORCECP(*days*) is specified. If FORCECP(*days*) is not specified or when FORCECP(*days*) is specified and the minimum *days* have not elapsed, message ADR298E is issued.

When a checkpointed data set is logically dumped, checkpoint bits DS1CPOIT and DS1DSGU are not reset, and message ADR297I is issued.

DUMP DATASET PHYSICAL: To be consistent with logical data set dump, users should check the checkpointed data set indicator before they perform a physical data set dump. The problem is that a physical data set restore may have as its source either a physical data set dump or a full volume dump. Therefore, to ensure that a checkpointed data set is not inadvertently destroyed, there must be a check on the physical restore side. Dump itself is nondestructive, unless DELETE is also specified. Since there is no other protection extended to checkpointed data sets to prevent deletion, it was decided that users should be able to physically dump with delete, if they so choose.

FULL/TRACKS RESTORE: Both full volume restore and tracks restore are allowed for a data set that has DS1CPOIT set, with or without DS1DSGU set. Neither DS1CPOIT nor DS1DSGU is checked or reset.

A full volume dump of an SMS-managed volume always is restored to a device with like track geometry and always uses the volser of the source volume as the volser of the target volume.

Tracks dump/restore ought to be reserved as a tool to be used for data repair. Misuse of tracks dump/restore can cause significant damage to data sets.

LOGICAL/PHYSICAL DATASET RESTORE: For logical data set restore, when DS1CPOIT is on, regardless of the setting of DS1DSGU, FORCECP(*days*) is required to restore the data set.

For physical data set restore, FORCECP(*days*) is required to restore a checkpointed data set only if both DS1CPOIT and DS1DSGU are set. IMSC data sets, those with only DS1CPOIT set, are usable for restart after a physical data set restore; therefore, FORCECP(*days*) is not required for the restore, and DS1CPOIT is not reset for the restored data set.

DFSMShsm invokes the DFSMSdss function of logical data set restore for RECALL, ARECOVER, and RECOVER (from BACKUP) and the DFSMSdss function of physical restore for RECOVER (from DUMP) if DFSMSdss is defined as the data mover in DFSMShsm.

The DFSMSdss logical data set restore function restores checkpointed data sets only if FORCECP(*days*) is specified. If FORCECP(*days*) is not specified or when FORCECP(*days*) is specified and the minimum *days* have not elapsed, the data set is not restored and message ADR298E is issued.

When the target of a logical data set restore is preallocated and indicated as checkpointed, the same eligibility criteria apply to the target. If the target data set is eligible to be replaced, DS1CPOIT and DS1DSGU are reset for the target.

The DFSMSdss physical data set restore function restores checkpointed data sets that have DS1DSGU set only if FORCECP(*days*) is specified. If FORCECP(*days*) is not specified or when FORCECP(*days*) is specified and the minimum *days* have not elapsed, the data set is not restored, and message ADR298E is issued.

When the target of a physical data set restore is preallocated and has DS1DSGU set, the same eligibility criteria apply to the target. If the target data set is eligible to be replaced, DS1CPOIT and DS1DSGU are reset for the target.

For physical data set restore of checkpointed data sets that do not have DS1DSGU set, FORCECP is not required, and DS1CPOIT is not reset. For preallocated checkpointed data sets having only DS1CPOIT set, the setting of DS1CPOIT is taken from the dumped data set rather than the preallocated data set when the settings are different.

For IMSC data sets that are physically restored, FORCECP(days) is not required, and the checkpoint indications are not removed from the data sets, whether or not FORCECP(days) is specified.

RELEASE: The DFSMSdss RELEASE function releases allocated but unused space from sequential data sets and PDSs.

The RELEASE command operates against SMS-managed physical sequential checkpointed data sets only when FORCECP(*days*) is specified. If

FORCECP(*days*) is not specified or when FORCECP(*days*) is specified and the minimum *days* have not elapsed, message ADR298E is issued.

Checkpoint bits DS1CPOIT and DS1DSGU are reset when allocated but unused space in a checkpointed data set is released, and message ADR297I is issued. If a checkpointed data set is eligible for RELEASE but has no allocated but unused tracks, the checkpoint bits are not reset.

DFSMSdss Enhancement Exit Control: The setting of the FORCECP parameter and the number of days that must elapse since the date last referenced before a checkpointed data set can be logically dumped or copied can be overridden by the DFSMSdss installationwide exit ADRUIXIT, the options exit routine, which is described in *DFSMS/MVS V1R3 Installation Exits* (SC26-4908-02).

Installations that support MVSC data sets and/or IMSC data sets and use either DFSMSdss or DFSMShsm functions must be aware of the support provided in DFSMS/MVS V1R4 to protect those data sets from inadvertent movement.

Selection of the optimal value for *days* from function to function and may vary from application to application, depending on job frequency and use of the affected data sets.

When a checkpointed data set is not processed because it is ineligible, DFSMSdss issues an error message to describe the reason the data set is not processed by the function. DFSMSdss then resumes processing with the next data set. Any changes in the setting of DS1CPOIT and DS1DSGU occur only for data sets that have successfully completed all eligibility checks.

2.4.1.3 DFSMShsm Support

One of the main purposes of implementing the DFSMShsm enhancements is to prevent premature migration of physical sequential SMS-managed checkpointed data sets.

There is no intent to prevent taking backup copies of a checkpointed data set, either for normal recovery purposes (RECOVER) or for disaster recovery purposes (ARECOVER). It is presumed that DFSMShsm users do not initiate a recover of a data set unless the current copy of the data set, if it exists at all, has been damaged in some way and is no longer usable. RECOVER is always command driven, never automatic.

DFSMShsm invokes the services of DFSMSdss for DFSMShsm functions MIGRATE, RECALL, BACKUP, RECOVER, ABACKUP, and ARECOVER. DFSMShsm uses the DFSMSdss application interface and passes the request in the form of a DFSMSdss logical data set dump command (for MIGRATE, BACKUP, or ABACKUP), a DFSMSdss logical data set restore command (for RECALL, RECOVER, or ARECOVER), a DFSMSdss physical volume dump command (for DUMP), or a DFSMSdss physical data set restore command (for RECOVER).

Checkpoint/Restart Patches: The following patch byte enables users to modify the number of days that must have elapsed since the date last referenced for a checkpointed data set (DS1CPOIT set, with or without DS1DSGU set) to be eligible for migration:

PATCH .MGCB.+112 X′nn′VERIFY(.MGCB.+112 X′05′)

Note: X′nn′ is the hexadecimal representation for the number of days. The default is 5.

Checkpoint/Restart Migration: For both command migration and automatic migration, DFSMShsm determines whether the data set to be migrated is a checkpointed data set (DS1CPOIT set), with or without DS1DSGU set. A checkpointed data set is eligible for migration when the date last referenced plus the number days the data set is to be treated as unmovable are less than or equal to the current date.

- If the data set is eligible, DFSMShsm builds logical data set dump command syntax for DFSMSdss including the FORCECP(0) parameter.
- If the data set is not eligible, and the patch for getting ARC0734I messages for data sets not selected during migration is set, DFSMShsm issues ARC1245I REASON=90 when space management is running.
- If the data set is not eligible, DFSMShsm issues message ARC1245I REASON=8 for (H)MIGRATE command migration.

DFSMShsm does not provide an interface whereby a user of the MIGRATE command can pass the FORCECP parameter in the command syntax. Instead, the installation must modify the supported patch byte to force the migration.

Checkpoint/Restart Recall: The migration of a checkpointed data set is delayed by at least the unmovable days in the migration path. Therefore any checkpointed data set that was migrated can be recalled without any additional delay. For RECALL, DFSMShsm calls DFSMSdss with the parameter FORCECP(0) in the command syntax for DFSMSdss logical data set restore.

DFSMSdss needs FORCECP(0) to indicate that the restore should be allowed to proceed.

Checkpoint/Restart BACKUP and ABACKUP: The backup of a data set with BACKUP or ABACKUP is not delayed for a checkpointed data set. DFSMShsm invokes DFSMSdss with FORCECP(0) in the command syntax to allow backup of any checkpointed data sets.

Checkpoint/Restart Volume Dump: Because DFSMSdss does not distinguish checkpointed data sets during full volume dump, a keyword is not required on the DFSMSdss invocation of a dump to cause a checkpointed data set to be dumped.

Checkpoint/Restart RECOVER and ARECOVER: The recovery of a data set, with RECOVER or ARECOVER, is not delayed for a checkpointed data set. DFSMShsm invokes DFSMSdss with FORCECP(0) in the command syntax to allow recovery, including replacement, of any checkpointed data sets.

2.4.1.4 Enhanced Checkpoint/Restart Migration and Coexistence

DFSMSdfp O/C/EOV and DFSMSdss are affected by the migration process if other DFSMS/MVS releases exist in the installation.

O/C/EOV Migration and Coexistence: The setting of DS1DSGU in the format 1 DSCB (product sensitive programming interface) of an SMS-managed physical sequential data set is incompatible with previous releases of DFSMS/MVS and DFP. Therefore, O/C/EOV has toleration PTFs to previous DFSMS/MVS releases and DFP 3.3.0.

DFSMSdss Migration and Coexistence: DFSMSdss does not supply compatibility or toleration PTFs to any previous releases. Previous releases do not honor the FORCEP(days) parameter.

A DEFRAG APAR, which allows extents to be moved when DS1CPOIT is set, will be maintained for previous levels of DFSMSdss.

Note: OEM vendor products or user applications using the format-1 DSCB fields should be aware of the use of the DS1DSGU field introduced by the enhanced protection of checkpointed data sets.

2.5 Optical Access Method SMF Recording Enhancement

Customers must be able to perform the following activities:

- Performance monitoring, analysis, and tuning
- Capacity planning, including:
  - CPU utilization and growth planning
  - Storage subsystem utilization and capacity planning for tape, DASD, and optical storage subsystems
  - Removable media capacity planning related to removable media such as tape and optical
  - Accounting for information systems use and chargeback to end-user departments for the use of information systems resources.

The SMF recording enhancement assists customers who are using OAM in their performance monitoring, analysis, tuning, and capacity planning activities.

This enhancement introduces a new SMF record for OAM at the OAM programming interface level (the OSREQ macro interface). In an ImagePlus environment, these records are used to account for the OAM portion of document image processing.

The MVS system operator or system programmer can dynamically select the OAM SMF record subtypes to be recorded.

The OAM SMF record enables customers to collect substantial statistical information about OAM usage.

Note: The OAM SMF recording enhancement applies only for the 3995 optical library dataserver. IBM ATL devices such as the 3494 and 3495 are not affected by this OAM SMF recording enhancement.

OAM records SMF records in the SMF data set to account for OAM activity. Each OAM SMF record contains three sections:

- Standard 48-byte SMF record header
- 112-byte OAM product section
- Variable length OAM data section

Table 2 lists the OAM SMF record subtypes that OAM SMF record type 85 (x ′55′) supports.

Table 1. OAM SMF Record Subtypes		
	Size	**Description**
1	324	OSREQ access
2	324	OSREQ store
3	324	OSREQ retrieve
4	324	OSREQ query
5	324	OSREQ change
6	324	OSREQ delete
7	324	OSREQ unaccess
32	336	OSMC storage group processing
33	336	OSMC DASD space management
34	336	OSMC optical disk recovery utility
35	336	OSMC MOVEVOL utility
36	296	OSMC single object recovery utility
37	184	OSMC library space management
64	256	LCS optical drive vary online
65	256	LCS optical drive vary offline
66	256	LCS optical library vary online
67	256	LCS optical library vary offline
68	284	LCS optical cartridge entry
69	284	LCS optical cartridge eject
70	284	LCS optical cartridge label
71	284	LCS optical volume audit
72	284	LCS optical volume mount
73	284	LCS optical volume demount
74	variable (min. 380, max. 32,744)	LCS optical write request
75	380	LCS optical read request
76	380	LCS logical delete request
77	variable (min. 380, max. 32,744)	LCS optical physical delete request
78	variable (min. 380, max. 32,744)	LCS object tape write request
79	380	LCS object tape read request
87	228	LCS Object tape volume demount (OAM usage)

2.5.1 Start Time and End Time Accuracy

Each OAM SMF record has a function start time and end time in the common OAM product section. The start time and end time are in System/390 store clock (STCK) instruction format.

Every attempt is made to obtain the start and end time of the OAM function as soon and as close as possible so that the elapsed time of the function includes as much OAM processing time as possible.

2.5.1.1 OSREQ Macro TTOKEN Keyword

A new optional keyword, TTOKEN, is added to the OSREQ macro. The OSREQ macro is the general-use programming interface provided with OAM. The new keyword is valid on all forms of the OSREQ macro:

- List (MF=L)
- Execute (MF=E)
- Modify (MF=M)

The format of the new keyword is:

{**TTOKEN** = ¬ *tracking-token* | (tracking-token-pointer)]}

where *tracking-token* specifies a 16-byte area containing a tracking token. The contents of the 16-byte tracking token may be anything. The tracking token supplied on the OSREQ macro with the **TTOKEN** keyword is placed in OAM SMF record subtypes 1 through 7. If no tracking token is supplied on the OSREQ macro, the tracking token field in record subtypes 1 through 7 will contain binary zeros.

A new reason code is returned in register 0, following the OSREQ macro, if the tracking token is contained in a virtual storage area to which the application program does not have both fetch and store authorization.

2.5.1.2 Invocation

Before activating OAM SMF recording following an MVS IPL, the OAM subsystem identification (OAM) must be defined in the active SMF PARMLIB (SMFPRMxx). If the OAM subsystem identification has not been defined to SMF, add the following statements to the SMFPRMxx:

```
SUBSYS(OAM,TYPE(85))
```

and then activate it by entering the following MVS operator SET command:

```
SET SMF=xx
```

If the OAM subsystem identification has been defined to SMF, the MVS system operator or MVS systems programmer can dynamically change OAM SMF recording, using one of the following methods:

- Update the SMF PARMLIB member to include the OAM SMF record subtypes, for example:

  ```
  SUBSYS(OAM,TYPE(85(2:3)))
  ```

  and activate the SMFPRMxx by entering the following MVS operator SET command:

  ```
  SET SMF(xx)
  ```

- Update the SMF options dynamically by entering the following MVS operator SETSMF command:

  ```
  SETSMF SUBSYS(OAM,TYPE(85(4:6)))
  ```

For additional information about the MVS operator SET SMF and SETSMF commands, see the *MVS/ESA System Commands Reference*, (GX22-0015-03).

2.6 Program Management 3 (PM3)

DFSMS/MVS provides program management services to create, load, modify, list, read, transport, and copy executable programs. DFSMS/MVS 1.1 introduced the program management binder and the program management loader. The binder extends the services formerly provided by the MVS/DFP linkage editor and batch loader. These enhancements include support for an executable unit called a *program object*, which includes all of the functions of a load module, with additional functional and usability improvements. The loader adds to the capabilities of program fetch and can load both program objects and load modules into storage for execution.

2.6.1 Program Management - Background

The linkage editor, batch loader, and program fetch programs are now stabilized and will not be enhanced. In DFSMS/MVS 1.1 they were replaced by the binder and loader, which compatibly include all functions of their predecessors and incorporate many enhancements and new functions. A more powerful binding algorithm has been added to the binder, greatly increasing the flexibility of DFSMS/MVS program management in support of the language products.

In addition to support for the load module, the binder introduced a new executable unit, the program object. The program object removes all structural limitations that have long restricted the load module. Flexibility in the design and structure supports ongoing enhancements.

Program objects have the following base features:

- Program object design allowed for the removal or increase of most size restrictions, including maximum text size (now 1 GB) and number of control sections (now unlimited).
- Because program objects reside exclusively in PDSEs, they can take advantage of that library technology and its many advantages.
- The program object structure is generalized and extendable. It has changed and will continue to change with each PM release.
- Program objects, physically organized as 4K fixed blocks, can be page mapped into storage (using DIV).
- Program objects support long names (up to 1K).
- Program objects contain many of the same enhancements supported in the new object file, generalized object file format (GOFF), which is currently created by the High Level Assembler (as well as the Binder as an intermediate structure). This includes support for C and C++ writable static.
- Program objects contain multiple classes of text, distinguished by attributes that control binding and loading characteristics and behavior. Classes are central to C and DLL support.
  - There are two types of classes - text (byte stream) and nontext (recordlike, IDR, ADATA)
  - The separate attributes assigned to each class include:
    - LOAD - The class is brought into memory at the time the module is loaded (typical case today).

- DEFERRED LOAD - The class is prepared for loading, but not instantiated until requested (new in PM3).
- NOLOAD - The class is not loaded with the program, that is, it is nontext.
- RMODE 24/ANY - Indicates placement of segments within virtual storage

– A *section* is the smallest unit that users can manipulate (replace, delete, order). Each section may supply contributions to one or more classes.

– Classes are bound into independently loadable *segments*. A segment contains classes with compatible attributes. A program object may have multiple segments.

– The loading characteristics of the class (and segment) determine the placement of the segment in virtual storage. Multisegment program objects can be loaded into noncontiguous areas of virtual storage, for example, when bound with the RMODE(SPLIT) option.

– Program objects contain a class of data specifically intended for users to save associated or application data (ADATA). It is not loadable (NOLOAD). This data includes source statements, debugging tables, user information, history data, and documentation. It is accessible through the binder API.

2.6.2 PM3 Enhancements for DFSMS/MVS V1R4

The basic functions of the binder were not modified in PM3; that is, the binder binds sections to create modules and saves them in libraries or prepares them for immediate execution. In addition, the binder API continues to provide support for calling programs to access data and to construct, bind, or copy modules.

PM3 addresses some current problems in support of C and C++. It simplifies the creation of an executable module and extends support for OpenEdition. Specifically PM3 includes:

- A conversion function added to the binder to process XOBJ structures built by the C and C++ compilers. The output from the binder on this path is a program object that must be stored in a PDSE. This removes the need for the LE/370 Prelinker ***in those cases where a PDSE is specified as the target load library*** (that is, SYSLMOD).
- Enhanced binder and loader to support OS/390 dynamic linking and DLLs. This enhancement provides for user-controlled special handling of external references in modules, to allow their resolution when loading the referenced module explicitly or at the time the module is referenced, that is, it supports forms of dynamic loading.
- Support for the above two enhancements has also been included in OpenEdition and the C89 shell command.
- The loader is enhanced to support DLLs and deferred loading, as well as the OS/390 R4 Dynamic LPA functions, extending this support to PDSEs and all program objects, including DLLs.

Chapter 3. DFSMSdfp VSAM

In this chapter we describe the following enhancements to VSAM delivered in DFSMS/MVS V1R4:

- VSAM system-managed buffering
- VSAM fast load
- Update of the VSAM last reference date at close
- VSAM RLS KSDS extended addressability
- Data class support for more VSAM attributes
- DCOLLECT

Note: Now is probably a good time to review all performance-oriented parameters that can be specified in the storage class, such as BIAS, MSR, and SDR. You should also review the characteristics of extended format (EF) sequential and VSAM KSDS data sets and their definition, using ISMF data class panels.

Because of the additional usability characteristics of data sets defined in EF, more and more function is being delivered through EF data sets. Data set allocation and performance are directly affected by appropriately setting the storage class values.

3.1 VSAM System-Managed Buffering

VSAM system-managed buffering (SMB) enables VSAM to determine the optimum number of index and data buffers, as well as the type of buffer management (sequential or direct). Where appropriate a switch to local shared resource (LSR) buffering occurs automatically without changes to JCL, as are required now to use Batch LSR processing.

Performance improvements could be dramatic, particularly where defaults for the number of buffers are used, or a switch from nonshared resource (NSR) processing to LSR is made.

SMB is available under the following conditions:

- The VSAM data set must be in EF

  EF data sets are required to be system managed.

- ACB MACRF must be NSR

  The ACB MACRF parameter must not contain LSR, GSR, RLS, ICI, AIX, or UBF.

- Record_Access_Bias must be set to SYSTEM in the data class, or the JCL AMP parameter ACCBIAS must be set to SYSTEM, SO, SW, DO, or DW. For a description of these parameter values, see 3.1.1, “Record Access Bias Specification” on page 36.

VSAM is guided on how to optimize buffers based on the Record_Access_Bias parameter specified in the data class or on the JCL AMP parameter. The manner in which buffers are processed refers to buffer management continuing

as NSR, or being changed internally to LSR to take advantage of LSR capabilities, which may include the use of hiperspace.

The order of precedence for specifying values is: JCL over data class, with data class having precedence over specifications in the ACB. The MACRF values of SEQ, DIR, and SKP are used with a specification of SYSTEM to determine how buffers will be acquired. Also used with SYSTEM are the storage class values for BIAS and/or MSR, should they be specified.

3.1.1 Record Access Bias Specification

Record Access Bias is specified on either the JCL DD statement or in the data class:

- **JCL**

  The AMP parameter can be used to specify access bias with a subparameter of ACCBIAS. This subparameter can have one of six specifications:

  USER Bypass SMB

  SYSTEM Force SMB and let the system determine the buffering technique according to the ACB MACRF and storage class specifications

  SO SMB with sequential optimization

  SW SMB weighted for sequential processing

  DO SMB with direct optimization (forces switch to LSR)

  DW SMB weighted for direct processing

- **Data class**

  A new subparameter for a DATA SET NAME TYPE of EXT is REC_ACC_BIAS. Either USER or SYSTEM can be specified for REC_ACC_BIAS.

Specifying the type of Record Access Bias through the AMP parameter in JCL will override anything specified in the data class for this parameter. If nothing has been specified for this parameter, the default is USER. USER indicates that VSAM will continue to use buffers as it now does without SMB. Record Access Bias is ignored if the data set is not in EF.

SMB either *weights* the buffer handling toward sequential or direct processing or *optimizes* the buffer handling for sequential or direct processing. Weighting means that when the Record Access Bias specifies SW (sequential weighted) most buffers will be used to support sequential processing but some will be reserved for index buffers to help any direct processing. Conversely when the Record Access Bias specifies DW (direct weighted), most buffers will be used to support fast direct access to the data, with relatively few buffers reserved for any sequential processing that might occur. The manner in which buffer selection may be affected by the user program is specified by the use of the ACB MACRF parameters of SEQ, SKP, and DIR and the use of the storage class BIAS and/or MSR values (see Table 2 on page 37).

A value of SYSTEM (including SO, SW, DO, or DW for ACCBIAS) specifies that VSAM is to determine the number of buffers to obtain for the data set when NSR processing is used. When NSR processing is specified or defaulted, and VSAM chooses DO (direct optimized) as the most appropriate type of access, the buffering technique changes from NSR to LSR. LSR is also used if direct optimization is forced by specifying ACCBIAS=DO on the AMP parameter.

When SMB determines that LSR buffer management is to be used, VSAM also determines, by default, the number of virtual storage buffers to use.

Table 2. ACB MACRF Parameters and Storage Class BIAS

ACB MACRF Parameters	Storage Class BIAS			
	SEQ	DIR	Both	None
MACRF=DIR	DW	DO	DO	DO
MACRF=SEQ (default)	SO	SW	SO	SO
MACRF=SEQ and SKP	SO	SW	SW	SW
MACRF=SKP	DW	DW	DW	DW
MACRF=(DIR,SEQ) or (DIR,SKP) or MACRF=(DIR,SEQ,SKP)	SW	DW	DW	DW

Note: DO=direct optimized, DW=direct weighted, SO=sequential optimized, SW=sequential weighted

3.1.2 ISMF Data Class Panels

Several ISMF panels have been updated for the data class to support RECORD_ACCESS_BIAS = S (System) or U (User).

Neither SO, SW, DO, nor DW are specified in the data class as these are determined from a consideration of the storage class BIAS and/or MSR values and the ACB MACRF values as shown in Table 2.

3.1.3 Buffers Used for Direct Optimization

If DO is set, SMB converts NSR buffering to LSR buffering, in which case three new optional AMP parameters can be specified to tell the LSR buffer manager how to handle the processing of the buffers. The three new subparameters are:

SMBVSP specifies the amount of virtual storage to obtain for buffers when opening the data set

SMBHWT used to allocate hiperspace buffers based on a multiple of the number of virtual buffers that have been allocated

SMBDFR allows the user to specify whether writing the data from a buffer to disk can be deferred until the buffer is required for a different request or the data set is closed. A CLOSE TYPE=T will not write the buffers to disk when LSR processing is used for direct optimization.

3.1.3.1 Virtual Buffers for Direct Optimization

The two new AMP subparameters, SMBVSP and SMBDFR, can be used to tell the LSR buffer management how to handle the processing of the virtual buffers.

The amount of virtual buffer space to be acquired when opening the data set for LSR processing can be specified with the SMBVSP subparameter of AMP. The format is:

```
SMBVSP=nnK
```

where nn is 1 to 204800
or

```
SMBVSP=nnM
```

where nn is 1 to 2048

The SMBVSP parameter can be used to override the default buffer space to be obtained, which generally speaking is calculated assuming 20% of the data will account for 80% of the accesses. The buffer space acquired is split across two LSR pools; one for the index and one for the data.

Note: The value specified is the total amount of virtual storage that can be addressed in a single address space. It does not take into consideration the storage required by the system or the access method.

When direct optimization is used, the write processing of modified buffers can be deferred until the data set is closed or the buffer is required for a different request. The SMBDFR subparameter of AMP is used to specify deferred write. The format is:

```
SMBDFR=Y|N
```

where Y is the default for SHAREOPTIONS (1,3) and (2,3), and N is the default for SHAREOPTIONS (3,3), (4,3), and (x,4)

3.1.3.2 Hiperspace Buffers for Direct Optimization

Direct optimization converts NSR buffering to LSR buffering, which facilitates the use of additional buffers that are kept in hiperspace.

The amount of hiperspace to be used for LSR buffers can be specified with the SMBHWT subparameter of AMP. The value specified for SMBHWT is used as the hiperspace weighting factor. The format is:

```
SMBHWT=nn
```

where nn is a whole decimal number between 1 and 99

The hiperspace buffer size will be a multiple of 4096 (4K). These buffers may be allocated for the base data component of the sphere.

If the control interval (CI) size of the database component is not a multiple of 4K, both virtual space and hiperspace are wasted. In addition, excessive use of this facility will have a negative affect on performance for both the system and the user program. Overhead is involved in locating a buffer for the user program that may not be in hiperspace storage because of system requirements.

The SMBHWT value is not a direct multiple of the number of virtual buffers that are allocated to the resource pool but act as a weighting factor for the number of hiperspace buffers to be established. If SMBHWT is not specified, hiperspace will not be used.

3.1.4 Location of Buffers and Control Blocks

Although SMB allows for the specification of buffer handling techniques, it does not allow the user to specify the location of buffers and control blocks. A new JCL AMP parameter, RMODE31, is provided to allow the user to specify the location of buffers and control blocks. It will override any specification for the ACB macro. This capability is necessary when there are a large number of data sets open to the job and/or the number of buffers may cause a storage shortage when kept in 24-bit addressable storage. The values that can be specified for RMODE31 with the JCL AMP parameter or the ACB are:

CB - Control blocks above 16 MB
NONE - control blocks and buffers below 16 MB
BUFF - Buffers above 16 MB (this is the default)
ALL - Control blocks and buffers above 16 MB

The RMODE31 parameter that is currently specifiable in the ACB will be merged with, or overridden by, the new JCL RMODE31= AMP subparameter.

The merge of the AMP=('RMODE31=...') and the ACB RMODE31= keyword is summarized in Table 3. The DEFAULT value results when RMODE31 is not specified in JCL, Data Class RECORD_ACCESS_BIAS is not USER, and JCL AMP ACCBIAS is not USER.

Table 3. RMODE31 Selection

ACB RMODE31 OPTION	AMP=('RMODE31=...') ON JCL				
	NONE	**BUFF**	**Control block**	**ALL**	**DEFAULT**
NONE	NONE	BUFF	Control block	ALL	BUFF
BUFF	NONE	BUFF	Control block	ALL	BUFF
Control block	NONE	BUFF	Control block	ALL	ALL
ALL	NONE	BUFF	Control block	ALL	ALL

Note: The use of the JCL AMP RMODE31= parameter is valid only on systems with DFSMS/MVS V1R4 installed. A JCL error will be returned for earlier releases.

3.1.5 LSR Resource Pools

Currently, when using local shared resources, each address space may have up to 16 index resource pools and 16 data resource pools independent of other address spaces. The SHRPOOL parameter is used to indicate which resource pool you are using. The SHRPOOL value must be a number between 0 and 15.

With DFSMS/MVS V1R4 the number of LSR pools has been increased from 16 to 256. The value specified for SHRPOOL will now be a number between 0 and 255. This will cause changes in the BLDVRP, DLVRP, ACB, GENCB ACB and MODCB ACB macro instruction SHRPOOL= values.

3.2 VSAM Load Enhancements

Changes have been made to improve the performance of loading an EF KSDS by reducing the number of I/O requests required to write the data.

As illustrated in Figure 5 on page 40, the conditions under which this enhancement take effect are:

- The KSDS must be in EF.
- The data set is defined with the SPEED parameter.
- SMB is requested in the data class or on the AMP parameter.

For more information about SMB, see 3.1, “VSAM System-Managed Buffering” on page 35.

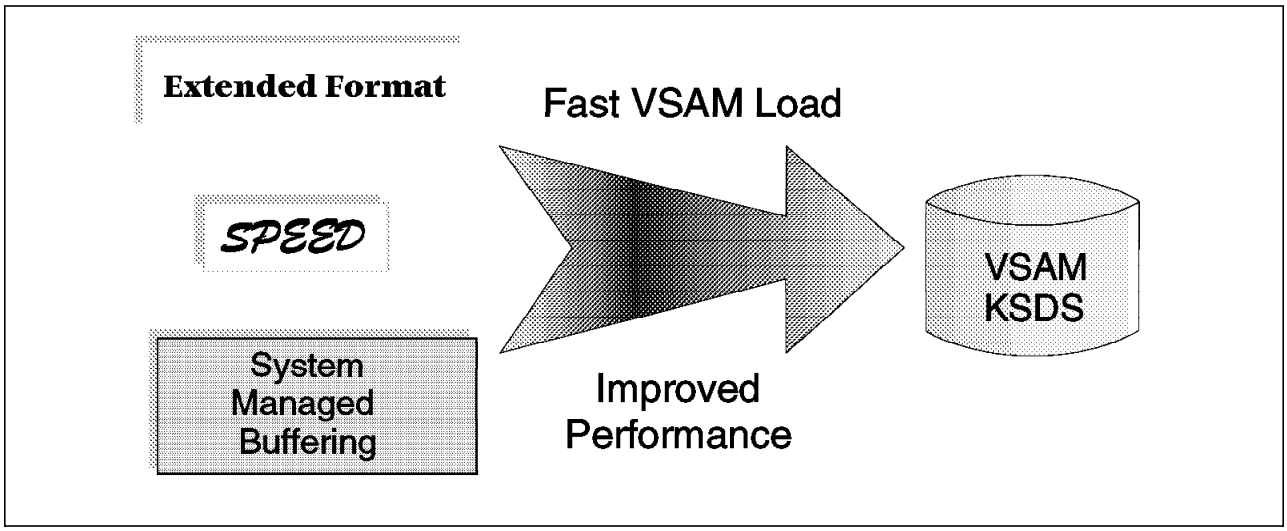


Figure 5. VSAM Loading Enhancements

The number of data buffers is optimized for load mode processing, which is sequential; system-managed buffers adjust to the load mode requirements by determining the current state of the data set. Sufficient buffers are acquired such that each control area (CA) is written with a single I/O request and full overlap of I/O and processing occurs. Before this release at least two I/O requests would have been needed for each CA, more if the data set was defined with the FREESPACE parameter specifying one or more free CIs per CA.

The index component (including the sequence set) should be updated only once per data CA area during load. Before this release the index was updated multiple times per CA. This change should improve load performance considerably.

Performance should also improve because the entire CA is written to cache with a single I/O request, and a device end is returned before all of the data is written to DASD. Even in cases where the algorithms bypass cache, performance should improve as the entire CA is written with a minimum of disk rotations.

3.2.1 Invocation

The following specifications are required for the load enhancements to take effect:

- System managed buffering
- SPEED specified in the IDCAMS DEFINE command

3.2.2 Interfaces

The interfaces used for this support are the IDCAMS DEFINE command and either the data class or the JCL AMP parameter.

3.3 Update VSAM Last Reference Date at CLOSE

The last reference date (LRD) is now updated in the format-1 DSCB on the first volume for the base data component of a VSAM sphere when the data set is closed in addition to open. The LRD is updated when the close is the last close for the data set. The current date must be greater than the date on which the data set was opened. It must also be greater than the DS1REFD date in the format-1 DSCB.

Before to DFSMS/MVS V1R4, the LRD for VSAM data sets was updated at open time. Thus there was always the potential for VSAM data sets that had been open for many days to be migrated unnecessarily.

Updating the LRD only at OPEN time is of great concern to some installations whose transaction-based systems may have a large number of data sets and/or databases open for many days or even weeks. When the transaction system is stopped, all of the data sets are eligible for migration by DFSMShsm according to the LRD set at OPEN time. Depending on the number and size of the data sets, the time to recall them could be long, thus delaying the time to bring up the transaction system again.

