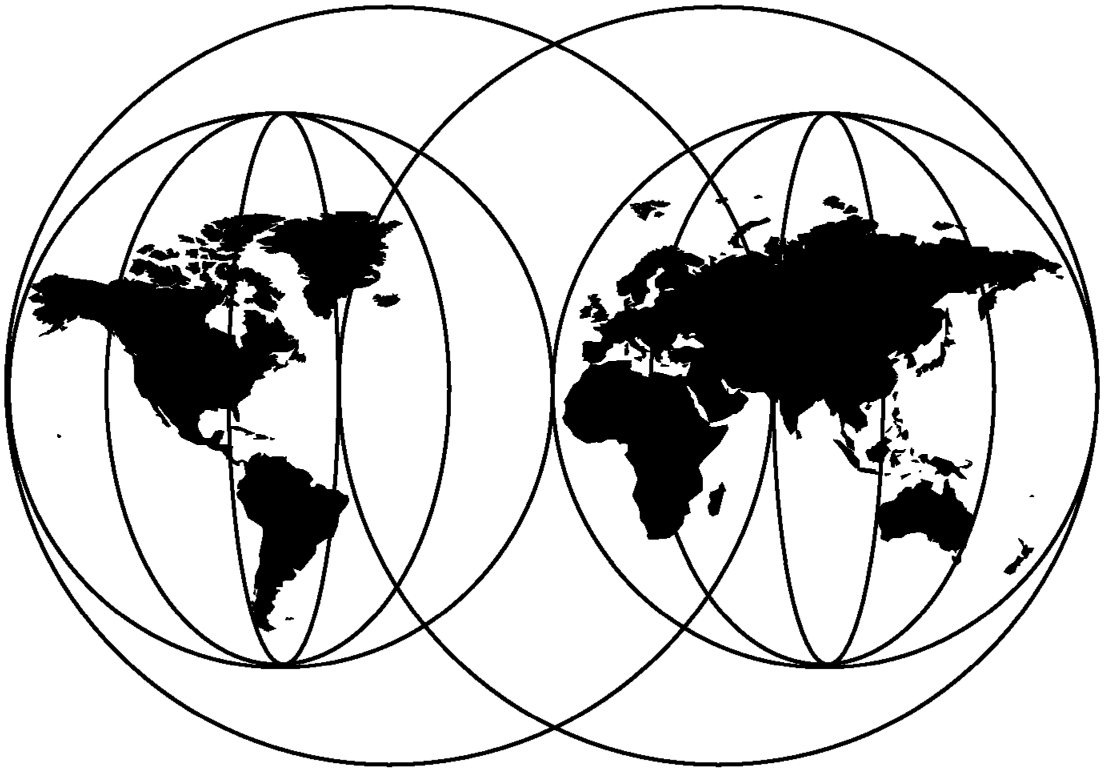
Implementing SnapShot

Alison Pate, Dean Allen, Roar Framhus, Richard Moore, Hannes Tekloth



International Technical Support Organization

http://www.redbooks.ibm.com

SG24-2241-01

International Technical Support Organization

SG24-2241-01

Implementing SnapShot

July 1999

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix E, “Special Notices” on page 187.

Second Edition (July 1999)

This edition applies to Version 1 Release 2 of IBM RAMAC SnapShot for MVS/ESA, Version 1 Release 2 of IBM RAMAC SnapShot for VM/ESA, and Version 1 Release 1 of IXFP/SnapShot for VSE/ESA.

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

Preface vii
The Team That Wrote This Redbook vii
Comments Welcome viii

Chapter 1. Introduction to RVA and SnapShot 1
1.1 Overview of RVA and the Virtual Disk Architecture 1
1.2 Overview of SnapShot 2
1.3 DFSMSdss Support for SnapShot 6

Chapter 2. SnapShot Technical Considerations 9
2.1 SnapShot for MVS/ESA Version 1 Release 2 9
2.2 RVA Extended Addressing 18
2.3 Snapping Data Sets 19
2.4 Supported Data Sets 23
2.5 Snapping Volumes 27
2.6 DFSMS/MVS 32
2.7 Concurrent Copy 33
2.8 DFSMSdss Fast DEFRAG 34
2.9 Data Mover Support 34
2.10 Net Capacity Load Considerations 35
2.11 Performance Implications 36
2.12 General Recommendations 37

Chapter 3. Using SnapShot in Batch 41
3.1 Identifying SnapShot Candidates 42
3.2 Using SnapShot for Data Set Backups 44
3.3 Operational Backup 49
3.4 VSAM KSDS Reorganization 53

Chapter 4. Using SnapShot with DB2 55
4.1 Requirements for Using Virtual Concurrent Copy with DB2 55
4.2 Benefits of Virtual Concurrent Copy for DB2 56
4.3 Benefits of SnapShot with DB2 56
4.4 General DB2 Considerations 57
4.5 Technical Considerations 60
4.6 Sample DB2 Backup and Recovery Procedures Using SnapShot 61
4.7 Application Testing 67
4.8 Using SnapShot for DB2 System Changes and Error Recovery 68
4.9 Cloning DB2 Data 70

Chapter 5. Using SnapShot with IMS/ESA 75
5.1 Using Virtual Concurrent Copy for IMS 75
5.2 Overview of IMS Databases 76
5.3 Overview of IMS Image Copy Procedures 76
5.4 Using SnapShot to Duplicate IMS Databases 85

Chapter 6. Using SnapShot for Disaster Recovery 87
6.2 Using SnapShot for Data Set Backups 90
6.3 Using SnapShot for Full Volume Dumps 90
6.4 Using SnapShot with Physical Full Volume Restores 98
6.5 Using SnapShot with DFSMShsm ABARS 101

Chapter 7. Using SnapShot for Year 2000 Testing 105
7.1 Year 2000 Test Environment 106
7.2 Year 2000 and SnapShot 107
7.3 Using SnapShot to Duplicate Database Environments 109
7.4 Snapping Database Volumes 112
7.5 Using SnapShot and XRC for Year 2000 Testing 113
7.6 Using SnapShot for DB2 Year 2000 Testing 114

Chapter 8. Using SnapShot for System Changes 115
8.1 System Backup 115
8.2 Recovery 116
8.3 Integrity 117
8.4 Catalogs 117

Chapter 9. Using SnapShot in a VM/ESA Environment 119
9.1 Installing SnapShot for VM/ESA 120
9.2 Running SnapShot for VM/ESA 120
9.3 Snapping Volumes 120
9.4 Snapping Minidisks 122
9.5 Instant Format 123
9.6 Guest Support 124
9.7 Integrity 125
9.8 Recovery 125
9.9 RVA Services Utility SIBVMRVA 126
9.10 SIBTRAP 126

| **Chapter 10. Using IXFP/SnapShot in a VSE/ESA Environment** 129
| 10.1 Installing IXFP/SnapShot for VSE/ESA 129
| 10.2 IXFP/SnapShot for VSE/ESA 131
| 10.3 Differences Between IXFP/SnapShot and IXFP/SnapShot for VSE/ESA 139

Appendix A. IXFP SMF Record Subtype 8 141

Appendix B. Sample SNAPDSSU Procedure 143
B.1 SNAPDSSU Command Syntax 143
B.2 Hints and Tips 148
B.3 Sample SNAPDSSU Output 150
B.4 SNAPDSSU REXX EXEC 153

Appendix C. DB2 Overview 169

Appendix D. Sample ABARS Procedure Using SnapShot and REXX 183

Appendix E. Special Notices 187

Appendix F. Related Publications 191
F.1 International Technical Support Organization Publications 191
F.2 Redbooks on CD-ROMs 191
F.3 Other Publications 191

How to Get ITSO Redbooks 193
IBM Redbook Fax Order Form 194

Index 195

ITSO Redbook Evaluation 197

Preface

Implementing SnapShot is written for storage administrators, system programmers, database administrators and technical operations managers and staff. It is an implementation guide for RAMAC SnapShot for MVS/ESA, RAMAC SnapShot for VM/ESA and IXFP/SnapShot for VSE/ESA.

SnapShot is a virtual data duplicator that enables you to make a copy of data by copying the logical pointers to the data. As no data movement takes place it is a very fast process and uses no additional disk space to make the copy.

This redbook provides detailed technical and operational guidance for implementing SnapShot in a variety of workloads and environments. It makes recommendations on the different ways you can implement SnapShot and positions SnapShot against current data duplication techniques and tools. Sample code is provided where applicable to facilitate the task of implementation.

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.

Alison Pate is a project leader in the International Technical Support Organization, San Jose Center. She joined IBM in 1985 after completing an MSc in Information Technology. A large-systems storage specialist since 1990, Alison has seven years of practical experience in using and implementing IBM DASD solutions. She has acted as a consultant to some of the largest, leading-edge customers in the United Kingdom, providing technical support and guidance to their key business projects.

Dean Allen is a Storage System Specialist in Australia. He has eight years of experience in storage management and system operations. His areas of expertise include support of DASD, DFSMS/MVS, and related storage products.

Roar Framhus is a Senior Systems Engineer in Norway. He has more than 30 years of experience in the field. He joined IBM in 1982 as a large systems specialist. His areas of expertise include many S/390 hardware and software products. He has written extensively and taught classes and workshops worldwide on S/390 hardware and software products.

Richard Moore is a DB2 system programmer in the United Kingdom. He joined IBM in 1979 as a courier/store man. His areas of expertise include systems operation, application support, and storage management. He holds a software patent for a storage automation product.

Hannes Tekloth is a Systems Engineer in Germany. He joined IBM in 1963 as a Customer Engineer. Since 1972 he concentrated his activities on the software service until he became an Instructor for SW Specialists. In 1981 he became a Systems Engineer, responsible mainly for VM and VSE customers, and since 1985, for MVS customers. During this time Hannes gained practical experience with storage hardware and software. Now he is an expert in storage performance and solutions and works with many large customers in Germany.

Thanks to the many reviewers and collaborators who were involved in this project. In particular thanks to:

Silvio Podcameni, International Technical Support Organization, San Jose Center
Ravi Kumar, International Technical Support Organization, San Jose Center
Geoff Nichols, International Technical Support Organization, San Jose Center
Mary Lovelace, International Technical Support Organization, San Jose Center
Toru Yamazaki, International Technical Support Organization, San Jose Center
Maggie Cutler, International Technical Support Organization, San Jose Center

Patrick Tomsula, Storage Technology Corporation
Keith Turner, IBM Global Services, UK
Gillian Docherty, Storage Solutions Division, UK
Pete Sadler, IBM UK
Roger Miller, Software Systems Division, Santa Theresa Laboratory
Janet Anglin, Storage Systems Division, San Jose
Terri Leamon, Storage Systems Division, San Jose
Mike Downie, IBM Education and Training, Boulder
Dale Freeman, IBM Global Services, San Jose
Bill Worthington, Advanced Technical Support, San Jose
Charlie Burger, Advanced Technical Support, San Jose
Brian Smith, Storage Systems Division, San Jose
John Thompson, DSS/SDM Development, Tucson
Andy Allen, Storage Solutions Divison, UK
Lee Siegmund, Relational Consultant, Dallas Systems Center, Dallas
Judy Ruby-Brown, Relational Consultant, Dallas Systems Center, Dallas
Brian Sherman, IBM Canada
Eneo Baborsky, IBM Italy
David Sacks, Storage Sales Specialist, Chicago
Dave Shepherd, Storage Technology Corporation, Boulder
Scott Ledbetter, Storage Technology Corporation, Boulder
Corinne Veltien, Storage Technology Corporation, Boulder
Mike White, Storage Technology Corporation, Boulder
Sheila Hanna, Storage Technology Corporation, Boulder
Bob Eskenberry, Storage Technology Corporation, Boulder
Jack Flynn, Storage Systems Division, San Jose
David Short, Storage Subsystems Division, San Jose

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 197 to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Send your comments in an Internet note to `redbook@us.ibm.com`

Chapter 1. Introduction to RVA and SnapShot

In this chapter we introduce the RAMAC Virtual Array (RVA) and SnapShot. Functions are explained here on a level that is, we believe, needed to understand the general concept of SnapShot. If you would like more detailed descriptions of this interesting virtual disk architecture, please see the RVA redbook, *IBM RAMAC Virtual Array,* , SG24-4951 for more details.

1.1 Overview of RVA and the Virtual Disk Architecture

Traditional storage subsystems, such as the 3880, 3990, and RAMAC Array DASD family, use the count key data (CKD) architecture. The CKD architecture defines how and where on the disk device the data is stored. Any updates to the data are written directly to the same position on the physical disk where the updated data was originally. This is referred to as *update in place*.

IBM′s RVA provides a high-availability, high-performance storage solution, thanks to its revolutionary *virtual disk architecture*. To the host, the RVA appears as up to four traditional 3990 subsystems, with up to 1024 3380 and/or 3390 volumes. These devices do not physically exist in the subsystem and are referred to as *functional devices*. In reality, the subsystem contains RAID-protected arrays of fixed block architecture (FBA) disk devices.

The RVA translates I/O requests from CKD format to FBA format.

1.1.1 Log Structured File

The RAID-protected FBA disk arrays that make up the RVA′s physical disk space are sequentially filled with data. New and updated data is placed at the end, as it is on a sequential or log file. We call this a *log structured file*.

Updates leave areas of data that are no longer needed in the RVA log file. A microcode recycle process called *freespace collection* ensures that these areas are removed so that there is always enough freespace for writing. You can control the freespace by observing the net capacity load (NCL) of the RVA and using the IBM Extended Facility Product (IXFP) program. See the RVA redbook *IBM RAMAC Virtual Array*, SG24-4951 for more information. The RVA′s physical disk space typically should not be filled above 75% NCL. Above that level, the freespace collection process runs with higher priority, and performance degradation may result.

The following tables are used to map the tracks of functional devices to the FBA blocks related to those tracks:

- *Functional device table*
  The functional device table (FDT) holds the information about the functional volumes that have been defined to the RVA.
- *Functional track directory*
  The functional track directory (FTD) is the collective name for two tables that together map each functional track to an area in the RVA′s physical storage:
  - *Functional track table*
    The functional track table (FTT) contains the host-related, that is, the functional-device-related track pointers of the FTD.

– *Track number table*
  The track number table (TNT) contains the back-end data pointers of the FTD. A *reference counter* is also part of this table.

Although the FTD consists of two tables, we discuss it as one entity. In fact, each functional track has an entry in the FTD. If a functional track contains data, its associated FTD entry has a pointer to the FBA block where the data starts (see Figure 1). The data of a functional track may fit on one or more FBA blocks.

If the data of a functional track is updated, the RVA puts the new data in a new place in the log structured file, and the FTD entry points to the new data location. There is no update in place.

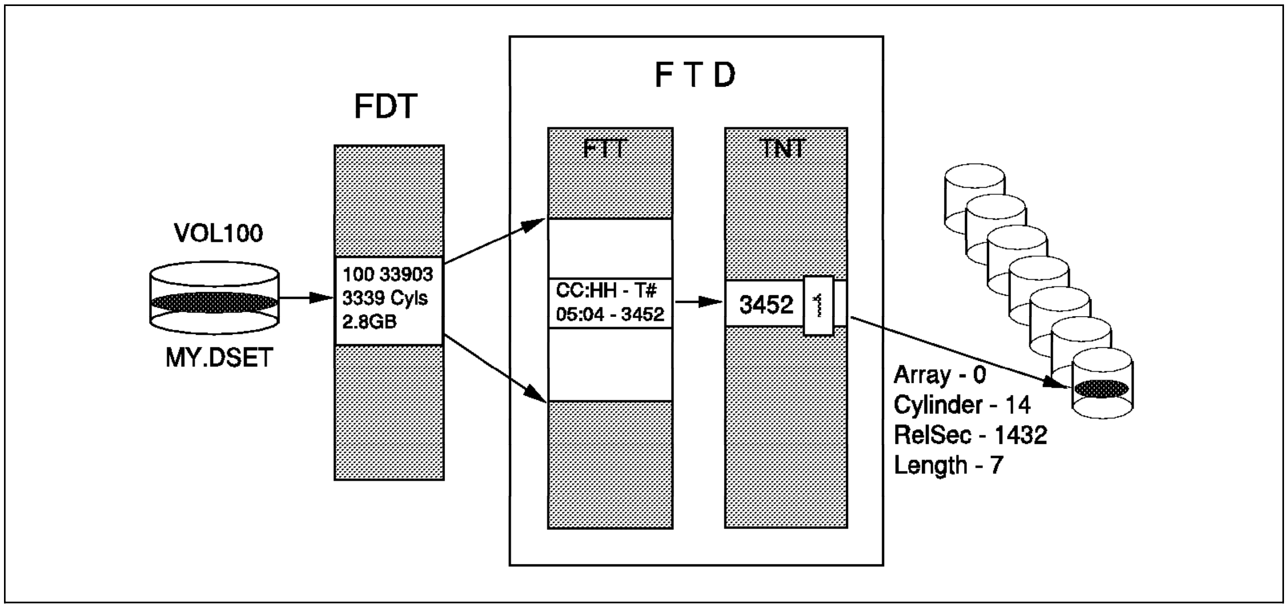


Figure 1. The RVA Tables

1.1.2 Data Compression and Compaction

Because update in place does not occur in the log structured file, data can be compressed. In fact, all data, as soon as it enters an RVA, is subject to compression. In addition, the gaps between the records, as they appear on the CKD devices, are removed before the data is written to disk. This is called *compaction*. Experience shows that RVA customers can achieve an overall compression and compaction ratio of 4:1.

1.2 Overview of SnapShot

The RVA′s virtual disk architecture enables you to produce almost instantaneous copies of data sets, volumes, and VM minidisks. We call this function *SnapShot*. Snapshot produces copies without data movement. We call making a copy with SnapShot a *snap*, the result of a SnapShot is also called a *snap*.

Conventional methods of copying data on DASD consist of making a physical copy of the data on either DASD or tape. Host processors, channels, tape, and

DASD controllers are involved in these conventional copy processes. Copying may take a lot of time, depending on available system resources.

With SnapShot, there is no movement of the actual data. Snapping can take just seconds rather than minutes or hours, because data is not moved and the host processor and the channels are not involved in the data transfer. Furthermore, additional physical disk space is not required to accommodate the snap until either the source or the target is updated. As far as the operating system is concerned, the snap is a real copy of the data; as far as the RVA hardware is concerned, it is a virtual copy of the data.

1.2.1 SnapShot Copy

A functional device is represented by a certain number of pointers in the FTD. Every used track has a pointer in the FDT to its back-end data.

The RVA′s virtual disk architecture enables you to make a copy of a data set, a VM minidisk, or volume, by copying its FTD pointers. As you can imagine, this is a very fast process. No data movement takes place, and no additional back-end physical space is used. Both FTD pointers, the original and the copy, point to the same physical data location (see Figure 2 on page 4).

Only when either the original or the copied track is updated is its associated FTD pointer changed to point to the new data location. The other FTD pointer remains unchanged. Additional space is needed in this case. As long as there is a pointer to a data block in the physical disk storage, the block cannot become free for the freespace collection process. A reference counter in the TNT prevents the block from becoming free.

On the RVA side, the SnapShot function is quite simple; on the MVS side, some additional operations are necessary. If a data set is snapped, a new data set name is required and given in the snap command. After the RVA performs the snap, MVS must update the catalog and VTOC, and the VVDS if the snap is performed on a VSAM data set.

Although the creation of the copy is a logical manipulation of pointers, the data exists on disk so in this sense is not a logical copy. As far as the operating system is concerned, the two copies are completely independent of each other.

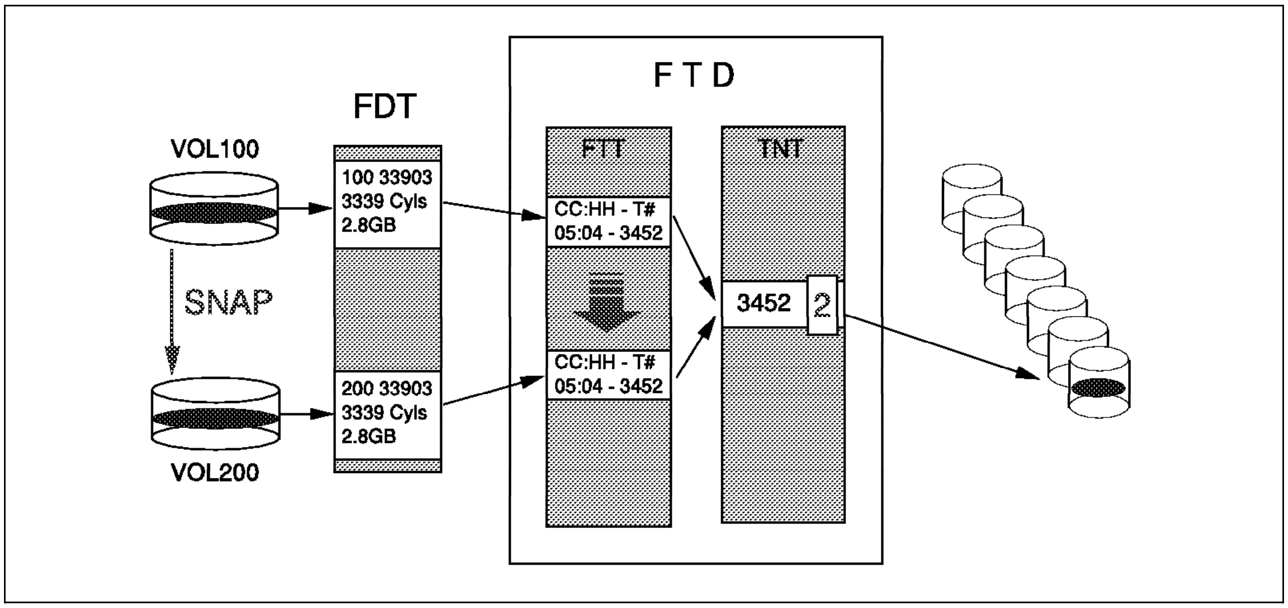


Figure 2. Data Set Snap. SnapShot creates a logical copy by copying the FTD pointers.

When you make a duplicate with SnapShot, both the source and the target must be within the same RVA. (If you are using test partitions, the source and the target must also be in the same partition.)

1.2.2 Invoking SnapShot

SnapShot is a combination of software and RVA microcode functions. It is invoked from the host and executed in the RVA. It can snap entire DASD volumes, VM minidisks, or data sets.

In an MVS/ESA environment, you invoke SnapShot commands with IXFP from:

- A batch job
- A REXX EXEC or CLIST
- The TSO environment

In a VM/ESA environment, you invoke SnapShot commands with IXFP from:

- A REXX EXEC or CLIST
- CMS command

In a VSE/ESA environment, you invoke SnapShot commands with IXFP from:

- An attention routine command
- A batch job
- A REXX EXEC or CLIST

SnapShot commands can be set up to be invoked by end users as well as by database administrators and system programmers. See the *IBM RAMAC SnapShot for MVS/ESA Installation* , SC26-7174 or *IBM RAMAC SnapShot for VM/ESA Installation and Using SnapShot* , SC26-7217 manuals for more information.

1.2.3 Hardware and Software Requirements for SnapShot

SnapShot is provided through a combination of hardware and software. You need to have both the correct hardware feature at the correct level of Licensed Internal Code (LIC) and the software program for your environment.

1.2.3.1 Hardware

To use SnapShot you need an RVA with the SnapShot enablement feature #6001 installed. This is a chargeable feature.

You also need to ensure that you are at the correct level of LIC. SnapShot Release 1.2 requires a minimum LIC level of 4.02.15 or later.

If you want to enable SnapShot V1.2 on a StorageTek 9200, you must submit an RPQ to IBM for each serial-numbered machine.

1.2.3.2 Software

To use SnapShot you need to have the correct SnapShot program product for your environment:

- In an OS/390 or MVS/ESA environment, SnapShot is provided by IBM RAMAC SnapShot for MVS/ESA, 5648-A12, which has the following prerequisites:

  – IBM Extended Facilities Product Version 2, 5648-A17, which in turn requires:
  – OS/390 Version 1 Release 1, 5645-001, or later

  or

  – MVS/SP 4.2.0 (5695-047/5695-048), and MVS/DFP 3.1.0 (5665-XA3) or MVS/DFP V3.1.0, 5665-XA3 or later

  Informational APARs II10922 (for IXFP) and II10923 (for SnapShot) detail the latest service recommendations.

- In an VM/ESA environment, SnapShot is provided by IBM RAMAC SnapShot for VM/ESA, 5654-A03, which has the following prerequisites:

  – IBM Extended Facilities Product, V2 R1, plus PTF L170471, 5648-A17, which in turn requires:
    - VM/ESA V1, R2.2, 5684-112
    - ISPF V3.2, 5684-043

- In an VSE/ESA enviroment, SnapShot is provided by IXFP/SnapShot for VSE/ESA which has the following prerequisites:

  – VSE/ESA 2.3.0 (5690-VSE) installed with VSE/AF Supervisor Level DY44820 or higher.
  – If IXFP/SnapShot for VSE/ESA is used in a VM/ESA environment, VM APAR VM61586 is required for mini-disk caching (MDC) support.

Before installing SnapShot, check with your local IBM support center to get the latest information about the product and its prerequisites.

1.3 DFSMSdss Support for SnapShot

Enhancements to DFSMSdfp and DFSMSdss provide an integration of concurrent copy and SnapShot. The new functions are delivered through a small product enhancement (SPE) to DFSMS/MVS.

1.3.1 DFSMSdss SnapShot

DFSMSdss SnapShot enables you to take advantage of SnapShot when you are copying data within a single RAMAC Virtual Array (RVA). When you specify DFSMSdss COPY, and both the source and target data sets are in the same RVA, SnapShot is automatically invoked.

Figure 3 shows the DFSMSdss SnapShot operation.

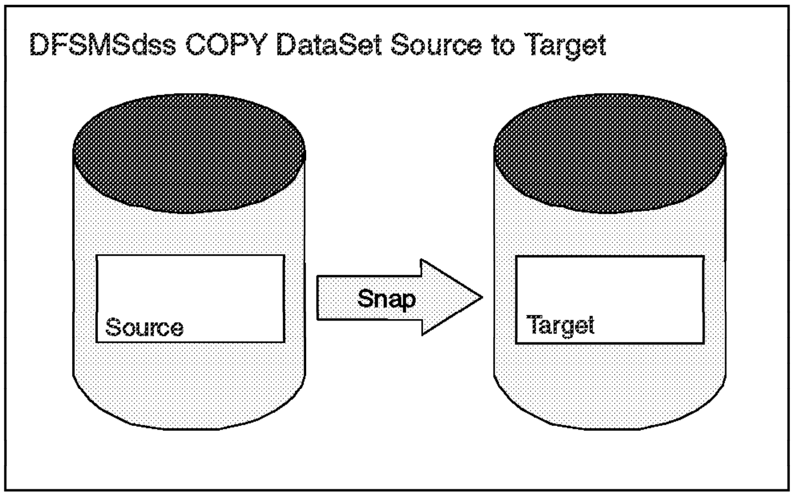


Figure 3. DFSMSdss SnapShot Operation

There are no changes to JCL, and your copy jobs instantly benefit from SnapShot, both in terms of the speed of the copies and the fact that SnapShot does not use any additional back-end space to make a copy.

1.3.2 Virtual Concurrent Copy

Virtual concurrent copy extends the benefits of concurrent copy to users who have RVA installed with SnapShot. When you specify the CONCURRENT keyword on a DFSMSdss COPY or DUMP statement, the software can detect whether you have a 3990 storage control or an RVA. If you have an RVA, the virtual concurrent copy function is invoked. If all the criteria are met for DFSMSdss SnapShot, a DFSMSdss SnapShot will be performed in preference to virtual concurrent copy.

The logical completion of the point-in-time copy occurs when the source data is snapped into an interim data set called the *working space data set* (WSDS). The physical completion occurs when the data is moved by DFSMSdss to the target tape or disk data set. Once the copy is logically complete, the data can be made available for application updates.

Figure 4 on page 7 shows the virtual concurrent copy operation.

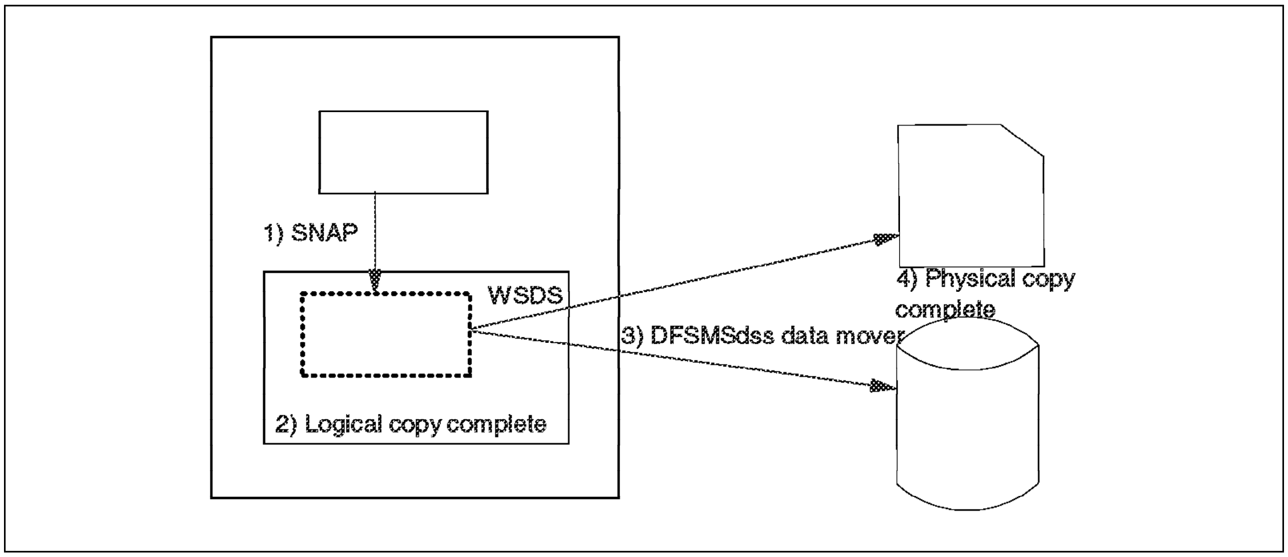


Figure 4. Virtual Concurrent Copy Operation

If you are already using concurrent copy, you do not have to make changes to your JCL to use virtual concurrent copy.

Concurrent copy support is incorporated into the backup and recovery utilities of DB2, IMS, and CICS, so virtual concurrent copy can take advantage of this support immediately and without any changes.

1.3.3 Concurrent Copy

Concurrent copy is a function of the 3990 Storage Control that essentially has similar benefits to SnapShot. Concurrent copy enables you to take a consistent copy or dump of data, using DFSMSdss, while applications are updating the data.

Concurrent copy is provided through a combination of 3990 Licensed Internal Code and System Data Mover and DFSMSdss software. It uses both 3990 cache and expanded storage to track updates that are made to the data, to ensure that a point-in-time copy is created, even though the source data may be updated.

Concurrent copy introduces the concept of a *logical* and *physical* completion to a copy operation. Logical completion occurs once a concurrent copy session is established. Serialization is held on the data until the session is established. Once all the extent information is secured, serialization is released. Physical completion occurs once the data is written to the output data set, whether on tape or disk. The process of physical copy can take place while the data is being updated by other applications, without affecting the point-in-time copy that has been made with concurrent copy.

For more details on the concurrent copy architecture and operation, see *Implementing Concurrent Copy,* GG24-3990.

1.3.4 Prerequisites for DFSMSdss SnapShot and Virtual Concurrent Copy

In order to take advantage of DFSMSdss Snapshot and virtual concurrent copy, you must have the required hardware and software support for SnapShot, as well as the software support for virtual concurrent copy and DFSMSdss SnapShot.