As illustrated in Figure 6 on page 42, updating the LRD at CLOSE time prevents the immediate migrate and recall activity and brings VSAM inline with updating the LRD for non-VSAM data sets at CLOSE time.

For non-RLS VSAM, the IDATMSTP routine is called during the open of the data set to retrieve a return code that specifies whether or not the date in the VTOC is to be changed. VSAM keeps this information until the data set is closed. For VSAM RLS, date stamp processing is always performed. Data stamp processing for close compares the date on which the data set was opened with the date on which the data set is closed to determine whether the date has changed.

3.4 VSAM RLS KSDS Extended Addressability

When VSAM RLS support was announced with DFSMS/MVS V1R3, it did not support VSAM KSDS extended addressability (EA) format. In other words VSAM RLS data sets retained the 4GB architectural limit for data set size imposed by using the 4-byte field for the relative byte address (RBA) addressing mechanism. DFSMS/MVS V1R4 removed this restriction by supporting RLS EA for VSAM KSDSs, and hence VSAM RLS now supports VSAM KSDSs up to the current multivolume limit of 59 DASD volumes.

Note: CICS was the first subsystem to exploit VSAM RLS, although batch VSAM programs could also access these data sets with read integrity. With DFSMS/MVS V1R4, DFSMShsm will use VSAM RLS for its control data sets.

DFSMS/MVS V1R4 supports EA VSAM KSDS only. It does not support other types of VSAM data sets for EA.

To use VSAM RLS the installation must have implemented a Parallel Sysplex environment. A coupling facility is therefore a prerequisite before VSAM RLS can be implemented. For a brief overview of a Parallel Sysplex and VSAM RLS, see Appendix A, “Parallel Sysplex and VSAM Record Level Sharing” on page 139.

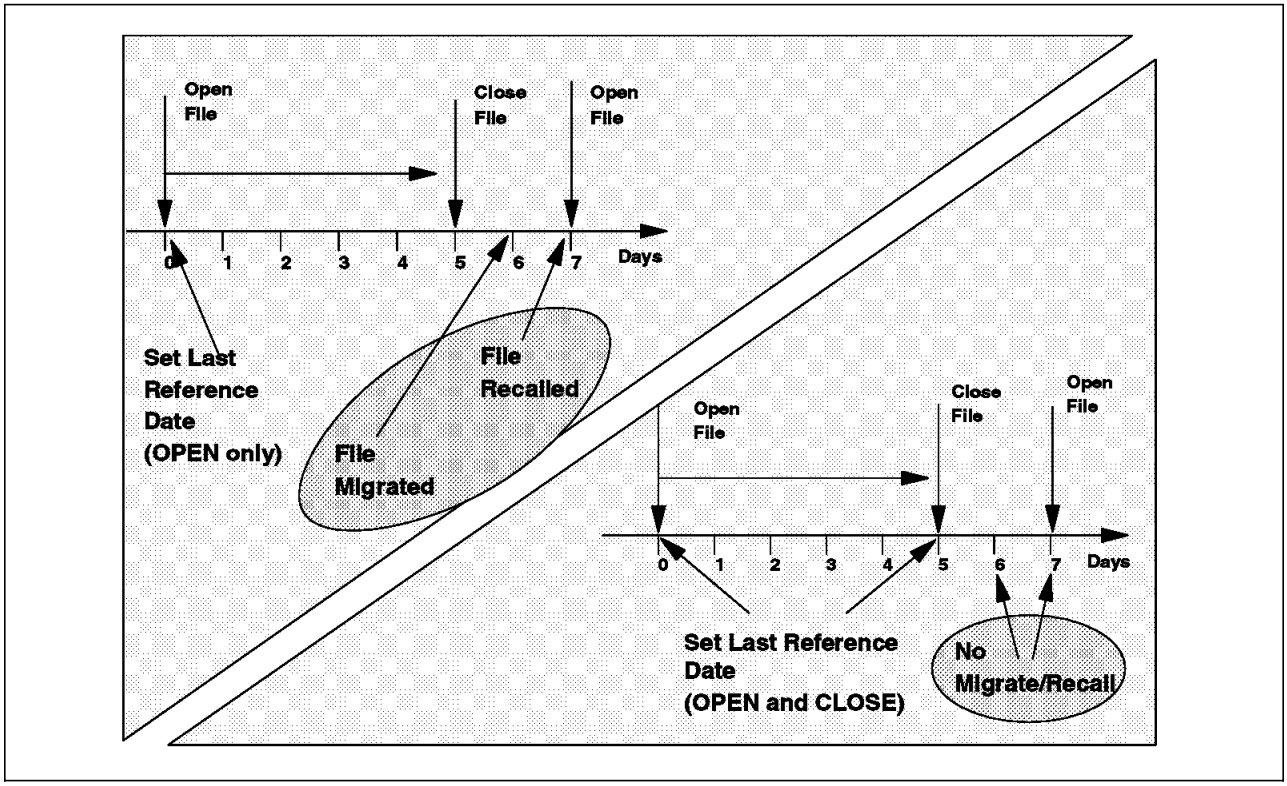


Figure 6. VSAM Last Reference Date at CLOSE. VSAM now updates the last reference date field at CLOSE time, thereby reducing the chance that a data set that has been open for many days would be migrated, only to be immediately recalled.

3.4.1 Objectives and User Requirements

VSAM RLS KSDS extended addressability satisfies these two requirements:

- A VSAM KSDS should be capable of being shared with integrity for concurrent updates from multiple processors in a parallel sysplex.
- The VSAM KSDS being shared should be capable of being much larger than 4GB.

This also satisfies the requirement of supporting the expansion of a compressed extended format VSAM KSDS into a decompressed VSAM RLS extended addressability KSDS that can be larger than 4GB.

3.4.2 Functional Characteristics

To accommodate data set sizes greater than 4GB changes have been made to VSAM RLS code from RBA fields to CI-number.

As in VSAM non-RLS extended addressability, a VSAM RLS extended addressability KSDS must have its data class defined with:

```
Data Set Name Type        =EXT
  If Ext                  =P or R
  Extended Addressability =Y
```

Just as with VSAM non-RLS EA, an application can use the TESTCB macro to test for extended addressability with VSAM RLS EA with, for example:

```
TESTCB  ACB=EFKSDS,ATRB=XADDR
```

Note: VSAM RLS (extended addressability or non-extended addressability) does not allow access to data through the RBA. RPL RBA or XRBA for GET and PUT are ignored without errors. However, control block manipulation macros GENCB, MODCB, and SHOWCB still work, even if the RBA or XRBA parameter is coded on the RPL.

Products that rely on the expanded data set size to determine the most efficient method of processing the data or return the logical data set size to the user can still do so for both EA and non-EA data sets, using an externally callable service. For example:

```
SHOWCB  ACB=EAKSDS,
        FIELDS=(XAVSPAC,XENDRBA,XHALCRBA)
```

could be used to return the amount of space, the last RBA value of the data set, and the high-used RBA of the data set. The field values are returned as 8-byte values instead of 4-byte values.

3.4.3 Migration and Coexistence

The following statements should help you understand the compatibility issues between DFSMS/MVS V1R3 and DFSMS/MVS V1R4 systems for processing a KSDS and between EA and non-EA versions of a KSDS:

1. The format of a non-EA extended format KSDS created on either DFSMS/MVS V1R3 or DFSMS/MVS V1R4 is identical. These extended format KSDSs will continue to have RBA values in the catalog and index sequence set, whether or not they are processed in RLS mode.
2. A VSAM Extended Addressable KSDS has an index that is incompatible with an extended format KSDS index without EA. As a result, a KSDS allocated as EA capable and processed in RLS mode from a DFSMS/MVS V1R4 system cannot be processed with VSAM RLS on a DFSMS/MVS V1R3 system. The DFSMS/MVS V1R3 code already supports this, failing OPEN with an IEC161I 005-731 message because it does not support VSAM RLS KSDS EA.

Use IDCAMS REPRO to migrate from non-EA KSDSs to EA KSDSs or vice versa. You can also use IMPORT/EXPORT as long as the data set is not larger than 4GB. It is not possible to use ALTER to change an existing extended format non-EA format KSDS into an EA format KSDS.

3.4.4 Software Prerequisites

There are two dependencies:

- DFSMS/MVS V1R4
- MVS/ESA SP Version 5.2 or OS/390 R4 and later

Only data sets on SMS-managed volumes are eligible for EA because the mechanism for obtaining EA is the specification of a Data Set Name Type of EXT and Extended Addressability set to Y in the data class.

3.4.4.1 Program Evaluation

Whether KSDSs with EA are used in RLS mode or not, programs accessing them must be investigated to determine whether they can be greater than 4GB.

Note: If an extended addressable KSDS is used in RLS mode, RBA processing is *not* allowed.

3.4.4.2 JCL Specification

To create a KSDS with EA, you can use either the LIKE or DATACLAS data definition keywords. If the LIKE keyword is specified, the data set used as the model must already be a KSDS with EA. If the DATACLAS keyword is specified or assigned by the ACS routines, the data class must include a Data Set Name Type of EXT with Extended Addressability set to Y.

3.4.5 Configuration Specifications

To support VSAM RLS EA, the data set must reside on a DASD subsystem that supports the extended platform and thus extended format data sets.

VSAM RLS KSDS EA will operate on all processors supported by MVS/ESA 5.2 or later. Coupling Facility hardware in the System/390 Sysplex is required. See Appendix A, “Parallel Sysplex and VSAM Record Level Sharing” on page 139 for further information.

3.5 Data Class Supports More VSAM Attributes

You can define most data set attributes, such as the organization, record format, and space required, in the data class by using ISMF. This allows data sets to be allocated via JCL or TSO ALLOCATE or dynamic allocation. The main exceptions have been the attributes of BWO, LOG, and LOGSTREAMID made available with DFSMS/MVS V1R3, and the SPANNED/NONSPANNED attribute available since 1974. If users needed these attributes, they had no option but to specify them using the IDCAMS DEFINE command or the IDCAMS ALTER command.

DFSMS/MVS V1R4 has enhanced ISMF and the Data Class Application to allow these attributes to be defined in the data class.

3.5.1 User Requirements

Users have requested that all VSAM data set definition attributes be available from a data class. DFSMS/MVS V1R4 meets this requirement now with the inclusion of BWO, LOG, LOGSTREAM ID, and the SPANNED/NONSPANNED attributes in the data class. Hence it is possible to define any VSAM data set with its required attributes through JCL, as illustrated in Figure 7 on page 45. This should be considered to be the preferred way of defining VSAM data sets in the future. In many cases a separate IDCAMS DEFINE or IDCAMS ALTER step can be eliminated.

IF JCL and the data class are used to set up a VSAM data set, attributes conflicting with each other, or with the data set definition, can be caught when the data set is created rather than when the data set is first opened.

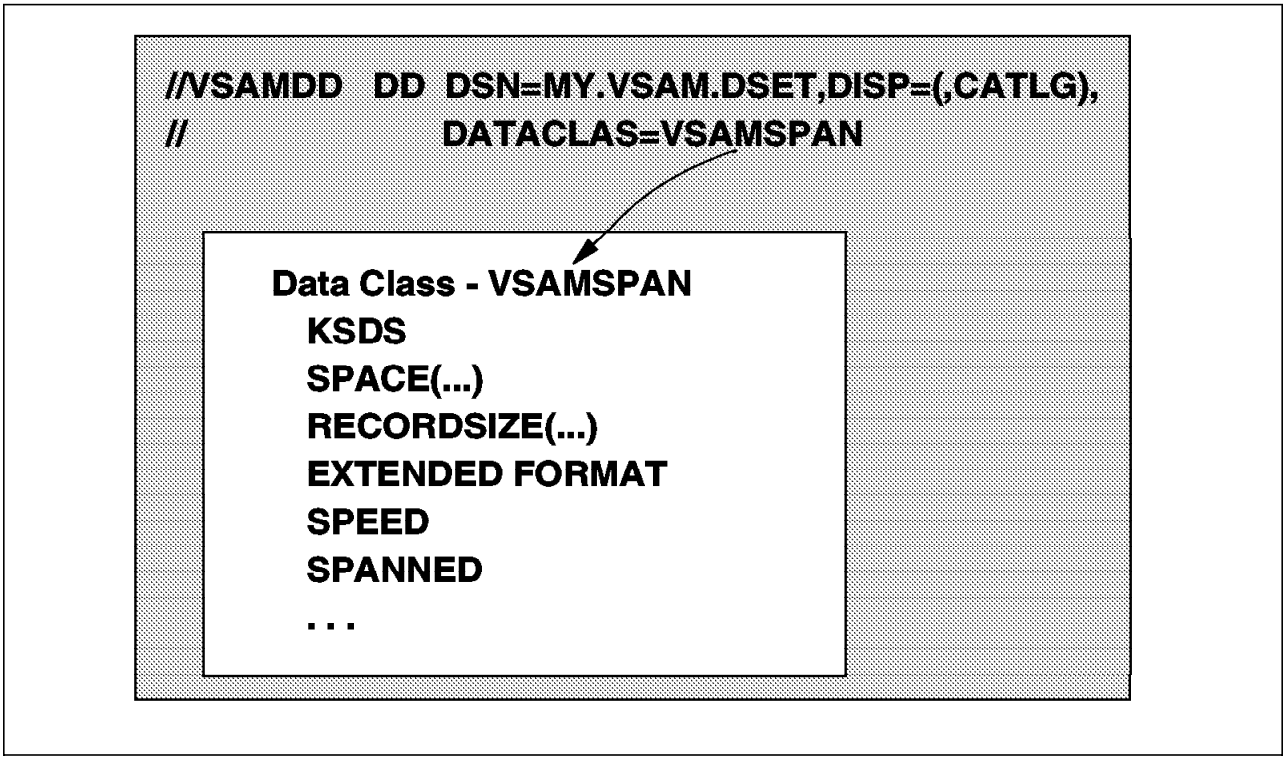


Figure 7. VSAM Data Class. The Data Class now allows more attributes to be specified, including the SPANNED attribute.

3.5.1.1 ISMF Support

ISMF has been modified to accept the BWO, LOG, LOGSTREAM ID, and SPANNED/NONSPANNED attributes in the Data Class Application:

- **BWO**(TYPECICS|TYPEIMS|NO|blank)

  BWO is an optional attribute in the data class. It should be specified if backup-while-open (BWO) processing is allowed for the VSAM sphere. BWO applies only to SMS VSAM data sets and cannot be used with TYPE(LINEAR).

  - TYPECICS (abbrev. TYPEC)

    Use TYPECICS to specify that BWO is to be performed for CICS VSAM file control data sets, to allow forward recovery using a BWO copy. The LOG attribute determines whether CICS will use this attribute from the data class or use values from the file control table (FCT).

  - TYPEIMS (abbrev. TYPEI)

    Use TYPEIMS to specify that BWO is to be performed for IMS VSAM data sets.

  - NO

    Use NO when BWO does not apply to the VSAM cluster.

  - blank (default)

    For ISMF, *blank* is the same as NO (that is, BWO is not used). When *blank* is defaulted, the attribute is not saved in the catalog.

- **LOG**(NONE|UNDO|ALL|blank)

  LOG is an optional attribute in the data class. It establishes whether the sphere to be accessed with VSAM RLS is recoverable or nonrecoverable. It

also indicates whether a forward recovery log is available for the sphere. LOG applies to all components of the sphere.

The LOG attribute determines whether CICS uses the FCT or the values in the catalog. If the LOG attribute exists in the catalog, CICS uses the LOG, LOGSTREAM ID, and BWO attributes from the catalog. If the LOG attribute does not exist in the catalog, CICS uses the FCT definitions for LOG, LOGSTREAM ID, and BWO.

- NONE

  Indicates that neither an external backout nor a forward recovery capability is available for the sphere accessed in RLS mode. If NONE is specified, RLS considers the sphere to be nonrecoverable.

- UNDO

  Specifies that changes to the sphere accessed in RLS mode can be backed out by using an external log. RLS considers the sphere to be recoverable when LOG(UNDO) is specified, but forward recovery is not possible.

- ALL

  Specifies that changes to the sphere accessed in RLS mode can be both backed out and forward recovered by using external logs. RLS considers the sphere recoverable when you use LOG(ALL).

- blank (default)

  When *blank* is defaulted, CICS use the FCT definition for non-RLS processing. OPEN is not allowed for RLS access.

- **LOGSTREAM ID**(dsname|blank)

  LOGSTREAM ID is an optional attribute in the data class. It is used to name the CICS forward recovery log stream. It applies to all components of the VSAM sphere.

  - dsname

    The name of the forward recovery log stream. The data set name (the logstreamid) has a maximum length of 26 characters, including separators.

  - blank (default)

    When this attribute is not specified in the data class, SMS fails dynamic or JCL allocation of data sets with the LOG(ALL) attribute. On an AMS DEFINE, after merging data class attributes, AMS fails the DEFINE if the LOG(ALL) attribute is specified or derived and LOGSTREAMID is not specified or derived.

- **SPANNED|NONSPANNED|**blank

  SPANNED|NONSPANNED is an optional attribute in the data class. It specifies whether a data record is allowed to cross CI boundaries. These parameters have been a part of standard VSAM record management since VSAM was first delivered in 1974 and are valid for SMS-managed and non-SMS-managed data sets.

  - SPANNED

Specifies that if the length of a data record is larger than a CI, the record will be in more than one CI. Thus VSAM can select a CI size that is optimal for the DASD.

- NONSPANNED

  Indicates that the record must be in one CI. VSAM selects a CI size that accommodates the maximum record size.

- blank

  Defaults to NONSPANNED. Hence records are not allowed to expand beyond the boundary of a single CI.

3.5.1.2 Attribute Precedence

AMS merges the BWO, LOG, LOGSTREAM ID, and SPANNED/NONSPANNED attributes from the data class if not specified on DEFINE.

When a data set is created, SMS applies the data class values for the attributes only if they apply to the data set. For example, none of these attributes is compatible with TYPE(LINEAR) and therefore none is applied to linear data sets.

3.5.1.3 Impact on the catalog

The BWO, LOG, and LOGSTREAM ID attributes are saved in the RLS cell of the data set in the catalog. The RLS cell is not created if these attributes are not specified or are not applicable. The cell is also not created for non-VSAM data sets and non-SMS data sets.

3.5.2 Migration and Coexistence

Upward compatibility is maintained.

The LOGSTREAM ID value can be up to 26 characters and increases the length of the data class construct. Toleration PTFs are required in ISMF and SMS components on systems with a release of DFSMS/MVS before DFSMS/MVS V1R4 installed to prevent failures when the construct is accessed. The four data class attributes cannot be displayed, listed, defined, or altered through ISMF on a down-level release.

The BWO, LOG, and LOGSTREAM ID attributes are saved in the RLS cell in the catalog. The RLS toleration PTFs will fail OPEN on releases before DFSMS/MVS V1R3 if the RLS cell exists for the VSAM sphere.

With DFSMS/MVS V1R4 most VSAM data sets can be defined completely through JCL without requiring a separate IDCAMS DEFINE or IDCAMS ALTER.

3.6 DCOLLECT Enhancements

DCOLLECT has been modified to provide the volser number from which the source is obtained for migrate or backup operations. The volser is the first original volume of a multivolume data set.

A new parameter for the DCOLLECT command enables you not to produce output for unneeded volumes. The new parameter, EXCLUDEVOLUMES, enables you to eliminate volumes identified by the STORAGEGROUP and VOLUMES parameters, which enable you to specify a set of volumes for which data is to be collected.

In addition, changes have been made to reporting the VSAM association information and the actual device type information. The device type shown in reports created on a system before DFSMS/MVS V1R4 is the logical device type for the volume (3380, 3390).

3.6.1 DCOLLECT OUTPUT

New fields are added to the output records produced by DCOLLECT.

3.6.1.1 New Volume Field

The **UMFRVOL** field is a new field for migrated data (UMBFRVOL for data that is backed up) that contains the first DASD volume as the source of migrated data or backup data. The new field is available in the following records:

- Migrated data set information (record type M)
- Backup data set information (record type B)

Note: The size of both records (M and B) has increased by six characters.

3.6.1.2 VSAM Association Information

Record type A has been expanded by two new fields. The first record type A for a data set is reported; this is the record for the primary, not the secondary, volume. The new fields are:

- DCACISZ, number of bytes in a CI
- DCACICA, number of CIs in a CA

3.6.1.3 Volume Information

The DCVDPTYP field in record type V reports the *true device type* rather than the generic 3380/3390. The *true device type* is the actual physical device type (for example, RAMAC), as opposed to the logical device type.

3.6.1.4 Data Class Information

For data class information, record type DC has been expanded by a new field as well as by a field extension:

- DDCRABS, the new bit - Record Access Bias is specified
- DDCRBIAS, the new field - VSAM Record Access Bias constant
  - DDCRABUS, value 0 = user
  - DDCRABSY, value 1 = system

3.6.2 DCOLLECT Command

With this DCOLLECT enhancement, you can restrict the amount of DCOLLECT output that is generated for the DCOLLECT report.

The DCOLLECT command has been enhanced with the optional parameter, EXCLUDEVOLUME. The EXCLUDEVOLUME parameter can be abbreviated as EXCLVOL, EXV, or EXCLUDE.

Figure 8 on page 49 shows the syntax of DCOLLECT with EXCLUDEVOLUMES.

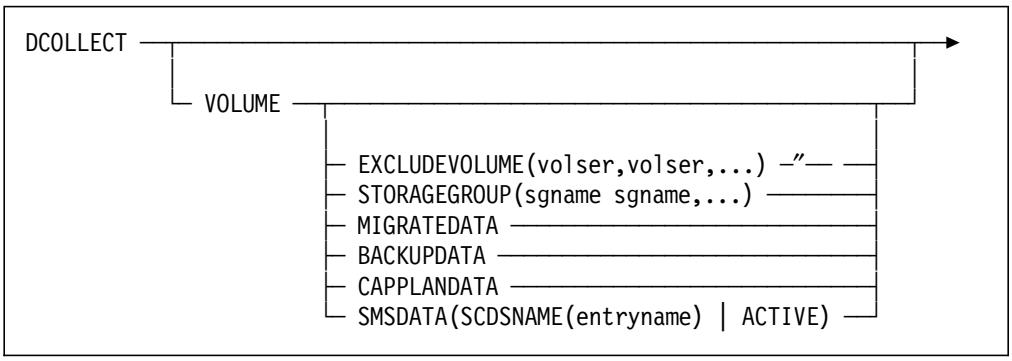


Figure 8. DCOLLECT Command Syntax

DCOLLECT can be abbreviated by DCOL. The options for EXCLUDEVOLUME are:

- A fully qualified volser, counting one to six characters
- A partially specified volser using trailing asterisks as a place holder for all remaining characters
- Any combination of the above

You can exclude information on a selected volume or group of volumes.

Chapter 4. DFSMShsm

DFSMShsm V1R4 enhances product useability as well as performance and throughput. In the disaster recovery and backup area, it provides more parallelism and better use of tape media. Therefore the DFSMShsm aggregate backup and recovery support (ABARS) function will need fewer tape cartridges than before to store the same amount of data. DFSMShsm V1R4 can exploit the Magstar technology. DFSMShsm can access its VSAM control data sets (CDSs) in RLS mode by using the coupling facility. This enhancement relieves the parallel sysplex constraint that DFSMShsm has today and enables you to take advantage of parallel sysplex benefits.

DFSMShsm V1R4 has been enhanced to write data to two tapes simultaneously. Thus you can make a backup copy or migrate data to tape on two different devices at the same time. With this new function, you can address a second, perhaps remote, Automated Tape Library (ATL).

DFSMShsm can now communicate with DFSMSrmm and the DFSMS/MVS Optimizer directly without exits. It no longer requires the exits to send requests to DFSMSrmm tape management or the DFSMS/MVS Optimizer. This frees the ARCTVEXT exit, allowing it to be used by other tape management products.

DFSMShsm will provide more account data as well as aliases for keywords on specified DFSMShsm commands.

DFSMShsm V1R4 provides the following set of enhancements:

- Duplex tape
- Alter without recall (IDCAMS ALTER)
- ABARS
- CDS access in RLS mode
- DFSMShsm and DFSMSrmm interface
- Dump analysis elimination (DAE)
- Serviceability support items

4.1 Duplex Tape

DFSMShsm V1R4 provides an alternative to TAPECOPY processing for backup and migration *cartridge* tapes. With duplex tape DFSMShsm creates two tapes concurrently. The original tape is intended to be kept onsite, and the alternate tape can be taken offsite or perhaps handled on a remote ATL. The pair of tapes will maintain the original versus alternate distinction. The original tape will have one of the following data set names:

- prefix.HMIGTAPE.DATASET
- prefix.BACKTAPE.DATASET

The alternate tape will have one of the following data set names:

- prefix.COPY.HMIGTAPE.DATASET
- prefix.COPY.BACKTAPE.DATASET

Thus the new tape can be compatible with the current tapes created by TAPECOPY. In an SMS environment, the ACS routines can direct the alternate tape to a different tape library, such as one at an offsite location. In a non-SMS environment, the output restriction (for example, SETSYS UNIT(3490)) will be used for both the original and the alternate tape.

Note: The alternate tape must have the same tape geometry as the original, for example, both must be 3490 Extended Capacity Cartridge Storage Tape (ECCST) tapes.

4.1.1 Automatic TAPECOPY Scheduling

DFSMShsm automatically schedules a TAPECOPY when necessary to ensure that valid alternate tapes exist. To this end, a new CDS record, TCN, is created in the OCDS. Automatic TAPECOPY commands are scheduled for certain infrequently occurring functions and exceptions. When a TAPECOPY completes successfully, the TCN record for that volume is deleted.

This new TCN record is type *E*, and the key is either *M-* for migration tape or *B-* for backup tape, followed by the 6-byte volume serial number, and padded with blanks.

During secondary space management, if any TCN records exist for migration volumes, another TAPECOPY is scheduled.

During autobackup on the primary host, if any TCN records exist for backup volumes, another TAPECOPY is scheduled. This copy occurs only on the primary host, to avoid multiple TAPECOPYs for the same volumes.

4.1.2 Supported DFSMShsm Functions

Duplex tape is supported for following DFSMShsm functions:

- Volume backup (including autobackup)
- Volume migration (including primary space management)
- Recycle
- Move backup copies from ML1 volumes
- Secondary space management
- Data set migration
- FREEVOL
  - Migration volumes
  - Backup volumes
  - ML1BACKUPVERSIONS
- SPILL processing
- ARECOVER ML2 tape

  Only one tape will be created. A TCN record will be created and written to the CDS to be processed by secondary space management to create the second (alternate) copy of the tape.

4.1.3 Invocation

Changes have been made to the SETSYS command and the QUERY SETSYS command in support of duplex tape.

- SETSYS command

  A new keyword, DUPLEX, has been added to the SETSYS command to allow the specification of duplexing for backup and/or migration volumes.

- QUERY SETSYS command

  The QUERY SETSYS command has been enhanced to display the current duplex status of migration and backup processing. The message is as follows:

  ```
  ARC0442I TAPE OUTPUT PROMPT FOR TAPECOPY=NO, DUPLEX
  ARC0442I (CONT.) BACKUP TAPES=YES, DUPLEX MIGRATION TAPES=YES
  ```

4.1.4 Initial Tape Selection

In a duplexing environment, DFSMShsm attempts to allocate two tape drives, mount two tapes, and write both tapes concurrently. The first tape is predesignated as the original, and the second, as the alternate.

DFSMShsm selects tapes, using the following selection process:

- In a duplex environment
  1. A partial tape and its alternate (partial tapes without alternates are excluded), if one exists
  2. If no partial tapes are found, an empty ADDVOLed volume, if one exists, with a scratch as an alternate
  3. Two scratch volumes
- In a Simplex environment:
  1. A partial tape that does not have a duplicate (partial tapes with alternates are excluded)
  2. If no partial tapes are found, an empty ADDVOLed volume, if one exists.
  3. A scratch volume

This selection process enables DFSMShsm to better support private scratch pools.

4.1.5 TAPECOPY Processing

TAPECOPY continues to function as before, with three exceptions:

- When any alternate is being replaced, that is, there is an external TAPECOPY request for its original, DFSMShsm calls the installationwide exit, ARCTVEXT, if active, to release the previous alternate tape volume through the tape management systems. A new flag in the ARCTVEXT parameter list indicates that the tape being released is an alternate tape being replaced by a user TAPECOPY; the original tape is not being released. The new flag is also defined in the interface to RMM (EDGTVEXT).
- When the internal TAPECOPY command is successfully processed, DFSMShsm deletes the TCN OCDS record for that volume and marks the volume as available.

- When an internal TAPECOPY MWE is being processed, if a valid alternate exists in the TTOC record, do not perform TAPECOPY but delete only the TCN record.

4.1.6 Tape Exception Processing

The actions that DFSMShsm takes in a duplex environment when a tape exception occurs are described below.

4.1.6.1 Allocating and Mounting an Existing Alternate

If DFSMShsm allocates and mounts an empty original tape for double output but is unable to allocate and mount a new alternate scratch volume, it continues to write the original volume. However, in this case, when the original volume (ADDVOLed or scratch) is demounted, DFSMShsm marks the volume available after a successful TAPECOPY.

4.1.6.2 I/O Error Processing

If the duplexed original tape encounters an I/O error, both tapes are marked full and demounted and two different tapes are selected. The user data set, related to the I/O error, is restarted from the beginning on the different tapes.

If the alternate tape encounters an I/O error, it is demounted and returned to scratch, but the original continues to be written until it is demounted. The original is marked as duplexed and unavailable, and DFSMShsm automatically schedules a TAPECOPY to create an alternate tape. DFSMShsm makes the volume available after a successful TAPECOPY.

4.1.6.3 End-of-Volume Processing

Should the original tape reach natural EOV, DFSMShsm markd it full and keeps both tapes if they contain any valid data. If the tapes do not contain any valid data, they are released. Regardless of whether the tapes contain valid data, the data set being processed when the natural EOV was encountered is restarted from its beginning on two new tapes.

If the alternate tape encounters natural EOV, it is demounted and returned to scratch, but the original continues to be written until it is demounted. The original is marked as duplexed and unavailable, and DFSMShsm automatically schedules a TAPECOPY to create an alternate tape. DFSMShsm makes the volume available after a successful TAPECOPY.

4.1.6.4 Percent-Full End-of-Volume Processing

When the original tape reaches its percent-full capacity, before performing a FEOV, DFSMShsm must flush all data to both tapes. If natural EOV is reached on either tape, DFSMShsm marks the tapes as *FULL* and restarts processing the current data set from its beginning on two new tapes.

4.1.6.5 Exception Handling

During some exception processing, DFSMShsm automatically schedules an internal TAPECOPY to create the alternate. See section 4.1.1, “Automatic TAPECOPY Scheduling” on page 52 for a description of automatic TAPECOPY processing.

4.1.7 AUDIT Processing

AUDIT reports discrepancies between the MCV/MCT record and the TTOC relating to the alternate volume. AUDIT does not process TCN records.

4.1.8 LIST Processing

The *(H)LIST MVOL* output includes the duplex status and the alternate volser.

The *(H)LIST BVOL* output includes the duplex status and the alternate volser.

4.1.9 Limitations

Some parameters of nongeneric TAPECOPY commands are not available with the duplex tape option.

The TAPECOPY command supports a private DFSMShsm scratch pool environment through the ALTERNATEVOLUMES and INDATASET parameters. Using those parameters, you can specify a target tape to which a specific input tape will be copied. With the INDATASET parameter, you can also specify a specific expiration date for each alternate tape. With the duplex option, all allocations are for a nonspecific tape, and all expiration dates are the same, subject to data class capabilities.

The TAPECOPY command provides an ALTERNATEUNITNAME parameter that can be useful in non-SMS allocations and SMS filtering. You can specify a unit name specifically for allocation of the alternate tape. No such capability is available when using the duplex option, so the unit name for the original and alternate will be the same. If ACS routines are used, the tape data set name can be used to direct the different allocations. The alternate tape data set name inserts a COPY qualifier that is not present in the original tape data set name.

4.1.10 Migration and Coexistence

Previous releases of DFSMShsm require toleration PTFs for all functions that cause movement of user data to tape. These functions must not attempt to extend original duplexed tapes. Such tapes will be bypassed in the selection process.

If changing to use duplexed output and you have been using the PARTIALTAPE(REUSE) option, consider marking *full* the existing partial tapes as they are not considered in a duplexing environment. Only by marking partial tapes *full* will they be considered by recycle.

4.2 Alter without Recall

Today, if you want to alter the catalog entry from a migrated data set, DFSMShsm performs a recall to bring back the data set on level 0. DFSMShsm performs a recall even if you do not change the contents of the data set.