The following prerequisites are listed below. See the MVSSNAP120, MVSIXFP210, and 9393DEVICE PSP buckets for the latest software and microcode requirements.

1. IXFP 2.1 with minimum PTF level at L170019
2. SnapShot 1.2 at the current field level
3. DFSMS/MVS 1.3 or higher with the dss license enabled:
    - OW28059 - device support
    - OW29883 - DFSMSdss support
    - OW30320 - SDM support
    - OW29881 - SDM support
    - OW26926 - DFSMS support
    - OW32415 - Target multi-volume VSAM support
    - OW33611 - WSDS sharing enhancement
4. RVA microcode level 4.3.26 or higher
    - The RVA also requires the SnapShot enablement feature 6001.

The activation of the above support requires an IPL. For the microcode upgrade, an IML is also required.

Chapter 2. SnapShot Technical Considerations

In this chapter we explain the enhancements to SnapShot that are available in Release 1.2. We also discuss the technical considerations of using SnapShot and some of the areas where you can use SnapShot.

2.1 SnapShot for MVS/ESA Version 1 Release 2

In this section we discuss the items introduced in SnapShot for MVS/ESA Version 1 Release 2, which was available in December 1997. The major enhancements in this release are:

- Support for individual VSAM data sets
- Support for multivolume data sets
- Support for extended format data sets

  APARs OW33143 (for IXFP) and OW33144 (for SnapShot) add SNAP DATASET support for extended format VSAM data sets.

- Alternate indexes (AIs)

  PTF L170650 for SnapShot provides support for AIs in the RELATE parameter.

- SMS support
- Command syntax changes
- Command syntax additions

There are also some enhancements to IXFP delivered by ptf L170017 that improve reporting for SnapShot:

- Enhancement to IXFP space utilization reports
- New IXFP SMF record subtype 8

2.1.1 VSAM Data Sets

You can use the SNAP DATASET command to snap the following types of VSAM data sets:

- Key-sequenced data sets (KSDSs)
- Entry-sequenced data sets (ESDSs)
- Linear data sets (LDSs)
- Relative record data sets (RRDSs)
- Variable relative record data sets (VRRDSs)

The snap is taken at the cluster level, and the source must contain data. The target is opened with SHAREOPTION(1,1). If you are snapping a VSAM data set that has an alternate index (AI), be aware that SNAP DATASET does not support AI for data set snap. Only the VSAM cluster will be snapped.

You can use SnapShot to snap individual AIs by name. You cannot snap a sphere (a related collection of base clusters and AIs) or a path (a unique catalog entry that provides a named relationship between the base cluster and the AI).

To snap a VSAM data set with an AI you must snap the base cluster and the AI separately and use the RELATE parameter to associate the AI with the cluster. For details on the RELATE parameter see 2.1.6, “Command Syntax Additions for SNAP DATASET” on page 14.

After snapping a base cluster and an AI you may need to use IDCAMS DEFINE PATH to define a path through which the AI can be used to access the base cluster. Because you cannot snap the base cluster and all its AIs at the same time, you must take care to maintain data integrity. Update activity to the base cluster and AI should be quiesced while the base cluster and AI are snapped.

SnapShot automatically generates target component names for VSAM. If DB2 naming conventions are used, SnapShot creates target component names according to the DB2 convention.

2.1.2 Multivolume Data Sets

You can use the SNAP DATASET command to snap multivolume data sets. Included in this support is space allocation optimization for dynamically allocated target data sets and provisions for when the source data set spans multiple RVAs or non-RVA subsystems.

For all multivolume snaps, the extent size, the number of extents, and the number of volumes on the target are different from those on the source data set and depend on the results of the target data set allocation.

The primary objective of SnapShot is to ensure that there is enough space in the target data set so that the snap can complete successfully. The secondary objective is to ensure that as much data as possible can be snapped.

The results that you achieve depend on whether the data set is SMS managed, whether the target is dynamically allocated or preallocated, and the data set type.

In general, for multivolume data sets, the data mover should always be specified. For SMS managed data sets, volume preferencing should be used.

If a data mover is not specified, and only part of the data set is eligible for a snap, the snap operation will fail, thereby preventing SnapShot from creating a partial snapped data set.

2.1.2.1 Target Data Set Allocation

When allocating a target data set, SnapShot attempts to maximize the probability that the allocation will be successful. As with all data set allocations, the ability to obtain space varies according to the number of volumes available, the amount of space on the volume, and the primary and secondary allocation quantities.

Here are the variables possible:

- SMS managed, dynamically allocated target data set

  SnapShot makes an initial storage request for the primary allocation amount of the source data set. Additional space requests use the extent processing functions of the system. Device allocation, with the aid of volume preferencing, determines the best placement, number of extents, extent size, and number of volumes for the target data set on the basis of the available space in the subsystem.

- Non-SMS-managed, non-VSAM, dynamically allocated target data set

  The consolidated allocation request is equally distributed across all candidate target volumes.

- Non-SMS-managed, VSAM dynamically allocated data set

  SnapShot makes an initial storage request for the primary allocation amount of the source data set. Additional space requests use the extend processing functions of the system. A list of volumes must be specified, through the VOLUME parameter, to allocate a multivolume data set.

- Preallocated target data set

  For preallocated VSAM data sets the control area (CA) and control interval (CI) size of the target data set must be the same as that of the source data set; otherwise the allocation will fail.

  If the target is too small to contain the source, SnapShot extends the target data set. SnapShot uses the normal extend processing functions, and the results are the same as for extending any data set of the target type. For example, an SMS multivolume data set extends to the current primary volume if space is available, or to the next candidate volume.

2.1.2.2 VOLUMECOUNT

The VOLUMECOUNT and VOLUME parameters help determine placement of target data sets.

The VOLUMECOUNT parameter specifies the maximum number of volumes on which the new target data set can be allocated. Not all these volumes are used if consolidation is possible.

VOLUMECOUNT is applicable for SMS managed data sets only. For further details on the VOLUMECOUNT parameter, please see ″VOLumeCOUNT(n)″ on page 14.

2.1.2.3 Volume Preferencing

Volume preferencing works the same for multivolume data sets as it does for single volume source data sets. Any volumes that are not on the preferred RVA are initially rejected. If the total amount of space is not available on the preferred RVA, the space is acquired wherever it is available.

Volume preferencing can only prefer a single RVA subsystem. For source data sets that are not contained on a single RVA, SnapShot prefers the subsystem that contains the largest amount of allocated space, to maximize its speed and resource benefits. Once the data set is allocated, SnapShot snaps all of the data that can be snapped, including all relative extents or portions of extents that reside on a common RVA. Where extents or portions of extents cross RVA subsystems (or include a non-RVA subsystem), the data mover is invoked for the tracks that cannot be snapped.

2.1.2.4 Guaranteed Space Considerations

Target data sets that request guaranteed space can experience unexpected allocation results under some conditions.

If you specify a value in VOLCOUNT that exceeds the number of volumes in the storage group, the allocation will fail.

If the source data set is close to the 4GB limit and a new volume is required to obtain sufficient space for the target data set, the allocation will fail if the primary allocation value for the new volume pushes the data set over the 4GB limit.

When a preallocated target data set is larger than the source data set, and the source data set is almost full, the following condition may occur: Multiple updates to the target data set may cause a CA split into the next CI. Because the new CI has not been previously formatted, the CI split will fail.

This is also true for nonguaranteed space requests under the same conditions. Use dynamic allocation when you expect to have multiple updates to the target data set.

2.1.2.5 Summary of Multivolume Scenarios

Table 1 shows the results that can be expected when snapping multivolume data sets.

Table 1. Multivolume Allocation Results

Source	Target	Result
SMS managed, same RVA	Dynamically allocated	Multivolume data set on same RVA. Extents are different from source and may be consolidated.
SMS managed, same RVA	Preallocated	Determined by preallocation. Will be extended if necessary to last primary or next candidate volume.
SMS managed, different RVAs	Dynamically allocated	Volume preferencing prefers RVA with largest amount of source allocated space. Will snap where possible and data move remainder.
SMS managed, different RVAs	Preallocated	Determined by preallocation. Will be extended where necessary to last primary or next candidate volume. Will snap where possible and data move remainder.
SMS managed, some RVA, some non-RVA	Dynamically allocated or preallocated	Same as for multiple RVAs. Data going to or from non-RVA will be moved with data mover.
Non-SMS-managed, same RVA, VSAM	Dynamically allocated	First allocation request is for primary allocation of source data set. Additional extents and volumes used as needed. Unused volumes remain as candidates.
Non-SMS-managed, same RVA, non-VSAM	Dynamically allocated	Multivolume data set on same RVA. Extents equally distributed among volumes in volume list.
Non-SMS-managed, same RVA	Preallocated	Determined by preallocation. Will be extended if necessary to last primary, or next candidate, volume.
Non-SMS-managed, different RVAs, VSAM	Dynamically allocated	First allocation request is for primary allocation of source data set. Additional extents and volumes used as needed. Unused volumes remain as candidates.
Non-SMS, different RVAs, non-VSAM	Dynamically allocated	Multivolume allocation equally divided among volumes in volume list. Snap where possible and call data mover otherwise.
Non-SMS, different RVAs	Preallocated	Determined by preallocation. Will snap where possible and call data mover for remainder.
Non-SMS, some RVA, some non-RVA	Dynamically allocated or preallocated	Same as for multiple RVAs. Data going to or from the non-RVA will be moved with data mover.

2.1.3 Extended Format Data Sets

You can now use the SNAP DATASET command to snap extended format data sets, except OS/390 UNIX System Services Hierarchical File System (HFS) files. Extended format data sets must be SMS managed. They are defined with the EXT parameter of the DATA SET NAME TYPE field in the data class ISMF setup panels.

The DATACLAS parameter is required to allocate target extended format data sets. The dataclass of an existing data set will not be updated.

Examples of extended format data sets that can be snapped are: extended sequential, extended partitioned data sets (PDSEs), and striped sequential. SnapShot does not support multivolume extended format striped sequential data sets with a stripe count of 1. Extended format sequential data sets with a stripe count of greater than 1 are supported, as are single volume data sets with a single stripe.

Support for PDSEs has been enhanced. The target will be extended if insufficient space is allocated. You can also allocate the target in blocks.

2.1.4 SMS Support

Calls to SMS are always taken from SnapShot during target data set allocation. The standard DFSMS/MVS allocation and catalog services are used. The automatic class selection (ACS) routines are run before allocation to determine whether the new target is an SMS-managed data set. If it is SMS managed, volume preferencing is called. Volume preferencing is now used only for allocation of SMS-managed data sets. SMS is not a prerequisite for SnapShot but is required for volume preferencing. For further details on volume preferencing, see 2.4.3, " Volume Preferencing" on page 25.

The DATACLAS parameter is introduced to specify the SMS data class to be assigned to the target data set. See 2.6.1.1, "Special Considerations for ACS Routines" on page 32 for further information.

2.1.5 Command Syntax Changes

The VOLUME keyword used with the TARGET parameter of the SNAP DATASET command now supports a volume list. This parameter is mutually exclusive with the VOLUMECOUNT parameter. Figure 5 shows an example of the volume list syntax.

```
VOL(SNAP01 SNAP02 SNAP03)
```

Figure 5. Volume List Syntax

For **SMS-managed** data sets, specifying a list of volumes is an alternative way to determine the maximum number of volumes for the target. The total number of volumes is used in the same way as VOLUMECOUNT, see ″VOLumeCOUNT(n)″ on page 14.

The actual volumes specified are used only as a hint to dynamic allocation, and only if guaranteed space is requested.

If neither VOLUME nor VOLCOUNT is specified, the maximum number of volumes used for the target data set is the same as for the source data set (total primary plus the candidate volumes).

For **non-SMS managed** data sets, the VOLUME parameter specifies a volume, or list of volumes, to be used for the dynamic allocation of the target data set. To dynamically allocate a multivolume data set in a non-SMS environment, a volume list must be specified.

For non-VSAM data sets, the target data set is spread equally across all the volumes in the list. For VSAM data sets, there must be enough space on the first volume in the list to contain the total amount of space occupied on the first primary volume of the source. Extend processing is used if additional space is needed. Volumes that are not needed for the creation of the initial target data set are left as candidate volumes for subsequent data set extensions of the target. A list of volumes must be specified in order to dynamically allocate a multivolume target data set.

2.1.6 Command Syntax Additions for SNAP DATASET

The following parameters are new to the SNAP DATASET command:

- DATACLASs(Class name)

  This parameter specifies the SMS data class to be assigned to the target data set. The parameter is required to provide SMS with a data class for allocation of extended format data sets. SnapShot does not copy the data class from the source, so the data class must be provided by either the data set naming convention and ACS routines or this parameter.

- TOLERATETRUNCATION(Yes|No)

  The target data set will be truncated (YES) only if SnapShot cannot allocate more space. The default value is NO. NO specifies that the data set will not be truncated and the snap will fail if more extents cannot be obtained. This parameter is active only when the target data set requires more space than its primary allocation. If secondary extents are needed to extend the target, but the extents cannot be obtained, SnapShot 1.1 truncates the data set and issues a warning message. The YES option for this parameter in SnapShot 1.2 enables truncation of a data set snapped to a smaller data set.

  Attention

  A warning message is issued to inform you that truncation has occurred. If you specify TOLERATETRUNCATION(NO), the error message is SIB4707E. If you specify TOLERATETRUNCATION(YES), the warning message is SIB4714W.

  TOLERATETRUNCATION applies only to non-VSAM partitioned, sequential, and direct access data sets. VSAM data sets and PDSEs do not allow truncation. Be aware that truncation may cause you to lose your data.

- VOLumeCOUNT(n)

  The VOLUMECOUNT parameter, together with the VOLUME parameter, helps determine the placement of target data sets. The maximum value for VOLCOUNT is 59. A value greater than 59 may cause unpredictable results.

For **SMS managed** data sets, VOLCOUNT specifies the maximum number of volumes that can be used to contain the data. Data is placed on the target volumes according to available space on the volumes chosen by device selection. If the target data set does not occupy all the volumes, the additional volumes remain as candidates for future extensions of the target data set.

It is not always possible to predetermine the number of volumes needed for a target data set. The total number of primary plus candidate volumes for a data set is limited to 59. However, this number cannot exceed the number of volumes in the storage group of the first primary volume.

If VOLCOUNT(59) is specified, SnapShot uses as many volumes as necessary to contain the data set, and any remaining volumes in the storage group will be available for future extensions of the target data set. The parameter is used only for multivolume data sets, not for striped sequential data sets.

This parameter is mutually exclusive with the VOLUME parameter.

| • RELate(dsname)

| This optional parameter allows you to relate an AI to a base cluster. The
| base cluster is an entry-sequenced or key-sequenced cluster to which the
| target AI is to be related. You cannot relate an AI to a reusable cluster or to
| a VSAM volume data set (VVDS).

| The dsname specified in the RELATE parameter must be cataloged in the
| same catalog as the target AI. If you snap an AI and do not specify the
| RELATE parameter, the target AI will be related to the same base cluster as
| the source AI.

| An SMS-managed AI has the same management class and storage class as
| its base cluster. You can use the DATACLASS parameter to assign a
| dataclass to the target AI.

2.1.7 Command Syntax Additions for SNAP VOLUME

The following parameter is new for the SNAP VOLUME command:

- CONDitionVOLume(ALL | LaBeL) (see Figure 6 on page 16)

  This parameter is processed only when you specify COPYVOLID(NO); otherwise it is ignored. If you specify CONDITIONVOLUME(ALL), the command works as it does currently if COPYVOLID(NO) is specified.

  The CONDITIONVOL(LBL) parameter enables you to snap a volume and retain the original volser of the target. The label is updated with the target volser. The target unit control block (UCB) is also updated to enable the device to remain online after the snap operation is complete. The VTOC, VVDS, and VTOCIX remain unchanged from the original source. The target volume can then be physically dumped to tape without being varied online to another system.

```
//*
//STEP1   EXEC PGM=SIBBATCH
//SYSTERM  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
 SNAP VOLUME( -
       SOURCE(VOL(9RMO1C))  -
       TARGET(VOL(9RMO1D))  -
       COPYVOLID(NO) -
       CONDVOL(LBL) -
            )
/*
```

Figure 6. Volume Conditioning

2.1.8 IXFP Space Utilization Reports

There are two new fields in the IXFP space utilization reports, under the Physical Capacity Used(MB) heading. The fields are SHARED and UNIQUE. These two values are added in the TOTAL column for each device.

The UNIQUE field shows the amount of back-end physical space that is unique for a single functional device.

The SHARED field shows the amount of back-end physical capacity that is shared among functional devices as a result of volume or data set snaps. When you snap a data set from one functional device to another, the functional capacity is reported in the SHARED field of *both* devices.

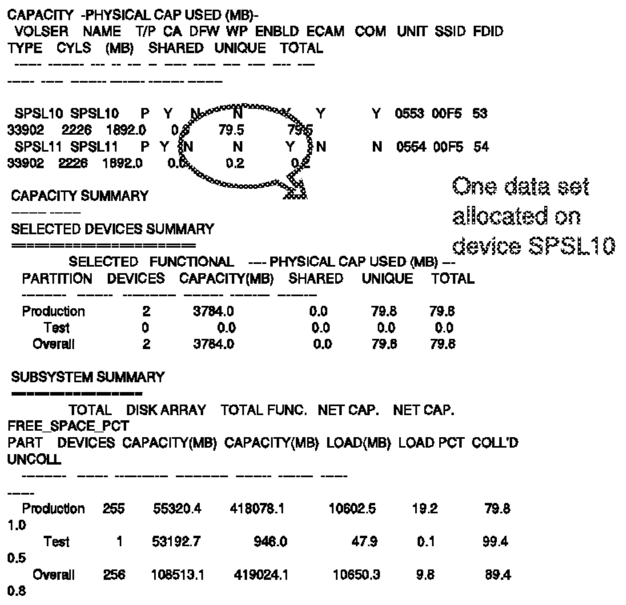
As each copy of the snapped data is updated, the tracks that are still ″common″ to both copies of the data are reported in the SHARED field, and the updated tracks are reflected in the UNIQUE field. Thus you have a mechanism for understanding the cost of keeping snapped data before deletion or refresh.

Device Summary Report

The TOTAL field sums the capacity used by both devices, whether SHARED or UNIQUE. This sum is a representation of the *logical* physical capacity, as the SHARED fields are totaled for both devices. The TOTAL field compared with the physical space used demonstrates the total capacity that is saved by SnapShot.

The NCL for the subsystem accurately represents the real amount of back-end physical capacity used by a functional device. Any NCL reporting tools that you use are unaffected by the new report fields. The NCL remains the same. NCL does not include the storage used by the mapping tables and functional track directories.

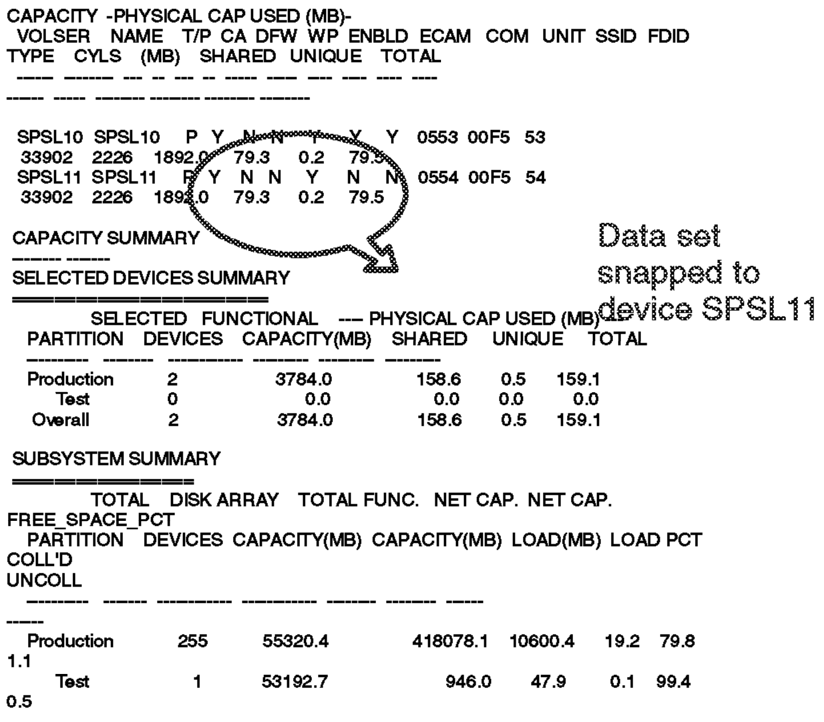
Figure 7 on page 17 and Figure 8 on page 17 show the effect of snapping on the SHARED and UNIQUE fields. Figure 7 on page 17 shows the space utilization reports before the snap has taken place.



```
CAPACITY  -PHYSICAL CAP USED (MB)-
 VOLSER   NAME   T/P CA DFW WP ENBLD ECAM  COM  UNIT SSID FDID
TYPE   CYLS   (MB)   SHARED  UNIQUE   TOTAL
 ---- ----- --- -- -- - --- --- -- -- --- --
---- ---  ----- ----- ----- ----

 SPSL10 SPSL10    P  Y   N    N      Y     Y       Y  0553 00F5  53
33902   2226  1892.0    0.0      79.5        79.5
 SPSL11 SPSL11    P  Y  N      N      Y    N       N  0554 00F5  54
33902   2226  1892.0    0.0       0.2        0.2

 CAPACITY SUMMARY
 ----- -----
 SELECTED DEVICES SUMMARY
 =================
          SELECTED  FUNCTIONAL   --- PHYSICAL CAP USED (MB) --
  PARTITION   DEVICES   CAPACITY(MB)   SHARED    UNIQUE    TOTAL
  ------  -----  --------  -----  -----  -----
  Production       2       3784.0            0.0       79.8      79.8
     Test          0          0.0            0.0        0.0       0.0
   Overall         2       3784.0            0.0       79.8      79.8

 SUBSYSTEM SUMMARY
 ===============
          TOTAL   DISK ARRAY   TOTAL FUNC.   NET CAP.   NET CAP.
FREE_SPACE_PCT
PART   DEVICES  CAPACITY(MB)  CAPACITY(MB)  LOAD(MB)  LOAD PCT  COLL'D
UNCOLL
  ------  ----  -------  ------  -----  -----  ----
----
  Production   255     55320.4     418078.1       10602.5      19.2        79.8
1.0
     Test        1     53192.7        946.0          47.9       0.1        99.4
0.5
   Overall     256    108513.1     419024.1       10650.3       9.8        89.4
0.8
```

Figure 7. IXFP Space Reporting before Snap

Figure 8 shows the change to the SHARED field once the data set on volume SPLS10 has been snapped to SPSL11.



```
CAPACITY  -PHYSICAL CAP USED (MB)-
 VOLSER   NAME   T/P CA DFW WP ENBLD ECAM  COM  UNIT SSID FDID
TYPE   CYLS   (MB)   SHARED  UNIQUE   TOTAL
 ---- ------ --- -- --- -- ---- ---- --- --- ---- ----
----- -----  ------ ------ ------ ------

 SPSL10  SPSL10    P  Y   N  N   Y     Y    Y   0553  00F5  53
 33902   2226   1892.0   79.3   0.2   79.5
 SPSL11  SPSL11    P  Y   N  N   Y     N    N   0554  00F5  54
 33902   2226   1892.0   79.3   0.2   79.5

 CAPACITY SUMMARY
 ------ ------
 SELECTED DEVICES SUMMARY
 ====================
        SELECTED   FUNCTIONAL    --- PHYSICAL CAP USED (MB) ---
  PARTITION   DEVICES   CAPACITY(MB)   SHARED    UNIQUE    TOTAL
  --------  -------  ---------  -------  -------  -------
  Production       2          3784.0         158.6      0.5    159.1
     Test          0             0.0           0.0      0.0      0.0
   Overall         2          3784.0         158.6      0.5    159.1

 SUBSYSTEM SUMMARY
 ==============
        TOTAL   DISK ARRAY   TOTAL FUNC.   NET CAP.  NET CAP.
FREE_SPACE_PCT
  PARTITION   DEVICES  CAPACITY(MB)  CAPACITY(MB)  LOAD(MB)  LOAD PCT
COLL'D
UNCOLL
  --------  ------  ----------  ----------  -------  -------  -----
-----
  Production        255       55320.4          418078.1   10600.4    19.2   79.8
1.1
     Test             1       53192.7             946.0      47.9     0.1   99.4
0.5
```

Figure 8. IXFP Space Reporting after Snap

These new fields are added to the LISTCFG device and subsystem reports, the space utilization report, and the space utilization SMF record.

2.1.9 IXFP SMF Records

IXFP Version 2 introduces a new IXFP SMF record subtype 8 and an updated record subtype 7. Events in the RVA subsystem are recorded by IXFP with an IXFP SMF record. The record is an SMF user type record and must be given a unique number between 200 and 255. The IXFP SMF record contains eight different subtypes depending on the event that took place in the RVA subsystem.

The IXFP SMF record subtype 7 has been updated to reflect the updates to space utilization reporting.

SnapShot events are recorded in the original subtype 6. With SnapShot 1.2, the SnapShot events recorded in subtype 6 are left unchanged. However, the new IXFP SMF subtype 8 record, the extended event record, is designed for VSAM data set support. All information in SMF record subtype 6 is copied to SMF record subtype 8, and additional information is added for VSAM, new keywords, and additional extents.

The subtype 8 record is a segmented SMF record. Each record consists of the record prefix, the basic segment, and one of four segments attached to it. The four segments are:

- Cluster Definition Segment

  Contains the names of the source and target VSAM clusters, the cluster organization, and the path name

- Data Set Name Segment

  Contains the snapped data set names, and the DSNTYPE from the catalog.

- Snapped Extents List Segment

  Contains information from the snap event, such as timing, return codes, data mover return codes, device addresses, and extents

- DDSR Extents List Segment

  Contains information about the extent.

The layout and contents of the record are documented in *IBM RAMAC SnapShot for MVS/ESA Using SnapShot*, SC26-7173.

We recommend that you start by using the subtype 8 record instead of the original subtype 6 record. To start by using the new subtype 8 record, you must update your SMPPRMxx member in SYS1.PARMLIB so that SMF will record subtype 8 of the IXFP SMF user record.

For details of the format of the new SMF record subtype 8, see Appendix A, “IXFP SMF Record Subtype 8” on page 141.

2.2 RVA Extended Addressing

The RVA has been enhanced to deliver extended addressability with up to 1024 addresses and support for 3390 Model 9 device types. For customers who were using all of the 256 addresses previously available, the extended addressability provides many more target addresses for SnapShot. Each of the four LCUs in the RVA now supports 256 addresses, and all of the existing addresses map to LCU 0.

You have the option to define 1024 3390 Model 3 devices, or 256 3390 Model 3 devices and 256 3390 Model 9 devices. Defining more than 512 devices may require you to increase your cache size. You will need to update your HDC or ICOP definitions for the new configuration.

The addressability enhancement is delivered in a new model of the RVA—the RVA X82—which is upgradeable from earlier models. The real capacity of the RVA is unchanged, with a maximum of 840GB, however you now have the ability to address all the capacity in a maximum configured RVA.

You must have software support in the operating system to allow 256 device addresses for each LCU. The following list shows the PTFs available for DFSMS/MVS 1.1 and later.

- 256 devices support
  - OW26616 (DFSMS 1.4)
  - OW26425 (DFSMS 1.1, 1.2, and 1.3)
- DEVSERV
  - OW27859
- RMF
  - OW31700
- Asynchonous Operations Manager (AOM)
  - OW32197

The following software support is required for IXFP and SnapShot:

- IXFP
  - OW36501
  - OW36513
- SnapShot
  - OW36507
- VM
  - VM62024

2.3 Snapping Data Sets

You use the SNAP DATASET command to snap a source data set to a target data set in the same RVA. You can copy only one data set with a single SNAP DATASET command. The target must be on the same device type as the source data set, but it can be on a different model. After the snap, the target data set is an exact copy of the source data set. SnapShot requires a data path to both the source and the target volume to perform the snap.

Target data sets are extended if insufficient space has been allocated. Excess tracks are released with DDSR if the target is larger than required.

Release is done for all data set types if they are larger than the source file. It does not matter whether the TARGET or OUTDD keyword is used. The request is made internally to SnapShot, using an ECAM request for DDSR release. Neither the dynamic nor interval functions are used; in fact they do not even have to be activated or running. Data class, management class, and storage class are not assigned from the source data sets automatically. You must ensure that these constructs are assigned by using the SNAP DATASET parameters accordingly or the ACS routines for specific data set names.

If you request selection of a target data set that is not in the same subsystem as the source, SnapShot can call a data mover to copy the data set. This call produces a conventional copy, not a snap. Refer to 2.9, “Data Mover Support” on page 34 for additional information.

For more information about the SNAP DATASET command and its options, see *IBM RAMAC SnapShot for MVS/ESA Using Snapshot*, SC26-7173.

Snapshot supports the following types of data sets:

- Sequential and extended format sequential data sets
- Partitioned data sets and PDSEs
- Direct access data sets
- Multivolume data sets
- VSAM data sets, ESDSs, KSDSs, LDSs, RRDSs, and VRRDSs
- | Extended format VSAM data sets
- | AIs

If the SnapShot dynamic caller allocates the data set, the extents can be merged into one extent where possible.

Snapshot does not support the following types of data sets:

- Catalogs, VTOCs, or indexed VTOCs
- Individual members of a partitioned data set
- ISAM data sets
- HFS data sets
- Generation data group (GDG) models
- Page data sets
- Concatenated data sets
- Undefined DSORG data sets
- VSAM volume data sets (VVDSs)
- VSAM data sets with IMBED or REPLICATE
- Extended format partitioned sequential multivolume data sets with a stripe count of 1

Figure 9 shows an example of the SNAP DATASET command used for a VSAM KSDS.

```
//* SNAP VSAM KSDS DATASET
//*
//SNAPDS   EXEC PGM=SIBBATCH
//SYSTERM  DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
 SNAP DATASET  (SOURCE(HLQ.AMSBM.KSDS)  -
               TARGET(HLQ.AMSBM.SN1.KSDS))
/*
```

Figure 9. Snapping a VSAM KSDS

2.3.1 DFSMSdss SnapShot

Functions have been added to DFSMSdss to support SnapShot for the COPY function. When both the source and target devices reside in an RVA subsystem and the format of the data does not have to be changed, DFSMSdss can use SnapShot to quickly move the data from the source to the target, regardless of whether the target has been preallocated or dynamically allocated. This process is referred to as DFSMSdss SnapShot.

Here is a summary of the advantages of DFSMSdss Snapshot:

- SnapShot fully benefits from the flexibility of DFSMSdss for such functions as filter lists and wildcard characters.
- DFSMSdss can use SnapShot to quickly copy data.
- SnapShot also benefits from DFSMSdss serialization, which ensures that consistency can be maintained across related data sets.
- DFSMSdss SnapShot extends the range of data set formats that can be copied with SnapShot.

DFSMSdss SnapShot support applies to SMS-managed and non-SMS-managed data sets and volumes, MVS formatted volumes, and VM formatted volumes that are shared with MVS.

To use DFSMSdss SnapShot, the following requirements must be met:

- Source and target devices must be of the same device type.
- Data does not have to be manipulated (reblocking, track packed to unlike).
- Source and target devices must be entirely in the same partition of the same RVA.
- A utility is not called to move the data.

If the source resides behind multiple RVAs, SnapShot is not performed on any part of the data set, so the rule is “all or nothing,” to prevent SnapShot from producing a partial data set. If a data set is spread across multiple RVAs, a traditional DFSMSdss copy is executed, so the data set is copied anyway.

In some cases, DFSMSdss internally invokes a utility to move a data set. When this occurs, the data cannot be copied by SnapShot. Table 2 lists the common data set types for which DFSMSdss invokes a utility for a data set copy operation and which therefore cannot be copied by SnapShot.