You thus waste time until the data set is recalled and incur unnecessary CPU and I/O use. Alter without recall relieves this constraint by not recalling a migrated data set to perform an IDCAMS ALTER entryname command for only the storage class and/or management class submitted by an end user. Thus, an IDCAMS ALTER entryname command for only the storage class and/or management class submitted by an end user will cause the storage class and/or

management class to be altered for a migrated data set without recalling the data set.

The DFSMShsm record used for SMS space management processing has been updated with the new storage class and/or management class to keep it consistent with the construct names in the catalog.

If DFSMShsm fails to process the catalog update request for the requested class names, catalog will not update the class name(s) in the catalog and the IDCAMS command will fail.

When catalog fails to update the class name(s) in the catalog after DFSMShsm′s updates, the DFSMShsm record will retain the updates made. The IDCAMS command will fail.

The new alter without recall function causes both DFSMShsm′s and catalog′s records to be updated and kept synchronized. The HALTER EXEC updates only DFSMShsm records, leaving DFSMShsm′s and catalog′s records unsynchronized. The HALTER EXEC has been modified to reflect the fact that IDCAMS ALTER can now perform the alter without recall function directly.

4.2.1 New IDCAMS Interface with Catalog

IDCAMS can now use the LOCATE macro without a DFSMShsm recall. The LOCATE macro results in an SVC 26 (IGG026DU) call.

In the IDCAMS ALTER *datasetname* for only the STORAGECLASS and/or MANAGEMENTCLASS path, IDCAMS calls catalog for both a LOCATE command and an ALTER request with a *do not recall the data set indicator*. IGG026DU will not recall the data set, even if it is migrated.

IGG026DU sees the command in both cases but due to the *do not recall the data set* indicator being set, the request will pass directly on the catalog.

4.2.2 New Catalog Notification of DFSMShsm

When catalog receives one of the following requests from IDCAMS:

```
ALTER dsn STORAGECLASS(storageclassname)
          MANAGEMENTCLASS(managementclassname)
ALTER dsn STORAGECLASS(storageclassname)
ALTER dsn MANAGEMENTCLASS(managementclassbame)
```

and it has successfully completed the security checking for the data set and the data set is migrated, it notifies DFSMShsm to update its inventory with these values.

4.2.3 DFSMShsm Processing Considerations

DFSMShsm updates the CDS MCD record for the data set with the new SMS constructs sent in the ARCHSEND text string from the catalog. DFSMShsm does not update the CDS backup records.

DFSMShsm sends a return code back to the catalog, indicating the success or failure of the CDS record update. When the return code from DFSMShsm is not zero, the catalog update will not be performed and the IDCAMS alter function will fail.

If the catalog updates should fail after DFSMShsm has successfully made its update, the DFSMShsm updates will be retained. The IDCAMS command will fail. An IDCAMS message informs you about the *out of Sync* status of catalog and DFSMShsm.

4.2.4 DFSMShsm Error Processing

If the update request sent to DFSMShsm from the catalog fails, there is no retry logic in DFSMShsm. If DFSMShsm is stopped or shut down or its address space is canceled while the update request from catalog is queued to DFSMShsm, the MWE will be posted complete with a nonzero return code. DFSMShsm will not process the request.

4.3 ABARS

The following ABARS enhancements are included in DFSMShsm V1R4:

- ABARS output files can be stacked on a minimum number of tape volumes
- Number of concurrent active ABARS requests has increased to 64.
- Invocation of ARCBEEXT has been extended to data sets that are being dumped by DFSMSdss processing so that installations can bypass data sets that fail.
- GDG base names can be specified in the ALLOCATE list. Thus you can back up and restore GDG base definitions without having to back up an associated generation data set (GDS).
- ABARS activity log on DASD or tape is automatically deleted when aggregate roll-off occurs, during either automatic roll-off or EXPIREBV processing.
- CPU time for processing ABACKUP or ARECOVER is maintained in the ABACKUP/ARECOVER Function Statistics record (WWFSR) and aggregate backup and recovery (ABR) record. A new 32-character accounting information attribute in the aggregate group definition is saved in the WWFSR and the ABR record.
- The TGTGDS and OPTIMIZE keywords used when invoking DFSMSdss to dump and restore data sets are externalized.

4.3.1 Output File Stacking

Today ABARS ABACKUP typically needs at least three tape cartridges for output: one for the control file ABARS ARECOVER needs for the recovery process, one to contain the activity log and instruction data set, and one or more for the data itself.

ABARS provides a new function that allows the ABACKUP output files to be stacked on a minimum number of tape cartridges. The minimum number of tape cartridges could be 1 if the aggregate is small.

4.3.1.1 Invocation of Output File Stacking

The SETSYS command has been updated with a new parameter, ABARSTAPES(STACK|NOSTACK), with *STACK* as default.

The *STACK* parameter indicates to ABACKUP to stack the ABACKUP output files onto a minimum number of tape volumes, *NOSTACK* indicates to ABACKUP not to stack the ABARS tape. Specifying NOSTACK causes ABACKUP to operate as it does in earlier releases of DFSMShsm.

4.3.1.2 QUERY SETSYS and QUERY ABARS

The *QUERY SETSYS* and *QUERY ABARS* commands have been enhanced to display the status of the *ABARSTAPES* parameter in message ARC6036I.

4.3.1.3 ABACKUP and ARECOVER DATASETNAME

The ABACKUP command has been enhanced with the new STACK|NOSTACK parameters to allow installations to override the SETSYS ABARSTAPES.

The ARECOVER command with the DATASETNAME parameter allows new STACK|NOSTACK keywords to be specified. Specifying STACK|NOSTACK indicates to ARECOVER DATASETNAME processing whether or not the ABACKUP output was stacked. Whether output was stacked must be known before mounting the control file tape because ARECOVER has to allocate file sequence number 4 if STACKED or not specify a file sequence number if not STACKED.

The STACK|NOSTACK parameter is unnecessary on the ARECOVER command specified with the AGGREGATE parameter. ARECOVER AGGREGATE obtains all information about ABACKUP output files from the catalog. Thus it will knows whether or not to allocate with an appropriate file sequence number.

4.3.2 Up to 64 Concurrent Requests

ABARS has been enhanced to allow up to 64 concurrent ABARS address spaces. The current restrictions remain; that is, you cannot simultaneously back up the same aggregate group name or recover using the same aggregate group name or control file data set name.

4.3.2.1 Invocation

The existing SETSYS MAXABARSADDRESSSPACE(n) parameter has been enhanced to allow the value of *n* to range from 1 to 64, with the default remaining as 1.

The started task identifier contains the task number as part of the identifier. The identifier is currently ABARnntt, where nn is the task number, and tt is a time stamp. The value of nn can now range from 1 to 64. For example, task 56 would get an identifier of ABAR56tt.

4.3.3 Invocation of ARCBEEXT Extended to DFSMSdss Processing

ABACKUP processing has been enhanced to extend the error conditions where the ARCBEEXT installationwide exit gets control. With this enhancement ARCBEEXT gains control when DFSMSdss errors occur while dumping level 0 DASD data sets in the INCLUDE list.

The ARCBEEXT installationwide exit is now called if SETSYS EXITON(BE) is specified and one of the following failures occurs:

- An *E* level DFSMSdss message is issued indicating that a data set has failed dump processing.
- A *W* level DFSMSdss messages is issued and DFSMSdss does not indicate that the warning is OK.

When one of the above failures occurs, the failing data set name and an indicator that a DFSMSdss failure occurred are passed to ARCBEEXT. Thus you can determine whether the data set should be skipped. If ARCBEEXT does not indicate that the data set be skipped, ABACKUP will fail at that point; otherwise the data set in error will be skipped and the next data set processed.

To enable this support, you must specify SETSYS EXITON(BE) before issuing the ABACKUP command.

4.3.4 GDG Base Name in ALLOCATE Statement

Customers have asked for a method where they can have ABARS define a GDG base at the recovery site without having to back up any of the GDSs. Older versions of DFSMShsm do not permit the specification of a GDG base name in the selection data set. If a GDS name was specified, however, ABARS automatically retrieved the GDG base definition and maintained it in the control file. This support will remain in effect.

ABARS has been enhanced so that you can specify a GDG base name in the ALLOCATE statement and have ABARS predefine the GDG base name during ARECOVER processing if it does not currently exist. You can specify a GDG base name even though you have not specified an associated GDS name in the selection data set.

During ARECOVER processing, all GDG base names are defined before any GDSs specified in the aggregate are restored.

4.3.5 Automatic Delete of ABARS Activity Log during Roll-Off Processing

A new SETSYS parameter, ABARSDELETEACTIVITY(Y | N), enables you to select whether or not you want to have DFSMShsm automatically delete the ABARS activity logs during ABARS roll-off processing or EXPIREBV ABARSVERSIONS processing.

If you select ABARSDELETEACTIVITY(Y), the ABARS activity log is deleted during either automatic roll-off of expired ABARS versions or EXPIREBV ABARSVERSIONS processing.

If you specify ABARSDELETEACTIVITY(N), there is no automatic deletion.

The default is ABARSDELETEACTIVITY(N).

4.3.6 CPU Time for Aggregate Processing in WWFSR

ABARS now maintains the CPU time for processing ABACKUP and ARECOVER requests in the WWFSR control block. A WWFSR is created at the completion of each individual ABACKUP and ARECOVER request and can be written as an SMF record if requested.

The ABR record has been enhanced to maintain an ABACKUP CPU time and an ARECOVER CPU time.

If an ARECOVER request fails and is reissued with a valid RESTART data set, the CPU time in the WWFSR reflects only the time to process the remaining data sets. The ABR record, however, accumulates the CPU times of each restart until the recovery of the aggregate is successful.

The ISMF aggregate group definition panels have been enhanced to allow specification of a 32-character accounting code. This accounting code is also written to the WWFSR control block, ABR record, and the ABACKUP control file. It is written to the control file so that the account code can be propagated to the recovery site without an aggregate definition at the recovery site.

If you specified SETSYS SMF(smfid), ABARS writes the WWFSR as an SMF record with the record number (smfid+1) in the SYS1.MANx or SYS1.MANy system data set. You can then use the reported CPU time to assist in chargeback of ABARS requests.

4.3.7 TGTGDS and OPTIMIZE Keyword Externalized

The TGTGDS and OPTIMIZE keywords that ABARS passes to DFSMSdss to dump and restore data sets are externalized.

4.3.7.1 TGTGDS Keyword

The TGTGDS keyword that ARECOVER processing passes to DFSMSdss during restore processing is externalized in this release of DFSMShsm. This provides greater flexibility managing SMS-managed GDSs that are being restored to level 0 DASD.

ABARS introduces a new SETSYS ARECOVERTGTGDS(option) parameter.

Valid values for `option` are:

- DEFERRED

  Specifies that the target data set is to be assigned the DEFERRED status

- ACTIVE

  Specifies that the target data set is to be assigned the ACTIVE status, for example, rolled into the GDG base

- ROLLEDOFF

  Specifies that the target data set is to be assigned the ROLLEDOFF status

- SOURCE

  Specifies that the target data set is to be assigned the same status as that of the source data set. This is the default; it maintains consistency with releases of DFSMShsm previous to DFSMShsm V1R4.

The installation can override the SETSYS ARECOVERTGTGDS with a new keyword on the ARECOVER command, TGTGDS(option). The values for `option` are the same as for ARECOVERTGTGDS.

4.3.7.2 OPTIMIZE Keyword

The OPTIMIZE keyword that ABACKUP processing passes to DFSMSdss during dump processing of level 0 data sets is externalized in this release of DFSMShsm. This keyword enables installations to adjust performance when backing up level 0 DASD data sets that are specified in the INCLUDE list.

ABARS introduces a new SETSYS ABARSOPTIMIZE(n) parameter. The value specified for *n* controls what ABACKUP will specify in the OPTIMIZE keyword when invoking DFSMSdss to back up level 0 DASD data sets.

Valid values for *n* are as follows:

- 1 DFSMSdss reads one track at a time.
- 2 DFSMSdss reads two tracks at a time.
- 3 DFSMSdss reads three tracks at a time.
- 4 DFSMSdss reads a cylinder at a time.

The default for *n* is 3.

The installation can override the specification for *n* in the SETSYS ABARSOPTIMIZE keyword by specifying a new OPTIMIZE(option) keyword on the ABACKUP command. The values for `option` are the same as for ABARSOPTIMIZE.

If ABACKUP OPTIMIZE is not specified, ABACKUP will use the value defined for *n* in the SETSYS ABARSOPTIMIZE(n) keyword.

4.3.8 ISMF Changes for ABARS Accounting Information

The following ISMF Aggregate Group Applications are affected by these ISMF changes:

- Aggregate Group Define/Alter
- Aggregate Group Display
- Aggregate Group List Display
- Aggregate Group List Sort
- Aggregate Group List View
- Aggregate Group List Print

4.3.9 Migration and Coexistence

There are migration and coexistence concerns if you have multiple systems with both DFSMShsm V1R4 and earlier releases of DFSMShsm installed.

4.3.9.1 Disaster Recovery Site Considerations

If you specify the NOSTACK parameter on the ABACKUP command when backing up on a DFSMShsm V1R4 system, you can successfully ARECOVER on a downlevel DFSMShsm.

4.3.9.2 Migration

Use the NOSTACK parameter on the ARECOVER DATASETNAME command to ARECOVER backups performed on a downlevel version of DFSMShsm.

4.3.9.3 Coexistence

Use the ARECOVER AGGREGATE command on a DFSMShsm V1R4 system to ARECOVER a backup performed on a downlevel DFSMShsm. This command also allows a downlevel version of DFSMShsm to ARECOVER an aggregate backed up on DFSMShsm V1R4, whether or not the output files are stacked.

4.4 CDS Record Level Sharing

DFSMShsm suffers data set contention against its CDSs when three or more hosts are processing in a shared environment. This is especially true when a work profile has a large amount of small data sets for DFSMShsm to process. DFSMShsm V1R4 accesses its CDSs in VSAM RLS mode, which relieves this contention and improves throughput and performance of DFSMShsm functions.

DFSMShsm V1R4 provides an option for its CDSs to be accessed in RLS mode. Thus DFSMShsm can take advantage of the S/390 coupling facility to reduce CDS contention.

For a brief description of the benefits of a Parallel Sysplex with a coupling facility and how it works, see Appendix A, “Parallel Sysplex and VSAM Record Level Sharing” on page 139, which also contains an overview of VSAM RLS and VSAM RLS locking.

DFSMShsm uses VSAM RLS with LOG(NONE), which means that DFSMShsm does not use the MVS Logger facility. The DFSMShsm VSAM RLS implementation uses the CACHE and LOCK coupling facility structures. The DFSMShsm CDS recover process is unchanged. DFSMShsm still uses its own journal for forward recovery.

4.4.1 Invocation

Startup procedure keyword CDSSHR has been enhanced to accept RLS as a parameter. When RLS is specified, the DFSMShsm CDSs are accessed in RLS mode. When the CDSs are accessed in RLS mode, any values specified for CDSQ and CDSR are ignored.

CDSSHR = {YES|RLS|NO}

```
YES  Perform multiple-processor serialization of the type requested
     by the CDSQ and SDSR keywords.

RLS  Perform multiple-processor serialization, using RLS.

NO   Do not perform mulitple-processor serialization.
```

If the CDSSHR keyword is not specified, multiple-processor serialization (non-RLS mode) is used if the index component of the MCDS resides on a DASD volume that has been SYSGENed as SHARED or SHAREUP. This process remains unchanged from the previous release.

4.4.2 CDS Access in Record Level Sharing Mode

When DFSMShsm is active, user programs and ISV products that access the CDSs must use the same access technique as that DFSMShsm currently uses.

4.4.3 QUERY CONTROLDATASETS Command

The QUERY CONTROLDATASETS command has been enhanced to display the CDS serialization technique in use and the boundaries being used for dynamic key boundary multiclusters.

4.4.4 Multicluster CDSs

When RLS is used to access the CDSs, they cannot be defined with key ranges. The multicluster support for the CDS has been enhanced so that the multiclusters can be defined without key ranges. DFSMShsm dynamically calculates the key boundaries used for each cluster.

DFSMShsm continues to support key ranges for CDSs not accessed in RLS mode.

4.4.5 CDS Creation or Redefinition

When you use dynamic key boundary multicluster support, the CDSs cannot be empty when you start DFSMShsm.

4.4.5.1 Sample JCL for Multicluster, Non-Key-Range, and RLS-Eligible CDSs

The following is sample JCL for redefining the CDSs as multicluster, non-key-range, and RLS eligible. This sample JCL is shipped in ARCTOOLS.

```
//HSMCDS   JCL ,MSGLEVEL=(1,1)
//*******************************************************************/
//* SAMPLE JCL THAT ALLOCATES NEW MULTICLUSTER, NONKEY-RANGE, RLS   */
//* ELIGIBLE CONTROL DATA SETS AND COPIES THE OLD KEY-RANGE CONTROL */
//* DATA SET INTO THEM                                              */
//*******************************************************************/
//*
//STEP1    EXEC PGM=IDCAMS,REGION=512K
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSIN DD *
 DELETE DFHSM.RLS.MCDS1 CLUSTER
 DELETE DFHSM.RLS.MCDS2 CLUSTER
 DELETE DFHSM.RLS.BCDS1 CLUSTER
 DELETE DFHSM.RLS.BCDS2 CLUSTER
 DELETE DFHSM.RLS.OCDS CLUSTER

 DEFINE CLUSTER (NAME(DFHSM.RLS.MCDS1) -
     STORAGECLASS(STRCLRLS) -
     CYLINDERS(2) NOIMBED NOREPLICATE -
     RECORDSIZE(200 2040) FREESPACE(0 0) -
     INDEXED KEYS(44 0) SHAREOPTIONS(3 3) -
     UNIQUE LOG(NONE)) -
   DATA -
     (NAME(DFHSM.RLS.MCDS1.DATA) -
     CONTROLINTERVALSIZE(12288)) -
   INDEX -
     (NAME(DFHSM.RLS.MCDS1.INDEX) -
```

```
    CONTROLINTERVALSIZE(4096))

 DEFINE CLUSTER (NAME(DFHSM.RLS.MCDS2) -
     STORAGECLASS(STRCLRLS) -
     CYLINDERS(2) NOIMBED NOREPLICATE -
     RECORDSIZE(200 2040) FREESPACE(0 0) -
     INDEXED KEYS(44 0) SHAREOPTIONS(3 3) -
     UNIQUE LOG(NONE)) -
   DATA -
     (NAME(DFHSM.RLS.MCDS2.DATA) -
     CONTROLINTERVALSIZE(12288)) -
   INDEX -
     (NAME(DFHSM.RLS.MCDS2.INDEX) -
     CONTROLINTERVALSIZE(4096))

 DEFINE CLUSTER (NAME(DFHSM.RLS.BCDS1) -
     STORAGECLASS(STRCLRLS) -
     CYLINDERS(2) NOIMBED NOREPLICATE -
     RECORDSIZE(200 2040) FREESPACE(0 0) -
     INDEXED KEYS(44 0) SHAREOPTIONS(3 3) -
     UNIQUE LOG(NONE)) -
   DATA -
     (NAME(DFHSM.RLS.BCDS1.DATA) -
     CONTROLINTERVALSIZE(12288)) -
   INDEX -
     (NAME(DFHSM.RLS.BCDS1.INDEX) -
     CONTROLINTERVALSIZE(4096))

 DEFINE CLUSTER (NAME(DFHSM.RLS.BCDS2) -
     STORAGECLASS(STRCLRLS) -
     CYLINDERS(2) NOIMBED NOREPLICATE -
     RECORDSIZE(200 2040) FREESPACE(0 0) -
     INDEXED KEYS(44 0) SHAREOPTIONS(3 3) -
     UNIQUE LOG(NONE)) -
   DATA -
     (NAME(DFHSM.RLS.BCDS2.DATA) -
     CONTROLINTERVALSIZE(12288)) -
   INDEX -
     (NAME(DFHSM.RLS.BCDS2.INDEX) -
     CONTROLINTERVALSIZE(4096))

 DEFINE CLUSTER (NAME(DFHSM.RLS.OCDS) -
     STORAGECLASS(STRCLRLS) -
     CYLINDERS(2) NOIMBED NOREPLICATE -
     RECORDSIZE(200 2040) FREESPACE(0 0) -
     INDEXED KEYS(44 0) SHAREOPTIONS(3 3) -
     UNIQUE LOG(NONE)) -
   DATA -
     (NAME(DFHSM.RLS.OCDS.DATA) -
     CONTROLINTERVALSIZE(12288)) -
   INDEX -
     (NAME(DFHSM.RLS.OCDS.INDEX) -
     CONTROLINTERVALSIZE(4096))
/*
//*****************************************************************/
//* COPY THE OLD CONTROL DATA SETS INTO THE NEWLY DEFINED          */
//* MULTICLUSTER CONTROL DATA SETS                                 */
//* NOTE: THE FROMKEY/TOKEY VALUES ARE ONLY SAMPLES.  THE ACTUAL   */
//* PARAMETERS USED FOR THESE KEYWORDS SHOULD BE DERIVED FROM      */
```

```
//* ACTUAL CDSS BEING USED.                                          */
//*******************************************************************/
//*
//STEP2    EXEC PGM=IDCAMS,REGION=512K
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSIN DD *
 REPRO INDATASET(DFHSM.MCDS) OUTDATASET(DFHSM.RLS.MCDS1) -
    FROMKEY(X'00') TOKEY(DFHSM)
 REPRO INDATASET(DFHSM.MCDS) OUTDATASET(DFHSM.RLS.MCDS2) -
    FROMKEY(DFHSN) TOKEY(DFHSM)
 REPRO INDATASET(DFHSM.BCDS) OUTDATASET(DFHSM.RLS.BCDS1) -
    FROMKEY(X'00') TOKEY(DFHSM)
 REPRO INDATASET(DFHSM.BCDS) OUTDATASET(DFHSM.RLS.BCDS2) -
    FROMKEY(DFHSN) TOKEY(DFHSM)
 REPRO INDATASET(DFHSM.OCDS) OUTDATASET(DFHSM.RLS.OCDS) -
/*
```

4.4.6 DCOLLECT

DCOLLECT has been changed so that it first attempts to open the CDSs in RLS mode. If the CDSs are not RLS eligible, OPEN error message IDC161I RC9 is issued. This message can be ignored for non-RLS-eligible data sets because the OPEN is retried for non-RLS mode. If both OPENs fail, a new DFSMShsm message is issued.

4.4.7 ARCIMPRT

The parameter passed to ARCIMPRT, the enhanced CDS recovery function utility, specifies which cluster of a multicluster CDS should be recovered.

If you are recovering a single-cluster CDS, the parameters are:

BCDS | BACKUPCONTROLDATASET

MCDS | MIGRATIONCONRTOLDATASET

OCDS | OFFLINECONTROLDATASET

If you are recovering a single cluster of a multicluster CDS, the parameters are:

BCDS*n* | MCDSCONTROLDATASET*n*

MCDS*n* | MIGRATIONCONRTOLDATASET*n*

where *n* is a value from 1 to 4. (The OCDS cannot be defined as a multicluster.)

4.4.8 CDS Backup

When DFSMShsm is active and accessing CDSs in RLS mode, *DFSMSdss* must be specified as the datamover for CDS backup. If directed to tape, the PARALLEL parameter must also be specified. If either of these conditions is not met during automatic CDS version backup, these values will override the existing values, and an informational message will be issued. If either of these conditions is not met when BACKVOL CDS is issued, the command will fail.

4.4.9 Migration

Changes to the DFSMShsm CDSs and the MVS/ESA or OS/390 systems are required before the CDSs can be accessed in RLS mode.

4.4.9.1 Required CDS Changes

Before the CDSs can be accessed in RLS mode, they must be defined or altered to be RLS eligible. The following required changes must be made to all three CDSs (BCDS, MCDS, OCDS), or DFSMShsm will fail at startup. All three CDSs are accessed in the same mode, either RLS or non-RLS.

- The CDSs must be defined or altered with the LOG(NONE) attribute.

  Example: **ALTER** cdsname **LOG(NONE)**

  Do not use the ALL or UNDO parameter of LOG.

  To change the CDSs back to non-RLS eligible, use the ALTER *cdsname* NULLIFY(LOG) command.

- The CDSs must be SMS managed to use RLS. The SMS configuration must include a storage class definition that indicates which coupling facility to use.
- All operating systems running with DFSMShsm must be coupling facility capable (MVS/ESA SP Version 5.2 and up), and the processors must have access to the coupling facility.
- The CDSs cannot be defined as key-range data sets. If the CDSs were previously defined as multicluster, they used key ranges and must be redefined without key ranges. DFSMShsm dynamically calculates the key boundaries used for the redefined CDSs.

4.4.9.2 Required CDS Backup and Datamover Changes

If CDS backup will be invoked while the CDSs are accessed in RLS mode, DFSMSdss must be the specified datamover. If the backup is directed to tape, the PARALLEL parameter of SETSYS CDSVERSIONBACKUP(BACKUPDEV ICECATEGORY) must be specified.

4.4.10 Coexistence

When one system accesses the DFSMShsm CDSs in RLS mode, all systems in the hsmplex must access the CDSs in RLS mode.

4.4.10.1 Processing Concurrently at DFSMShsm V1R4 Level

DFSMShsm fails at startup if the selected serialization technique of the starting DFSMShsm differs from the serialization technique of other DFSMShsm systems already active in the hsmplex.

4.4.10.2 Processing Concurrently with a Downlevel Version of DFSMShsm

DFSMShsm fails at startup on a downlevel system if an active DFSMShsm ViR4 system is accessing the CDSs in RLS mode. A toleration PTF must be installed for this support.

DFSMShsm fails at startup on a downlevel system if the CDSs are defined by using multicluster dynamic key boundaries. No toleration PTF is needed for this, as the downlevel system determines that the CDSs are invalid and fails startup with the ARC0900I RC8 message.

4.5 DFSMShsm and DFSMSrmm Interface

With DFSMS/MVS V1R4, DFSMShsm can now communicate with DFSMSrmm directly without exits. It no longer requires the exits to send requests to DFSMSrmm tape management. Thus the ARCTVEXT exit is free and other tape management products can use it.

4.5.1 Current Processing

DFSMSrmm ships source code and object code for ARCTVEXT. The DFSMSrmm-supplied code links to the DFSMSrmm programming interface, EDGDFHSM, to notify DFSMSrmm of tapes being released from DFSMShsm.

EDGDFHSM checks whether DFSMSrmm is in use and issues the EDG8021E message warning that DFSMSrmm code is in use on the system when it should not be. If DFSMSrmm should be active, it issues a WTOR, EDG8011D, to make DFSMSrmm active.

4.5.2 New Processing

DFSMSrmm no longer ships source code and object code for ARCTVEXT. Instead, a new general-use programming interface, EDGTVEXT, is provided, which is called from DFSMShsm with the same parameter list as ARCTVEXT. The EDGTVEXT is basically the same code as the old DFSMSrmm-supplied ARCTVEXT but is now OCO and must use DFSMSrmm macros and check whether DFSMSrmm is in use. If DFSMSrmm is not in use, it no longer calls EDGDFHSM.

DFSMShsm always invokes EDGTVEXT before it determines whether the call to ARCTVEXT is to be made. The ARCTVEXT exit is still invoked based on the SETSYS EXITON() / EXITOFF() command. The EDGTVEXT general-use programming interface frees ARCTVEXT from DFSMSrmm usage, so you can use ARCTVEXT for other tape management systems.

4.5.3 Running DFSMShsm with DFSMSrmm

You do not use the ARCTVEXT installationwide exit to communicate between DFSMShsm and DFSMSrmm. DFSMShsm automatically calls DFSMSrmm to process tapes that are to be returned to the DFSMShsm tape pool or deleted from DFSMShsm. If DFSMSrmm is not in use, the DFSMSrmm code returns to DFSMShsm taking no action.

4.5.4 EDGDFHSM Program Interface

DFSMSrmm uses the EDGDFHSM program interface to release DFSMShsm tape volumes, but you do not have to take any action to activate or call it. The interface between DFSMShsm and DFSMSrmm is automatically in place when you install DFSMS/MVS V1R4.

4.5.5 EDGTVEXT

DFSMSrmm does not use the DFSMShsm tape volume exit, ARCTVEXT, to manage tapes that DFSMShsm uses. It has its own interface from DFSMShsm to avoid conflicts over use of the ARCTVEXT installationwide exit.

If you have a product with similar requirements for releasing tapes from DFSMSrmm as DFSMShsm, you can use either the EDGDFHSM or EDGTVEXT

program interface. The difference between EDGDFHSM and EDGTVEXT is that EDGTVEXT accepts the ARCTVEXT parameter list, and the EDGDFHSM interface accepts only the first volume in the ARCTVEXT parameter list.

EDGTVEXT is a callable programming interface to DFSMSrmm that is used from DFSMShsm, or you can use it from any other APF-authorized program that needs to obtain the same service as the DFSMShsm ARCTVEXT exit. To run DFSMShsm with DFSMSrmm, DFSMShsm must be defined to RACF. The DFSMShsm user ID should be other than the default user ID and should be defined in the started procedures table, ICHRIN03, or with the RACF 2.1 STARTED class.

4.5.5.1 Invocation

EDGTVEXT can be invoked from either the LOAD, CALL, or LINK macro.

4.5.5.2 Input

The input is a parameter list that describes the volumes DFSMShsm is releasing and the actions required. The parameter list is identical to the list DFSMShsm passes to ARCTVEXT.

On entry, register 1 contains a pointer to the ARCTVEXT parameter list.

On entry, register 13 contains the address of a standard 18-word save area, and register 14 contains the return address.

4.5.5.3 Output

EDGTVEXT and EDGDFHSM issue messages when problems are encountered. EDGTVEXT always sets the ARCTVEXT return code value to zero. It does not always pass a zero register 15 return code back to the caller. A nonzero return code is the register 15 return code from the subsystem request attempted by EDGDFHSM. You can obtain information about the return codes in *MVS/ESA SP Version 5 Using the Subsystem Interface*.

EDGTVEXT ensures that DFSMSrmm is in use on your system before continuing with DFSMSrmm processing. If DFSMSrmm is not in use, it sets return code zero and returns to its caller. If DFSMSrmm is in use or should be in use, it calls EDGDFHSM to process the volumes passed to it in the ARCTVEXT parameter list.

4.5.5.4 Environment

EDGTVEXT must be link edited in an APF-authorized library. It runs in AMODE(31) RMODE(ANY).

4.5.5.5 Migration

There are migration concerns to consider if you invoked ARCTVEXT for DFSMSrmm support. The DFSMSrmm code that was previously shipped in ARCTVEXT exit must be removed, or DFSMSrmm could be invoked twice by DFSMShsm to remove the tape volume from its inventory. If the ARCTVEXT exit was only invoked with the previously shipped DFSMSrmm ARCTVEXT code, turn off the ARCTVEXT exit with the SETSYS EXITOFF(TV) command. ADSM customers who previously invoked the ARCTVEXT exit to use DFSMSrmm services should now invoke the new DFSMSrmm general-use programming interface, EDGTVEXT, directly from the ADSM installationwide deletion exit. Change the DELETIONEXIT option from ARCTVEXT to EDGTVEXT in the MVS server options file. For more information about the ADSM deletion exit, refer to the *ADSTAR Distributed Storage Manager for MVS/ESA Administrator's Guide*.

4.6 Dump Analysis Elimination

Dump analysis elimination (DAE) is an MVS function that eliminates duplicate dumps in MVS systems or across MVS systems.

DFSMShsm generates dumps to gather first-failure diagnostic information. Problems that occur multiple times, perhaps on different hosts, typically generate a storage dump. The initial dump is helpful for diagnosing the problem; additional dumps for the same problem usually are not needed.

Controlling DFSMShsm storage dumps is a major issue with customers; the ability to properly suppress duplicate dumps is an important step toward successfully managing dump data sets. DFSMShsm supplies additional information to the DAE function to build a symptom string to be used to suppress duplicate dumps for functions that run in the same or different systems.

4.6.1 Setup Requirements

The setup requirements are:

- *SETSYS SYS1DUMP* must be used. (SYS1DUMP is the DFSMShsm default.)
- SYS1.PARMLIB member ADYSETxx must be coded with the keyword *SUPPRESSALL*, for example:

```
DAE=START,RECORDS(400),
    SYSMDUMP(MATCH,UPDATE),
    SVCDUMP(MATCH,UPDATE,SUPPRESSALL)
```

- MVS/ESA SP V4.3 introduced the ability to use DAE in a multihost environment, thereby allowing a single DAE data set to be shared across systems in a sysplex. The coupling services of the cross system coupling facility (XCF) and GRS must be enabled for the DAE data set to be shared in a sysplex environment and for dumps to be suppressed across MVS systems.

4.6.2 Programming Systems

The applicable product release level is DFSMShsm V1R4.

Only those hosts with DFSMShsm V1R4 will use the DAE suppression for dumps for the DFSMShsm product.

DFSMShsm V1R4 uses DAE for functions that run in both the primary address space and the ABARS secondary address space.

4.6.3 Dumps Not Eligible

DAE does not suppress SYSABEND, SYSUDUMP, SYSMDUMP, or SNAP dumps or dumps that originate from SLIP or DUMP operator commands. Because these dumps are taken only on demand, suppression is not desirable.

This support does not apply to dumps produced by DFSMShsm as a result of the TRAP command.

4.6.4 DFSMShsm Processing in the Primary Space

For dumps taken in the primary address space that are eligible, DFSMShsm will fill in additional fields in the SDWA to build a symptom string before taking a dump. The following variables are filled in by DFSMShsm:

- Load module name
- Csect name
- Name of the recovery routine
- DFSMShsm component identifier
- DFSMShsm component base number (prefix)

The DAE function uses the symptom string to decide whether the dump requested is a duplicate dump.

4.6.5 DFSMShsm Processing in the ABARS Secondary Address Space

For dumps taken in the secondary address space that are eligible, DFSMShsm uses the SYMREC parameter on the SDUMP macro that requested the dump to build a symptom string. The parameter contain the following information:

- Load module name for the secondary address space
- Abend code
- Csect name of failing module
- Offset in failing module
- Name of the recovery routine for the secondary address space
- DFSMShsm component identifier
- DFSMShsm component base number (prefix)

The DAE function uses the symptom string to decide whether the dump requested is a duplicate dump.

4.7 Serviceability Support Items

DFSMShsm V1R4 includes several seviceability items. They provide improved reporting ability and facilitate DFSMShsm command use.

4.7.1 Four New SMF Functional Statistical Record Types

Four new SMF FSR subtypes have been created for the following events:

- Type 17 - Expire data set (from L0, ML1, and ML2)
- Type 18 - Partial release data set on L0 DASD
- Type 19 - Expire incremental backup version
- Type 20 - Delete incremental backup version

Two FSRs are now a subtype 17 instead of a subtype 6. Thus record subtype 6 can be written only for user-requested migrated data set deletes. FSRs are created when the scheduled delete of the migrated data set takes place. An ARC0734I message is issued when the delete is scheduled for:

- Expire data set from ML1
- Expire data set from ML2

There is no change from the current processing.

Two new bits have been added to one of the flag bytes in the FSR mapping to indicate whether the data set that is being expired is from ML1 or ML2. If neither of these bits is set, the data set being expired is from L0. Thus you can map the expired data set FSR subtype 17 to the different volume categories of L0, ML1, or ML2.

Three new bits have been added to the flag bytes in the FSR mapping to indicate:

- The data set is being deleted by its expiration data in the catalog or by the management class attributes
- Whether the incremental backup version is being deleted by the EXPIREBV command
- Whether the incremental backup version being deleted is on a tape volume

There may be migration concerns for the conversion of two FSRs (expire from ML1 and expire from ML2) from a subtype 6 to a subtype 17. If the SMF subtype 6 information was previously collected, the values will change because the expires from ML1 and ML2 are no longer included. To continue to collect the information for expires from ML1 and ML2, subtype 17 must be collected.

4.7.2 Aliases for DFSMShsm Keywords

DFSMShsm V1R4 introduces four short aliases for long keywords. This enhancement will simplify and shorten commands.

The following changes have been made:

- BACKUP can be used as an alias for BACKUPCONTROLDATASET on the LIST and the HLIST commands. The BCDS alias that exists today will be kept.

  For example, the command

  ```
  LIST LEVEL(qualifier) BACKUP
  ```

  will give you the backup version of all data sets with the initial characters of this *qualifier*. A command using the DSNAME option instead of the LEVEL option also applies:

  ```
  LIST DSNAME(datasetname) BACKUP
  ```

- MIGRAT can be used as an alias for the MIGRATIONCONTROLDATASET on the LIST and HLIST commands. The MCDS alias that exists today is still valid. The LIST LEVEL and the LIST DSNAME commands are also applicable to the MIGRAT alias.
- The abbreviations UUT and NOUUT can be used, respectively, for USERUNITTABLE and NOUSERUNITTABLE on the SETSYS command.

4.7.3 AUDIT DATASETCONTROLS

The granularity of the AUDIT DATASETCONTROLS restart (RESUME) function has been improved by saving all 44 characters of the data set name.

4.7.4 DFSMS/MVS Optimizer HSM Monitor/Tuner

Currently DFSMShsm communicates with the Optimizer Monitor/Tuner through the DFSMShsm initialization exit ARCINEXT. In DFSMShsm V1R4, DFSMShsm directly invokes GFTEMINX. It utilizes a new interface, GFTEMSDX, to the DFSMS/MVS Optimizer component.

DFSMShsm always invokes GFTEMINX during startup. If the invocation fails, DFSMShsm continues without any error messages. The ARCINEXT exit is still invoked based on the SETSYS EXITON() / EXITOFF() command. Thus the ARCINEXT is freed from Optimizer use, so you can use ARCINEXT for other reasons.

If the invocation of the GFTEMINX is successful, GFTEMSDX is always be invoked during DFSMShsm shutdown. If the GFTEMSDX invocation fails, DFSMShsm continues without any error messages.

There are migration concerns to consider if ARCINEXT was previously invoked for Optimizer support. The code that invokes the GFTEMINX exit must be removed from the ARCINEXT installationwide exit; otherwise the Optimizer GFTEMINX exit could be invoked twice during startup. If the ARCINEXT exit was invoked only for the Optimizer GFTEMINX exit, turn off the ARCINEXT exit with the SETSYS EXITOFF(IN) command.

Chapter 5. DFSMSdss Multivolume Selection Enhancements

The DFSMSdss multivolume selection changes affect logical data set DUMP, data set COPY, and CONVERTV processing. The support adds a new selection keyword, SELECTMULTI(ALL|ANY|FIRST). This keyword replaces and improves on the current ALLMULTI keyword, which will become obsolete.

For logical data set dump, data set copy, and CONVERTV processing, the new keyword enables a multivolume data set to be selected according to whether the volume list includes all of the volumes on which the data set resides (ALL), some of the volumes (ANY), or at least the volume that contains the first extent (FIRST).

Current DFSMSdss jobs that use the ALLMULTI keyword will continue to run as before with the specification of ALLMULTI mapping to SELECTMULTI(ANY), unless CONVERTV to SMS is specified. In this case ALLMULTI will map to SELECTMULTI(FIRST). If neither ALLMULTI nor SELECTMULTI is specified, the default will be SELECTMULTI(ALL).

5.1 ALLMULTI

The ALLMULTI keyword can cause multiple dumps of multivolume data sets to be taken. Consider the situation where you have two jobs, one that dumps VOL001 and VOL002, and the other that dumps VOL003. A large multivolume data set is stored on VOL001 and VOL003. When both jobs include ALLMULTI, the multivolume data set is included in the dump for volumes VOL001 and VOL002 as well as in the dump for VOL003 (see Figure 9 on page 74).

Although this limited case can be ″handled″ by changing the volumes that the jobs reference, in a real-life environment, this is often impossible to achieve. To avoid missing any multivolume dumps, most installations using logical dumps code ALLMULTI on each DFSMSdss job.

A more realistic example illustrates the difficulty in obtaining a single dump of a data set when performing systemwide, logical dumps for disaster recovery purposes:

- Eighty volumes are to be dumped as fast as possible to eight tape drives.
- Eight DFSMSdss jobs are to run.
- Each job is to reference 10 volumes.
- The installation makes extensive use of multivolume data sets, to the extent that the minimum set of volumes required to ensure 100% coverage of all the multivolume data sets is 80.

In the above situation, to ensure that each data set is dumped once, only one DFSMSdss job can be run, which is impractical because:

- The total dump will be excessively long in terms of elapsed time.
- The total restore time will be unacceptably long in terms of restoring all of the data within a disaster recovery window.

If multiple jobs are run:

- The jobs will take longer than they need to because of the multiple dumps of the same data set.
- There will be errors during the restore: The second and subsequent restores of a data set will because of a ″duplicate data set″ condition.

DFSMSdss users want to select data sets during a ″logical volume dump″ according to the following criteria:

- The entire data set, if the first primary extent is found on any volume that is referenced, even though not all volumes are on the list.
- The entire data set, if any part of the data set is found on any volume that is referenced, even though all of the volumes are not on the list.

The first method of selection is what most DFSMSdss users really want and need, but DFSMSdss until now has only supported the second method through the ALLMULTI keyword.

5.2 SELECTMULTI

The SELECTMULTI keyword, is applicable during logical data set DUMP, data set COPY, and conversion of volumes to or from SMS management. It is a functional replacement for the ALLMULTI keyword. Because SELECTMULTI is meaningful only when there is a volume list from which to select, SELECTMULTI, like ALLMULTI, requires that either LOGINDDNAME or LOGINDYNAM be specified. If SELECTMULTI is present but both LOGINDDNAME and LOGINDYNAM are absent, existing message ADR138E is issued.

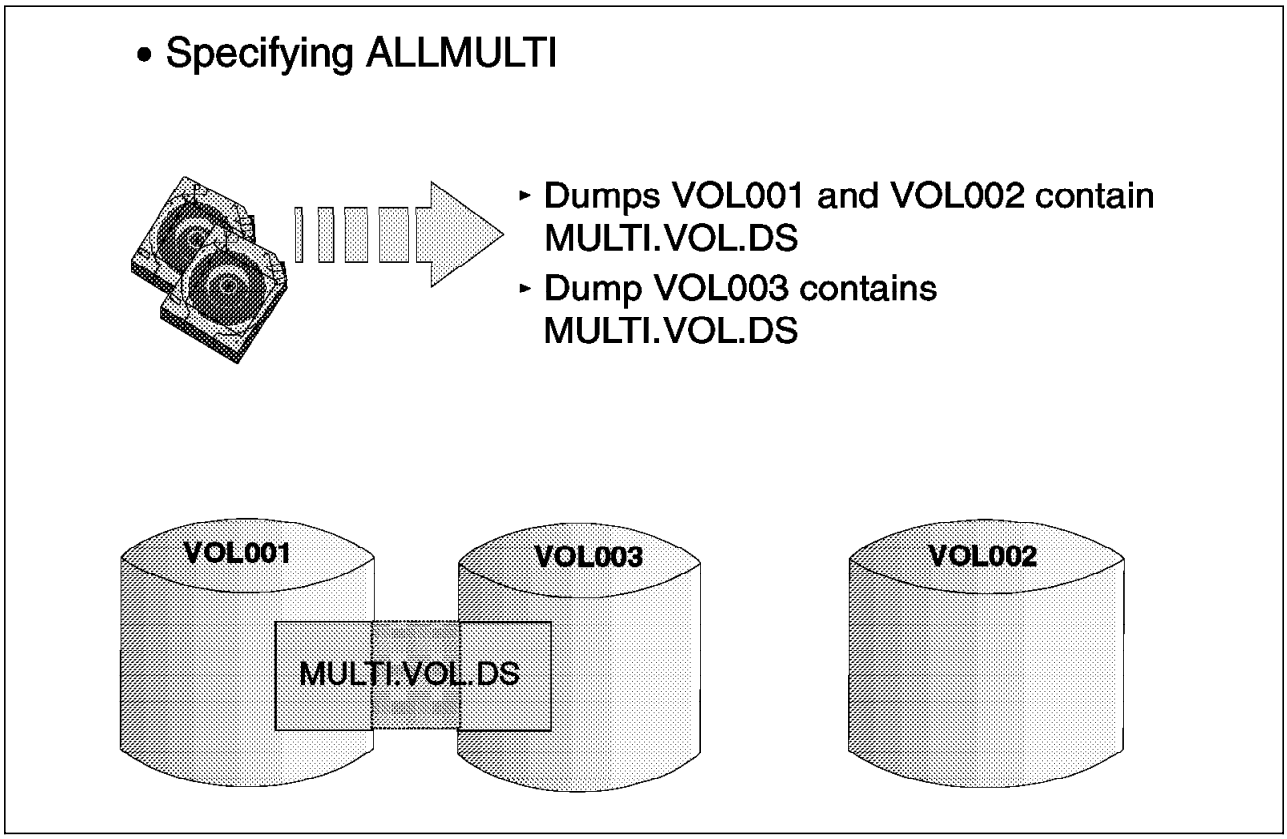


Figure 9. Problems with ALLMULTI. When two DFSMSdss jobs are run, one to dump VOL001 and VOL002, and the other to dump VOL003, with both specifying ALLMULTI, the large multivolume data set is dumped twice.

5.2.1 SELECTMULTI for DUMP and COPY

SELECTMULTI(ALL|ANY|FIRST) specifies the method for determining how cataloged multivolume data sets are to be selected during a logical data set dump or copy operation. SELECTMULTI is accepted only when logical volume filtering is also used. Logical volume filtering is used when LOGINDDNAME or LOGINDYNAM is specified during dump or copy. If logical volume filtering is not used, the specification of SELECTMULTI is not accepted.

ALL is the default and indicates that DFSMSdss is not to dump or copy a multivolume data set unless the volume list specified by LOGINDDNAME or LOGINDYNAM lists all volumes that contain a part of the data set or VSAM cluster. For VSAM data sets:

- If SPHERE is not specified, the data set in this context is any cluster, base, or AIX.
- If SPHERE is specified, all parts of the base cluster and all AIXs must be contained in the volume list.

ANY indicates that DFSMSdss is to dump or copy a multivolume data set when any volume in the volume list specified by LOGINDDNAME or LOGINDYNAM contains a part of the data set or VSAM cluster. For VSAM data sets:

- If SPHERE is not specified, the data set in this context is any cluster, base, or index.
- If SPHERE is specified, any part of the base cluster or any part of any associated AIX may be contained in the volume list.

FIRST indicates that DFSMSdss is to dump or copy a multivolume data set only when the volume list specified by LOGINDDNAME or LOGINDYNAM lists the volume that contains the first part of the data set. For VSAM data sets:

- If SPHERE is not specified, the volume list must include either the volume containing the first extent of the data component for the base cluster or the volume containing the first extent of the data component of an AIX.
- If SPHERE is specified, the volume list must include the volume containing the first extent of the data component for the base cluster.

5.2.2 SELECTMULTI for CONVERTV

SELECTMULTI(ALL|ANY|FIRST) specifies the method for determining how cataloged multivolume data sets are to be selected during conversion to or from SMS management. The volume list is the list of volumes supplied by the DDNAME or DYNAM keyword:

ALL is the default and indicates that DFSMSdss is not to process a multivolume data set unless the volume list specified by DDNAME or DYNAM lists all volumes that contain a part of the data set or VSAM sphere.

ANY indicates that DFSMSdss is to process a multivolume data set when any part of the data set or VSAM sphere is on a volume in the volume list specified by DDNAME or DYNAM.

FIRST indicates that DFSMSdss is to process a multivolume data set only when the volume list specified by DDNAME or DYNAM lists the volume that contains the first part of the data set or the primary data component of the base cluster for a VSAM sphere.

5.2.3 Using the ADRUIXIT Exit

The setting of the SELECTMULTI keyword can be overridden by the DFSMSdss installationwide Exit, ADRUIXIT, the Options Exit Routine, which is described in *DFSMS/MVS V1R3 Installation Exits*.

5.2.4 Interactions

The specification of SELECTMULTI(ANY) has the same effect as the specification of ALLMULTI on previous levels of DFSMSdss except when you convert a volume to SMS. When converting a volume to SMS, specify SELECTMULTI(ANY) to select a data set for processing if any part of the data set is on a volume in the volume list.

The specification SELECTMULTI(ALL) has the same effect as the absence of the specification of ALLMULTI on previous levels of DFSMSdss.

Specifying SELECTMULTI(FIRST) enables you, for logical data set DUMP, data set COPY, and CONVERTV to non-SMS, to select a data set for processing only when the first volume on which it resides is included in the volume list. The specification of SELECTMULTI(FIRST) has the same effect as the specification of ALLMULTI on previous levels of DFSMSdss when you convert a volume to SMS.

5.2.5 Errors

- If you specify ALLMULTI but not SELECTMULTI, SELECTMULTI(ANY) is set, and a new message, ADR146I, is issued.
- If you specify both ALLMULTI and SELECTMULTI(xyz), SELECTMULTI(xyz) is set, and message ADR146I is issued.
- If you alter the setting of ALLMULTI, either from OFF to ON or from ON to OFF, the SELECTMULTI(xyz) set by the keyword is not altered, and a new message, ADR147W, is issued.
- If the installation exit, ADRUIXIT, altered the setting of SELECTMULTI(xyz), whether set by keyword or by default, the setting is altered and existing message ADR035I is issued.

5.2.6 New Messages

The following messages are added in support of the SELECTMULTI changes:

ADR146I *(xxx)-mmmmmm(yy),* OBSOLETE KEYWORD *keyword1* SPECIFIED. *keyword2* WILL BE USED

ADR147W *(xxx)-mmmmmm(yy),* INSTALLATION EXIT ATTEMPTED TO ALTER OBSOLETE KEYWORD *keyword.* REQUEST IGNORED.

ADR148I *(xxx)-mmmmmm(yy),* MULTIVOLUME DATA SET *dsname* NOT SELECTED.

ADR732E *(xxx)-mmmmmm(yy),* DATA SET *dsname* ON VOLUME *volume_serial_number* WAS NOT SELECTED BECAUSE IT WAS NOT CATALOGUED.

ADR878E *(xxx)-mmmmmm(yy),* THE FOLLOWING DATA SETS ON VOLUME *volume_serial_number* {WILL NOT BE | WERE NOT} SUCCESSFULLY PROCESSED *dsname* CATALOG: *catalog_name*

ADR886E *(xxx)-mmmmmm(yy),* DATA SETS EXIST ON VOLUME *volume_serial_number* WHICH ARE NOT CONVERTIBLE {TO | FROM} SMS MANAGEMENT, THE VOLUME {WILL BE | WAS} {PLACED | LEFT} IN INITIAL STATUS

5.2.7 Migration and Coexistence

After you install DFSMS/MVS V1R4, you must change existing jobs to take advantage of the multivolume selection enhancements in DFSMSdss. There are coexistence considerations if the SELECTMULTI keyword is used in a job on a system with a prior release of DFSMS/MVS installed.

5.2.7.1 Migration Considerations

To select a data set for processing when only the volume containing its first extent is in the volume selection list, you have to change existing jobs to add SELECTMULTI(FIRST).

Specification of ALLMULTI maps to the functionally equivalent SELECTMULTI(ANY) unless CONVERTV to SMS is specified, in which case ALLMULTI maps to SELECTMULTI(FIRST) so that you are not forced to change existing jobs.

Absence of the specification of ALLMULTI maps to the functionally equivalent SELECTMULTI(ALL) without forcing you to change existing jobs.

Warning message ADR147W is issued when the installation exit attempts to modify the settings of ALLMULTI.

5.2.7.2 Coexistence Considerations

The SELECTMULTI keyword is not supported on releases previous to DFSMS/MVS V1R4.

No other incompatibilities exist. Data sets dumped on releases that do not support the SELECTMULTI keyword can be restored with DFSMSdss VR40. Use of the new keyword during dump with DFSMSdss V1R4 does not introduce any incompatibility during restore with a previous release.

No toleration PTFs are expected to be released for this announcement. The only potential issue is in a shared spool environment with processors or LPARs running different levels of DFSMS/MVS. Jobs executing DFSMSdss using the new SELECTMULTI keyword must not be allowed to run on levels of DFSMS/MVS previous to DFSMS/MVS V1R4.

5.2.8 Support Use Information

DFSMSdss manuals that describe or give examples of using the ALLMULTI keyword have changed all descriptions and examples to use the SELECTMULTI keyword together with the appropriate subparameter.

5.2.9 Performance

This enhancement to DFSMSdss does not change the performance of any function. However, more efficient selection may result in a shorter elapsed time for a group of data set dumps where the work is spread across multiple jobs. A shorter elapsed time is possible because in many circumstances the same multivolume data set could have been dumped multiple times with levels of code before DFSMS/MVS V1R4.

5.2.10 Resources

No new control blocks have been added or deleted as a result of the multivolume selection enhancement announcement. The sizes of existing control blocks are not changed, and the announcement does not change the DFSMSdss storage requirements.

Chapter 6. DFSMSrmm

In this chapter we cover the following announcements and functional enhancements for DFSMSrmm:

- Journal usage threshold
- Nonintrusive backup of the CDS
- Problem determination aid (PDA) trace
- Support for DFSMShsm alternate tape processing
- Recognition of external data managers
- Inventory management trial runs

Note: Some of the enhancements are not unique to DFSMS/MVS V1R4. They are currently available through SPEs or APARs but are included in this chapter for completeness.

6.1 Journal Usage Threshold

The DFSMSrmm journal data set contains a chronological record of changes to the DFSMSrmm control data set. It allows the DFSMSrmm control data set to be forward recovered. Before the journal usage threshold was available, the journal could fill up unexpectedly.

As shown in Figure 10 on page 80, DFSMSrmm can now display information messages to the operator, indicating how full the DFSMSrmm journal is becoming. The first message to be displayed (EDG2107E) indicates that the usage threshold, specified in the DFSMSrmm PARMLIB member, for this journal has been reached. This message can be used to act as a trigger for a procedure to be run automatically to back up both the control data set and journal and then clear the journal. Subsequent messages are issued as the journal becomes ever closer to becoming filled.

Customers have become concerned that they are not informed when the journal is becoming full. When the journal is full, all tape processing must stop until the journal and the control data set have been backed up and the journal cleared. This procedure is disruptive and could involve a tape processing outage.

6.1.1 Update PARMLIB Member EDGRMMxx

Specify the JOURNALFULL threshold and BACKUPPROC parameters in PARMLIB member EDGRMMxx on the OPTION command as follows:

6.1.1.1 JOURNALFULL(nn)

Specify JOURNALFULL to define a percentage-full threshold for the journal data set. When DFSMSrmm detects that the journal has reached this threshold, it issues message EDG2107E. DFSMSrmm also issues message EDG2107E at DFSMSrmm startup if the journal has already reached the threshold specified. If you specify a value of 0, DFSMSrmm issues no warnings on that system. Different threshold values can be specified on systems sharing the RMM control data sets.

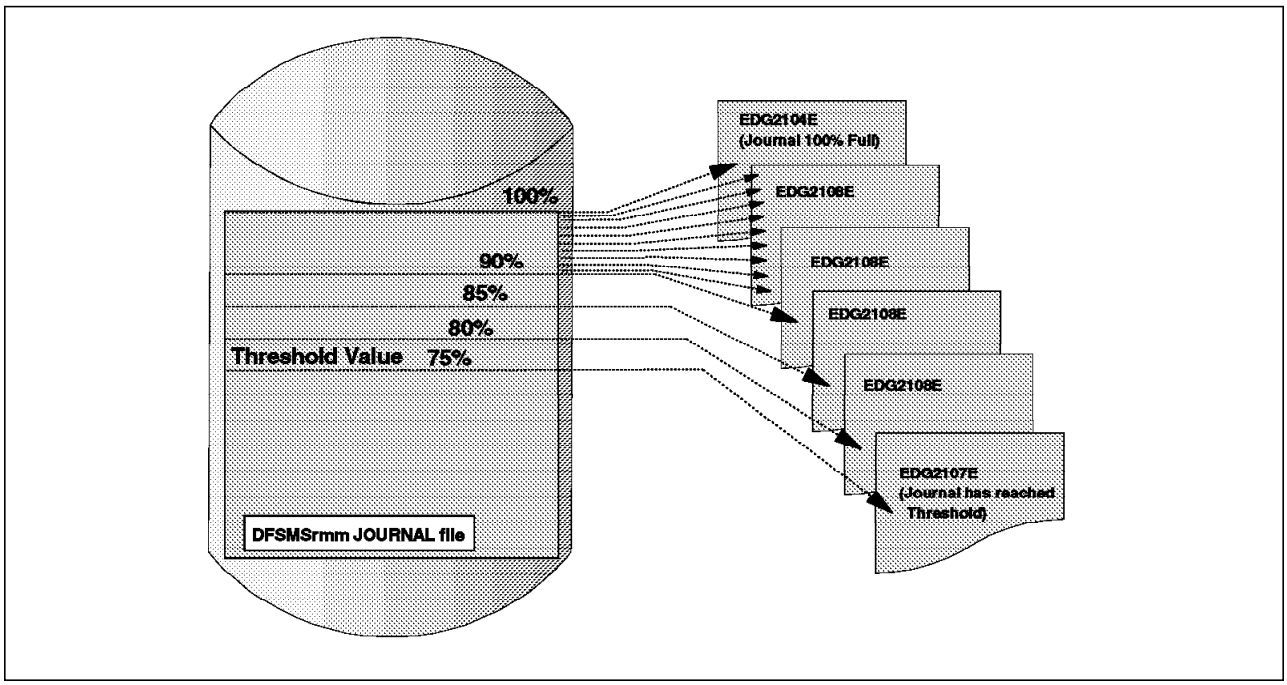


Figure 10. Journal Usage Threshold. Messages are produced when the threshold is reached and then every 5% until the 90% threshold is reached. Messages are then produced for every additional 1% usage.

Note: A backup procedure can be started automatically when the journal percentage full threshold is reached. See **BACKUPPROC** for further information. Specify a value in the 0-99 range. The default is 75.

6.1.1.2 BACKUPPROC(procname)

BACKUPPROC specifies the name of the procedure that you want started automatically when the journal percentage full threshold is reached.

Specify a valid alphanumeric procedure name from one to eight characters. There is no default value. If you do not specify a name, an automatic start command is not issued.

To find out the setting of the threshold and the current journal utilization, either issue this command:

```
RMM LISTCONTROL CNTL OPTION
```

or use the DFSMSrmm ISPF panels to show the System Options Panel Display.

If a journal is not allocated, the journal is disabled, or the utilization is less than 0.5%, a value of zero is returned for the current utilization value.

6.1.2 Automating Control Data Set Backup and Journal Clearing

You can use DFSMSrmm to automate the backup of the control data set and the clearing of the journal. Although you can continue to run regular backups during inventory management, automating control data set backup and clearing the journal help ensure that the journal does not fill up. A full journal can impact tape usage on your system.

To automate control data set backup and clearing of the journal you can:

- Specify a value for the JOURNALFULL operand on the OPTION command in PARMLIB member EDGRMMxx as described in 6.1.1.1, “JOURNALFULL(nn)” on page 79 or use the default value of 75%.
- Specify a procedure name for the BACKUPPROC operand as described in 6.1.1.2, “BACKUPPROC(procname)” on page 80. The submission of this procedure is triggered when message EDG2107E is issued.

A procedure with the same name as specified on the BACKUPPROC parameter must be stored in the system procedure library.

Figure 11 shows a sample procedure for backing up the control data set and clearing the journal.

```
//EDGHSKP  EXEC PGM=EDGHSKP,PARM='BACKUP(DSS)'
//MESSAGE  DD  DISP=SHR,DSN=RMM.MESSAGE
//SYSPRINT DD  SYSOUT=*
//BACKUP   DD  DISP=(,CATLG),UNIT=TAPE,DSN=BACKUP.CDS(+1),
//             LABEL=(,SL)
//JRNLBKUP DD  DISP=(,CATLG),UNIT=TAPE,DSN=BACKUP.JRNL(+1),
//             LABEL=(2,SL),VOL=REF=*.BACKUP
//DSSOPT   DD  *
   CONCURRENT OPTIMIZE(1) VALIDATE
/*
```

Figure 11. CDS and Journal Backup and Clearing

Notes:

1. The names of the control data set and journal are obtained from the active DFSMSrmm subsystem.
2. EDGHSKP backs up and clears the data set. If the RMM subsystem is not active, you must use EDGBKUP, which does not clear the journal.
3. This example shows how DFSMSdss and DFSMSrmm work together to back up the CDS and journal using concurrent copy; see 6.2, “Nonintrusive Backup of the CDS” on page 82 for further information.

6.1.3 Messages

The following messages are displayed in relation to journal threshold processing:

EDG2103E PERMANENT JOURNAL ERROR - REPLY ″R″ TO RETRY, ″I″ TO IGNORE, ″D″ TO DISABLE, OR ″L″ TO LOCK

- Look for a previous message with the EDG prefix that shows the error.
- Notify your system programmer.

EDG2104E JOURNAL FILE IS FULL - SCHEDULE CONTROL DATA SET BACKUP TO CLEAR IT

- Manually start the DFSMSrmm backup job to reset journal.
- Notify your system programmer.
- There is no reply for this message. This message is followed by message EDG2103D, to which you must reply.

EDG2107E JOURNAL THRESHOLD REACHED - JOURNAL IS *percentage_value%* FULL. *tracks* TRACKS(*kilobytes*K) AVAILABLE

- The journal has reached the specified threshold value. If an autostart procedure for backup is defined, RMM starts it automatically. Otherwise follow your installation-defined backup procedure.
- Notify your system programmer.

EDG2108E JOURNAL IS *percentage_value%* FULL. *tracks* TRACKS (*kilobytes*K) AVAILABLE

- This message is issued for every additional 5% full, or every 1% once over 90% full. If a backup procedure has not been started, follow your installation-defined backup procedure.
- Notify your system programmer.

6.1.4 TSO Subcommand Variables by Name

The variables in Figure 12 are available and can be used in REXX procedures for managing the journal if required. For example a REXX procedure might be developed that would display the current percentage usage of the journal.

Variable Name	Subcommand	Content	Format
EDG@JDS	LC	Journal name	44 characters
EDG@JRNF	LC	JOURNALFULL PARMLIB operand value	Numeric 0-99
EDG@JRNU	LC	Journal percentage used	Numeric 0-100

Figure 12. DFSMSrmm REXX Variables for the Journal

6.1.5 Migration and Coexistence

There are no migration or coexistence issues concerning the JOURNALFULL operand and journal space utilization support.

6.2 Nonintrusive Backup of the CDS

The standard method of backing up DFSMSrmm′s control data set is to use the DFSMSrmm backup utilities instead of other backup utilities, such as Access Method Services EXPORT, because DFSMSrmm provides the necessary serialization and forward recovery functions. DFSMSrmm backup utilities check whether the control data set is in use, tell the DFSMSrmm subsystem that backup or recovery is in process, and provide a way to forward recover using the DFSMSrmm journal data sets.

Serialization prevents updates from occurring to the control data set and journal until the backups are complete, thus ensuring the integrity of the backups so that recovery is possible. While the backups are taking place, however, any tape

processing involving updates to the DFSMSrmm CDS or journal has to wait until the backups are complete. If the control data set is large, this outage could last for an unacceptably long time.

DFSMSrmm now allows an installation to use concurrent copy to minimize the outage. With using concurrent copy, the backups are logically complete in just a few seconds, after which normal DFSMSrmm tape management functions can continue while the physical dumps take place. Concurrent copy enables nonintrusive backup of the DFSMSrmm control data set as follows:

- The control data set and journal can be backed up without impact on tape use.
- Updates resulting from tape use or from commands are permitted when backup is in progress and do not affect the integrity of the resulting backup copy.
- Backup may now be made directly to tape.

  Note: This is true even if concurrent copy is not being used.

6.2.1 Functional Characteristics

The backup utilities of DFSMSrmm have been enhanced such that DFSMSdss can be invoked instead of Access Method Services. This is achieved by coding BACKUP(DSS) in the parm field of the JCL's EXEC statement, and by optionally coding a DSSOPT DD statement. The DSSOPT statement allows the installation to code any appropriate parameters that would normally be coded on the DFSMSdss DUMP and RESTORE control cards. The DSSOPT DD statement can be used to specify customized operands for the DUMP and RESTORE commands used by DFSMSrmm. You might customize the operands according to the media used for the backup and the resources that are available. The operands specified with the DSSOPT DD statement override the default DFSMSdss DUMP and RESTORE commands issued by DFSMSrmm. DFSMSrmm reads all the records specified with the DSSOPT DD statement and uses them to replace its default command operands beginning at the second line.

The default DFSMSdss DUMP and RESTORE commands issued by DFSMSrmm when the EDGBKUP EXEC parameter BACKUP(DSS) is specified are:

```
DUMP DS(INCLUDE(cds_name)) OUTDD(BACKUP) SHARE -
   COMPRESS CONCURRENT VALIDATE OPTIMIZE(1)

RESTORE DS(INCLUDE(**)) INDD(BACKUP) -
    REPLACE
```

Figure 13 on page 84 shows an example of using the DSSOPT DD statement.

```
//EDGHSKP  EXEC PGM=EDGHSKP,PARM='BACKUP(DSS)'
//MESSAGE  DD  DISP=SHR,DSN=RMM.MESSAGE
//SYSPRINT DD  SYSOUT=*
//BACKUP   DD  DISP=(,CATLG),UNIT=TAPE,DSN=BACKUP.CDS(+1),
//             LABEL=(,SL)
//JRNLBKUP DD  DISP=(,CATLG),UNIT=TAPE,DSN=BACKUP.JRNL(+1),
//             LABEL=(2,SL),VOL=REF=*.BACKUP
//DSSOPT   DD  *
   CONCURRENT OPTIMIZE(4) VALIDATE COMPRESS
/*
```

Figure 13. DSSOPT DD Statement

The DUMP command operands you can specify in the DSSOPT DD statement are controlled and validated by DFSMSdss, not by DFSMSrmm. If unsupported command operands are specified, DFSMSdss fails the dump operation. Comments can be included in the DSSOPT records, but they must be comments acceptable to DFSMSdss. Refer to the *DFSMS/MVS V1R3 DFSMSdss Storage Administration Reference* for command operands that are supported.