Table 2. Utilities Called by DFSMSdss

Data Set Type	**Utility**
Partitioned data set extended (PDSE)	IGWFAMS
Extended format VSAM	IDCAMS(REPRO)
Integrated catalog facility user catalogs(1)	IDCAMS(EXPORT/IMPORT)
CVOL	IEHMOVE
Indexed sequential(2)	IEBISAM

Notes: 1: DFSMSdss cannot be used to move an active master catalog. If the master catalog is not active, it is considered to be just another user catalog.
2: DFSMSdss invokes IEBISAM to move an ISAM data set unless the target data set can be allocated on the same tracks (CCHHs) on which the source data sets reside, and the target volume has an active VTOC index.

SnapShot does not support the use of the IDCAMS DEFINE command parameters, IMBED and REPLICATE. Data sets defined with these parameters are supported by DFSMSdss SnapShot as long as the data set is not multivolume and the primary index has not been extended.

2.3.2 Virtual Concurrent Copy

Concurrent copy is a function of DFSMSdss and the 3990 storage control that enables you to take a consistent copy or dump of your data while applications update the data. Virtual concurrent copy provides a copy operation comparable to concurrent copy by using a combination of RVA, SnapShot, and DFSMSdss. You invoke virtual concurrent copy by specifying the CONCURRENT keyword on a DFSMSdss COPY or DUMP statement.

When the CONCURRENT keyword is specified on a DFSMSdss COPY command and if all the requirements for DFSMSdss SnapShot are met, DFSMSdss tries to perform a DFSMSdss SnapShot, as described in 2.3.1, “DFSMSdss SnapShot” on page 21. In this case the logical and physical completion of the copy occur at the same time.

The benefit of virtual concurrent copy is that it can be performed in the following circumstances, when a DFSMSdss SnapShot cannot:

- The data needs to be manipulated (DUMP command used, reblocked, track packed to unlike) and the source resides on an RVA.
- The source and target reside on different RVAs and DFSMSdss is the data mover.
- The source resides on an RVA, the target is a non-RVA device, and DFSMSdss is the data mover.
- A multivolume source data set spans multiple RVAs, and DFSMSdss is the data mover.
- A multivolume source data set resides on a combination of RVAs and devices connected to a 3990 storage control, and DFSMSdss is the data mover.

Virtual concurrent copy is very beneficial because DB2, IMS/ESA, CICS, and DFSMShsm can use concurrent copy for their backups. If you already use concurrent copy for your backups, no changes are necessary to use virtual concurrent copy. See the appropriate database chapters in this book for more details.

To use virtual concurrent copy, the following requirements must be met:

- Source must be a SnapShot candidate.
- The concurrent copy (CC) keyword must be specified.
- WSDS space is available.
- A utility is not called to move the data.

See Table 2 on page 21 for the data set types for which DFSMSdss invokes a utility for a data set copy operation.

Virtual concurrent copy provides some additional flexibility when compared with SnapShot DFSMSdss SnapShot:

- The target can be non-RVA.

- The data set format can change.
- CONCURRENT can be used for DUMP and COPY.
- Multivolume data set processing is more comprehensive.

If virtual concurrent copy is not possible, DFSMSdss dumps the data, using traditional data movement techniques.

2.4 Supported Data Sets

Table 3 lists the data sets that are supported by SnapShot, DFSMSdss COPY, and DFSMSdss SnapShot.

DFSMSdss DUMP can be used for all data set formats.

Table 3 (Page 1 of 2). DFSMSdss COPY: Supported Data Sets

	SnapShot 1.2 SNAP DATASET	**DFSMSdss Using Traditional I/O**	**DFSMSdss SnapShot**
Catalog data sets	**N**	Y	**N**
ISAM data sets	**N**	Y	**N**
OpenEdition Hierarchical File System (HFS) data sets	**N**	**N**	**N**
Generation data group (GDG) base names	**N**	Y	Y
Page data sets	**N**	Y	Y
Concatenated data sets	**N**	Y	Y
Undefined DSORG data sets	**N**	Y	Y
VTOCs	**N**	Y	Y
Indexed VTOCs	**N**	Y	Y
Extended format VSAM data sets	Y	Y	**N**
VSAM data sets with IMBED or REPLICATE	**N**	Y	Y
Alternate indexes	Y	Y	Y
Extended format sequential data sets	Y	Y	**N**
Extended format sequential multivolume data sets with a stripe count of 1	**N**	Y	**N**
Partitioned data set extended (PDSE)	Y	Y	**N**
Physical sequential (PS) data sets	Y	Y	Y
Partitioned (PO) data sets	Y	Y	Y
Direct access (DA) data sets	Y	Y	Y
Striped sequential data sets	Y	Y	Y
Multivolume data sets	Y	Y	Y (1)
Entry sequenced data sets (ESDSs)	Y	Y	Y
Key sequenced data sets (KSDSs)	Y	Y	Y
Linear data sets	Y	Y	Y
Relative record data sets (RRDSs)	Y	Y	Y

Table 3 (Page 2 of 2). DFSMSdss COPY: Supported Data Sets

	SnapShot 1.2 SNAP DATASET	**DFSMSdss Using Traditional I/O**	**DFSMSdss SnapShot**
VSAM variable record RRDSs (VRRDSs)	Y	Y	Y
Legend: **N=** No **Y=** Yes **Notes:1:** Valid only when entire source and target data set are in the same RVA.			

2.4.1 Data Set Allocation

When allocating devices for the data set snap, you can code a DD statement for either or both the source and target, or you can allow the SNAP DATASET command to perform the allocation.

If you use DD statements:

- You must supply all required parameters.
- You can increase the size of the target data set.
- You can snap an uncataloged source data set.

If SnapShot performs the allocation:

- The source data set attributes are used for the target.
- Space remains the same.
- Multiple extents can be merged.
- The source data set must be cataloged.
- The target may have a different number of volumes than the source.

2.4.2 Multivolume Allocation Hints and Tips

The following are some suggestions to ensure that you get the maximum benefit from snapping your multivolume data sets:

- You can derive the best benefits from SnapShot if you keep your multivolume data sets within a single RVA subsystem. This can be achieved through SMS.
- Consider using dynamic allocation for target data sets that are expected to have multiple updates.
- Volume preferencing can prefer only a single RVA subsystem. For data sets that are not in the same RVA, SnapShot prefers the subsystem with the largest amount of the allocated space, to maximize the amount of data that can be snapped.
- Given the availability of space on the target volumes, you can achieve extent or volume consolidation as a result of SnapShot.
- If you are continuously snapping a source into the same preallocated target, ensure that the last volume on the candidate list has sufficient unused space to hold any additional data that might be added to the source. Alternatively, delete the target volume, and use dynamic allocation each time you snap.
- To maximize the probability of finding available space for a dynamically allocated data set, use VOLCOUNT(59).
- Where space availability and other allocation restraints prevent an entire multivolume data set from being allocated onto the same RVA as the source data set, SnapShot snaps as much data as it can, using the data mover for data that crosses physical subsystems.

2.4.3 Volume Preferencing

Volume preferencing is a function of IXFP for SMS managed data sets. SnapShot uses volume preferencing to check whether the volumes selected by SMS for newly allocated data sets can be used for the intended operation. A storage group can cover more than one RVA, but the allocation must be made on the RVA where the source is located.

When the SNAP DATASET command is executed, IXFP device mapping is used to determine the volumes that are in the same RVA subsystem and are the same device type as the source volume. When SMS selects a volume, if it matches a device in the same RVA, allocation continues. If there is no device match in the same RVA, volume preferencing rejects the volume, and control is returned to SMS to provide another possible target volume. This process continues until a suitable target volume is found or the list is exhausted.

If a volume match is not found, and the data mover is not specified, the allocation and the Snap will fail.

If the data mover is specified, the allocation is redriven, and any volume chosen by SMS is used. DFSMSdss is then called to copy the data set physically to the target volume.

Volume preferencing is activated only when you specify the TARGET parameter in the SNAP DATASET command and the data set is SMS managed.

Volume preferencing is automatically activated during IXFP installation. It can optionally be deactivated. See *IBM RAMAC SnapShot for MVS/ESA Installing Snapshot*, SC26-7174 and *IBM RAMAC SnapShot for MVS/ESA Using Snapshot*, SC26-7173 for more information.

2.4.4 VSAM Considerations

The SNAP DATASET command can be used to snap VSAM data sets. You use the name of the VSAM cluster to specify the data set to be snapped. With the SNAP DATASET command, the VSAM data set is copied. In the case of a KSDS cluster, both the data and the index are copied. Sphere processing is not supported, but you can snap a base cluster and its associated AIs, and use the RELATE parameter to associate the AI with the base cluster. The target data set must be placed on a volume of the same device type as the source and must be located in the same RVA.

SnapShot does not support the use of IMBED or REPLICATE. These do not provide any performance advantage in an RVA subsystem, so they can be discontinued. However, if you have data sets that were defined with REPLICATE and IMBED, you can use DFSMSdss SnapShot to make a SnapShot copy.

The TARGET data set name must be a unique name, different from the source data set name. Volume preferencing can help to direct allocation to the correct RVA subsystem and device type.

When snapping to an existing VSAM target data set, SnapShot extends an existing target data set to match the size of the source data set. The target attributes are altered to match the source attributes if required.

If indexes or data components of a VSAM cluster reside on different RVAs, SnapShot consolidates the target allocation to the RVA with the most space, usually the RVA containing the data component of the data set. SnapShot is used when the source and target are in the same RVA, and the data mover is used for a component of a VSAM data set where the source and target are not on the same RVA. You can override this consolidation with preallocation, in which case when the data source and target are on one RVA, and the index source and target are in another, SnapShot is used for both components.

2.4.5 Striping

You can use the SNAP DATASET command to make a copy of a striped data set. A striped data set is an SMS-managed extended format sequential data set with a data class attribute, a Data Set Name Class of EXT, and the appropriate storage class attribute for the sustained data rate ensure the striping. See the *DFSMSdfp Storage Administration Reference* , SC26-4920 for more information. As the SMS constructs are not automatically copied from the source to the target, they must be provided by either data set naming conventions and ACS routines or the data class or storage class parameters in the SNAP DATASET command. The number of stripes cannot be changed with the SNAP DATASET command.

Multivolume striped data sets with a stripe count of 1 are not supported.

2.5 Snapping Volumes

You can use the SNAP VOLUME command to snap a functional volume to another functional volume within the same RVA. Only one volume is copied with a single SNAP VOLUME command. The target volume must be of the same device type and model as the source volume with the same number of tracks. The source and the target volume must be online to the system that initiates the snap; the target should not be online to another MVS. After the snap, the target is an identical copy of the source volume.

With the SNAP VOLUME command, you can request selection of a target volume that is not in the same subsystem as the source volume. In this case, SnapShot can call a data mover to copy the source volume. A conventional copy is produced, not a SnapShot. Refer to 2.9, “Data Mover Support” on page 34 for additional information.

When you choose the COPYVOLID(YES) option, the volser of the source is also copied to the target, and the target is set offline. In this case the target is a complete identical copy of the source. You cannot access the target from the system that has made the copy because the volser will be a duplicate, but you can vary it online and use it with another MVS that shares access to the RVA.

When you choose the COPYVOLID(NO) option, the volume is completely copied, but the target volser is not changed. After the copy is completed, SnapShot updates the VTOC and VVDS to reflect the new volser. The data sets on the target are not cataloged.

The new CONDVOL(LBL) option enables you to snap a volume and give the target volume a different volser, thereby allowing it to remain online after the snap operation is complete. The target volume can then be physically dumped to tape without being varied online to another system.

The REPLACE option enables you to overwrite existing data on the target volume. If the target volume is not empty, and you have not specified this parameter, the operation terminates. The volume is not empty when it contains other data, except the VTOC, VTOC index, and VVDS.
Use REPLACE(NO) to prevent production data from being overwritten. If you use REPLACE(YES), the SNAP VOLUME runs faster, because the VTOC is not queried for user data set names.

For more information about the SNAP VOLUME command and its options, see *IBM RAMAC SnapShot for MVS/ESA Using Snapshot*, SC26-7173.

Figure 10 on page 28 shows a sample batch job using the SNAP VOLUME command.

```
//*
//STEP1   EXEC PGM=SIBBATCH
//SYSTERM  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
 SNAP VOLUME( -
       SOURCE(VOL(9RM01F))  -
       TARGET(VOL(9RM01B))  -
       COPYVOLID(NO) -
            )
/*
```

Figure 10. Snap Volume

2.5.1 Volume Selection

You can select the device to which the selected source is to be snapped by using a DD statement and the OUTDD parameter in the SNAP VOLUME command. In this case the volume is preallocated.

Alternatively you can use the TARGET parameter of the SNAP VOLUME command with the volser or unit definition. In this case the volume is selected by SnapShot.

2.5.2 Catalog Implications

The SNAP VOLUME command produces uncataloged data sets, but you have several options for using the output.

2.5.2.1 COPYVOLID(NO)

If you use COPYVOLID(NO) with CONDVOL(ALL), the target volume stays online, but you cannot access its data sets by its data set names. You can still access them with a DD statement with a volser definition, so you have to do it manually, this is not recommended in an SMS environment,

If you need a single data set from a snapped volume, you have to rename or delete the original and then recatalog the snapped one (see Figure 11 on page 29 and Figure 12 on page 29).

```
//*
//* 1. ALTER DS NAME OF THE ORIGINAL NONVSAM FILE
//* 2. RECATALOG NONVSAM FILE ON SNAPPED VOLUME 9RM01B
//*
//RECAT    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *

 /*1*/   ALTER -
           HLQ.SNDS31.OUTLIST -
           NEWNAME(HLQ.SNDS31R.OUTLIST)

 /*2*/   DEFINE NONVSAM (-
           NAME(HLQ.SNDS31.OUTLIST) -
           DEVICETYPE(3390) -
           VOLUMES(9RM01B) -
           RECATALOG )
/*
```

Figure 11. Recataloging a Non-VSAM Data Set

```
//*
//* 1. DELETE ORIGINAL VSAM FILE
//* 2. RECATALOG VSAM FILE ON SNAPPED VOLUME
//*
//RECAT    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *

 /*1*/   DELETE HLQ.VSAM1.KSDS

 /*2*/   DEFINE CLUSTER (NAME(HLQ.VSAM1.KSDS) -
                RECATALOG              -
                VOLUME(9RM01B) )       -
         DATA   (NAME(HLQ.VSAM1.KSDS.DATA) ) -
         INDEX  (NAME(HLQ.VSAM1.KSDS.INDEX) )
/*
```

Figure 12. Recataloging a VSAM KSDS

If you need all data sets on the snapped volume to be accessible by their original names, the data set names must be uncataloged first (the catalog entries point to the former volser) and recataloged with the new volser.

If the snapped volume contains a user catalog, after EXPORT DISCONNECT of the current volume, you can connect and use the snapped volume. This procedure can be used only if your original user catalog is destroyed. But be careful, the contents of the snapped catalog are the same as the source was at SnapShot time. The pointers refer to the source volumes, not to the targets, which have different volsers. Sample JCL is shown in Figure 13 on page 30.

```
//*
//*  1. DISCONNECT CURREND UCAT.PRIJOSE ON ORIGINAL VOLUME 9RM01F
//*  2. IMPORT CONNECT UCAT.PRIJOSE ON SNAPPED VOLUME 9RM01B
//*  3. DEFINE ALL ALIASES AGAIN IN THE MCAT
//*
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD *

 /*1*/ EXPORT UCAT.PRIJOSE DISCONNECT

 /*2*/ IMPORT OBJECTS -
         ((UCAT.PRIJOSE VOLUME(9RM01B) DEVICETYPE(3390))) -
           CONNECT

 /*3*/ DEFINE ALIAS (NAME(JOSE) -
         RELATE(UCAT.PRIJOSE) )
       DEFINE ALIAS ....
/*
```

Figure 13. IMPORT CONNECT a User Catalog on a Snapped Volume

If the snapped volume is the source for a subsequent backup to tape with DFSMSdss, you must consider the following:

- If you make a logical dump of the snapped volume, all data sets are uncataloged by default. The uncataloged non-VSAM data sets are dumped to tape; VSAM data sets are not dumped. If an original data set becomes corrupted, you can restore it from the backup tape. You cannot take a logical dump of the uncataloged VSAM data sets, so we recommend taking a physical dump instead. The data sets are not cataloged as a result of a SNAP VOLUME.
- You can take a physical dump of the snapped volume. From this backup tape you can restore a single data set. If it is a VSAM data set, you must define it before, or recatalog it after, the restore.

  See Figure 14 for an example of a DFSMSdss physical dump. Figure 15 on page 31 shows the recataloging of the VSAM KSDS from the dump.

```
//*
//STEP01 EXEC PGM=ADRDSSU,REGION=8M
//SYSPRINT DD SYSOUT=*
//INVOL1   DD VOL=SER=9RM01B,UNIT=3390,DISP=SHR
//OUTDD1   DD DSN=HLQ.DUMP01.D9RM01B,DISP=(,CATLG),UNIT=3390,
//    SPACE=(CYL,(500,100),RLSE)
//SYSIN    DD    *
      DUMP FULL -
      INDDNAME(INVOL1) -
      OUTDDNAME(OUTDD1)
/*
```

Figure 14. DFSMSdss Physical Dump

```
//*
//*  1. ALTER DATASETNAME OF THE ORIGINAL VSAM KSDS
//*  2. RESTORE A SINGLE VSAM DATASET FROM A PHYSICAL DUMP
//*     OF A SNAPPED VOLUME
//*  3. RECATALOG THE RESTORED VSAM FILE
//*
//ALTDSN   EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
    ALTER -
      HLQ.VSAM1.KSDS -
      NEWNAME(HLQ.VSAM1R.KSDS)
    ALTER -
      HLQ.VSAM1.KSDS.* -
      NEWNAME(HLQ.VSAM1R.KSDS.*)
/*
//*
//RESTO  EXEC  PGM=ADRDSSU,REGION=8M
//SYSPRINT DD  SYSOUT=*
//INDD1    DD  DSN=HLQ.DUMP01.D9RM01B,DISP=SHR
//SYSIN    DD  *
 RESTORE DATASET(INCLUDE(HLQ.VSAM1.KSDS)) -
      INDDNAME(INDD1) -
      OUTDYNAM(9RM01E)
/*
//*
//RECAT    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
         DEFINE CLUSTER ( -
                NAME(HLQ.VSAM1.KSDS) -
                RECATALOG             -
                VOLUME(9RM01E) -
                )
/*
```

Figure 15. Restoring and Recataloging a VSAM Data Set

Other aspects of physical volume dump and restore are discussed in Chapter 6, "Using SnapShot for Disaster Recovery" on page 87.

2.5.2.2 COPYVOLID(NO) with CONDVOL(LBL)

If you use COPYVOLID(NO) with CONDVOL(LBL), the volser of the target is preserved together with the VTOC and VVDS of the source. These then do not match, which presents you with the following restrictions:

- A logical dump cannot be produced from this volume because the VSAM volume records (VVRs) do not match.
- A physical dump with the DATASET INCLUDE option dumps only non-VSAM data sets.
- A physical dump with the DUMP FULL option dumps all data sets.
- You cannot restore a single VSAM data set from the dump. If you want to restore a VSAM data, you have to restore the entire volume.

This procedure is useful for disaster recovery. Usually you do not restore a single data set from these backup tapes, but you can restore the complete volume in case of a disaster. Then the volser is clipped to restore its original volser, and the volume and its data sets are accessible. Assuming you have used this procedure with your catalog volumes, the data sets are cataloged as well.

See also Chapter 6, “Using SnapShot for Disaster Recovery” on page 87 for additional information.

2.5.2.3 COPYVOLID(YES)

If you use COPYVOLID(YES), the target volume is varied offline. However, you can use another MVS where you can set the snapped volumes online and make backup copies or work with the data sets. If you have snapped the related user catalog volumes as well and brought them online on the second MVS, you do not have to recatalog the data sets on the second MVS. (This is valid if the second MVS has an identical copy of the first MVS′s master catalog.) It may therefore be an advantage if you increase the number of user catalogs.

If you do not have a second MVS, you can change the volser, using the ICKDSF utility and the REFORMAT command. This is called *clipping*. (The *Change Label Information Program* is no longer available because ICKDSF performs this function.)

Using the ICKDSF REFORMAT command to clip the volser to a unique label enables you to bring the volume back online (see Figure 16). The VVDS and VTOC are not changed with the REFORMAT; they are identical to those on the original volumes.

```
//REFORMAT EXEC PGM=ICKDSF
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 REFORMAT UNITADDRESS(uuuu) VERIFY(PRD001) VOLID(UNQ001)
```

Figure 16. Reformatting a Snapped Volume

After the volser is changed to a unique volser, you can vary the volume online and dump it to tape. The restrictions that apply to COPYVOLID(VOLCOND) also apply when the volser is clipped.

2.6 DFSMS/MVS

When you use the target operand in a SNAP DATASET command, the DFSMS/MVS ACS routines are used and the allocation is done accordingly. Data class, management class, and storage class are not assigned from the source data sets automatically. You must ensure that these constructs are assigned by using the SNAP DATASET parameters accordingly or the ACS routines for specific data sets.

2.6.1.1 Special Considerations for ACS Routines

When the target of a data set snap does not exist, SnapShot first runs the ACS routines to determine whether the data set will be DFSMS/MVS managed or not. The target data set is allocated according to the ACS routines unless specific variables are specified in the JCL. The data set snap is then performed, and ACS routines are called again, with all parameters initialized as part of the standard data set allocation process.

The parameters passed to SMS ACS routines are compatible with those used by IDCAMS DEFINE. The SNAP DATASET command uses the VOLUME and VOLCOUNT parameters to determine input to the SMC ACS routines for dynamically allocated target data sets. See 2.1.5, “Command Syntax Changes” on page 13 and 2.1.6, “Command Syntax Additions for SNAP DATASET” on page 14 for further details.

If one or more volumes are supplied in the VOLUME parameter, these will be passed to the ACS routines in the &ALLVOL ACS variable. If a volume count is specified in the VOLCOUNT parameter an asterisk (*) is passed to the ACS routines for each volume candidate. If neither a VOLUME nor VOLCOUNT parameter is specified the value passed to the ACS routines is the same as the total number of volumes on the source data set.

IDCAMS DEFINE works in the same way as described above when volume candidates are provided in the VOLUME parameter. However, when you do not specify a volume, IDCAMS DEFINE supplies a blank to the ACS routines in the &ALLVOL parameter. You should ensure that your ACS routines check for either a blank or an asterisk to determine whether any volume has been specified.

2.6.2 DFSMShsm

There is no direct way for DFSMShsm to use SnapShot. One possibility is to define an SMS storage pool with the DISABLE NEW option, so that data sets are not allocated on the volumes in that pool. In this sense it would be a pool of empty functional devices. The SNAP VOLUME with COPYVOLID(NO) is used to copy all volumes needed for disaster recovery to the volumes of that pool. Then DFSMShsm is used to dump the volumes to tape. In case of a disaster, the volumes are restored and the data sets are recataloged. Alternatively individual data sets can be restored from the dump or can be snapped back to the original source.

Another procedure for using DFSMShsm is described in 3.3.1.2, “SnapShot and Backup in a Second MVS” on page 50.

2.6.2.1 Using SnapShot with ABARS

Using SnapShot with ABARS backups is discussed in Chapter 6, “Using SnapShot for Disaster Recovery” on page 87.

| DFSMShsm exploits concurrent copy, for CDS backup, ABARS, and DFSMShsm
| managed volume backups. Using virtual concurrent copy DFSMShsm
| automatically invokes SnapShot using the concurrent copy interface. These
| procedures will work unchanged with virtual concurrent copy. See *Implementing
| DFSMSdss SnapShot and Virtual Concurrent Copy*, SG24-5268 for details.

2.7 Concurrent Copy

Concurrent copy, an extended function in the IBM 3990 control unit, enables you to copy or dump data while applications are updating the data. To invoke concurrent copy, specify the *CONCURRENT* keyword on a DFSMSdss COPY or DUMP command.

| Virtual concurrent copy extends the benefits of concurrent copy to users who
| have an RVA with SnapShot. When you specify the CONCURRENT keyword on a
| DFSMSdss COPY or DUMP statement, the software can detect whether you have

a 3990 storage control or an RVA. If you have an RVA the virtual concurrent copy function is invoked.

Concurrent copy support is integrated into the backup and recovery utilities of DB2, IMS, and CICS, so virtual concurrent copy can take advantage of this support immediately and without any changes.

Refer to *Implementing DFSMSdss SnapShot and Virtual Concurrent Copy*, SG24-5268 and *Implementing Concurrent Copy*, GG24-3990 for more information about concurrent copy.

In order to use virtual concurrent copy you must define WSDS according to a naming convention. PTFs UW50234 (for DFSMS 1.3) and UW50235 (for DFSMS 1.4) provide a new naming convention that must be used for the WSDS.

The naming convention is:

```
SYS1.ANTMAIN.Ssysname.SNAPnnnn
```

Where nnnn is a four-digit decimal number in the 0001—9999 value range, and sysname is the system name used by JES. If the system name has 8 characters, the first letter is replaced by ′S′.

2.8 DFSMSdss Fast DEFRAG

APAR OW36322 provides fast DEFRAG support for DFSMSdss. When DFSMSdss performs a DEFRAG of a volume, it interrogates the VTOC and builds an internal map of the used and unused space on the volume. It then does some analysis of the best extents to move to consolidate used and free space.

If the volume which you are defragging is SnapShot capable, DFSMSdss uses SnapShot to move the extents instead of physical I/O. The time for physical data movement is reduced, thus reducing the elapsed time for the DEFRAG operation. If the snap fails for one or more of the extents, that extent will be moved using physical I/O.

2.9 Data Mover Support

When a SNAP DATASET or SNAP VOLUME command is processed, every attempt is made to resolve the source and target to the same RVA subsystem so that a SnapShot can be performed. If the SNAP command cannot resolve the source and the target to the same subsystem, a data mover can be called to complete the snap request. In this case a conventional copy is made, not a snap, and the conventional copy may take much longer than a snap. You can activate the data mover option by specifying the name of the data mover to be used at the DATAMOVERNAME parameter in the SNAP DATASET or the SNAP VOLUME command. The default is NONE.

Where a snap would result in the snap of only part of a multivolume data set, and a data mover is not specified, the SnapShot operation ends in error to prevent the creation of partially snapped data sets.

2.10 Net Capacity Load Considerations

Initially, when a snap copy is created, it consumes no extra back-end storage. Both copies are completely identical, so every track is common between the two copies. Until you have updated every track, you will always use less NCL capacity than a traditional copy uses.

Predicting the amount of NCL a snap copy will use is difficult. Regardless of whether the copy is a snapped volume or a snapped data set, the considerations are the same.

Once a snap copy has been created, there is no distinction between the primary and secondary copies. Both can be regarded as completely separate entities that happen to share some common storage space.

The amount of extra NCL used after the snap is based on several factors:

- Number of track updates
- Frequency of track updates
- Amount of time that the snap copy is retained or refreshed
- Compression ratio of the data on the updated tracks
- Frequency of refreshes if the snap is periodically refreshed

2.10.1 Number of Track Updates

As either copy of the data is updated, the modified data is no longer common between the two copies, so the modified data is rewritten, and NCL usage increases. Like all writes in an RVA subsystem, these updates occur at the functional track level. If one record is changed, the entire track is rewritten and consumes some NCL. If three records are changed, and all three records are contained within one track, that track is rewritten, and the increase in NCL is the same as if only one record in the track had been updated.

2.10.2 Frequency of Track Updates

The frequency of the track updates is directly related to the rate at which the amount of NCL used increases. If track updates occur at a high rate, NCL usage grows quickly. If track updates occur at a slow rate, the NCL climbs slowly.

2.10.3 Amount of Time Copy Is Retained

Many snap copies of data sets are temporary copies that are copied to tape and then deleted. The amount of time that the copy is required to be retained on DASD is a factor in NCL increase. Over time, as the two copies diverge, the NCL grows. As soon as a temporary snap of a data set or volume has been copied to tape, it should be deleted. Any extra back-end storage the copy is using will then be deleted, thus reducing the NCL.

2.10.4 Compression Ratio of Updated Track Data

All back-end data stored in an RVA is compressed. Therefore, if the updated track contains data that compresses well, it will occupy less NCL. If the updated track contains data that cannot be compressed well, such as data that has already undergone software compression, it will occupy more NCL.

2.10.5 Frequency of Refreshes

Initially when a snap copy is created, it uses no extra back-end storage. Some snap copies are periodically refreshed from the original data set. Each time a snap is refreshed to the same data set name, any extra storage in use is released. The more frequently the copy is refreshed, the less the impact on NCL.

2.11 Performance Implications

In this section we discuss some performance aspects of using SnapShot with the RVA subsystem. The two areas where you might notice better performance than with traditional copy techniques are:

- Elapsed time taken for the snap itself
- Additional subsystem activity

2.11.1 SnapShot Elapsed Time

The process of taking a snap copy of a volume or a data set involves making a copy of the RVA pointers and is very fast indeed, as there is no data movement. An average time to snap a volume is 3-5 secs, for a data set snap, the time for each data set is similar.

SnapShot has to do some additional work before the snap takes place. It must check that all data set extents are located on the same RVA, the SMS routines are processed to determine whether the data set is SMS managed or not, and, for VSAM data sets, the cluster components are identified and located. All this checking takes place for all data sets regardless of size. Therefore the proportion of time spent on this checking is greater for smaller data sets than for larger ones, and data sets with a large number of extents may also take longer to process. However, the time saved over traditional copies may be significant.

Table 4 shows the savings in elapsed time from SnapShot when compared to a DFSMSdss disk to disk copy.

Table 4. SnapShot and DFSMSdss Performance Comparison

Copy Technique	CPU Time (sec)	I/Os	Elapsed Time
DFSMSdss	42	796000	23 min
SnapShot	0.93	5	14 sec

Table 5 compares the performance of SnapShot and IDCAMS REPRO when copying 24GB of VSAM RRDSs. Four concurrent SnapShot jobs were run, each duplicating eight data sets.

Table 5. SnapShot and REPRO Performance Comparison

Copy Technique	CPU Time (min)	I/O	Elapsed time (HH:MM:SS)
REPRO	17.48	7,024,000	02:15:40
SnapShot	0.08	2,680	00:01:39

It is important to compare the results of the SnapShot jobs with the DFSMSdss elapsed times; SnapShot is a lot faster, and so for almost all data sets will be faster than DFSMSdss. Implementing SnapShot will save you more time in your schedule for the largest data sets, so these should be targeted first to realize the greatest elapsed time savings with the minimum of effort.

You need to understand the critical path of your batch to identify where you should put your effort in order to get the most benefit from SnapShot in elapsed time savings. Use OPC/ESA to identify the critical path through a suite of jobs.

2.11.2 Additional Subsystem Activity

If you intend to make backup copies of the snapped volumes or data sets in parallel with your other production processing, remember that you must consider the additional load the backup run places on the RVA subsystem. Running dump jobs concurrently with the application affects performance of the entire RVA subsystem. Here are some guidelines regarding the number of dump jobs to run concurrently to minimize the impact on the application:

- On an RVA2 subsystem, run two dumps jobs concurrently with the application.
- On an RVA Turbo 4-path subsystem, three or four dump jobs can be run concurrently with the application depending on the load on the RVA.
- On an RVA Turbo 8-path subsystem, up to six dump jobs can be run concurrently with the application with minimum impact on response times.

Depending on the type of RVA and tape hardware, the importance of application response times, and the urgency of getting the dump tapes offsite, adjust the number of concurrent DFSMSdss jobs to suit your requirements.

You also have the option with SnapShot to reschedule tape activity to another time; the snapped copies of the data will remain until they are dumped to tape.