The DFSMSdss parameter keywords specified in the DFSMSrmm housekeeping job in Figure 13 have the following meanings:

CONCURRENT specifies that the data is to be processed with concurrent copy if possible; otherwise, the data will be processed as if CONCURRENT were not specified.

OPTIMIZE(4) specifies that the data is to be dumped as fast as possible by reading a cylinder's worth of data at a time and overlapping the reading of the data with writing the previous cylinder's worth of data to tape.

VALIDATE specifies that all indexed VSAM data sets are to be validated as they are dumped.

COMPRESS specifies that the dumped data is to be written in compressed form to the output medium. In this example the output medium is tape. Even though the output tape may support hardware compression performance will benefit from using COMPRESS as less data will be transferred across the channel.

6.2.2 Migration and Coexistence

When DFSMSdss is used as the backup method, DFSMSrmm must run with journaling active. Backups taken using the new BACKUP(DSS) option cannot be restored using previous levels of DFSMSrmm code. However, a backup taken by DFSMSrmm using DFSMSdss can be restored directly with the ADRDSSU utility and the EDGBKUP utility for forward recovery from the journal and journal backups.

Backups taken on either level of code that do not use DFSMSdss are eligible on either level of code.

When BACKUP(DSS) is requested, updates to the control data set are allowed during journal backup. In addition, updates during control data set backup are allowed if concurrent copy is used. The journal backup includes records that are not included in the control data set backup. For this reason the restore process now involves forward recovery from the latest journal backup as well as the

active journal. If only the active journal is used, some records may not be recovered. EDGBKUP detects this situation and issues message EDG6424E.

To use DFSMSrmm nonintrusive backup, an installation must have a DASD subsystem controller that supports concurrent copy. The IBM 3990-6 DASD subsystem controller supports concurrent copy, as do the IBM 3990-X03 controllers with the Extended Platform.

Using DFSMSdss with concurrent copy can dramatically reduce the time during which DFSMSrmm is unavailable for tape management processing. The dump output is fully compatible with dumps produced without concurrent copy.

Dumps produced on a system using concurrent copy can be restored on systems that do not support concurrent copy.

6.2.3 Performance

Although DFSMSrmm nonintrusive backup does not speed up the actual dump, it dramatically reduces the time during which DFSMSrmm is unavailable for performing tape management functions. With concurrent copy, DFSMSrmm is unavailable for only a few seconds rather than the minutes before the option was available.

6.3 Problem Determination Aid Trace

The DFSMSrmm PDA trace provided with DFSMS/MVS V1R4 will gather diagnostic data at specific internal module points and record it in a circular file within storage and (optionally) on DASD (see Figure 14 on page 86). The purpose of the trace is to gather sufficient information during DFSMSrmm execution to enable subsequent analysis to pinpoint module flow and resource usage related to any given problem.

From a customer perspective, PDA trace support addresses the serviceability issue of first-time data capture. Currently, if a DFSMSrmm defect occurs, it requires multiple re-creations, traps, and dumps to understand the failing scenario. The PDA trace decreases the amount of time to diagnose defects should they occur.

6.3.1 Functional Characteristics

The overall design objective of the PDA trace is to maximize data capture with minimum performance impact. The PDA trace facility consists of the following:

- Entry and exit trace requests throughout DFSMSrmm
- A trace data accumulation function
- A trace data recording function

The DFSMSrmm trace recording function receives the trace data scheduled for output and writes it to a file on DASD. The PDA trace consists of two separate log data sets. DFSMSrmm recognizes these log data sets by their DD names, EDGPDOX and EDGPDOY. Recording takes place in the data set defined by EDGPDOX. When that data set is filled, the two data set names are swapped, and recording continues on the newly defined data set.

When the newly defined data set is filled, the names are again swapped, and the output switches to the other data set, thus overlaying the previously recorded

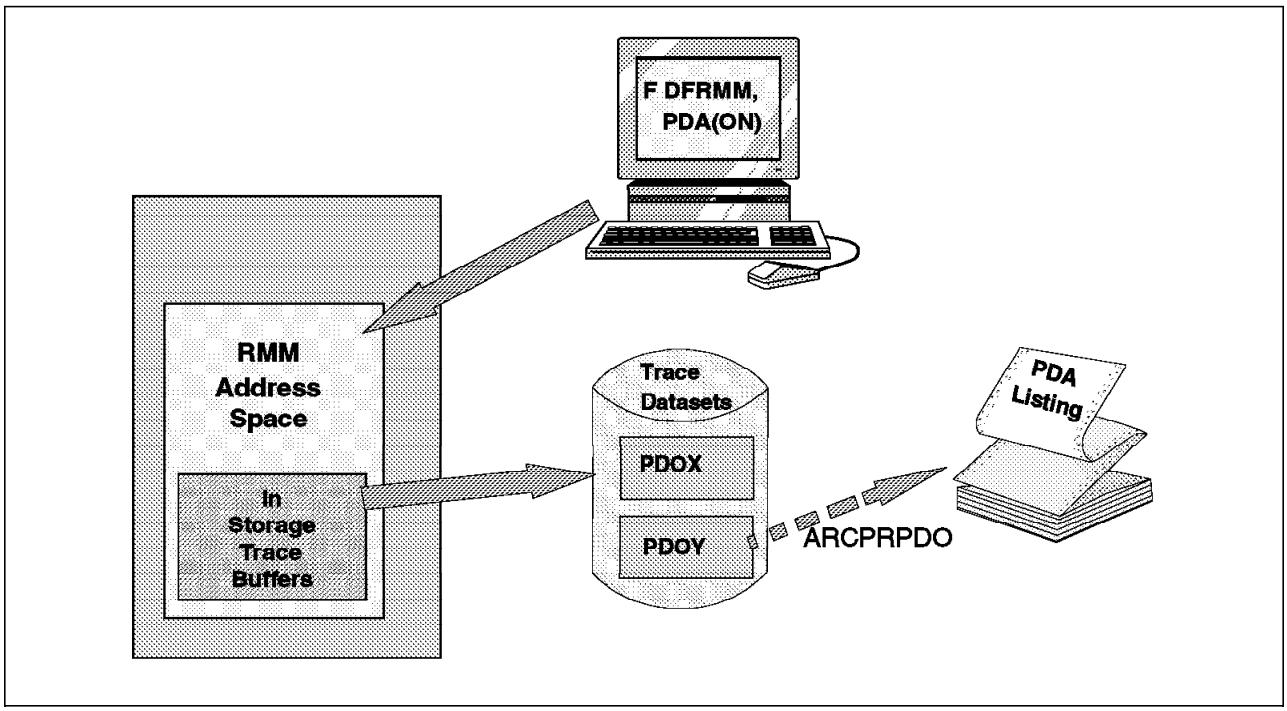


Figure 14. DFSMSrmm Problem Determination Aid

data. The larger the file, the longer the period of time that will be represented by the accumulated data.

The preferred implementation of the PDA trace is to establish a protocol that automatically copies the EDGPDOY data set to tape as a GDG data set each time message EDG9117I is issued. This practice provides a sequential history of trace data over time so that the data is available when needed for resolving problems. Sample JCL will be provided for dumping this trace data as a series of GDGs.

The in-storage circular file is a contiguous area of storage divided into blocks. Each block contains multiple variable-length trace entries. Each block is written to DASD as a separate unit of data. The size of each block and number of blocks in the file are controlled by user-defined values within an in-storage circular file. The DASD block size for the EDGPDOX/EDGPDOY data sets and the number of blocks or buffers that make up the trace wrap table can be changed with PARMLIB options:

```
PDABLKSZ(min=1, max=31)  and
PDABLKCT(min=3, max=255)
```

The total size of the in-storage trace wrap table is the product of these two items in kilobytes (KB). For example, if you specify PDABLKSZ(4) and PDABLKCT(12), you get a total in-storage trace table of 48KB. The default in-storage trace wrap table sizes if PDABLKSZ and PDABLKCT are not specified are based on where the RMM trace data sets are placed. These defaults are:

```
EDGPDOX on 3380         = 5610K
EDGPDOX on 3390         = 6885K
default if no EDGPDOX   = 3060K
```

The PDA trace came from DFSMShsm. The reuse of this trace facility will allow consistency in how the two products implement a trace facility. Customers and support groups familiar with the DFSMShsm trace facility will be able to use the

DFSMSrmm trace facility without having to learn anything radically new. Use of the DFSMSrmm trace facility requires no new keywords.

6.3.1.1 Invocation

The trace facility is enabled at the end of the DFSMSrmm startup unless specifically inhibited by command in the startup procedure.

The default operating mode for the trace facility is:

- PDA ON
- PDALOG ON
- PDALOG SWAP PDOX and PDOY at initial startup
- PDABLKCT(255)
- PDABLKSZ(22) for 3380, 27 for 3390, or 12 for any other device

Typical commands that can be used from the console to control the DFSMSrmm PDA trace are:

```
F DFRMM,PDA=ON          (turns on tracing)
F DFRMM,PDALOG=SWAP      (used to SWAP the DASD trace data sets)
```

Frequently, only several minutes to one hour of the PDA trace will be required to analyze a problem. To reduce the amount of data to be sent for analysis, use the DFSMSrmm trace formatter program, ARCPRPDO, to select and copy all trace entries created during the time of interest.

6.3.2 Migration and Coexistence

If an installation does not update its DFSMSrmm startup procedure to include the new DD EDGPDOX and EDGPDOY statements, it will receive informational message EDG9115I, and the tracing will be recorded only in the in-storage trace buffer.

6.3.3 Support Use Information

Two new data sets are required for PDA trace output data logging to DASD. They are not required if trace data is not to be written to DASD. These data sets are physical sequential, in variable-length record format, with a maximum record length of 256 bytes and a maximum block size of 31748 bytes. The actual block size is controlled by PARMLIB options or the default values. The data sets should have a disposition of SHR so that they can be examined online, concurrent with DFSMSrmm operation. Because of the data set swap and rename process, both the EDGPDOX and EDGPDOY data sets must reside on the same volume. Care must therefore be taken if these data sets are to be allocated to SMS managed volumes. In a multihost environment, each different host must have its own unique EDGPDOX and EDGPDOY data sets. The PDA output data sets must not be shared with any other system component for trace data accumulation.

In the DFSMSrmm startup procedure, the following new DD statements are required for the EDGPDOX and EDGPDOY data sets:

```
//EDGPDOX DD DSN=hlq.RMMPDOX,DISP=SHR
//EDGPDOY DD DSN=hlq.RMMPDOY,DISP=SHR
```

IEFBR14 could of course be used to preallocate these data sets. An initial size recommendation for these data sets is 20 cylinders. Then adjust them according to installation requirements.

When DFSMSrmm switches the output data sets (indicated by message EDG9117I), the trace data before the switch is available in the data set defined by the EDGPDOY DD statement.

The DFSMShsm PDA formatter (ARCPRPDO utility) can be used to format the EDGPDOX and EDGPDOY files. Users are not required to have the DFSMShsm license to use the ARCPRPDO utility to format the DFSMSrmm PDA files.

6.3.4 Performance

Although the trace circular file processing is performed inline, the logging to DASD is an independent task. This task is not merged inline with DFSMSrmm function, has minimal dependence on the product structure or design, and competes separately for time and resources. Therefore, the trace′s potential for degrading DFSMSrmm performance and reliability is minimal.

6.4 Support for DFSMShsm Alternate Tape Processing

If APAR OW21287 is not installed, DFSMSrmm prevents an installation from using the alternate copy of a backup or migration tape to recall or recover a user data set. DFSMSrmm checks that the full 44-character DFSMShsm data set name matches the name recorded in DFSMSrmm′s control data set. Because the data set name of the alternate tape has **COPY** as its second qualifier, it fails the check.

To work around this, an operator issues the RMM CHANGEVOLUME subcommand to change the data set name known to DFSMSrmm to the alternate data set name. DFSMSrmm then allows the alternate tape to be used by DFSMShsm.

The commands are:

```
RMM CHANGEVOLUME altvol INIT(Y) CRLSE(INIT)
RMM CHANGEVOLUME altvol DSN('prefix.xxxxTAPE.DATASET')
```

where:

altvol

specifies the volser of the copy that is being used to replace the original

prefix

specifies the DFSMShsm-defined backup or migration prefix

xxxx

specifies either BACK or HMIG, depending on the DFSMShsm tape type

When APAR OW2157 is installed, DFSMSrmm validates only the last 17 characters of the data set name used by DFSMShsm when it reads the tape for input processing. This DFSMSrmm validation allows DFSMShsm to use the alternate tape without having to inform DFSMSrmm about the change of data set name to be used.

6.5 Recognition of External Data Managers

DFSMSrmm can now be instructed to manage a tape volume but not to track all the data sets that are written to that volume. The EDGUX100 installation exit now supports a new function: RMM records data set details for only the first file on a tape volume but keeps track of the statistics at the volume level.

Customers may have applications that manage their own pool of tapes and the data sets on these tapes. Until now these applications (known as *external data managers* to some other tape management system) have stored information about these data sets, and DFSMSrmm has duplicated it in its control data set. Thus more disk space was required (for the data set information being duplicated), and the backup for DFSMSrmm′s control data set and journal was taking longer than necessary.

When DFSMSrmm is told through the EDGUX100 installation exit to record only information for the first file on a tape volume, it still performs normal volume validation, but it can no longer perform data set name checking for the second and subsequent files on the volume.

DFSMSrmm can still be used to manage the volume based on the first file details and the statistics maintained at the volume level.

6.5.1.1 Requesting Recording of First File Data Set Details

This option is available to the EDGUX100 installation exit only during the OPEN processing for the first file on a volume. The sample exit retrieves the data set name, job name, and job step program name from the system control blocks, before scanning a table for a match. If there is a match, the sample exit sets the PL100_SET_IGNORE_FILE2_TON bit to request that DFSMSrmm maintain data set details only for the first file.

6.5.1.2 Updating the EDGUX100 Exit

To use the EDGUX100 exit for this function, update the sample table with the job name, data set name, and program name for each application that can manage its own data set inventory on tape. To make the job name, data set name, and the program name generic, include either a % or an *. The % represents a positional character, and the * can be used to tell the exit to ignore all remaining characters in any of the three pieces of information per table entry.

Figure 15 on page 90 shows a sample EDGUX100 installation exit with the table coded to include the names of four external data managers.

```
EDMTAB  DS    OF            *START OF TABLE
*                 JOBNAME         DATA SET NAME
*
        DC    CL8'*        ',CL44'BACKUP*'
        DC    CL8'ABC*'               PROGRAM NAME
*
        DC    CL8'STSGWD* ',CL44'*'
        DC    CL8'A*'                 PROGRAM NAME
*
        DC    CL8'STSG%D* ',CL44'STSG%%.BACKUP.*'
        DC    CL8'DEF%MAIN'           PROGRAM NAME
*
        DC    CL8'STSGDPW ',CL44'DAVE.TOOMUCH.DATA.*'
        DC    CL8'AB999*'             PROGRAM NAME
*
        DC    CL8'EDMEND'   *END OF TABLE MARKER
```

Figure 15. EDGUX100 Installation Exit. The table entries are coded to include four external data managers.

Using the EDGUX100 exit to specify external data managers allows quite a lot of flexibility in specifying the type of jobs, data sets, and programs for recording only the first file on a tape volume. For example, the first entry in Figure 15 asks the exit to look for all data sets with a high-level qualifier starting with the characters BACKUP that is being written by any program starting with the characters ABC. In this particular example, the job name is specified with an *, so all jobs would be considered.

6.6 DFSMSrmm Inventory Management Trial Runs

It is now possible to check what DFSMSrmm will do during VRSEL processing in a trial run of inventory management vital record processing, without DFSMSrmm actually making any changes to the control data set or journal. If required, this trial run can be made for any specified date in the future. The trial run lets the installation see how the vital record specifications, which have been defined or changed, would be processed in a production run, except that DFSMSrmm does not make any changes to the control data set. DFSMSrmm can be set up to require that tape volumes and data sets are ″covered″ by a specified minimum number of vital record specification (VRS) policies. If an installation has added, changed, or deleted VRS policies, the system can be set up such that a production inventory management run cannot be performed until a successful trial run has been made.

To reduce the possible exposures occurring from changes made to VRS policies, such as causing changes to tape data set and volume expiration processing, customers have requested the following changes to DFSMSrmm:

- A trial run capability for inventory management
- A report of updates made by inventory management
- The option to enforce a trial run if policy changes are made
- The provision of thresholds for inventory key actions

This release of DFSMSrmm addressed each of the above requirements although threshold processing has been restricted in this release to checking that a minimum number of VRS policies cover the tape data sets and volumes. This will catch the situation where either several VRS policies have been deleted by mistake or an error on the control data set has prevented all policies from being

read into storage. In the past such mistakes or problems could have led to a large number of data sets and/or tape volumes expiring as they would no longer be covered by a valid VRS.

Now that the date can be specified on the trial inventory management run, an installation can check what effect the existing or changed VRS policies would have at some time in the future. For example, it would be possible to show how many tapes would be moved offsite after month end processing had completed. It would also be possible to see whether sufficient scratch tapes would be released and available for use on the weekend.

6.6.1 Inventory Management Processing

If VRSCHANGE(VERIFY) is specified in the EDGRMMxx PARMLIB member and changes have been made to retention and movement policies since the last successful run of inventory management, any attempt at running VRSEL (without VERIFY also being specified), DSTORE, or EXPROC processing will fail. If VRSEL, DSTORE, or EXPROC is requested and changes have been made to retention and movement policies since the last successful run of inventory management, message EDG2308I is issued to the MESSAGE file regardless of the VRSCHANGE option. If a trial inventory management run is requested (VERIFY is coded as a parameter on the EXEC statement), neither DSTORE nor EXPROC can be specified; otherwise processing stops and message EDG6002E is issued.

The ACTIVITY file is now supported, and, during VRSEL processing, details of changes made to data set information are added to it. If the ACTIVITY file is not allocated for a VERIFY run, existing message EDG6101E is issued, and processing fails.

Figure 16 on page 92 shows what you must do to set up the environment for inventory management trial runs.

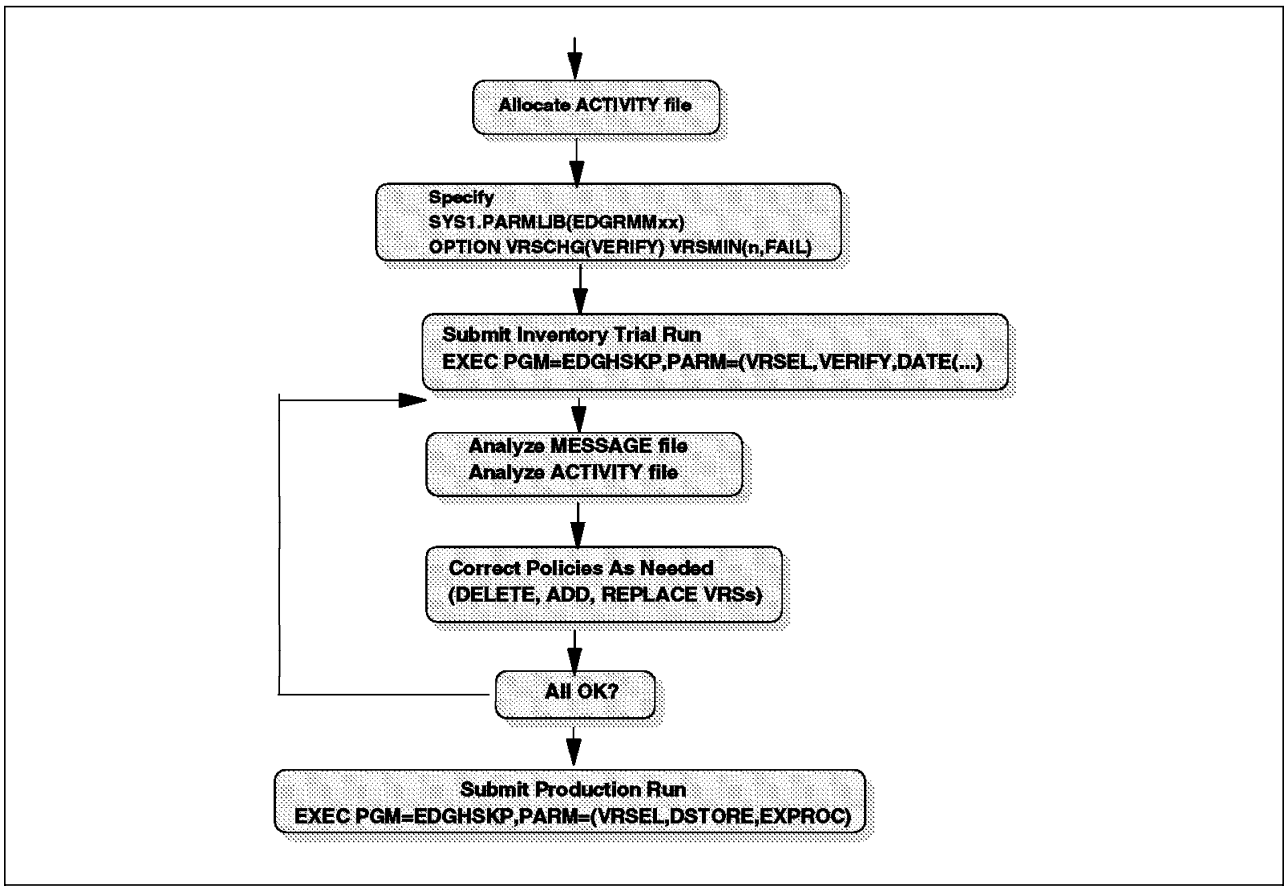


Figure 16. Inventory Management Trial Runs

Once a successful inventory management trial run has completed (VERIFY was specified as a parameter on the EXEC statement), a production inventory management job can be submitted, even if the VRSCHANGE PARMLIB option specifies VERIFY, because DFSMSrmm knows that no changes to VRS have been made.

6.6.2 Inventory Management VRSEL Processing

During processing DFSMSrmm makes no control data set changes during a VERIFY run, other than to show that VERIFY has been run successfully.

DFSMSrmm now counts all VRSs (volume, data set name, and name) that it does not delete and writes this number with message EDG2229I to the MESSAGE data set. DFSMSrmm also compares this number to the number specified through the VRSMIN PARMLIB option and uses the VRSMIN action value (FAIL, WARN, or INFO) to decide whether, and how, processing is to continue.

If the ACTIVITY file is allocated and open, details of changes to data set records and their vital record status are added to the file during processing.

When a NEXTVRS is missing (VRS chain is broken), a new message, EDG2230I, is issued. If VRSCHANGE(VERIFY) is specified in the EDGRMMxx PARMLIB member, this NEXTVRS chain must be repaired and a new VERIFY job run successfully before a production inventory management job can be run.

6.7 Migration and Coexistence

When converting to DFSMS/MVS V1R4, consider the following points:

- Several new messages are issued to the MESSAGE file during inventory management (see 6.7.3, “New and Changed Messages” on page 94 for details).
- The default for the VRSCHANGE PARMLIB option is that VERIFY is required. Therefore, unless this default is changed to INFO, an inventory management trial run will be required following any changes made to VRS, before the next production run for inventory management is made.
- The default for VRSMIN is that one VRS is required. The recommendation is that VRSMIN be coded as a PARMLIB option to a more realistic value on the basis of how many VRS policies exist for the installation. If changes to VRS policies are made such that they expire or become deleted, this number should be set to a number lower than the current number being used and periodically reviewed.

  If a VRSMIN value is defined and that number of VRSs does not exist, processing stops, unless an alternative action is specified (see 6.7.2, “New PARMLIB Options” for details of coding this operand).

6.7.1 New EXEC Parameters for EDGHSKP parameters

DFSMSrmm provides the EDGHSKP utility to help you perform inventory management. The following new EDGHSKP EXEC parameters are provided in support of inventory management trial runs:

DATE(date|+nnn) The date to be used for VERIFY processing. Dates can be provided in the format yyddd, such as 98030, or yyyy/ddd, such as 1998/030. Both formats indicate 30 January 1998. For the year 2000 and higher, the yyyy/ddd format must be used.

The alternative is to enter the number of days to add to the current date to determine the actual date to be used for VERIFY processing. For example, DATE(+7) tells DFSMSrmm to use the date of seven days in the future.

The default is the current date.

VERIFY Specify this parameter to request that DFSMSrmm not update the control data set on this run of inventory management. This parameter is supported only for VRSEL processing. Neither DSTORE nor EXPROC can be specified along with VERIFY.

Note: With VRSCHANGE(VERIFY) specified in the PARMLIB option statement, only a single, successful VERIFY run is required to enable full inventory management processing. If the expected results are not achieved despite a successful run, the policies must be modified and further VERIFY runs made until results are acceptable.

6.7.2 New PARMLIB Options

The following options are added to the PARMLIB member EDGRMMxx in support of inventory management trial runs:

VRSMIN(count,action) Use this operand to specify a minimum number of VRSs and the action to be taken by DFSMSrmm when the minimum number of VRSs is not defined.

Also, to control what DFSMSrmm should do when the count is not reached, specify one of the following:

FAIL Issue message EDG2229I to the MESSAGE file and stop inventory management processing. A return code of 8 is set. This is the default.

WARN Issue message EDG2229I to the MESSAGE file and continue processing. A return code of 4 is set.

INFO Issue message EDG2229I to the MESSAGE file and continue processing.

VRSCHANGE This operand determines which action DFSMSrmm should take during inventory management following changes to VRS policies using ADD or DELETE subcommands. Optionally one of the following can be specified:

INFO Changes to VRS policies will not force a VERIFY run to be made

VERIFY VRS policy changes must be verified by running EDGHSKP vital records selection with the VERIFY parameter. This program must run successfully before a production run can be performed.

VERIFY is the default

6.7.3 New and Changed Messages

The following new and changed messages have been introduced in support of inventory management trial runs:

EDG6101E REQUIRED DDNAME *ddname* NOT SPECIFIED

EDG2229I NUMBER OF VRS RECORDS READ IS *number*

EDG2230I NEXTVRS *name_vrs* DOES NOT EXIST. CHAINING *vrs_type* VRS IS *vrs_mask*.

EDG2308I CHANGES HAVE BEEN MADE TO VRS POLICIES SINCE THE PREVIOUS INVENTORY MANAGEMENT RUN

EDG2309I THE PARMLIB OPTIONS CURRENTLY IN USE ARE VRSJOBNAME(*job_option*) VRSMIN(*count_value,action_value*) VRSCHANGE(*change_option*) CATRETPD(*hours*)

EDG2310I INVENTORY MANAGEMENT STOPPING BECAUSE OF VRSMIN(*count_value*,FAIL) OPTION

EDG2311I INVENTORY MANAGEMENT STOPPING BECAUSE OF VRSCHANGE(*change_option*) OPTION

EDG2312I CHANGES HAVE BEEN MADE TO VRS POLICIES SINCE THE START OF THE VERIFY RUN - A FURTHER VERIFY RUN IS REQUIRED

EDG6001I INVENTORY MANAGEMENT STARTING ON *date* AT *time* - PARAMETERS IN USE ARE *parameters*

Chapter 7. Distributed Data Access

In today′s distributed computing environment, applications must often access data residing on different computers in a network. In many cases, the most effective data access services occur when applications can access remote data as if it were local data.

To provide such services, DFSMS/MVS offers the following solutions:

- Network File System Server, which supports client platforms using TCP/IP communication protocols and NFS file access protocols
- Distributed FileManager/MVS, which supports client platforms using APPC communication protocols and DDM file access protocols

These data access services enable users and applications on heterogeneous client computers in your network to take advantage of storage resources on MVS/ESA and OS/390 systems, including system-managed storage, high-performance storage access, file access security, data sharing, and centralized data access.

7.1 Network File System

The DFSMS/MVS Network File System is an MVS/ESA implementation of the industry standard Network File System (NFS) Version 2 protocol originally created by Sun Microsystems, Inc.. The DFSMS/MVS NFS server enables your MVS/ESA system to act as a file server or client to any authorized NFS clients on a TCP/IP network.

The NFS protocols have become one of the most pervasive methods for interoperability between different types of operating systems. The NFS protocol is designed to allow transparent access between a client system and the server system, with the files contained on the server system appearing as if they were available locally on the client system. The NFS client program on the client system maps local file system calls into network calls and sends a request for data or control information to the NFS server. In this way, NFS client and server deliver remote data access to an application without special interfaces in the application.

The files from the server are mounted on the client′s file system. Data transfer occurs when files are read or written and the data is transferred using TCP/IP facilities across the network. A client can see much more data than can be held locally, and many clients can share the same data, thus avoiding distribution of data, downloading, and needless duplication.

7.2 DFSMS/MVS Network File System

IBM′s DFSMS/MVS NFS feature adds significant capabilities that enable OS/390 and MVS systems on a TCP/IP network to access data on other NFS server machines (for example, OS/390, RS/6000, SUN, HP, and PCs using OS/2).

Several functional changes were introduced with DFSMS/MVS V1R3 NFS, which was announced in September 1996:

- Provide DFSMS/MVS NFS client support

- Retrieve the attributes of a migrated data set without recall

7.2.1 DFSMS/MVS NFS Client Function

Evolving from a UNIX environment, many systems support both NFS server capabilities and NFS client capabilities. For some time, the DFSMS/MVS product has had NFS server support as a feature. NFS client support will now be included in the DFSMS/MVS V1R3 NFS feature for the OS/390 and MVS platforms. The DFSMS/MVS NFS client program supports the SUN NFS Version 2 protocols. It uses remote procedure call (RPC) requests to communicate with remote NFS servers over a TCP/IP network (see Figure 17). Thus one OS/390 (or MVS) system can exchange data files with another using the NFS client/server technology. The DFSMS/MVS NFS client provides the full capability for OpenEdition applications to read or write MVS conventional data sets and OpenEdition Hierarchical File System (HFS) files on another OS/390 system that has the DFSMS/MVS NFS feature installed.

With the DFSMS/MVS NFS client support, DFSMS/MVS has the complete client and server implementation of NFS for MVS.

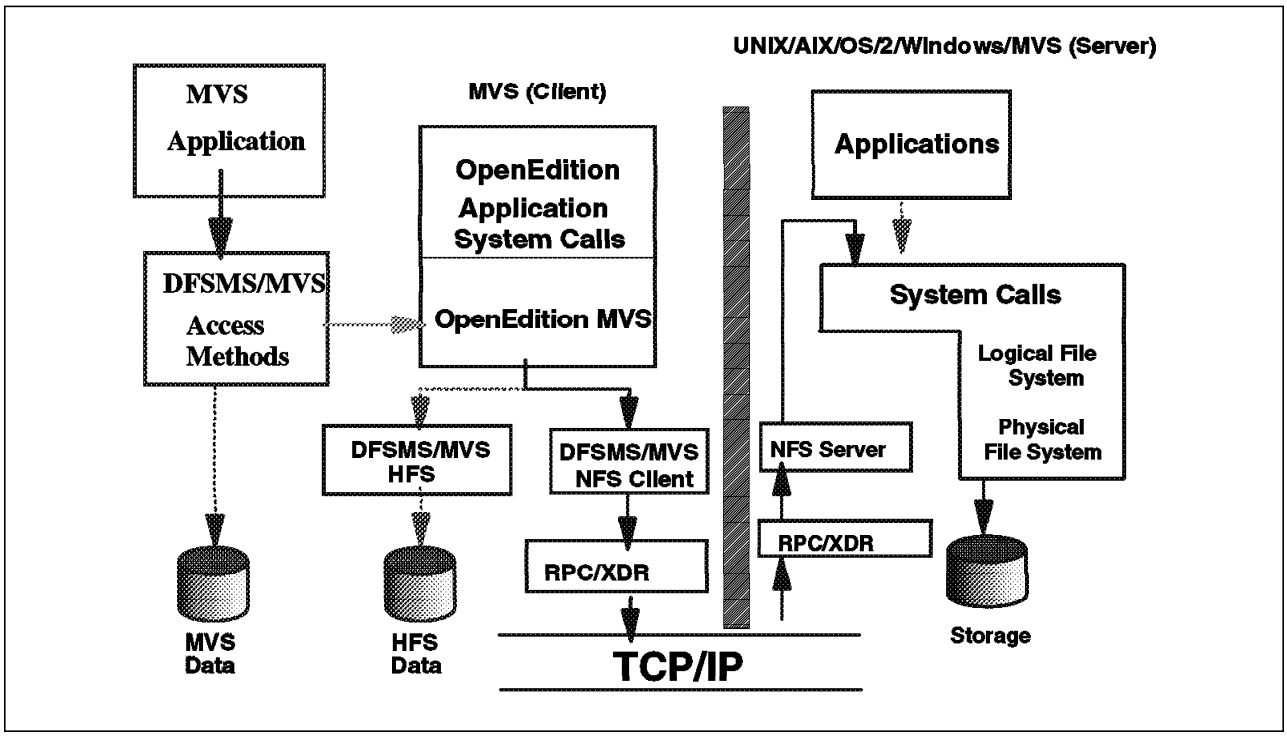


Figure 17. DFSMS/MVS NFS client feature. The NFS MVS client allows you to read and write data on UNIX and other MVS machines.