You might also consider eliminating some of the tape activity altogether. A lot of batch processing involves dumping data to tape because of historical reasons: either the JCL was already available, or there was not at that time enough disk space to consider dumping temporary backups to disk. With the RVA you may want to temporarily back up data sets to disk, using SnapShot, and then delete them at the end of the batch processing.

2.12 General Recommendations

In this section we provide some general recommendations and hints you should consider when using SnapShot.

2.12.1 Capacity

Initial snap copies of data take no NCL. Over time, as updates are processed against the data, the NCL increases, as described in 2.10, “Net Capacity Load Considerations” on page 35.

Delete temporary snap copies of data as soon as they are no longer needed or have been copied to another medium.

Monitor the NCL percentage closely. The IBM recommendation is to run the RVA at 75% NCL under MVS or VM.

On systems without IXFP, DDSR is not available. When a data set is deleted, the NCL allocated to that data set is still occupied by the deleted data. When a new data set is created in the same place on that disk, the NCL allocated to the old data is released. This is functionally equivalent to traditional storage subsystems. For this reason, systems without IXFP should be sized to operate at an NCL of 60%—65%. After a volume snap, when the data has been dumped to tape, and you no longer need the snapped copy, you can initialize the functional device, or use IXFP to delete it.

If you delete the device, its space is immediately available and becomes global freespace. However, deleting a functional device may be operationally more complex, especially in a multisystem environment.

If you initialize the functional device, its used space becomes RVA freespace only at the next interval DDSR run. This may not be a problem if you use the same logical space on the volume immediately. In this sense, the initialized volume contains freespace, which is still related to the volume, but it is not seen by the RVA, and it is not free for global usage.

The recommendation is to perform a minimal INIT followed immediately by an interval DDSR run against the device.

2.12.2 Serialization

SnapShot serialization is similar to DFSMSdss. For a DATASET SNAP, SnapShot holds an enqueue on the data set itself; for a VOLUME SNAP, the enqueue is against the VTOC. Therefore, with a VOLUME SNAP, data sets may not be allocated or deleted from the volume, but existing data sets may be updated.

A SnapShot operation is completed in fewer I/O operations than a traditional DFSMSdss copy. So, although in theory a data set could be updated, in practice it is unlikely until after the Snap has completed. This may significantly improve data consistency on a single volume copy because the entire volume is at the same point of consistency. However, each extent of the data set is snapped as a separate operation, so, for data integrity, you must ensure that the data cannot be updated during the snap. Typically you would ensure that the data cannot be updated by quiescing the application.

DFSMSdss can process multiple volumes in a single command, and SnapShot can process only a single volume per command. From a serialization point of view, however, this does not make too much difference as each volume in the DFSMSdss volume list is processed serially, so there is no serialization across all volumes in the list.

The big benefit of SnapShot in a batch environment is that most customers require a point of consistency across all volumes they are dumping. To ensure consistency, all activity is usually quiesced. When SnapShot is used instead of DFSMSdss, the period during which applications have to be quiesced is much reduced because SnapShot is not performing data movement. The data movement to tape may take place asynchronously if necessary once the applications are restarted, thus extending application availabilty.

For details on SnapShot serialization in a shared DASD environment, see 8.3, “Integrity” on page 117.

2.12.3 Security

Access to both the source and target functional volumes occurs in the requester′s address space and uses standard system services for certain operations, such as open or catalog services. These system services use the MVS System Authorization Facility (SAF) provided by OS/390 to call the system security product, such as the OS/390 Security Server.

There are two levels of security with SnapShot: the volume and data set level, and the SnapShot command level. The different classes are used with the OS/390 Security Server. Other security products have similar facilities. The SNAP VOLUME command uses the DASDVOL class with the volser as the resource name. The SNAP DATASET command uses the DSN class. The command level uses the FACILITY class.

The SNAP VOLUME command calls SAF with a DASDVOL read access request to verify access to the source volume. For the target volume, SnapShot checks for a DASDVOL alter access permission.

The SNAP DATASET command opens the source for input and the target for output to verify that the requester has proper access authority. For the source data set, you need DSN READ authority. For an existing target data set, you need DSN UPDATE authority. For a new target data set, you need DSN ALTER authority.

The SNAP command security is optionally implemented through the FACILITY class. You can define resource rules at the command level, and they can be expanded down to the volume or data set level.

With the REPLACE(YES) parameter of the SNAP command, you can easily overwrite existing data sets or volumes. You must apply procedures to prevent the overwriting of production data by mistake. If you set aside a contiguous range of device numbers and always use the same device numbers or functional devices for volume snapping, you can minimize the risk of overwriting current data.

2.12.4 Hints and Tips

You can reduce the IOS queuing on read-only data sets as well as on load libraries, by making several snap copies of them and assigning different copies to different applications.

If you have to format many volumes or data sets, you can save time by formatting one volume or data set as a template and snapping it as many times as necessary. Some databases require many formatted volumes.

You can SnapShot for VM/ESA to instantly format minidisks. See Chapter 9, “Using SnapShot in a VM/ESA Environment” on page 119 for more information.

It is essential to maintain a good naming convention when snapping data sets and volumes. A good naming convention enables you to maintain version control of snapped copies and to ensure that snapped copies that are no longer needed are deleted to free NCL.

You can also use SnapShot to clear SMF data sets quickly without losing data. Inactive SMF data sets can be snapped and then cleared when they are full. HOSTCOPYMODE(SHARED) must be specified on the SNAP DATASET command.

This application and SnapShot can be useful when all SMF data sets are full and the SMF data is being buffered, as it is significantly quicker than using the IFASMFDP utility to dump and clear the data sets.

2.12.5 Evaluating SnapShot

To evaluate your need for and the benefits of SnapShot, the factors for consideration are shown in Table 6.

Table 6. Evaluating SnapShot

Business requirement	**SnapShot benefits**
Your business objectives for implementing SnapShot.	SnapShot can save time, disk space, and money.
Level of data currency. • A few seconds old? • A few minutes old? • A few hours old?	SnapShot reduces backup time so the frequency of backups can be increased.
Data consistency • Multiple table consistency • Transaction consistency • Subsystemwide consistency	SNAP DATASET allows you to copy application related data sets. SNAP VOLUME allows you to replicate a complete system.
The timescale for rollout of SnapShot	SnapShot is available now. SNAP DATASET now supports VSAM and multivolume data sets. Many customers are already realizing savings in time and space from SnapShot.
Current backup window	SnapShot replaces traditional dump and image copy techniques and cuts hours from your backup window.
Application testing	SnapShot allows you to re-engineer your test processes to achieve greater efficiency and reduced costs.
Hardware and Software availability	SnapShot requires an RVA with the SnapShot enablement feature, and the IBM RAMAC SnapShot program product.
Human resources	SnapShot is easy to implement. IBM global services can help you with your implementation.

Chapter 3. Using SnapShot in Batch

To satisfy the requirements of today′s IT business and to meet online user needs, reduction in batch running time is essential. SnapShot improves batch throughput and reduces the batch running time.

In this section we present some examples of using SnapShot in a batch environment.

Snapshot is invoked in batch through the SIBBATCH program. The commands and their operands are not identical to the DFSMSdss commands. So you cannot easily replace the DFSMSdss commands without changes to procedures and JCL.

Some common components of batch processing are:

- Data Backup
  Data is backed up during batch processing for many different reasons. Data that is copied to protect against a component failure in the computer center, either hardware, software, or application failure, can be considered as an ″operational backup″. The backup might be a complete copy of all data, or an incremental copy, that is a backup of the changes made since the last backup.

  Backups may be taken of data between processing steps to protect against data loss if a subsequent job or step fails. These backups are called interim backups.

  In many cases the backups made here are also used for disaster recovery. Many of the considerations for disaster recovery are also valid for operational backup. See the Chapter 6, "Using SnapShot for Disaster Recovery" on page 87 for additional information.

- Data set reorganization
  Data set reorganization is very often a time-consuming part of nightly batch. We show how SnapShot can dramatically reduce the running time.

- Report generation
  A large part of batch processing is often dedicated to generating output reports from production data. Often read access only is required by the applications. SnapShot can be used to decrease the contention of multiple read jobs accessing the same data set by replicating critical files and allowing parallel access to multiple copies of the data.

- Application processing
  There may be other data copy steps during batch processing which can be speeded up by SnapShot.

There are different ways of using SnapShot:

- Data set snap
  Data set snaps can be used for almost all data set types. A new data set name is required to perform the DATASET SNAP operation. After the logical copy of the data set is produced in the RVA, MVS performs the necessary catalog and VTOC updates so that the new data set is accessible.

- Volume snap with COPYVOLID(NO)

After the logical volume copy is made in the RVA, MVS updates the VVDS and VTOC to reflect the new VOLID. In this sense the snapped copy of the volume is not identical to the original. The data sets are not automatically cataloged.

- Volume Snap with COPYVOLID(YES)
  The logical copy is identical to the original. After the snap, the copied volume is set offline, as it would be after a DFSMSdss full volume copy, with the COPYVOLID(YES) option. The snapped volume cannot be used by the MVS that made the snap, as long as the original volume is online. Another MVS may of course set this volume online and use it.
- Volume snap with CONDVOL(LBL) and COPYVOLID(NO)
  The original volser of the target volume is preserved, but the VTOC and VVDS are identical to the source volume. Therefore the volume is accessible from the same MVS system and can be dumped. However, individual VSAM data sets cannot be restored from such a dump.

3.1 Identifying SnapShot Candidates

You can use any of the following methods to identify candidates for SnapShot.

3.1.1.1 Team of Experts

Use a team of experts from application programming, data management, job scheduling, and system programming who know the time-critical applications, the possible timing problems with the data set backup procedures, the most time-consuming jobs, and whether those jobs are in the critical path of the batch processing. After you introduce the RVA and SnapShot, you can discuss potential areas for implementation.

3.1.1.2 SNAPAID

SNAPAID is a program to estimate SnapShot benefits and identify candidates for SnapShot use. It is available on the IBM MKTTOOLS disk. If you do not have access to the disk, ask your IBM representative for a copy of SNAPAID.

SNAPAID identifies the potential benefits in terms of CPU and elapsed time savings that would accrue from replacing data copying programs with SnapShot. The program has a built-in list of standard copying programs. Other programs can be specified through input parameters. It uses SMF data as input and therefore accurately calculates the data.

Parameters are used to limit the time analyzed or to limit the analysis to specific jobs.

The following example is part of a SNAPAID output report.

```
                    SNAPAID - PROGRAM TO ESTIMATE SNAPSHOT BENEFITS PAGE 0001

INPUT PARAMETERS

 NO PARAMETERS SPECIFIED
 DEFAULTS USED - ALL JOBS IN SAMPLE, STANDARD PROGRAM LIST,
 STEP ELAPSED > 5 MINS
 EARLIEST RECORD DATE & TIME = 97134, 12:23:42  LATEST = 97140, 18:13:10
```

```
JOBNAME  STEPNAME PGM NAME ELPSD TIME CPU TIME   EXCPS   MOUNTS     PAGE 0002

DCUGE151 DB2BATCH DSNUTILB       37        5     1246        0
     TOTALS FOR LISTED STEPS     37        5     1246        0

     TOTALS FOR JOB        00H 00M 37S     5     1246        0


ESDAG168 COPYIT   IEBCOPY        21        1     1431        0
     TOTALS FOR LISTED STEPS     21        1     1431        0

     TOTALS FOR JOB        00H 11M 34S    31    32769        0    1)


FIDAG483 MODIFYIC DSNUTILB        9        1      132        0
         DUMPSTP  ADRDSSU       621       17    27140        1
     TOTALS FOR LISTED STEPS    630       18    27272        1    2)

     TOTALS FOR JOB        00H 10M 30S   118    27272        1
```

Times of less than 1 sec are shown as zero.

Notes:

1. The report lists only those steps that use a recognized copying program. Job ESDAG168 has multiple steps, only one of which used a copying program (step COPYIT). The other steps, which are not detailed in the report, accounted for 11min 13sec of elapsed time (and 30sec CPU + 31338 EXCPs) which is reported as the total for the ESDAG168 job. The IEBCOPY step needed 21sec elapsed time, so this job is not likely to be a candidate for SnapShot.
2. The last job, FIDAG483, only had two steps, both of which used a copying program. This job might be a candidate for SnapShot, because 10 min and 30 sec elapsed time were used for copying data.

3.1.1.3 DFSMS/MVS Optimizer

IBM′s DFSMS/MVS Optimizer can help identify the SnapShot candidate data sets that will give you the biggest benefits with the least amount of effort.

The SnapShot Optimizer report lists extremely large data sets that are backed up or read for long periods of time. The report also identifies the type of data set being accessed and the job and program name processing that particular data set.

The SnapShot Optimizer report can be sorted by ″open time,″ so the data sets read for the longest period of time trickle to the top. These are the data sets to initially focus on when determining which data sets are good SnapShot candidates. The Optimizer can also be used to identify situations where multiple jobs are accessing the same data for report generation purposes. SnapShot can be used to allow parallelization of this access and thus reduce the elapsed time of the process.

3.1.1.4 IBM Global Services

The following services are available from IBM:

- Planning for RAMAC Virtual Array (RVA) utilization
- Establishing an operational RVA environment
- Implementing a pilot SnapShot application

3.2 Using SnapShot for Data Set Backups

In this section we discuss how to use SnapShot to copy data sets and reduce your backup window.

SnapShot reduces the outage required for applications by reducing the amount of elapsed time backups take to complete.

If you have many data sets to back up, it is probably impractical to invoke a SNAP DATASET command for each data set individually from a SIBBATCH batch job.

You should analyze your backup batch stream. Many installations will find that a large proportion of the elapsed time of the backup window is spent backing up a minority of the data sets. Using SnapShot to back up the minority of your application data sets may significantly reduce your backup window.

3.2.1 Scenario

The installation has used SNAPAID and the DFSMS/MVS Optimizer to determine which of its DFSMSdss backup jobs take the most time to complete.

Analysis of the DFSMSdss output of this job, which backs up many data sets using DFSMSdss wildcards, shows that it takes 20 min to back up many small data sets, whereas it takes 1 hr to back up three relatively large data sets (see Figure 17).

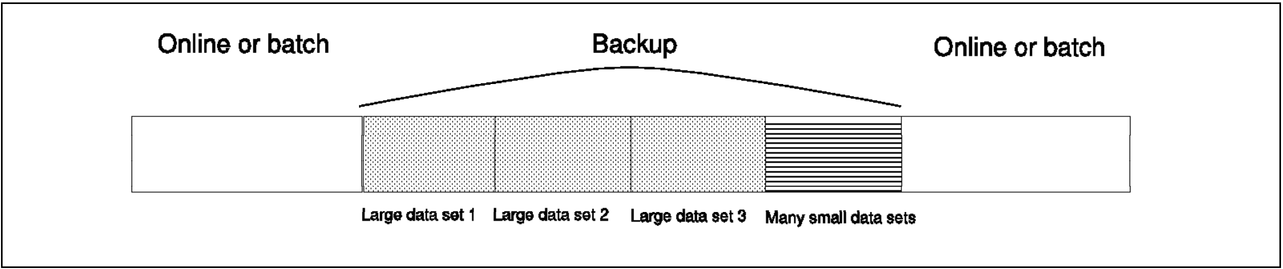


Figure 17. Backup Flow without SnapShot

If SnapShot is used to back up the few large data sets, the backup window is dramatically reduced, and the installation does not have to convert its entire backup procedure to issue individual SNAP DATASET commands (Figure 18 on page 45).