Some of the functions supported by the DFSMS/MVS NFS client include:

- Create and delete files
- Read and write files
- List directories
- NFSSTAT and SHOWMOUNT commands
- Support for UNIX-style credentials authentication, such as UID, GID, and permission bits

- Additional Security Authorization Facility (SAF) protection to a remote DFSMS/MVS NFS server with the MVSLOGIN command
- Support for simple data conversion of single-byte character sets

7.2.1.1 Configuration Tasks

The tasks to configure the DFSMS/MVS NFS client are listed below. For detailed parameter descriptions refer to the manual *DFSMS/MVS V1R3 NFS Customization and Operation*, SC26-7029.

1. Note DFSMS/MVS NFS client considerations

   The DFSMS/MVS NFS client does not share the same address space with OpenEdition MVS. During OpenEdition MVS file system initialization, the DFSMS/MVS NFS client is started and run in the Logical File System (LFS) colony address space.

2. Set up the DFSMS/MVS client authorization

   All programs that come with the DFSMS/MVS NFS client must reside in an authorized program facility (APF)-authorized program library. The IEAAPFxx PARMLIB member must be updated with the DFSMS/MVS NFS client library name defined in the GFSCPROC STEPLIB. A sample GFSCPROC is provided as a member of library NFSSAMP.

3. Define the DFSMS/MVS NFS client to RACF

   Define a RACF userid for the DFSMS/MVS NFS client. Because the client runs as a started task, you must define an entry in the RACF-started procedures table that associates the DFSMS/MVS NFS client userid with the client startup procedure name.

4. Update the IFAPRDxx or IGDDFPKG PARMLIB member

   For OS/390 Enable/Disable support add the following to the IFAPRDxx member of PARMLIB:

   ```
   PRODUCT OWNER('IBM CORP')
           NAME('OS/390')
           FEATURENAME('DFSMS/MVS NFS')
           VERSION(*)
           RELEASE(*)
           MOD(*)
           ID('5645-001')
           STATE(ENABLE)
   ```

   If you are installing DFSMS/MVS V1R3 or higher on a system other than OS/390, add the following statement to the IGDDFPKG member of PARMLIB to enable the DFSMS/MVS NFS feature product:

   ```
   DFSMS_FEATURE=(systemid.NFS)
   ```

5. Update the OpenEdition MVS PARMLIB member

   The NFS Client is started during OpenEdition MVS initialization. The ***FILESYSTYPE*** statement must be present in OpenEdition PARMLIB member ***BPXPRMxx***. It defines the type of file system.

   ```
   FILESYSTYPE
       TYPE(NFS)                  <----- NFS Client
       ENTRYPOINT(GFSCINIT)       <----- Entry point for NFS Client
                                         initialization
       PARM('installation parms') <----- Installation parameters for
                                         NFS Client
   ```

```
ASNAME(proc_name)          <----- Procedure name in
                                  SYS1.PROCLIB to start
                                  colony address space
```

The name of the **TYPE** operand must be NFS; otherwise the utility program *nfsstat* may fail.

The **ASNAME(proc_name)** operand specifies the procedure name in SYS1.PROCLIB that is used by OpenEdition MVS to start the address space in which the DFSMS/MVS NFS client will be initialized. The **proc_name** is also used for the name of the address space.

6. Update SYS1.PROCLIB

   Add the procedure name specified in the **ASNAME(proc_name)** operand to the system PROCLIB.

   The DFSMS/MVS NFS client is required to start in a separate and stand-alone LFS colony address space for data integrity and data isolation among different physical file systems (PFSs). The sample DFSMS/MVS NFS client startup procedure can be found as member GFSCPROC in the NFSSMP library.

7. Allocate DFSMS/MVS NFS client log data sets

   The NFS stores messages in the DFSMS/MVS NFS client data sets specified in the NFSCMSG1 and NFSCMSG2 statements of the clients startup procedure. The log data sets must be preallocated with the following attributes:

   - DSORG=PS
   - RECFM=VB
   - LRECL=137
   - BLKSIZE=6144

8. Mount remote file systems

   OpenEdition does not support NFS mounts in the SYS1.PARMLIB member statement. You can use the OpenEdition MVS automount facility or the TSO MOUNT command to make a connection between a mount point on your local MVS HFS and one or more files on a remote MVS, AIX, UNIX, OS/2, OS/390, or other file system.

   The remote file system must be mounted on the OpenEdition MVS file system before any reference is made to the remote data. Once mounted, the remote file system can be treated as an extension of the local OpenEdition MVS file system.

7.2.2 OpenEdition Interface

The DFSMS/MVS NFS client support operates under OpenEdition (UNIX Services). Thus applications can use the OpenEdition POSIX-compliant system interfaces to utilize the NFS client functions with few changes to those applications. In addition conventional DFSMS/MVS Basic Sequential Access Method (BSAM), Queued Sequential Access Method (QSAM), Virtual Storage Access Method (VSAM) ESDS applications, as well as OpenEdition applications and users, have transparent access to data on any remote NFS server platform supporting SUN NFS Version 2 protocols.

The NFS Client is implemented on OpenEdition MVS as a new PFS. It implements the client portion of the NFS Version 2 protocols and uses TCP/IP to communicate with a remote NFS server.

7.2.2.1 Access Method Support for OpenEdition Files

DFSMS/MVS V1R3 access methods offer a new opportunity to access enterprise data in an open systems environment. Many applications written to use the BSAM/QSAM access methods, and some applications written to use the VSAM ESDS access, now have transparent access to files associated with OpenEdition. With this support, the MVS conventional application using the normal record or block interfaces provided by the DFSMS/MVS access methods can access files residing in remote NFS server platforms. This support will map between the record interface or block interface and the byte-oriented structure of the OpenEdition or remote files. The following files can be accessed by specifying the PATH=pathname parameter on the BSAM, QSAM, or VSAM JCL DD statement or SVC 99:

- Local OpenEdition files

  OpenEdition regular files are managed by the DFSMSdfp HFS. Each HFS file is an identified unit of text or binary data and is byte-oriented (versus record-oriented). The HFS supports file names and pathnames up to 255 characters long. Within the HFS file name, special characters and lower-case characters can be embedded.

- Remote files

  A remote file is any regular file residing in a remote NFS server platform that supports the SUN NFS Version 2 protocols. With this function, MVS applications written to use BSAM, QSAM, and VSAM ESDSs can access remote files through the DFSMS/MVS NFS client and the network environment just as if the remote files were local.

- Local FIFO special files

  An FIFO (First-In-First-Out) special file, or named pipe, is an OpenEdition special file. An FIFO special file is defined in a DD statement with both PATH= and DSNTYPE=PIPE. An FIFO special file provides data connections between programs and typically sends data from one program (or process) to another. Both ends of an FIFO special file can be used in a single program task. All data in an FIFO special file vanishes when the last program using it closes it.

- Local character-special files

  A character-special file is an OpenEdition special file that defines a null or a dummy file (for example, /dev/null).

7.2.3 DFSMS/MVS NFS Client Commands

The following commands are provided with NFS client support on the OS/390 and MVS platforms:

mvslogin Login function to DFSMS/MVS NFS server

mvslogout Logout function from DFSMS/MVS NFS server

showattr Display attributes established for a site or mount point of the DFSMS/MVS NFS server

nfsstat Display NFS client statistical information, or reset the statistical information to zero

showmount Display remote NFS server mount information

os22mvs Convert line delimiter from OS/2 format to MVS format. The carriage-return line-feed pairs (CR, LF) in the input line are converted to a newline (NL). No other conversions are performed. Input and/or output files can be local or remote files.

mvs2os2 Convert line delimiter from MVS format to OS/2 format. The NL in the input file is converted to CR LF pairs. No other conversions are performed. Input and/or output files can be local or remote files.

The command programs are intended to run in an OpenEdition MVS shell environment. They are not implemented as TSO commands. The input/output file must be an HFS file object. Conventional MVS data sets are not supported.

7.2.4 DFSMS/MVS NFS Server Enhancements

With DFSMS/MVS V1R3, data set attributes such as size and record format are accessible for migrated SMS-managed data sets, without having to recall the data set if the data set is migrated under DFSMS/MVS V1R3. Supported SMS-managed data set types are: ,

- Physical sequential (PS) data set
- VSAM data set
  - Entry-sequenced data set (ESDS)
  - Key-sequenced data set (KSDS)
  - Relative-record data set (RRDS)
  - Variable-length relative-record data set
- Partitioned data set extended (PDSE) and partitioned data set (PDS) members, treating each member as a separate file.

Migrated PDS and PDSE members are not supported because a migrated member′s attributes are not accessible through this interface.

7.2.4.1 Obtaining Migrated Data Set Attributes without Recall

The DFSMS/MVS V1R3 NFS server can obtain the attributes of a supported SMS-managed migrated data set without recalling the data set. To take advantage of this function, the data set must have been accessed by the DFSMS/MVS NFS server before migration.

Attributes such as file size and time stamp are saved to DASD. Subsequent file size requests do not cause a recall of the supported SMS-managed migrated data set, thus improving performance.

If the data set is modified outside the server by a non-NFS application (for example, by ISPF edit) before it was migrated, the stored file size could be incorrect. When the data set is accessed again by the DFSMS/MVS NFS server, it must be recalled to determine the correct file size.

7.2.4.2 Deleting Migrated Data Sets

When a request is made by an NFS client to delete a supported SMS-managed migrated data set on the DFSMS/MVS NFS server, the data set will be deleted without recalling the data set. For PDS and PDSE migrated data sets, the data set will be recalled to read its member information.

7.3 DFSMS/MVS Distributed FileManager

When developing applications in a distributed environment, have you ever wanted to run your OS/2 programs using the existing host-based data sets, without having to write additional code on the host side, translate data and record formats, or transfer the data sets to your workstation? Using the Distributed FileManager (DFM) set of products, you can access and manipulate existing host data sets. If the data sets are under the control of an operating system that supports DFM, all you need to do is write your application, define the logical connection to your host system, and define the data translation that you need. Then let DFSMS/MVS DFM do the rest.

DFM as a client/server concept is implemented as a set of products that act as the interface between your applications and the file data, irrespective of where the data resides. This implies that the data may be local to the application or remote to the application on either a LAN server or a mainframe server on a WAN. This product set comprises client products and mainframe-based products.

The DFM product set is based on the Distributed Data Management (DDM) Architecture that has introduced the terms *source* and *target* for data requests. These terms are equivalent to the terms *client* and *server*, respectively. One of the primary ingredients of the Distributed FileManager suite of products is Data Access Services (DAS) for both record-oriented and stream-oriented data on local systems and remote servers. With DFM you can invoke commands to create, delete, rename, and copy data sets on a remote system.

DFSMS/MVS DFM uses the DDM protocol, which enables like and unlike computer systems to share file systems across a network. It enables your MVS/ESA system to act as a server (target) to remote client (source) systems. DFSMS/MVS DFM is designed to work with operating system platforms that support DDM requests.

DFSMS/MVS DFM enables users on authorized client systems to remotely access data on MVS/ESA server systems as if the data were local to the users. Users can remotely perform such tasks as creating, reading, and deleting MVS/ESA data sets through local commands. Because DDM clients and servers communicate through common DDM commands, users on the client systems do not need to use MVS/ESA command language.

DFSMS/MVS DFM accepts access with a record application programming interface (API) or byte stream access with a byte stream API. DFSMS/MVS DFM allows client systems to access the following data set types on MVS/ESA server systems:

- PS data set
- VSAM data set
  - ESDS
  - KSDS
  - RRDS
  - Variable-length RRDS
- PDSE and PDS members, treating each member as a separate file.

Note: DFM also supports path processing through an alternate index for VSAM data sets.

7.3.1 Distributed FileManager DataAgent

The DFM DataAgent, shown in Figure 18, is an extension to the DFSMS/MVS DFM component of DFSMSdfp and to the DFM component of SMARTdata UTILITIES that enables remote callers on OS/2, AIX, and Windows NT to invoke DFM DataAgents to execute specified routines on MVS. The functions that may be performed using this facility include the execution of TSO and REXX commands as well as user-written programs. A set of model routines is provided that illustrates specific uses of this function. The DFM DataAgent enhancement represents a significant extension of the functionality of the remote workstation application beyond basic data access.

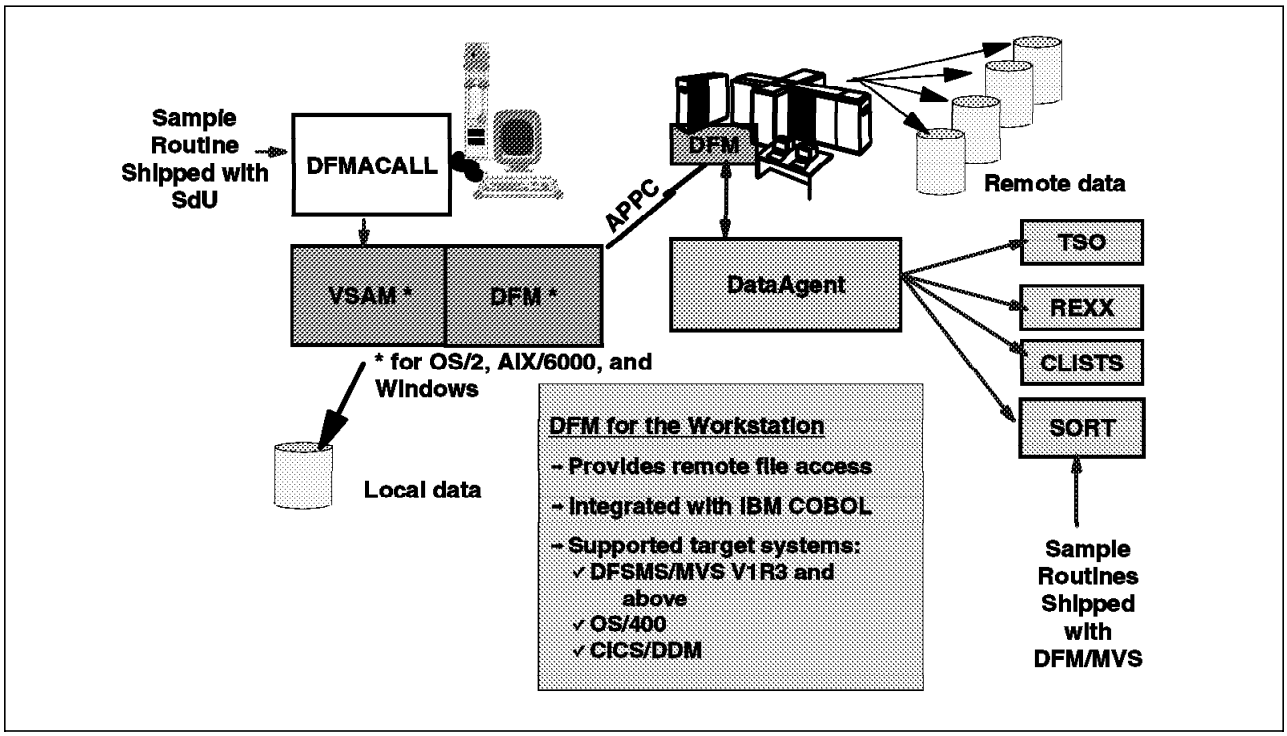


Figure 18. The DFSMS/MVS DFM DataAgent.

The DFSMS/MVS DFM DataAgent can be called by a remote workstation that uses SMARTdata UTILITIES to specify the DataAgent routine to be invoked. The DataAgent routines can be user written, IBM written, or vendor written. Several sample routines are provided, showing how to write a DataAgent routine in a high-level language, how to invoke DFSORT, and how to invoke TSO functions such as TSO CLISTs, REXX EXECs, and TSO commands.

The advantage of the DFSMS/MVS DFM DataAgent approach over some other client/server packages is that it allows client applications to expand their function beyond basic data access (read, write, update, and delete) by actually executing MVS jobs that range from running simple TSO and REXX commands to fairly elaborate programs.

This ability to initiate jobs on remote MVS/ESA systems provides the basis for much greater flexibility to tailor access from workstation applications to specific business requirements. For instance, one DataAgent could be invoked to preprocess data by making extracts from various files and repositories and placing the data into a temporary work file. The remote workstation application could then process the data in the temporary file and, upon completion, another

DataAgent could do the postprocessing on the temporary file and put the updates and changes into the appropriate files and repositories.

The DFSMS/MVS DataAgent provides the ability to initiate jobs on the remote MVS/ESA platform in a manner similar to remote procedure call. This represents a significant increase in capability as the workstation application's processing ability is extended beyond basic data access by allowing the actual initiation of jobs from the remote platform.

The DFSMS/MVS DFM DataAgent requires that the SMARTdata UTILITIES be installed on the workstation.

7.3.1.1 SMARTdata UTILITIES

The SMARTdata UTILITIES offer a solution to some of the complexities of developing applications and accessing distributed data in a client/server environment. The utilities include a record-oriented access method, an interface to remote server data, data description and conversion services, and data sorting and manipulation tools. The SMARTdata UTILITIES are embedded in some popular HLLs (COBOL and PL/I) to simplify client/server application programming.

The workstation utilities and languages work together with complementary products on the server system (for example, DFSMS/MVS DFM on the MVS/ESA platform) to provide transparent access to distributed data.

The SMARTdata UTILITIES are:

- *Virtual Storage Access Method (VSAM) Local File System*. This utility provides an API and a workstation local file system that can be used to create, access, and modify record-oriented files on local (OS/2 or AIX) systems.
- SMARTdata UTILITIES DFM. This utility, using the same application programming interface as the local VSAM file system, can be used to create, access, and modify record-oriented files on remote systems from client OS/2, AIX, and Windows NT systems. In addition, SMARTdata UTILITIES DFM for OS/2 provides "stream access" to execute OS/2 native commands (for example, dir and type) directed against MVS files.
- Data Description and Conversion (DD&C). This utility offers data conversion services that can be used when complex data is exchanged between different operating systems, programming languages, and international locations.
- SMARTsort. This utility's data manipulation tools can help you sort, merge, extract, and organize your data, including record and stream data, whether it is local or remote or both.

7.3.1.2 Simplified Application Development

The SMARTdata UTILITIES simplify client/server application development by offering these features:

- Integration with several HLLs
- Callable application programming
- Transparent access to local and remote data
- A single interface to any kind of record data

- Server-compatible data management capabilities
- Application portability across several platforms

7.3.1.3 Calling the UTILITIES

To simplify client/server application programming, the SMARTdata UTILITIES are embedded in several HLLs (currently COBOL and PL/I). You can write a client application in one of the HLLs that exploit the utilities, and let the HLLs call the utilities; or your application can call the utilities directly, using the APIs that come with the utilities. First, you use the administrative functions of the utilities to define your data environment. Then, whether you call the utilities from an HLL, the APIs, or both, the utilities handle the complexities of finding, accessing, converting, and sorting your data.

7.3.1.4 Addressing Distributed Data

The SMARTdata UTILITIES facilitate developing and maintaining client/server applications that access distributed data, by offering transparent data access and access to record files.

The data routing and conversion services let you write client applications without concern for the location and format of the server data. You do not have to know whether the data server is local or remote, you do not have to know how to address the data formats on the server systems, and you do not have to change your application if the data is relocated.

The record access utility can help you access record-oriented data, in general used in mainframe environments, even though workstation data is typically stream-oriented. You can use exactly the same interface to access record data anywhere, so you do not have to remember, or learn, how to use several different data management systems.

7.3.1.5 Client/Server Development

The SMARTdata UTILITIES facilitate integrating workstation applications with server heritage systems, particularly when used with an associated HLL.

The record access utility and the data sorting tools provide you with new data management capabilities on the workstation that were only available on the mainframe in the past. They can help you establish a more complete development environment on the workstation by letting you test against server data or simulate server data on the workstation.

The new workstation version of server (mainframe) programming languages, together with the embedded mainframe-style utilities, can help you run mainframe applications on a workstation. The mainframe compatibility of the languages and the utilities give you new delivery and development options:

- You can easily move an existing application from the mainframe to the workstation without moving your heritage data.
- You can develop an application on the workstation and deliver it on the mainframe without changing the application.

7.3.1.6 Simplified Systems Management

The SMARTdata UTILITIES simplify client/server systems management by offering these features:

- Data access independent of data administration
- Remote data access at the record and stream level

7.3.1.7 Independent Applications and Data

The SMARTdata UTILITIES include data access functions, which can be called from a client application program, and administrative functions, which can be performed independently of an application. The data access functions let you write a client application without concern for the location or format of your data; and the administrative functions let you define and change the data environment without modifying your application. The ability to access and administer data independently gives you the flexibility to redistribute your applications and data as your business requirements evolve, without requiring any changes in your application. Some of the utilities include both data access and administration functions; others are primarily one or the other.

7.3.1.8 Sharing Distributed Data

The SMARTdata UTILITIES, by providing data access at the record and stream level, virtually eliminate the need to download and upload entire files. As a result, using the utilities can reduce the problems and costs associated with multiple copies of files spread across the client/server environment. Sharing distributed data, rather then replicating it, can help simplify client/server data administration tasks, improve data integrity and currency, and reduce the use of system and network resources.

7.3.1.9 Benefits of the SMARTdata UTILITIES

From an application development perspective, the SMARTdata UTILITIES handle the complexities of client/server data access and bring the power of mainframe data management capabilities to the workstation. In addition, the utilities are embedded in the workstation version of some mainframe programming languages, thereby simplifying multiplatform application programming.

From a system management perspective, the SMARTdata UTILITIES give you the flexibility to develop, migrate, and run your applications independently of the way you distribute your data. By making it easier to share data without replicating it, the SMARTdata UTILITIES can help simplify data administration, enhance data integrity, and reduce the overall costs of a client/server environment.

7.3.2 DFM DataAgent Invocation

All of the functions and benefits of the SMARTdata UTILITIES do not alleviate the need to process data on the mainframe that does not fit in a predefined transaction and the need to have a way of invoking TSO functions such TSO CLISTs, REXX EXECs, and TSO commands. This lack of functionality will be relieved with the DFSMS/MVS DFM DataAgent. A sample DataAgent is provided to invoke a TSO CLIST, REXX EXECs, or TSO commands.

A DFSMS/MVS DFM DataAgent can also be used to extract data from MVS files and databases at the beginning of a workstation application in preparation for subsequent retrieval by the client through normal SMARTdata UTILITIES DFM interfaces. You could, for example, use DFM DataAgent to access data sets not otherwise supported by DFM and copy all or some of the data to a temporary file

that is supported. If the DataAgent returns the name of the temporary file to DFSMS/MVS DFM, an SMARTdata UTILITIES DFM application could then work with this temporary file as if it were the real file. The DataAgent could then copy any changes to the real file or databases when the declaration for the temporary file name is deleted.

When a DDMOpen API is presented to SMARTdata UTILITIES DFM, the leading characters in the file name are used to:

- Determine whether the file is local or remote
- Associate the remote file with the specific DDM server

In this way, all client platforms that have SMARTdata UTILITIES DFM installed can use the DDMOpen remote file request to MVS/ESA or OS/390 to trigger DataAgent processing on the host.

For both stream and record access, parameter declarations pertain to the object for which a DCLFIL (declare file) is done. This could be a file, a directory, or a drive.

Parameter declarations from multiple sessions are treated independently, even if they are for the same object. For example, if session one defines a DataAgent for directory A.B, a different session can define a DataAgent for A.B while session one′s agent is still active. Depending on which session is used to access the remote data associated with A.B, one or the other agent will be used.

In any case, two different agents cannot run in the same conversation or session for a given object. Also, no more than 32 DataAgent routines can be active concurrently on the DFSMS/MVS system and not more than one instance of a given DataAgent routine can be run at a time if the DFMXLPRM (locate extended DataAgent) parameter is used.

A DFM DataAgent can only be triggered from a SMARTdata UTILITIES DFM application. The DFMACALL sample program is distributed with SMARTdata UTILITIES. It demonstrates the ability to invoke DFM DataAgent functions from C applications on workstations running SMARTdata UTILITIES.

7.3.2.1 DDMOpen

The current DDMOpen function is used to provide a file name suffix that can be used to trigger DFM DataAgent processing on MVS.

Use the DDMOpen function as follows:

```
DDMOpen  (MVS_FileName, FileHandle, AccessMethod, AccIntList,
          FileShare, EABuf,  ........)
```

The MVS_FileName can contain a parameter suffix as follows:

```
MVS_FileName <,parm1(value1)<,parm2(value2)<,.....>>>
```

where *MVS_FileName* and *parm1* through *parmn* are passed unchanged to DFSMS/MVS DFM for processing.

7.3.2.2 DDMClose

The current DDMClose function is used to terminate agent processing. It issues the DDM commands, CLOSE and DELDCL (Delete Declaration), which actually terminate agent processing by invoking the exit with the DELDCL code point in the parameter list, if requested.

7.3.2.3 File Name Suffix Parameters

The new DataAgent file name suffix parameters currently recognized by MVS are:

- AGENT(agent_name<,procedure_parameter>) - Invokes the indicated agent when a file is declared (at DDMOpen) and, optionally, when the file declaration is deleted (at DDMClose). The agent_name must specify the name of a member in SYS1.PROCLIB. Parameters can optionally be provided for symbolic substitution in the PROCLIB member.

  However, allocation will run under the authorization assigned to started tasks. The agent (running under the user′s authorization) may have to use dynamic allocation to allocate files that cannot be allocated by started tasks.

  Because the agent begins as a started task, unless DFM00 (PROCLIB member) specifies RESTRICT_START(NO), the first three characters of the procedure member name must be DFM.

  The maximum length of the agent name and its parameters is 107 bytes. Each parameter in the list of procedure parameters is subject to the MVS limit of 44 bytes.

  The agent runs synchronously. If the PROCLIB member has multiple steps, any file name changes or return code settings will be propagated to the later steps and will be returned to DFM only after the last step is executed.

- PARM(agent_parameter_list) - Parameters to be passed to the agent routine when it begins running in the DataAgent address space. Parameters are converted to upper case and concatenated to any parameters provided in the PROCLIB member.

  If PARM is specified, the PROCLIB member must contain the JCL statements as provided by the sample DFMX0001, which specifies the DFMINIT parameters so as to run a DFSMS/MVS DFM DataAgent address space initialization routine as the first program in the address space. This causes DFSMS/MVS DFM to pass a supplementary run-time parameter list to the DataAgent routine to allow the routine to return an error code and additional reason codes to DFM. The agent parameter list specified is concatenated with DFMINIT before the DataAgent routine is invoked.

  This parameter is ignored if AGENT is omitted.

- PGM(prgram_name) - Name of program (DataAgent routine) that DFM should invoke after initialization. If omitted, the program invoked will default to the agent name requiring that you have identically named MVS load module and PROCLIB members.

- START(job_name<,job_parameters>) - Name of PROCLIB member representing a job or procedure to be started asynchronously. Unless DFM00 specifies RESTRICT_START(NO), the first three characters of the procedure or command must be DFM.

  Optional parameters can also be provided. The MVS limit for the total length of the job name and its parameters is 124 bytes.

DFSMS/MVS DFM verifies that an address space for running the procedure was created but does not verify that the procedure exists or that it ever completes successfully; that is, the started job runs asynchronously.

It is possible to run some existing PROCLIB members that may not have particular initialization requirements by using only the AGENT keyword, we do not recommend that you do so because return codes will not be passed back to DFM. Also, there will usually be a need for extended parameter passing. For these two reasons, use the AGENT parameter in conjunction with the PARM and/or PGM parameter, even if the PARM parameter is the value of PARM() or the PGM name is the same as the agent name.

DFSMS/MVS DFM imposes a limit of 255 bytes for the file name and file name suffix and therefore for the total length of the parameter (AGENT, PARM,PC_CCSID,START) that can be passed.

7.3.3 DataAgent Processing (DFM Target)

The DataAgent is first called when a stream or record file is declared with a file name suffix specifying the agent name. For example, assume the remote file name sent to MVS is ″A.B,AGENT(DFMXUTIL)″. The suffix specifying the agent name is AGENT(DFMUXITL).

The comma, which is invalid as an MVS file name character, is used to distinguish the MVS file name from its suffix. If the file name sent to MVS is ″A.B.C,PARM1(xxx),PARM2(yyy)″, the comma is used to distinguish the MVS parameters from each other as well as the file name.

On the client system, a sample SMARTdata UTILITIES application called *DFMACALL* is available. On an MVS system you are responsible for ensuring that the DataAgent name is unique in SYS1.PROCLIB and in its load library. The prefix *DFM* has been reserved for DFSMS/MVS DataAgent names to help ensure uniqueness. Sample DFM agents use the prefix *DFM* or *DFMX* and can be modified and/or renamed as appropriate.

If the DataAgent routine does not exist or an error occurs while loading a DataAgent, a VALNSPRM (parameter value not supported) will be returned. If the agent returns an error as a result of a DCLFIL or later call, an invalid request error reply message (INVRQSRM) will be returned.

7.3.3.1 DataAgent Interface

This discussion of DataAgent processing assumes that the PARM keyword was specified in conjunction with the AGENT parameter. Otherwise, processing is triggered once at DCLFIL (declare file) time, and processing is the same as for any other synchronous, started task.

The DataAgent can be called a number of times. It can be called at DCLFIL time for preprocessing necessary to build a temporary file for use by an application or for one-time processing. It can also be called at DELDCL (delete declaration) time for postprocessing necessary to move temporary file data to its final destination. Note that this temporary file is located on the server site, so there is no data traffic over the network.

A second parameter is provided to allow DataAgent routines to request postprocessing or additional return and/or reason code processing.

See sample agent DFMXSORT for a DataAgent routine that does both preprocessing and postprocessing.

The exit to the DataAgent routine has the following interface:

- Program status: AMODE(31), RMODE(ANY), KEY(8), TCB mode, primary mode, running in its own address space. The program runs either APF-authorized or non-APF-authorized, as appropriate.
- Register: Standard MVS register conventions are followed.

  Register 1 points to a pointer parameter list consisting of a halfword length field followed by the character string from the workstation′s PARM field. In addition, immediately following the first parameter pointer, a 4-byte pointer to a structure can be used by the DataAgent routine to request that it be called again, to provide additional error feedback, or to allow the routine to change the file name.

  Existing applications are likely to run unchanged because they will be unaware of the second parameter pointer. However, it should be fairly easy to write a new DataAgent application that can access the additional parameter list, either by looking for a second parameter list pointer after the first or by using the DFMXLPRM function described in “DataAgent Locate Extended Parameter List (DFMXLPRM)” on page 110.

DataAgent Parameter List: At DCLFIL and DELDCL time, register 1 points to a fullword address of a 2-byte length field and a variable length character string representing the PARM parameter as specified at DCLFIL time. Immediately following the pointer to the first parameter is a pointer to the DataAgent Extended Parameter List.

DataAgent Extended Parameter List: DCLFIL and DELDCL processing calls the DataAgent routine with a second parameter consisting of an area of storage containing the following:

1. Halfword length field

   The length of the parameter list that follows

2. Reserved halfword

   Reserved for future use

3. Two-byte command code point

   Refer to the DDM architecture for the definitions of code points. In the first release the agent will only be taken for DCLFIL, code point X′102C′, and DELDCL, code point X′102D′. This code point can be used by the DataAgent routine to determine whether to do preprocessing, postprocessing, or one-time processing.

4. Two-byte code point representing the type of object to be declared

   The agent can be taken for an object whose type is DRCNAM (directory name), code point X′1165′, and for FILNAM (file name), code point X′110E′.

5. Original MVS file or directory name

   Two-byte length field and a 54-byte character string containing the file or directory name padded with blanks. This is the name as provided by the workstation.

6. Modified MVS file name

Two-byte length field and a 54-byte character string containing a file name padded with blanks.

The DataAgent is allowed to return a new length and file name here for DFSMS/MVS DFM processing using the altered file name (attempts to change directory names are ignored).

At DELDCL time this field will contain the file name as modified by any previous DataAgent processing.

7. Eight-byte extended reason code area (for optional use by the DataAgent if setting register 15 to nonzero).

DataAgent Locate Extended Parameter List (DFMXLPRM): A new callable system service is provided to allow DFM DataAgent routines written in an HLL to locate the DataAgent parameter list.

Function syntax:

```
DFMXLPRM(AGENTNAME,PARMPTR)
```

- AGENTNAME (input)

  This variable contains the name of the agent routine (which is not necessarily the same as the procedure name) whose extended parameter list is to be located. It is eight characters long and padded with blanks, if necessary.

- PARMPTR (output)

  A 4-byte pointer to the extended parameter list. Set to zero if the parameter list cannot be found.

For example, a DataAgent exit written in C could call this function and test the result as follows:

```
DFMXLPRM("DFMXAGENT",&p_extra);
 if (p_extra == NULL) {
```

The DFMXAGNT sample DataAgent routine is distributed in source form in SYS1.SAMPLIB, member DFMXAGNT, with the DFSMS/MVS DFM DataAgent.

Other DataAgent Routine Considerations: By default DataAgent routines are run only once, when the file or directory is declared. However, if a DataAgent routine wants to do any further processing for the request (for example, if it needs to do some processing at delete declaration time) it can set the first word of the first word of the extended reason code area to -1 before returning with register 15 set to zero.

A nonzero value in register 15 will terminate any further processing for the DataAgent routine regardless of the extended reason code setting.

If the DataAgent routine is running in an HLL environment (in which it is not possible to access the second parameter directly), DFMXLPRM can be used to return a pointer to the extended parameter list. To use DFMXLPRM SYS1.CSSLIB must be specified in the SYSLIB DD statement concatenation when the DataAgent routine is link edited.

7.3.3.2 Sample DataAgent Routines

Sample DataAgent routines are provided with the DFSMS/MVS DFM DataAgent function. These routines can be run as is, or they can be used as models to create your own routines.

PROCLIB Sample DFMX0001: DFMX0001 is a sample PROCLIB member that can be used as a template for setting up PROCLIB members required by DataAgent routines. It is set up to discard the JOBLOG listing that is produced. If you do not discard the output, you may have to issue the ′$ps′ command periodically.

The general procedure for converting an old PROCLIB member so that it can run as a DataAgent is to:

1. Copy the member to a new PROCLIB member (preferably with a prefix of DFMX).
2. Add DFMINIT = to the PROC statement.
3. Change the program name on the EXEC statement to PGM=&DFMINIT
4. Invoke the DataAgent routine by specifying AGENT(new_procname) PGM(original_program_name).

DataAgent for Full TSO Support: DFMXTSO is provided as a general-purpose TSO agent. It is intended to be used with the IKJEFT01 function so its restrictions are as documented by TSO for batch execution. The agent can be, for example, invoked from an OS/2 command file or an SMARTdata UTILITIES DFM application to force execution of a CLIST or REXX EXEC on an MVS/ESA or OS/390 system.

IKJEFT01 will obtain commands to issue from both the parameter list and the userid.DFMXTSO.SYSTIN file. Output will be written to the userid.DFMXTSO.SYSTSPRT file.

SYSTPRT can be browsed from the workstation to determine the results of the commands executed.

DataAgent for Quick TSO Support: DFMQTSO is provided as a general purpose TSO agent. It links to the IKJTSOEV function so its restrictions are as documented by TSO (for example, it is restricted to nonauthorized commands and programs). The agent can be, for example, invoked from an OS/2 command file or an SMARTdata UTILITIES DFM application to force execution of a CLIST or REXX EXEC.

DFMQTSO is available as a source file in SYS1.SAMPLIB.

The agent routine is written in such a way as to obtain commands to issue from the parameter list and not from a SYSTIN file. Output is written to userid.DFMQTSO.SYSTSPRT.

SYSTSPRT can be browsed from the workstation to determine the result of the command executed. Remember that the SYSTSPRT file may be empty if the PARM field specifies an invalid command.

SMARTdata UTILITIES-Based MVS DFM DataAgent Caller: A sample SMARTdata UTILITIES application called *DFMACALL* is available as a sample C program, which facilitates invoking TSO CLISTs and DataAgent routines from the workstation.

The application will accept an assigned drive letter and a file name containing the TSO command input. For example:

```
DFMACALL QTSO  driveletter:  TSOcommandline <DISPLAY>
DFMACALL TSO   driveletter: <TSOcommandline> <DISPLAY>
DFMACALL AGENT driveletter: <remote_filename> MVS procedure<,proc_parm>
           <PGM program_name> <PARM program_paramaters> <DISPLAY>
DFMACALL START driveletter: MVS procedure<procedural_parms>
DFMACALL       driveletter: remote_filename<,filename_suffix> <DISPLAY>
```

Note that the remote file name is optional for AGENT, disallowed for QTSO, TSO, and START, and required for the free-form invocation. If a remote file name is specified, it must exist but other than checking that it can be opened for input, it is not used by DFMACALL itself.

QTSO invokes quick TSO exit DFMQTSO, TSO invokes regular TSO exit DFMXTSO, AGENT invokes the specified MVS procedure, START runs the specified MVS command, and the last is the free form getting its parameters from the filename_suffix, if any.

To use this application you have to initialize the procedure's input file on MVS. The provided samples use userid.AGENT_NAME.SYSIN or userid.AGENT_NAME.SYSTSIN, depending on whether the agent is DFMXTSO. For example, when you use the TSO DataAgent routine DFMXTSO, the SYSTSIN input file might contain something like %MYCLIST PARM1 ....PARMn.

After running the application, you can view the results in the MVS SYSOUT file, for example, in userid.AGENT_NAME.SYSPRINT (or userid.AGENT_NAME.SYSTSPRT for TSO agents DFMQTSO and DFMXTSO).

Because TSO batch defaults to no prefixing of data set names, you might want to specify "profile prefix(userid)" in either the PARM or SYSTSIN.

Other MVS DFM Sample DataAgent Routines: Sample agent DFMXAGNT, written in C, provides a starting point for user- or vendor-written routines and demonstrates that HLLs can be used to write DataAgent routines.

Sample agent DFMXSORT sorts an input file based on column numbers specified in the parameter list, builds a temporary file consisting of the sorted output, and returns the name of the temporary file to DFM for subsequent retrieval by SMARTdata UTILITIES DFM.

7.3.3.3 START Command Consideration

By default, the start function, START(job_name), will ensure the job_name (or procedure name) starts with DFM before issuing the start command. Because some customer might not have the appropriate RACF (or other vendor security product) definitions defined to tolerate certain commands or procedures, the function defaults to only starting jobs or procedures whose prefix is DFM.

The system PARMLIB DFM00 member contains a new DFSMS/MVS DFM parameter, Start command control. If this parameter is set to *YES*, DFSMS/MVS DFM AGENT and START command support is limited to those jobs or procedures with names that are at least four characters long and that begin with DFM. The variable *RESTRICT_START* defaults to *YES*.

7.3.3.4 Installation Considerations

You will have to determine the appropriate setting for the RESTRICT_START and run the installation jobs for the sample applications. You may also have to preallocate some input or output files for TSO processing.

Because DataAgent routines can initiate long-running processes, Advanced Program-to-Program Communication (APPC) message limit and timeout settings should be adjusted accordingly.

7.3.3.5 DFSMS/MVS System Messages

The new system message issued by DFSMS/MVS DFM is:

GDE007E module ERROR INVOKING FUNCTION:function return_code reason_code

Explanation: During DFM DataAgent processing, the MVS function shown failed with the indicated return and reason codes.

However, common errors have more specific text. For example, if the DataAgent routine cannot be found in JOBLIB, STEPLIB, or LPALIB, ″LOCATING MODULE DataAgent_routine_name″ will be substituted for function, return_code, and reason_code.

System Action: DFM DataAgent processing is terminated.

Operator Response: Notify your System Programmer.

System Programmer Response: Refer to documentation of the indicated function to determine the meaning of the return and reason codes. Refer to the job log for any related console messages that may have occurred at the time of error.

Descriptor Code: 4 (informal system status)

Routing Information: 10 (system/error maintenance)

Chapter 8. DFSMS/MVS Optimizer

The DFSMS Optimizer is not part of the DFSMS/MVS V1R4 Preview announcement; it was announced in November 1994. With DFSMS/MVS V1R4 the Optimizer will be a priced feature, and the code will be shipped with the base product. The Optimizer code has been totally refreshed and it provides support for OS/390 in addition to MVS/DFP Version 3 Release 3.

In this chapter we review the main features of the DFSMS Optimizer.

The DFSMS Optimizer uses historical and real-time data to provide an overall picture of data usage on each system. It gives an installation the ability to understand how it is managing storage today. With that information the installation can then make informed decisions about how it should manage this storage in the future.

The Optimizer uses the following analyzers to provide detailed and summarized information about storage use (Figure 19 on page 116):

- Performance analyzer
- Management class analyzer

The DFSMS Optimizer uses input data from several sources and processes it, using an extract program that merges the data and builds the Optimizer database.

By specifying different filters, you can produce reports that help you build a detailed management picture of your enterprise. You can use the charting facility to produce color charts and graphs from the report data to greatly enhance the value of this information.

In addition, the HSM Monitor/Tuner is available to keep you informed about the status of DFSMShsm′s activity and provides dynamic control and tuning capability.

By utilizing a historical database, the Optimizer can provide information about how the system was used previously and project how it may be used in the future. Real-time data can provide information about how the system is currently being used. The Optimizer provides what-if simulations for performance, capacity planning, system tuning, and the application of SMS technology, both hardware and software.

8.1.1 Input Data for the Optimizer

As shown in Figure 19 on page 116, the Optimizer uses data from Resource Measurement Facility (RMF), Cache RMF Reporter (CRR), system management facility (SMF) (including HSM information), and SMS to help an installation develop a picture of how storage is being managed. The Optimizer builds its database by correlating data from different record types to produce a complete picture of the data processing environment. All of these records are required in order to do that. The data kept in the DFSMSopt database is usually 85-90% reduced from the raw SMF data.

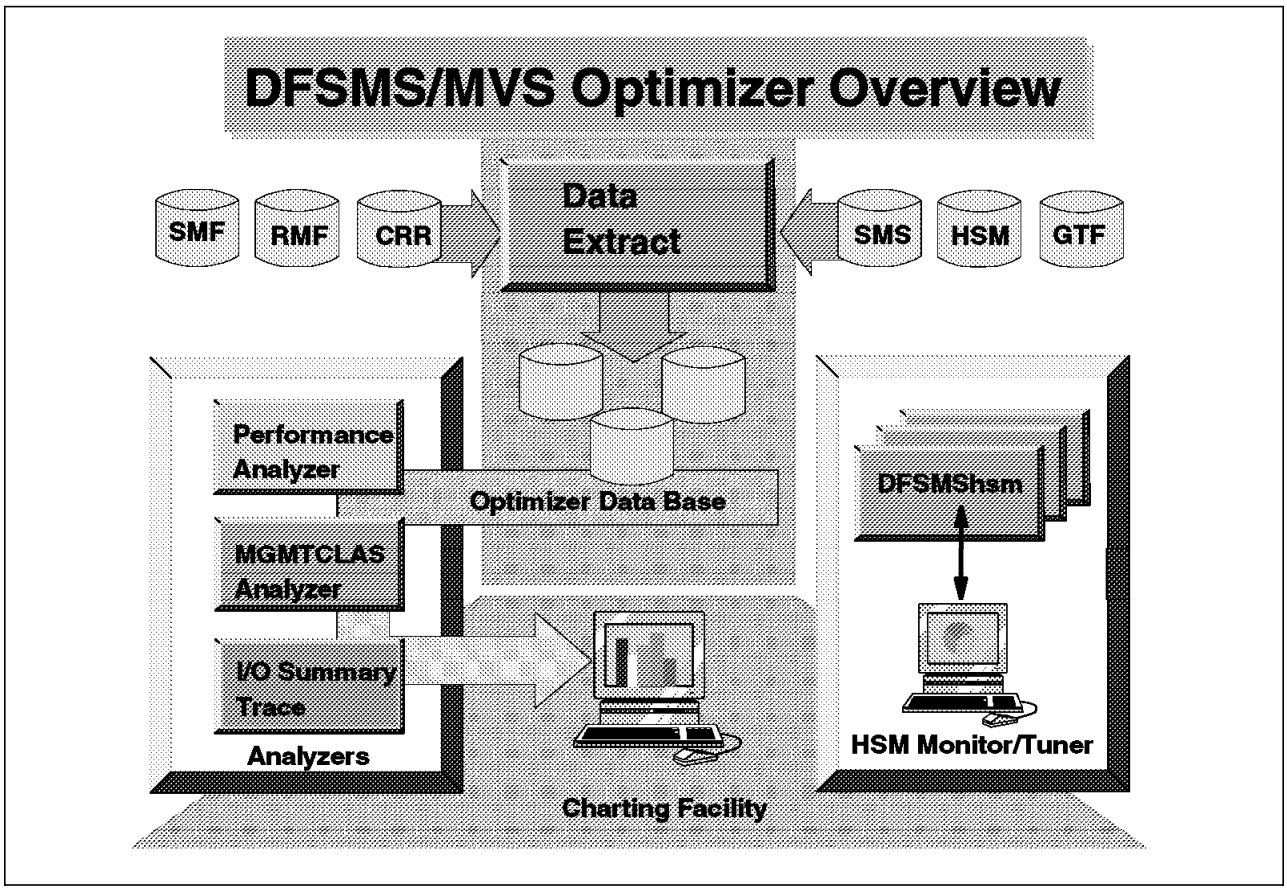


Figure 19. Elements of the DFSMS/MVS Optimizer

8.1.1.1 Data Extract

The required records are as follows:

- SMF records
  - OPEN/CLOSE records (type 14, 15, 62, and 64)
  - STEP/JOB end (type 30, subtypes 4 and 5)
  - Delete/rename (types 17, 18, 65, 66)
  - Data set performance (type 42, subtypes 5 and 6)
  - Tape demount (type 21)
- RMF record types 70 and 74
- Cache RMF Reporter - record 245 (default)
- HSM Functional Statistic Records (FSR) - record 241 (default)

8.1.2 Performance Analyzer

The performance analyzer allows an installation to analyze performance data of the subsystem, volumes, and data sets. Using the data from this analyzer, an installation can determine the causes of performance problems such as device skew, I/O constraints, or I/O-intensive periods. With this information, an installation can make informed decisions about capacity planning, isolate I/O performance problems, and perform what-if simulations for all subsystems in the sysplex.

The performance analyzer helps with:

- *What-if* simulations - identifying the effects of various changes, such as

  - Changing the size of cache storage
  - CPU hardware compression
  - Data set striping
  - Data in memory placement
  - Conversion to extended format data sets
- Current state analysis, including
  - Batch window analysis
  - Reporting on average holding time and cache hit ratios
  - Sorting top data sets by
    - I/O rate
    - Megabytes transferred
    - I/O response time
    - I/O intensity
- Performance cost components
  - Daily megabyte cost of cache and nonvolatile (NVS) storage
  - Daily megabyte cost of expanded storage

8.1.3 Management Class Analyzer

The management class analyzer provides cost-benefit analyses and what-if simulations of management class policies for SMS and non-SMS data. With this analyzer the storage costs and data access patterns are used as input when a report is run.

The management class analyzer can also be used to predict the amount of level 0 (L0) and migration level (ML1) storage that would be required if a new set of migration attributes is used.

Additionally, the management class analyzer can be used to determine the most cost-effective management class assignments for data sets that are to be converted to SMS or to tune the current management class settings to reduce possible DFSMShsm thrashing.

8.1.4 Charting Facility

The charting facility is an OS/2 Presentation Manager program that provides an easy-to-use graphical user interface to download files, view and print color charts, and create slide shows.

While viewing each chart, you can dynamically set attributes—style of chart, colors, text placement—to create the charts that best fit the needs of your installation.

The charting facility accepts any of the report files and automatically converts each file into multiple charts. Color charts are available in various formats including column, tabular, and three dimension.

8.1.5 HSM Monitor/Tuner

Real-time data is the most current data activity on a system. To identify real-time data, an installation must monitor data and system resources. The HSM Monitor/Tuner is a client/server solution that provides real-time monitoring of all major HSM functions on multiple systems.

In addition, the HSM Monitor/Tuner can automatically monitor and tune DFSMShsm through the use of more than 70 trigger events (20 of which are threshold-based) that invoke OS/2 REXX EXECs. These EXECs can be customized to issue commands directly to the host that turned on the trigger event, issue NetView alerts, or perform automated responses to the event.

The HSM Monitor/Tuner OS/2 interface enables all DFSMShsm functions running on each of the associated MVS hosts. It helps you set up system resources correctly to ensure that HSM runs in an efficient and timely manner.

Chapter 9. NaviQuest and Catalog Search Interface

In this chapter we provide an overview of NaviQuest and the Catalog Search Interface (CSI) capabilities. The NaviQuest component of DFSMSdfp assists customers in their initial migration to system-managed storage (SMS) and their ongoing configuration testing assurance and maintenance.

NaviQuest enables customers to perform many storage-related tasks in batch, in support of an SMS environment. Batch support includes:

- Selected DFSMS configuration creation and modification
- Testing - performing regression on changes to ACS routines and the DFSMS configuration
- Translation of ACS routines
- Validation of the DFSMS configuration
- Storage reporting
- Additional automation for ongoing, routine storage tasks

CSI was developed to give users easy read-only access to information stored in ICF catalogs. It provides an alternative to using the AMS LISTCAT command.

9.1 NaviQuest

SMS provides comprehensive and cost-effective management of an installation′s most valuable asset, data. Recognizing that some customers have not yet exploited SMS through the DFSMS/MVS product, additional assistance is being provided through the NaviQuest component of DFSMSdfp.

The NaviQuest component was introduced in DFSMSdfp as an SPE in DFSMS/MVS V1R3.

NaviQuest helps customers who are implementing SMS for the first time. It enables the storage administrator to test policies and configurations before running production data. NaviQuest also helps customers who have already implemented SMS and want to restructure their SMS routines to add new data types to be managed or exploit their SMS investment by using new or additional SMS functions.

NaviQuest offers solutions for simplified DFSMS testing and additional storage management capabilities:

- Familiar ISPF/ISMF panel interface
- SMS implementation assistance
- Fast, easy, bulk test case creation
- Automatic class selection (ACS) testing automation (that is, regression testing)
- Storage reporting assistance
- Additional tools to aid with storage administration tasks.

NaviQuest provides batch execution of several key SMS functions, including:

- Creation of data set and volume listings
- ACS routine translation

- ACS routine validation

The DFSMSdfp NaviQuest component supports the DFSMS Fast Implementation Techniques (DFSMS FIT) methodology but is not restricted to being used with this methodology. DFSMS FIT is documented in these ITSO redbooks:

- *Get DFSMS FIT* (SG24-2568)
- *DFSMS FIT Installation Examples* (SG24-2569)
- *DFSMS FIT Forms and Foils* (SG24-2570)
- *DFSMS FIT Process Guide* (SG24-4478)

Both NaviQuest and DFSMS FIT were developed in conjunction with actual customer system-managed storage implementations.

9.1.1 Enchanced ACS Management Option Panels

In this section we review some of the enhanced ACS Management Option Panels that the NaviQuest component brings in support of SMS. For a more detailed description of all of the fields on each panel, refer to the *DFSMS/MVS Version 1 Release 4 NaviQuest User's Guide* (SC26-7194).

Rather than presenting every panel and discussing every field, in this section we show the main panels as they apply to two main areas for which the NaviQuest component is likely to be used:

- Regression testing for changes to ACS routines and the DFSMS configuration
- Interactive selection of storage functions for batch processing

9.1.1.1 Regression Testing Using NaviQuest

The sequence of panels we present in this section shows how NaviQuest can be used to identify any changes to data class, storage class, management class, and storage group assignments, for a particular grouping of data sets from a data type (for example, TSO, batch production, and test), referred to as a *data subtype*, after an update to the ACS routines.

Your installation probably has an ISMF Saved List that contains the names and attributes of many data sets that are directly associated with TSO. The ISMF Saved List is created from ISMF's Data Set Selection Entry Panel and then on the resulting Data Set List panel by typing SAVE TSOLIST on the command line. When using NaviQuest, it would be normal practice to limit the ISMF Saved List to a single grouping of data sets (or data subtype) that have the same data class, storage class, management class, and storage group policy names.

On the ISMF Primary Option Panel shown in Figure 20 on page 121, choose option 11, Enchanced ACS Management, to go directly to the NaviQuest Primary Option Menu (POM), which includes seven NaviQuest options, plus the option to Exit (see Figure 21 on page 121).

```
 Panel  Help
------------------------------------------------------------------------------
 DGTSMMD2          ISMF PRIMARY OPTION MENU - DFSMS/MVS 1.4
 Enter Selection or Command ===> 11

 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
  6  Storage Group             - Specify Volume Names and Free Space Thresholds
  7  Automatic Class Selection - Specify ACS Routines and Test Criteria
  8  Control Data Set          - Specify System Names and Default Criteria
  9  Aggregate Group           - Specify Data Set Recovery Parameters
 10  Library Management        - Specify Library and Drive Configuration
 11  Enhanced ACS Management   - Perform Enhanced Test/Configuration Management
  C  Data Collection           - Process Data Collection Function
  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 or X to Exit.
```

Figure 20. ISMF Primary Option Menu

```
 Panel  Help
------------------------------------------------------------------------------
 ACBSMDPO ENHANCED ACS MANAGEMENT - NAVIQUEST PRIMARY OPTION MENU
 Enter Selection or Command ===> 1

 Select one of the following options and press Enter:

  1  Test Case Generation
  2  ACS Test Listings Comparison
  3  Enhanced ACS Test Listing
  4  Test Case Update with Test Results
  5  SMS Report Generation
  6  Model Commands Generation
  7  Batch Testing/Configuration Management
  X  Exit









 Use HELP Command for Help; Use END Command or X to Exit.
```

Figure 21. DFSMSdfp NaviQuest Component Primary Option Menu

Choose option 1, Test Case Generation, to get to the Test Case Generation Selection Menu, which, as you can see in Figure 22 on page 122, enables you to create bulk test cases from one of four different sources of data.

Option 3, SMF Data, assumes that the IGDACSSC storage class exit, provided by the CBIPO, which creates and stores SMF Type 127 records and stores them in the SMF data set, is already installed. The IGDACSSC storage class exit is invoked every time a data set is allocated. Hence the data collected reflects only new data sets allocated since the SMF data set itself was defined. Choose option 3 to invoke the ACSTST program, also provided by the CBIPO, to process these Type 127 records to generate the test cases.

You would choose option 4, VMA Extract Data, to generate test cases for tape data sets and tape volumes that should be stored and accessed from a system-managed tape library (that is, the IBM 3494 and/or 3495 Automated Tape Library Data Servers).

In the example in Figure 22, we use option 1, Saved ISMF List. We call the list *TSOLIST*.

```
 Panel  Help
------------------------------------------------------------------------------
 ACBSFLG4             TEST CASE GENERATION SELECTION MENU
 Enter Selection or Command ===>1_____________________________________________

 Select the input data to be used and press Enter:

  1  Saved ISMF List
  2  DCOLLECT Data
  3  SMF Data
  4  VMA Extract Data









 Use HELP Command for Help; Use END Command or X to Exit.
```

Figure 22. Test Case Generation Selection Menu

Figure 23 on page 123 shows the Test Case Generator from Saved ISMF List Entry Panel from which test cases can be generated from a Saved ISMF List. In this example, the DFSMS policies are assigned on the basis of the information contained in the Saved ISMF List, which contains such information as data set name, data set organization, the disk volume on which the data set was stored on, and the size of the data set.

It is also possible to specify other variables on this panel that would apply to all test cases in the Saved ISMF List, such as ddname and program name.

We continue to use the name of the previously Saved ISMF List: TSOLIST. Each test case is stored as a separate member in a library; in this example, the library is 'NAVIQ.ONL.TESTLIB.' Each test case is given a member name (for example, TSOL1, TSOL2, ...).

```
 Panel  Help
------------------------------------------------------------------------------
 ACBDFLG1    TEST CASE GENERATOR FROM SAVED ISMF LIST ENTRY PANEL
 Command ===> ____________________________________________________________

 To generate test cases, specify the following information and press Enter:
   Saved ISMF List . . . . . . TSOLIST_ (Data set list)

   Member Name Prefix  . . . . TSOL     (1 to 4 alpha characters)
   Test Case PDS . . . . . . . 'NAVIQ.ONL.TESTLIB'_____________________
     Replace Existing Prefix . N        (Y or N)

 ACS Test Case Variables:
                                                                  More:    +
   Applic  . . . . ________            DD . . . . . . ________
   Def_dataclas  . ________            Def_mgmtclas . ________
   Def_storclas  . ________            Filenum  . . . ____
   Group . . . . . ________            Job  . . . . . ________
   Label . . . . . ________            Libname  . . . ________
   Mvsgp . . . . . ________            Pgm  . . . . . ________
   Retpd . . . . . ________            Storgrp  . . . ________

 Use DOWN Command to Scroll Forward; Use UP Command to Scroll Backward:
 Use HELP Command for Help; Use END Command to Exit.
```

Figure 23. Test Case Generator from Saved ISMF List Entry Panel

After these test cases have been generated, if a typical member is browsed, it would contain information such as:

```
BROWSE NAVIQ.ONL.TESTLIB(TSOL1)
    DESCRIPTION1:
    TEST CASE CREATED 96/11/04 at 19:58:5 BY LITTLEP
    DSN:      NAVT001.DAILY.REPORT
    DSORG:    PS
    DSTYPE:   PERM
    ACSENVIR: RECALL
    JOB:      DFHSM
    SIZE:     55
    STORCLAS: STD
    MGMTCLAS: STD
    NVOL:     1
    VOL:      01
    MIGRAT
    UNIT:     3390
```

Once the test cases have been generated, you can test them, using option 7, Automatic Class Selection (see Figure 20 on page 121). Save the output from this run in the LISTING data set as it will act as the baseline against which future runs with modified ACS routines are checked.

The full procedure for running regression testing is documented in the *NaviQuest User's Guide* (SC26-7194). Basically whenever the ACS routines are to be changed such that a new data subtype is to be managed, you must do the following:

1. Ensure that the test case library contains the new data subtype.
2. Test these test cases against the new ACS routine, using option 7, Automatic Class Selection, and write the output to a new listing file.

3. Use NaviQuest to perform the regression test as illustrated in Figure 24 on page 124.

To perform the comparison, choose option 2, ACS Test Listings Comparison, from the NaviQuest POM (Figure 21 on page 121), to bring up the ACS Test Listings Comparison Panel as shown in Figure 24.

This panel contains the names of the listing files entered, from before and after the changes were made to the ACS routines, the name of the library used to hold the test cases, and the names of two output data sets:

1. Comparison Results Data Set, a sequential report file used to highlight the differences for the exception cases
2. Exception Test Case PDS, a data set that holds copies of the members that were the *exception* test cases.

The Exception Test Case PDS also holds details of all test cases that either have never been run before or have been given different data class, storage class, management class, or storage group assignments. If these exceptions identify errors in assigning the DFSMS policies, then the ACS routines must be recoded and the process rerun.

Note: The only valid exceptions should be for the new data subtype that has never been through this process before and for which, therefore, expected results are not stored in the test case library for the new test cases.

```
 Panel  Help
------------------------------------------------------------------------------
ACBDFLC1           ACS TEST LISTINGS COMPARISON PANEL
Command ===> ____________________________________________________________

To compare ACS listings, specify the following information and press Enter:
  Input Data Sets:
    Base ACS Test Listing (Before latest ACS routine changes)
      ===> LISTING_____________________________________________
    New ACS Test Listing (After latest ACS routine changes)
      ===> NEW.LISTING_________________________________________
  Reference Data Set for Compare:
    Test Case PDS (Test source for listings above)
      ===> 'NAVIQ.ONL.TESTLIB'_________________________________

  Output Data Sets:
    Comparison Results Data Set (Summary of exception test cases)
      ===> COMPARE.REPORT______________________________________
      Replace Contents if DSN Exists  . . Y (Y or N)
    Exception Test Case PDS (Contents of exception test cases)
      ===> EXCEPTNS____________________________________________
      Replace Contents if DSN Exists  . . Y (Y or N)

Use HELP Command for Help; Use END Command to Exit.
```

Figure 24. ACS Test Listings Comparison Panel

If the ACS routines have logic errors that produce incorrect assignments for any of the data types (or data subtypes), you can use the Enhanced ACS Test Listing Entry Panel (select option 3, Enchanced ACS Test Listing, on the NaviQuest POM) to find further information for each data set that was tested. As you can see in Figure 25 on page 125, you can choose which additional information should be included in the output listing.

```
 Panel  Help
------------------------------------------------------------------------------
ACBDFLX1            ENHANCED ACS TEST LISTING ENTRY PANEL
Command ===>_____________________________________________________________

To generate enhanced ACS test listing, specify the following and press Enter:
  ACS Test Listing
    ===> NEW.LISTING_____________________________________________

  Enhanced ACS Test Listing
    ===> ENHANCED.LISTING________________________________________
    Replace Contents if DSN Exists  . . Y (Y or N)

  Fields to Include in Enhanced ACS Test Listing: (Y or N)

    DSN  . . . . . Y              JOBNAME  . . . . Y
    EXPDT  . . . . N              SIZE . . . . . . N
    UNIT . . . . . Y              PROGRAM  . . . . Y




Use HELP Command for Help; Use END Command to Exit.
```

Figure 25. Enhanced ACS Test Listing Entry Panel

Once you have corrected all of the ACS routine errors and ensured that all exceptions are valid (that is, only for the new data subtypes), you can use NaviQuest to place the results of the test into the test case definitions, as the *saved expected results* for later regression testing.

To save the test results, choose option 4, Test Case Update with Test Results, from the NaviQuest POM to get the Test Case Update with Test Results Entry Panel. Figure 26 on page 126 shows this panel filled in with the:

1. Latest listing (NEW.LISTING) after the changes to the ACS routines
2. Names of the comparison results and exception test case data sets
3. Test case library

The test case members for the exceptions are read and copied into the test bed library. The saved expected results are obtained from the comparison report and are also saved in the test bed library.

```
 Panel  Help
------------------------------------------------------------------------------
ACBDFLU1        TEST CASE UPDATE WITH TEST RESULTS ENTRY PANEL
Command ===>_____________________________________________________________________

To update test cases with test results, specify the following and press Enter:

  Input Data Sets:
    New ACS Test Listing (After latest ACS routine changes)
      ===> NEW.LISTING_____________________________________________
    Comparison Results Data Set (Summary of exception test cases)
      ===> COMPARE.REPORT__________________________________________
    Exception Test Case PDS (Contents of exception test cases)
      ===> EXCEPTN_________________________________________

  Input/Output Data Set:
    Test Case PDS
      ===> 'NAVIQ.ONL.TESTLIB'_____________________________




Use HELP Command for Help; Use END Command to Exit.
```

Figure 26. Test Case Update with Test Results Entry Panel

9.1.1.2 Running Storage Administration Functions in Batch

Running storage administration functions in batch results in a huge productivity improvement. From NaviQuest POM, select option 7, Batch Testing/Configuration Management (see Figure 21 on page 121), to get to the Batch Testing/Configuration Selection Menu (Figure 27 on page 127), where you can choose an appropriate job skeleton to customize, with the option to save it to a JCL library, and run it.

The batch jobs are grouped into four major divisions by function as shown in Figure 27 on page 127. All of the jobs, complete with sample JCL, are stored in the SYS1.SACBCNTL library. Once you have selected the specific type of job to be run, you can edit a copy of the sample JCL from an ISPF Edit panel.

Let's look at an example using NaviQuest to set up a batch job, in this case, creating a data set listing. On the Batch Testing/Configuration Selection Menu, select ISMF option 1, Saved ISMF List Operations Batch Samples, to get to the Saved ISMF List Operations Batch Samples Selection Menu (Figure 28 on page 127).

```
 Panel  Help
------------------------------------------------------------------------------
 ACBSMDJ1    BATCH TESTING/CONFIGURATION SELECTION MENU
 Enter Selection or Command ===> 1

 Select an option or enter Data Set to Edit and press Enter:

   1  Saved ISMF List Operations Batch Samples
   2  DCOLLECT Data Operations Batch Samples
   3  Configuration Changes Batch Samples
   4  VMA and SMF Batch Samples









 Data Set to Edit . .

 Use HELP Command for Help; Use END Command to Exit.
```

Figure 27. Batch Testing/Configuration Management Selection Menu

The panel shows the names of the different sample jobs related to data set and volume lists. You can scroll down to show three further jobs related to producing tape volume lists. In our example, we type *S* next to Create Data Set List.

```
 Panel  Help
------------------------------------------------------------------------------
 ACBSMDJ2     SAVED ISMF LIST OPERATIONS BATCH SAMPLES SELECTION MENU
 Command ===>

 Select an option by typing 'S' or enter Data Set to Edit and press Enter:

                                                                More:     +
   S Create Data Set List
   _ Create Data Set List and Save Query
   _ Create Data Set List from a Saved Query
   _ Generate Test Cases from a Data Set List
   _ Generate Model Commands from a Saved List
   _ Generate Data Set Report
   _ Create Data Set List and Generate Data Set Report
   _ Create DASD Volume List
   _ Create DASD Volume List and Save Query
   _ Create DASD Volume Query from a Saved Query
   _ Generate DASD Volume Report
   _ Create DASD Volume List and Generate DASD Volume Report

 Data Set to Edit . .

 Use HELP Command for Help; Use END Command to Exit.
```

Figure 28. Saved ISMF List Operations Batch Samples Selection Menu

Pressing Enter will put you into ISPF Edit for the sample JCL. Once you have modified the JCL and/or commands, you can submit the job. If you press PF3 (or

END), a panel will be displayed asking you whether you want to save the modified JCL. If you provide the name of the data set and member, you can submit the same job in the future without having to go through the NaviQuest panels.

If you want to define or modify the SMS configuration in batch, select option 3, Configuration Changes Batch Samples, from the Batch Testing/Configuration Selection Menu to get to the Configuration Changes Batch Samples Selection Menu (Figure 29). This panel lists the sample jobs that you can select to define or change various DFSMS policies in the DFSMS configuration. In our example, we select Define/Alter Management Class.

```
 Panel  Help
------------------------------------------------------------------------------
ACBSMDJ5      CONFIGURATION CHANGES BATCH SAMPLES SELECTION MENU
Command ===> ____________________________________________________________

Select an option by typing 'S' or enter Data Set to Edit and press Enter:


  S Define /Alter Management Class
  _ Define/Alter Pool Type Storage Group
  _ Define/Alter Tape type Storage Group
  _ Change Storage Group Volume Status
  _ Translate ACS Routines
  _ Validate SCDS
  _ Test ACS Routines
  _ Generate Enhanced ACS Test Listing
  _ Compare ACS Test Listing
  _ Update Test Cases with Test Results



Data Set to Edit . . ____________________________________________________

Use HELP Command for Help; Use END Command to Exit.
```

Figure 29. Configuration Changes Batch Samples Selection Menu

Selecting this option will put you into ISPF Edit, and the JCL will be displayed as shown in Figure 30 on page 129 and Figure 31 on page 130. Changes will have to be made to the JOBCARD, prefix, and available parameters.
Here are the parameters for defining or altering a management class:

NaviQuest Key Word	ISMF Field Name
SCDS	CDS Name
MGMTCLAS	Management Class Name
DESCR	Description
EXPNOUSE	Expire after Days Non-usage
EXPDTDY	Expire after Date/Days
RETNLIM	Retention Limit
PARTREL	Partial Release
PRINOUSE	Primary Days Non-usage
LV1NOUSE	Level 1 Days Non-usage
CMDORAUT	Command or Auto Migrate
PRIGDGEL	# GDG Elements on Primary
GDGROLL	Rolled-off GDS Action
BACKUPFR	Backup Frequency
NUMBKDSE	Number of Backup Vers (Data Set Exists)

```
NUMBKDSD            Number of Backup Vers
                      (Data Set Deleted)
RETDYDSD            Retain days only Backup Ver
                      (Data Set Deleted)
RETDYEXT            Retain days extra Backup Vers
CMDBKUP             Admin or User command Backup
AUTOBKUP            Auto Backup
BKUPTECH            Backup Copy Technique
```

```
//IBMUSERA JOB (ACCT),'IBMUSER',MSGCLASS=H,
//      NOTIFY=IBMUSER,CLASS=A,MSGLEVEL=(1,1),TIME=(0,10)
//MYLIB JCLLIB ORDER=SYS1.SACBCNTL
//*******************************************************************/
//*$MAC(ACBJBAJ1) COMP(5695DF123): BATCH - MC DEFINE/ALTER           */
//*                                                                 */
//* PROPRIETARY V3 STATEMENT                                        */
//* LICENSED MATERIALS - PROPERTY OF IBM                            */
//* 5695-DF1                                                        */
//* (C) COPYRIGHT 1996  IBM CORP.                                   */
//* END PROPRIETARY V3 STATEMENT                                    */
//*                                                                 */
//*  CHANGE ACTIVITY:                                               */
//*                                                                 */
//*$LO=NAVIQUEST,HACS120,96/06/18,SNJTCS: INITIAL VERSION        @LOA*/
//*                                                                 */
//*******************************************************************/
//*****************************************************************
//*                                                                 *
//* SAMPLE JCL TO DEFINE/ALTER MANAGEMENT CLASSES IN BATCH          *
//*                                                                 *
//*   INSTRUCTIONS BEFORE SUBMITTING:                               *
//*                                                                 *
//*     CHANGE JOBCARD                                              *
//*     CHANGE PREFIX                                               *
//*     CHANGE PARAMETERS                                           *
//*                                                                 *
//*   PARAMETER FOLLOWING ACBQBAJ1 - DEFINE OR ALTER                *
//*   SCDS - SCDS IN WHICH MANAGEMENT CLASS IS TO BE DEFINED/ALTERED *
//*   MGMTCLAS - MANAGEMENT CLASS TO BE DEFINED OR ALTERED          *
//*                                                                 *
//*****************************************************************
//*                                                                 *
//*   STEP1 - SET UP THE PARAMETERS
//*
//*****************************************************************
//STEP1   EXEC ACBJBAOB
//SYSUDUMP DD  SYSOUT=*
//TEMPFILE  DD  DSN=&&TEMPFILE,DISP=(NEW,PASS),
//  SPACE=(TRK,(1,1)),LRECL=300,RECFM=F,BLKSIZE=300
```

Figure 30. First Edit Screen

```
//SYSTSIN  DD *
PROFILE PREFIX(IBMUSER)
ISPSTART CMD(ACBQBAJ1 DEFINE +
SCDS(TEST.CDS) +
MGMTCLAS(STANDARD) +
DESCR(DEFINING STANDARD MGMT CLASS) +
EXPNOUSE() +
EXPDTDY() +
RETNLIM() +
PARTREL() +
PRINOUSE() +
LV1NOUSE() +
CMDORAUT() +
PRIGDGEL() +
GDGROLL() +
BACKUPFR() +
NUMBKDSE() +
NUMBKDSD() +
RETDYDSD() +
RETDYEXT() +
CMDBKUP() +
AUTOBKUP() +
BKUPTECH() +
)
//*********************************************************************
//*
//*   STEP2 - EXECUTE THE DEFINE/ALTER
//*
//*********************************************************************
//STEP2   EXEC ACBJBAOB
//SYSUDUMP DD  SYSOUT=*
//SYSTSIN  DD DSN=&&TEMPFILE,DISP=(OLD,DELETE,DELETE)
//
```

Figure 31. Second Edit Screen. Shows the command and parameters that may be changed

For reference purposes the parameters for changing a pool storage group in batch are:

NaviQuest Key Word	ISMF Field Name
SCDS	CDS Name
STORGRP	Storage Group Name
DESCR	Description
AUTOMIG	Auto Migrate
MIGSYSNM	Migrate Sys/Sys Group Name
AUTOBKUP	Auto Backup
BKUPSYS	Backup Sys/Sys Group Name
AUTODUMP	Auto Dump
DMPSYSNM	Dump Sys/Sys Group Name
DUMPCLAS	Dump Class
HIGHTHRS	Allocation/migration Threshold: High
LOWTHRS	Allocation/migration Threshold: Low
GUARBKFR	Guaranteed Backup Frequency
SGSTATUS	SMS SG Status

DUMPCLAS can accept up to 5 values separated by commas.
SGSTATUS can accept up to 32 values separated by commas.

Note: You can use an asterisk (*) as a subparamter of SGSTATUS to leave the status unmodified for a particular system or sysgroup. So, for example, SGSTATUS(*,*,ENABLE) will change the status of the third system or sysgroup alone, leaving unaltered the status of systems 1, 2, 4, and above.

For reference purposes the parameters for changing a tape storage group in batch are:

NaviQuest Key Word	ISMF Field Name
SCDS	CDS Name
STORGRP	Storage Group Name
DESCR	Description
LIBNAME	Library Names
SGSTATUS	SMS SG Status

LIBNAME can accept up to 8 values separated by commas.

SGSTATUS can accept up to 32 values separated by commas.

Just as with a pool storage group, you can use an asterisk (*) as a subparameter of SGSTATUS to leave the status unmodified for a particular system or sysgroup.

9.1.2 NaviQuest Help Panels

Field level help is provided for all NaviQuest panels. If you press PF1 on an input field, help specific to that field will display. If the cursor is not on an input field, panel help will display.

9.1.3 Migration and Coexistence

For those who have licensed IBM NaviQuest for MVS, the DFSMSdfp NaviQuest component can use the test cases created. However, to use the new fields introduced in DFSMS/MVS V1R3 in the ACS routines and to test for them, new test cases must be created.

9.2 Catalog Search Interface

The CSI was developed as an MVS read-only general-use programming interface (5799-CSI) to enable fast extraction of data from catalogs. It provides an alternative to the AMS LISTCAT command. With DFSMS/MVS V1R4 CSI has been incorporated into DFSMSdpf free of charge to make it available for all customer and vendor applications.

The CSI is used to quickly obtain information about entries in the ICF catalogs. The ICF catalog entries are selected by using a generic filter key provided as input. The generic filter key can be either a fully qualified name that returns one entry or a partially qualified name (wild cards) that returns multiple entries on a single invocation. Several sample programs are provided to facilitate fast and easy use of the CSI programming interface (see 9.2.5, “CSI Sample Programs” on page 137).

9.2.1 Overview

To obtain information about ICF catalog entries, you request field information for each entry by specifying field names. Thus you do not have to know whether the information is in the Basic Catalog Structure (BCS) or the VSAM Volume Data Set (VVDS).

A work area, whose address and size are provided on input, is used to return the selected entries and the field information for those entries. If the work area cannot accommodate all of the entries, as many entries as possible are returned in one invocation, and an indicator is set to reflect that more entries exist. Subsequent invocations can specify that the request is being resumed, and the additional entries can be obtained. This can be repeated until all entries have been returne d. The resume process allows the ferrying of large amounts of information to the user program without affecting ICF catalog resources.

9.2.2 Invocation

CSI can be invoked in either 24-bit or 31-bit addressing mode. CSI is reentrant and reusable. CSI can be invoked in any protection key and in either supervisor or problem state.

The invocation can be done through user application programs written in assembler language or an HLL such as REXX.

- In an assembler program, CSI module IGGCSI00 can be invoked by one of the following methods:
  - A CALL statement that resolves to a V-type address constant
  - A LINK macro
  - LOAD and CALL macros
- In an HLL, CSI can be invoked by instructions that resolve to one of the above mentioned assembler language invocations.

9.2.3 The Parameter List

On invocation, general purpose register (GPR) 1 points to a parameter list, GPR 13 points to a standard register save area 72 bytes long, GPR 14 contains the return address in the caller′s program, and GPR 15 contains the entry point of CSI.

Figure 32 on page 133 shows the CSI paramter list.

- The first word in the parameter list contains an address that points to a 4-byte reason area. On return from CSI, the reason area contains a module identification, reason code, and return code.
- The second word contains the address of the selection criteria data fields. These fields supply information to CSI on invocation and contain information for resume if there is more informati on to be returned than can fit in the user provided work area.
- The third word contains the address of a user work area in which entry information will be returned. The caller must place the size of the work area in the first word of the work area. Although this area has no fixed size, it must be between 1024 bytes and 1,048,575 bytes, inclusive. CSI returns the minimum size necessary to contain one entry, its catalog and fields included. A reason code and return code will be set if a larger work area is needed.

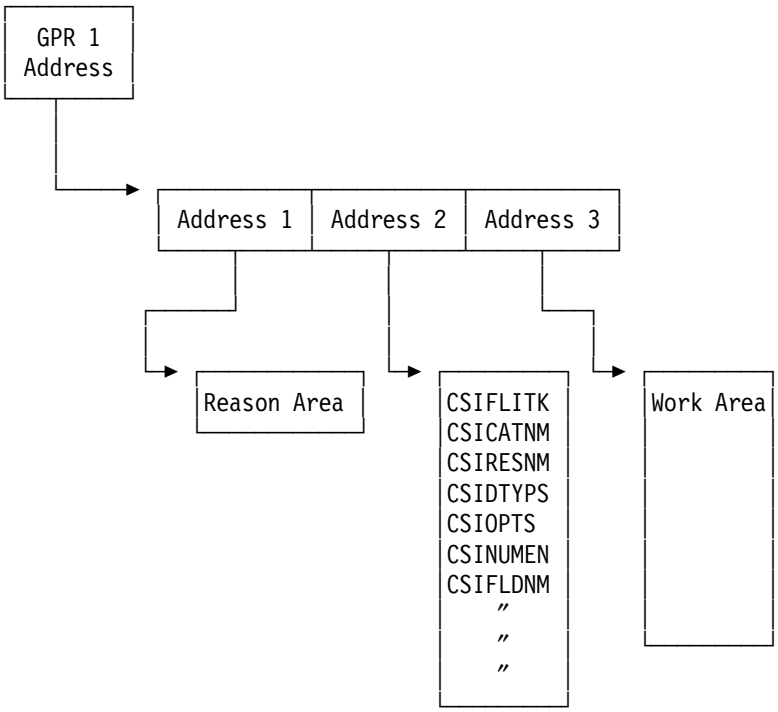


Figure 32. Catalog Search Interface Invocation Structure (Parameter List)

9.2.3.1 Reason Area

On return from CSI, the 4-byte reason area contains a module identifier, reason code, and return code. The pattern of the information in the reason area is:

Description	***Length in Bytes***
Module identification	2
Reason code	1
Return code	1

The contents of the reason area depend on the contents of GPR 15.

Refer to the "Return Status Information" section in the *Catalog Search Interface User's Guide* for a full explanation of return codes in GPR 15.

9.2.3.2 Selection Criteria Data Fields

The selection criteria data fields are your means of communicating with CSI. These fields supply information to CSI during invocation and contain information for a resume if there is more information to be returned than can fit in the user-provided work area. The name of the selection criteria data fields are described below.

Generic Filter Key Field (CSIFILTK): CSI uses a generic filter key supplied in CSIFILTK. A generic filter key is a character string that describes the catalog entry names for which you want information returned. The generic filter key can contain the following symbols:

***** A single asterisk by itself indicates that either a qualifier or one or more characters within a qualifier can occupy that position. An asterisk can precede or follow a set of characters.

****** A double asterisk indicates that zero or more qualifiers can occupy that position. A double asterisk cannot precede or follow any character; it must be preceded or followed by either a period or a blank.

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

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

Catalog Name (CSICATNM): CSICATNM is used for catalog selection. CSI uses the catalog name supplied in the CSICATNM field to search for entries if CSICATNM is not blank. If this field is blank, catalog management attempts to use the high-level qualifier of CSIFILTK to locate an alias or multiple aliases that match. If an alias is found, the user catalog for that alias is searched. Otherwise, the master catalog is searched.

If a tape volume catalog library entry type or a tape volume catalog volume entry type is specified in CSIDTYPS, a tape volume catalog is searched. Tape volume catalog library entry types and tape volume catalog volume entry types should not be mixed with ICF catalog entry types.

Resume Name (CSIRESNM): CSIRESNM is the name of the catalog entry on which the search resumes if the work area space for return information is used up. If the request can be resumed, CSIRESNM is set to *Y* on return from a call to CSI and contains the name of the entry on which to resume, and CSICATNM contains the name of the catalog in which the entry to be resumed was found. Typically, the application program tests CSIRESNM; if it is *Y*, the application program reissues the call to CSI without changing CSIRESNM and CSICATNM.

ICF Catalog Entry Types (CSIDTYPS): CSIDTYPS determines which type of catalog entries will be returned. Valid types for CSIDTYPS are:

A	Non-VSAM data set
B	GDG
C	Cluster
G	Alternate index
H	GDS
R	Path
X	Alias
U	User catalog connector entry
L	ATL library entry
W	ATL volume entry

VSAM components *data* and *index* are returned with the cluster. Thus there are no type specifications for them. However, *D* and *I* types will appear in the output information.

The valid types can be mixed and in any order. Blanks cannot separate the types. For instance, *ABH* might be specified to get only non-VSAM, GDG, and

GDS entries. All other positions in CSIDTYPS must be blank (X′40′) following the specification of the entry types.

All blanks for CSIDTYPS can be set and will get types A, B, C, G, H, R, X, and U. These are the ICF catalog types. L and W must be explicitly specified to get the tape volume catalog entry.

CSI Options (CSIOPTS)

- **CSICLDI** determines whether information will be returned if the data and index names of a cluster do not match the filter key but the cluster name does. If CSICLDI is set to *Y*, the components are returned if the cluster name matches the filter key.
- **CSIRESUM** is set to *Y* if CSI detects that there are more catalog entries than meet the search criteria. This field must be blank on the first invocation. CSIRESUM is tested, and CSI can be called again to obtain more entries. CSIRESUM will be set to blank (X′40′) if no more entries meet the search criteria.
- **CSIS1CAT** causes only one catalog to be searched. It is used in conjunction with CSICATNM to determine which catalogs to search. This option is useful if the filter key could cause catalog management to select more than one catalog for searching. Also, if not set, the master catalog is searched in addition to a selected user catalog. In cases where the catalog name is supplied as input and resume is done, CSIS1CAT causes only that catalog whose name is supplied as input to be searched. Otherwise, on resume, catalog management cannot tell whether this is a search across many catalogs and the resume caused the CSICATNM to be set or the name was supplied by the caller.

Number of Field Names (CSINUMEN): CSINUMEN is a binary number indicating the number of field names following in the subscripted area, CSIFLDNM.

It is possible to get a list of names without any field information. In this case, set CSINUMEN to zero.

There is a limit of 100 field names per invocation of CSI. CSINUMEN cannot be greater than 100.

Field Names (CSIFLDNM): CSIFLDNM is a list of 8-byte field names. If the field name is not eight characters long, it must be padded on the right with blanks to make eight characters. Refer to the “Field Name Directory” section in the *Catalog Search Interface User's Guide* for valid field names that can be used in the list and the information returned for each field name.

9.2.4 Work Area

The third word contains the address of the work area in which entry information will be returned. The caller must place the size of the work area in the first word or the work area. Resume information is returned in the selection criteria data fields. If on return CSIRESUM is set to *Y*, it will be set to the next entry to be processed, and CSICATNM will be the catalog in which that entry will be found. When the application program has processed returned entry information, it should reinvoke CSI to resume at the next entry.

The caller must set CSIUSRLN (length of user-provided work area) before invoking CSI. CSIUSRLN must be a fixed fullword value between 1,024 and 1,048,575 bytes, inclusive.

9.2.4.1 Format Description

The first field is the length of the work area and is set by the user to tell CSI how much data can be returned (see Figure 33 on page 137).

The second field is the minimum required length for a catalog entry and one entry′s worth of returned data. If the minimum length is greater than the work area length, then the work area length must be increased to at least as much as the minimum length. If this is not done, and a resume condition occurs, the user program will appear as if in an endless loop because the same information will be returned for each resume until the first length is increased to contain the entire entry. CSI sees a GDG with all of its associated GDSs as one record. Thus the required length for this entry is apt to be large.

The third field is the amount of space that was used in the work area. CSI always returns a full entry. If the last entry does not fully fit in the remaining work area space, the resume flag is set and the space at the end of the work area is unused. The unused space is usually small.

Next follows the number of field names plus one.

A catalog name entry is returned for every catalog processed. A catalog entry can be identified because its type is X′F0′. This is an artificial type invented so that the next catalog entry can be found. The catalog entry is always followed by return information. The return code portion will be zero if problems were not encountered while processing the catalog during the call.

Following the catalog entry is one or more entries contained in the catalog that match the search criteria (filter key). Each entry has flags, followed by its type and name. If the flags indicate, a module ID, reason code, and return code follow the entry name; otherwise, the field information for the entry follows.

The *MODULE ID / RSN / RC* returned in the work area for each entry is information returned by catalog management.

If no errors or messages occurred, the field information for the entry is returned as a set of lengths with the data corresponding to the lengths.

The first length field is the total length of the length fields including this field and all of the data returned for this entry. The total length field is 2 bytes long. After that is a reserved 2-byte field.

Next are a set of lengths corresponding to the number of fields passed in. Each length is used to determine the length and position of the returned data of the entry. For example, if three field names were supplied on input, there will be three field lengths. Each length will be for the data immediately following the lengths. If the lengths had values 4, 6, and 8, then, following the last length, there would be 4 bytes worth of data for the first field, 4 bytes from the last length field would be 6 bytes of data for the second field, and 10 bytes from the last length would be 8 bytes worth of data for the third field.

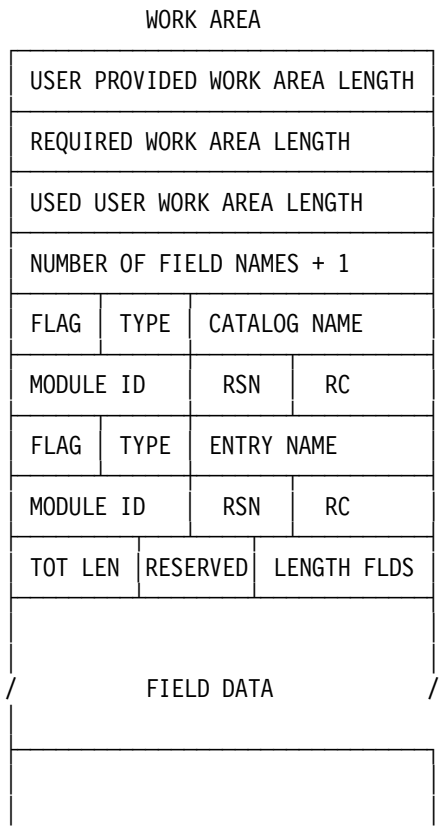


Figure 33. The Catalog Search Interface Work Area

9.2.5 CSI Sample Programs

Three sample assembler programs and one sample REXX EXEC come with the CSI product. You can use the sample programs to try out CSI without having to write your own program. You should be able to get CSI up and running, using these programs with only a few JCL changes. The sample programs are delivered *as is* and are not intended to be part of the licensed product. You may want to modify them to fit your specific needs.

IGGCSILC: IGGISLC is an assembler program that produces output similar to AMS (IDCAMS) LISTCAT NAME.

IGGCSIVG: IGGCSIVG is an assembler program that identifies unused space at the end of a VSAM data set. Basically, it computes the difference of the high-used and the high-allocated RBAs for each VSAM data set in a given catalog. This program is useful for identifying overallocated space.

IGGCSIVS: IGGCSIVS is an assembler program that identifies which data sets on a given volume reside in a particular catalog. In the event of a disk drive failure, this program would be useful in identifying which entries in a catalog have to be cleaned up if the data sets were recovered to a different volser.

IGGCSIRX: IGGCSIRX is a REXX EXEC that uses CSI. When executed, IGGCSIRX prompts you for a filter key. This should be a partially qualified data set name as described for the selection criteria data field CSIFILTK. The data set name, its type, and volser are returned to your TSO session.

Appendix A. Parallel Sysplex and VSAM Record Level Sharing

VSAM RLS extends the DFSMS/MVS storage hierarchy to support data sharing across multiple systems in a System/390 parallel sysplex. Enhancements in DFSMS/MVS V1R4 take advantage of VSAM RLS data access and thus exploit the capabilities of a parallel sysplex. In this appendix we review VSAM RLS and parallel sysplex technology.

A.1 Overview of Parallel Sysplex

A parallel sysplex is a collection of MVS/ESA systems that cooperate using certain hardware and software products to process work in multiple CPCs. Hardware required for a parallel sysplex includes:

- The Sysplex Timer for clock synchronization
- Shared DASD
- One or more coupling facilities and coupling facility links for enabling parallel processing and improved data sharing

The benefits of a parallel sysplex include:

- High-performance data sharing with integrity
- Single system image and point of control
- Continuous availability
- Reduced cost of computing

Note: The coupling facility effectively becomes another part of the storage hierarchy. It is shared by all processors in the sysplex and enables data buffers and locks to be shared across the sysplex.

Major transaction and database systems that can exploit the coupling facility technology within a parallel sysplex are IMS/ESA, DB2, and CICS VSAM RLS. DFSMShsm can now also exploit the functions and performance of VSAM RLS by allowing the migration, backup, and offline control data sets (MCDS, BCDS, and OCDS) to be cached in the coupling facility.

A.2 Overview of VSAM Record Level Sharing

VSAM RLS, a data set accessing mode introduced in DFSMS/MVS V1R3, exploits the locking and caching capabilities of the coupling facility in a parallel sysplex environment. It enables VSAM data to be shared with full update capability among many applications running in many CICS regions.

Other subsystems, such as DFSMShsm, can exploit the performance and functions of VSAM RLS.

VSAM RLS relies on the following four functions, which are provided by other software:

- Central logging

  VSAM data can be accessed by multiple users. For integrity, backup, and recovery purposes, the exploiting subsystem software can choose to log changes to the VSAM data set by using the MVS Logger.

- Central locking

Locking must be provided from a central facility, which is used by all subsystems accessing VSAM data.

- Logical unit of work control

  Unit of work control must be understood by all parties so that they do not work with uncommitted data.

- Retained locks

  Active locks must be retained if a subsystem fails and leaves updated records in an uncommitted state. Retained locks ensure that data integrity is maintained until the failing subsystem (for example, CICS, HSM) has completed its recovery processing.

To use VSAM RLS, the VSAM data set must be SMS managed. The storage class assigned to the VSAM data set determines which cache structures to use in the coupling facility, if the VSAM data set is opened in RLS mode.

VSAM RLS is an access mode that determines a particular manner of buffer management and control, and is mutually exclusive with NSR, LSR, or GSR buffer management. The RLS option is specified by either using a new JCL parameter (RLS=) on the DD card or specifying MACRF=RLS in the ACB.

VSAM RLS supports the KSDS, ESDS, RRDS, and VRRDS file organizations. KSDS and ESDS access is also supported through path access.

RLS specifies that VSAM record level sharing protocols are used and implies that VSAM uses cross-system record level locking, as opposed to CI locking, uses the coupling facility for buffer consistency and performance across the sysplex, and manages a systemwide local cache. With VSAM RLS the coupling facility is used as a store through cache and so the data on the shared DASD always reflects the most recent updates to the data set. When a VSAM data set is accessed in RLS mode, the SHAREOPTION values are totally ignored as the VSAM RLS code always assumes multiple readers and writers and takes responsibility for data integrity.

Currently, when a VSAM data set is opened in RLS mode, it cannot be accessed by any other application in any other mode (NSR, LSR, or GSR) at the same time. NSR reads will be allowed by 3Q97.

DFSMS supports a data sharing subsystem, first introduced in DFSMS/MVS V1R3, called SMSVSAM, which runs in its own address space to provide the RLS support required by the VSAM RLS applications within each MVS image in a parallel sysplex environment.

The SMSVSAM server uses the coupling facility for its cache and lock structures. With VSAM RLS, the application buffers and most VSAM control blocks are allocated in the associated 2GB data space with the SMSVSAM address space. The only VSAM control blocks still in the application address space are the ACB, RPLs, and EXLST.

The SMSVSAM servers (one per MVS image) coordinate with each other to logically maintain a single control block structure for each open VSAM RLS data set across the sysplex and assumes responsibility for maintaining synchronization across the sysplex.

A.3 VSAM RLS Locking

VSAM RLS allows VSAM data to be shared with locking performed at the record level across the sysplex. Thus multiple transactions can update different records even within the same CI without waiting for one another.

Because a data set CI can reside in more than one local buffer, it can be updated concurrently. Because record-level locking is performed, any concurrent updates are against different records in the same CI. The changed records from the multiple copies of the CI are merged to create a copy of the CI that contains all of the changed records. This avoids the update problem where an update can be lost when multiple independent tasks attempt to update the same block of data from their own buffers, not realizing that they do not have exclusive control over this block of data.

Locking at the record level rather than at the CI, CA, or data set level provides a much finer granularity of locking than would otherwise be the case. This helps performance by limiting exclusive lockouts to cases where multiple applications attempt to update the same record.

Appendix B. Special Notices

This publication is intended to provide storage administrators, system programmers, and experienced storage personnel with a technical preview of the functions and enhancements of DFSMS/MVS V1R4. The information in this publication is not intended as the specification of any programming interfaces that are provided by DFSMS/MVS V1R4. See the PUBLICATIONS section of the IBM Programming Announcement for DFSMS/MVS V1R4 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

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

AIX	BookManager
BookMaster	C/370

CBIPO
Common User Access
DB2
DFSMSdfp
DFSMShsm
Hiperspace
MVS/ESA
OS/2
Parallel Sysplex
RAMAC
RMF
S/390
System/390
CICS
CUA
DFSMS/MVS
DFSMSdss
DFSMSrmm
IMS
OpenEdition
OS/390
RACF
Resource Measurement Facility
RS/6000
Sysplex Timer

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, and the Windows 95 logo
are trademarks or registered trademarks of Microsoft Corporation.

Java and HotJava are trademarks of Sun Microsystems, Inc.

Other trademarks are trademarks of their respective companies.

Appendix C. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

C.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 147.

- *DFSMS Optimizer Presentation Guide*, SG24-4477
- *Converting to RMM: A Practical Guide*, SG24-4998
- *DFSMS/MVS V1R3.0 Presentation Guide*, GG24-4391
- *NaviQuest Demonstration and Hands-On Usage Guide*, SG24-4720

C.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

C.3 Other Publications

These publications are also relevant as further information sources:

- *Remote Copy Administrator's Guide and Reference*, SC35-0169
- *DFSMS/MVS V1R3 NaviQuest User's Guide*, SC26-7194
- *DFSMS Optimizer User's Guide and Reference*, SC26-7047
- *Using the Interactive Storage Management Facility*, SC26-4911
- *DFSMS/MVS V1R3 DFM/MVS Guide and Reference*, SC26-4915
- *DFSMS/MVS V1R3 NFS User's Guide*, SC26-7028
- *DFSMS/MVS V1R3 NFS Customization and Operation*, SC26-7029
- *Catalog Search Interface User's Guide*, SC26-7015
- *DFSMS/MVS V1R3 Access Method Services for ICF*, SC26-4906
- *DFSMS/MVS V1R3 Installation Exits*, SC26-4908
- *DFSMS/MVS V1R3 DFSMSdfp Storage Administration Reference*, SC26-4920

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at http://www.redbooks.ibm.com.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO: type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Web Site on the World Wide Web**

  http://w3.itso.ibm.com/redbooks

- **IBM Direct Publications Catalog on the World Wide Web**

  http://www.elink.ibmlink.ibm.com/pbl/pbl

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (http://www.redbooks.ibm.com/redpieces.htm). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1)001-408-256-5422 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com
- **On the World Wide Web**

Redbooks Web Site	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (http://www.redbooks.ibm.com/redpieces.htm). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number
- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

List of Abbreviations

ABARS aggregate backup and recovery support

ABR aggregate backup and recovery

ACB access control block

AMP access method parameter

APA all points addressable

API application programming interface

APPC Advanced Program-to-Program Communication

ATL automated tape library

BCDS backup control data set

BCS basic catalog structure

BWO backup while open

CA control area

CCW channel control word

CDS control data set

CI control interval

CPC central processing complex

CSI catalog search interface

DADSM direct access device space management

DAE dump analysis elimination

DAS data access services

DCB data control block

DFDSS Data Facility Data Set Services

DFHSM Data Facility Hierarchical Storage Manager

DFM Distributed FileManager

DSCB data set control block

EA extended addressibility

EF extended format

FCT file control table

GDG generation data group

GDS generation data set

GPR general purpose register

GRS global resource serialization

GTF generalized trace facility

FEOV force end of volume

HLL high level language

HLQ high level qualifier

IBM International Business Machines Corporation

ICF Integrated Catalog Facility

IOST I/O summary trace

ISMF Interactive Storage Management Facility

ISV independent software vendor

ITSO International Technical Support Organization

JFCB job file control block

KSDS key-sequenced data set

LACS layout analysis and checking system

LAN local area network

LCS library control system

LFS logical file system

LLA library lookaside

LPA link pack area

LPAR logically partitioned mode

LRD last reference date

LSR local shared resource

MCDS migration control data set

MWE management work element

NFS Network File System

NSR nonshared resource

OAM Object Access Method

OCDS output control data set

OCO object code only

OSMC OAM Storage Management Component

OSR object storage and retrieval

PDA problem determination aid

PDS partitioned data set

PFS physical file system

PDSE partitioned data set extended

PO partitioned organization (value of DSORG)

PPRC peer-to-peer remote copy

PROFS Professional Office System

PS physical sequential (value of DSORG)

PSU physical sequential unmovable (value of DSORG)

RBA relative byte address

RLD relocation dictionary

RLS record level sharing

SAF security authorization facility

SDM system data mover

SDR sustained data rate

SdU SMARTdata UTILITIES

SDWA system diagnostic work area

SMB system-managed buffering

SMS system-managed storage

STCK store clock

SVC supervisor call instruction

TCN transportation control number

TIOT task input/output table

TMM tape mount management

TTOC tape table of contents

UCB unit control block

UTC universal time, coordinated

VRS vital record specification

VRSEL vital record selection

VVDS VSAM volume data set

WTOR write to operator with reply

WWFSR ABACKUP/ARECOVER function statistics record

XCF cross system coupling facility

XRC extended remote copy

Index

E

F

G

I

L

M

N

O

P

R

S

T

V

W

ITSO Redbook Evaluation

DFSMS/MVS V1R4 Technical Guide
SG24-4892-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@vnet.ibm.com

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

_____

_____

_____

_____

_____

Printed in U.S.A.