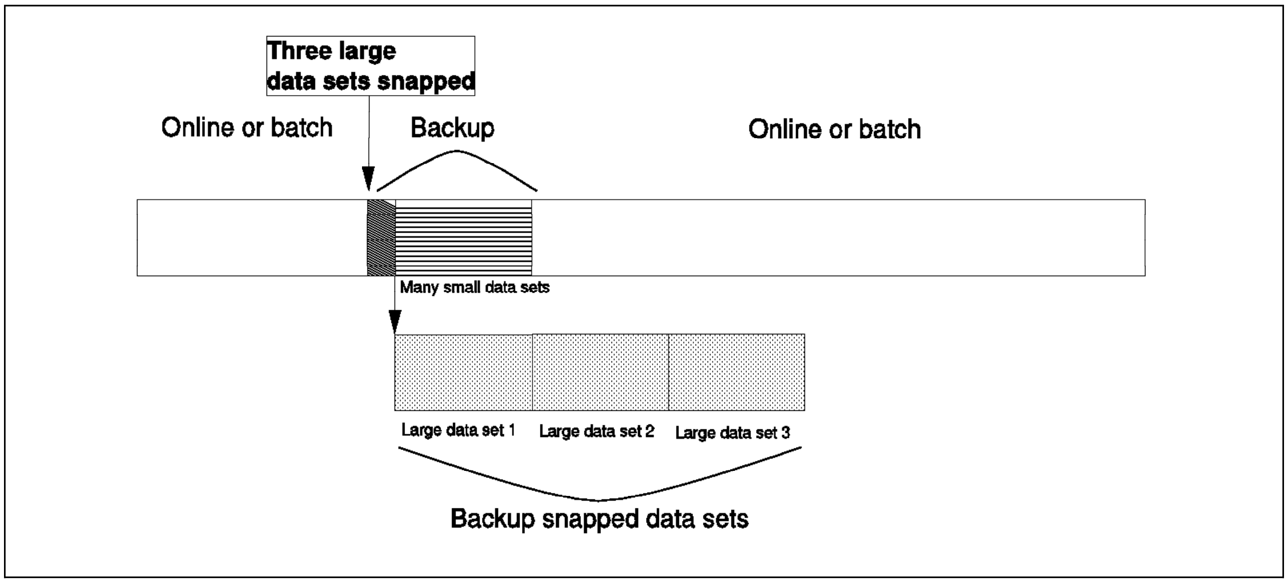


Figure 18. Backup Flow with SnapShot

If you have the support for DFSMSdss SnapShot, any DFSMS COPY command that copies data within the RVA, will automatically invoke SnapShot. This gives you the time and space saving benefits of SnapShot without requiring any programming changes.

3.2.1.1 Backup Procedure

The backup procedure is modified to perform the following steps:

1. Application access to the data sets is quiesced.
2. The largest data sets are snap copied by using SIBBATCH. In Figure 19, three data sets are snapped. SnapShot creates the target data sets if they do not already exist. If they do already exist, SnapShot overwrites them.

```
//SNAPBIG  EXEC PGM=SIBBATCH
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSIN    DD *
 SNAP DATASET(SOURCE(DATASET(APP.DATABASE.LARGE1)) -
             TARGET(DATASET(APP.DATABASE.LARGE1.SNAPPED)) -
             TOLERATENQFAIL(NO) -
             REPLACE(YES))
 SNAP DATASET(SOURCE(DATASET(APP.DATABASE.LARGE2)) -
             TARGET(DATASET(APP.DATABASE.LARGE2.SNAPPED)) -
             TOLERATENQFAIL(NO) -
             REPLACE(YES))
 SNAP DATASET(SOURCE(DATASET(APP.DATABASE.LARGE3)) -
             TARGET(DATASET(APP.DATABASE.LARGE3.SNAPPED)) -
             TOLERATENQFAIL(NO) -
             REPLACE(YES))
```

Figure 19. SIBBATCH Used to Snap Several Large Data Sets

3. DFSMSdss is used to back up the remaining small data sets (Figure 20 on page 46).

```
//DSSSMALL EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//TAPEBKUP DD UNIT=3480X,DISP=(NEW,CATLG),
//         DSN=APP.DFDSS.BACKUP.SMALL(+1)
//SYSIN    DD *
  COPY DATASET(INCLUDE(APP.DATABASE.**) -
               EXCLUDE(APP.DATABASE.LARGE*) -
       TOLERATENQFAILURE(NO) -
       OUTDDNAME(TAPEBKUP)
```

Figure 20. DFSMSdss Used to Back Up Small Data Sets to Tape

4. Application access to the data sets is enabled.
5. DFSMSdss is used to back up the large snapped data sets (Figure 21).

```
//DSSLARGE EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//TAPEBKUP DD UNIT=3480X,DISP=(NEW,CATLG),
//         DSN=APP.DFDSS.BACKUP.LARGE(+1)
//SYSIN    DD *
  COPY DATASET(INCLUDE(APP.DATABASE.**.SNAPPED) -
       TOLERATENQFAILURE(NO) -
       OUTDDNAME(TAPEBKUP)
```

Figure 21. DFSMSdss Used to Back Up Large Snapped Copies to Tape

6. Backup tapes are transported to a secure location.

   As either copy of the data set is changed after the snap, NCL usage will grow. If the available NCL is limited, the snap copies of the data sets should be deleted as soon as they have been successfully copied to tape.

   You can use a variety of methods to delete the data sets:

   - You can use a traditional data set deletion method, such as IDCAMS DELETE (see Figure 22), or a JCL DD statement with DISP=(OLD,DELETE), or you can specify the DELETE parameter with the DFSMSdss dump job so that DFSMSdss will delete the snap copies immediately after they are copied to tape.

```
//DELETE   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DEL 'APP.DATABASE.LARGE1.SNAPPED'
 DEL 'APP.DATABASE.LARGE2.SNAPPED'
 DEL 'APP.DATABASE.LARGE3.SNAPPED'
```

Figure 22. Using IDCAMS to Delete the Snapped Copies

     Dynamic DDSR ensures that any back-end storage allocated to the data set is released immediately.

   - You can delete the functional device.

     If you have a particular functional device dedicated to storing the target data sets, you can delete the device, using SIBBATCH, when the snapped copies are no longer needed (Figure 23 on page 47). The device must be varied offline to all systems before it can be deleted.

```
//DELVOL   EXEC PGM=SIBBATCH
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSIN    DD *
 DELETE DEVICE( -
       SUBSYS(RVASYS) -
       UNIT(1010) -
       FORCE -
       VERIFY(FDID(10)))
```

Figure 23. Deleting a Functional Device

The SIBBATCH commands listed in Figure 23 deletes the functional device with unit address 1010. The FORCE parameter allows the functional device to be deleted, even though it still contains user data. The VERIFY parameter ensures that the functional device ID and the unit address are correct.

The back-end storage allocated to this volume is released immediately when the volume is deleted.

If you use this method, you must redefine the functional device before it is used again.

- You can use ICKDSF to initialize the functional device.

  ICKDSF does not call DADSM, so dynamic DDSR will not notice that the data sets on the volume have been deleted. The space will not be released until the next time the interval DDSR function is triggered. The storage administrator or system programmer sets the frequency of Interval DDSR.

3.2.1.2 Recovery Procedure

Using the method described in 3.2.1.1, “Backup Procedure” on page 45 can greatly reduce your backup window, but there are extra considerations to take into account when restoring the application databases.

The larger data sets have been backed up with different names and are separate from the backup of the rest of the application. They must be restored separately and renamed to their original names:

1. DFSMSdss is used to restore the snapped copies of the large data sets from tape (Figure 24).

```
//DSSLARGE EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//TAPEBKUP DD UNIT=3480X,DISP=OLD,
//         DSN=APP.DFDSS.BACKUP.LARGE(0)
//SYSIN    DD *
  RESTORE DATASET(INCLUDE(**)) -
      REPLACE CATALOG
```

Figure 24. DFSMSdss Used to Restore Snapped Copies of Large Data sets

2. Application access to the data sets is stopped.

3. The restored copies of the data sets are snapped back to their original names (Figure 25 on page 48).

```
//SNAPBIG  EXEC PGM=SIBBATCH
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSIN    DD *
 SNAP DATASET(SOURCE(APP.DATABASE.LARGE1.SNAPPED) -
             TARGET(APP.DATABASE.LARGE1)
             TOLERATENQFAIL(NO) -
             REPLACE(YES))
 SNAP DATASET(SOURCE(APP.DATABASE.LARGE2.SNAPPED) -
             TARGET(APP.DATABASE.LARGE2)
             TOLERATENQFAIL(NO) -
             REPLACE(YES))
 SNAP DATASET(SOURCE(APP.DATABASE.LARGE3.SNAPPED) -
             TARGET(APP.DATABASE.LARGE3)
             TOLERATENQFAIL(NO) -
             REPLACE(YES))
```

Figure 25. SIBBATCH Used to Snap Back Large Data Sets

4. DFSMSdss is used to restore the remaining small data sets (Figure 26).

```
//DSSSMALL EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//TAPEBKUP DD UNIT=3480X,DISP=OLD,
//         DSN=APP.DFDSS.BACKUP.SMALL(0)
//SYSIN    DD *
  RESTORE DATASET(INCLUDE(**)) -
          REPLACE CATALOG
```

Figure 26. DFSMSdss Used to Restore Small Data Sets

5. Application access to the databases is enabled.

If NCL is limited, the snapped copies of the large data sets should be deleted as soon as they are no longer needed.

3.2.2 SNAPDSSU REXX EXEC for Data Set Backup

SNAPDSSU provides some functions that give SnapShot some of the same flexibility as DFSMSdss commands, such as wildcards and filter lists.

The SNAPDSSU procedure consists of three steps:

1. Delete SNAPDSSU data sets
2. Execute the DFSMSdss DUMP command in simulation mode, using the PARM=′TYPRUN=NORUN′ parameter, to obtain a list of data set names to be processed
3. Execute SNAPDSSU.

   During execution SNAPDSSU performs the following tasks:

   a. Checks the catalog status of source and target data sets
   b. Generates a target data set name, based on parameters passed to SNAPDSSU

c. Creates JCL to snap all source data sets listed in Step 3. This is referedr to as the SNAPTO JCL or job.
d. Creates matching JCL to snap back data sets. This is referred to as the SNAPBACK JCL or job.

With DFSMSdss′s data set filtering capabilities and SNAPDSSU′s ability to generate JCL, this procedure makes it possible to generate large numbers of data set snap commands in no time at all.

For details of this EXEC, see Appendix B, "Sample SNAPDSSU Procedure" on page 143.

3.3 Operational Backup

The reasons for making backup copies of data sets have changed in recent years. Some years ago you had to be prepared to lose a complete volume and its data sets from a head crash. This will not happen on todays DASD storage because it is RAID protected. On RAID DASD you will not lose a volume or a single data set for physical reasons. But you may lose a data set for logical reasons, that is, it may be deleted or partly deleted by mistake or by a program error. Therefore you still need backup copies, but less frequently.

When planning your SnapShot implementation, you have to review your operational backup procedures. Database utilities are used for operational backups where databases are involved. These procedures are covered in the specific database sections of this book.

Other backup procedures are volume backups, which are infrequently (weekly) performed, and incremental backups, which run daily. DFSMShsm does the volume and incremental backups automatically.

Volume backups are required, even if you expect no physical volume loss. They reduce the number of incremental backups you must save. They can also be used as a base for disaster recovery.

An *incremental backup* is a process in which data sets are backed up only if they have changed since their last backup. If you want to perform incremental backups with DFSMSdss, you can filter with BY(DSCHA,EQ,1) to dump only data sets that have changed since the last dump was taken. If you also code the RESET keyword, DFSMSdss changes the data-set-changed (DSCHA) indicator after successfully dumping the data set.

3.3.1 Using SnapShot for Operational Backup

Operational backups are those that an installation takes for protection against disaster, or against failure of hardware at the production data center.

3.3.1.1 SnapShot and Backup in a Single MVS Environment

In a single MVS environment, use SNAP VOLUME with COPYVOLID(NO) to leave the snapped volumes online in your MVS. If you do physical dumps to tape from those volumes, you can restore a single data set from tape if the original becomes corrupted. We recommend using physical dumps, because that is the only way to get all the noncataloged data sets on tape. See also 2.5.2, "Catalog Implications" on page 28.

You may not want to change all of your backup procedures, mainly not the incremental backups, because they do the necessary work automatically. If you need to save time, however, you can try to find the largest data sets of your incremental backup procedures, do a SNAP DATASET, and make the tape copy thereafter. This approach may significantly reduce your backup running time, but beware: You have to handle these data sets separately. For more details see 3.2, “Using SnapShot for Data Set Backups” on page 44.

3.3.1.2 SnapShot and Backup in a Second MVS

You can use a second MVS system (see Figure 27). It can run in an LPAR, and it must have shared access to the RVA.

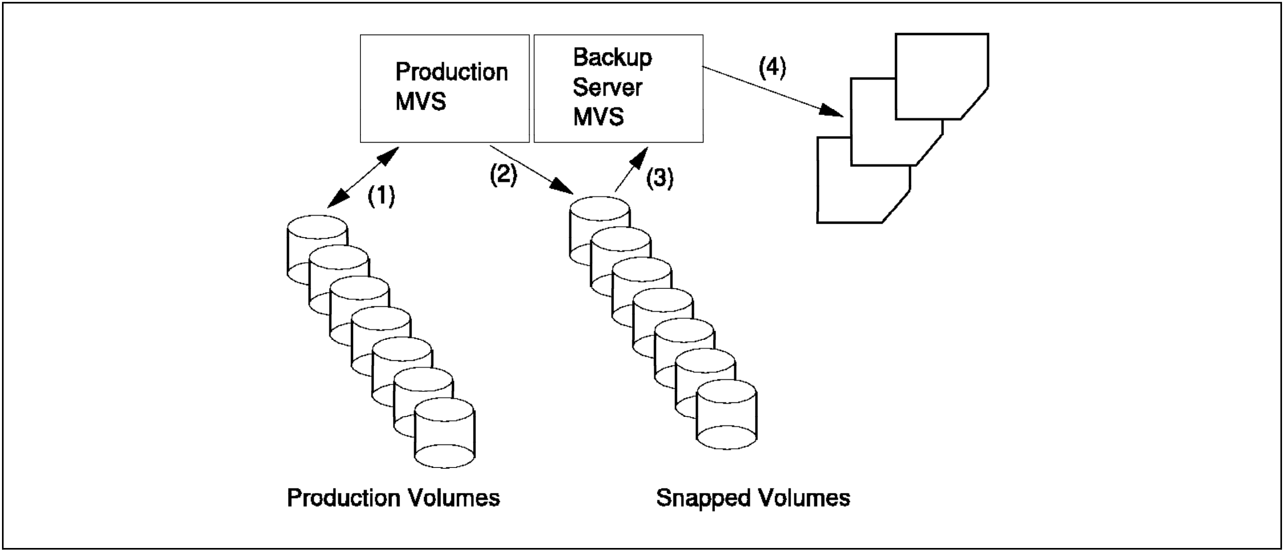


Figure 27. SnapShot and Backup in a Second MVS

As illustrated in Figure 27:

1. The production volumes are always online at the production MVS, but not at the backup server MVS.
2. The target volumes for the snap are online to the production MVS only during the snap.
3. The snapped volumes are varied online to the backup server MVS for the backup. They are initialized thereafter.
4. The tape drives are needed during backup at the backup server MVS.

To set up the additional MVS requires some work, but surely less work than changing your backup procedures.

Here are some guidelines:

- The backup server should be a minimum MVS. Only the functions required for the backup service should be included.
- Only the DFSMSrmm and the DFSMShsm control data sets are to be shared.
- The volumes used for snapping should not be defined as shared to the MVS systems, because they are online only to one MVS at a time, and only to either the production MVS or the backup server MVS.
- The device numbers used for these volumes should always be the same, and contiguous, to prevent a snap over a production volume by mistake.

- The master catalog of the backup server MVS is an identical copy or a superset of the production MVS′ master catalog. If the production MVS master catalog is changed, the backup server master catalog must also be changed.
- The volumes containing the user catalogs required the data sets to be backed up are also snapped.
- DFSMShsm backup runs in the backup server MVS, DFHSM recovery (HRECOVER) in the production MVS. DFSMShsm migrate processing remains unchanged in the production system.

In general, when you are operating in a shared DASD environment, you should use a serialization program, like the global resource serialization (GRS) component of MVS/ESA. The backup server MVS can be a member in the complex. There might be other serialization systems that do not allow duplicate volsers in their domain. You can use GRS if it is needed for other reasons, but the recommendation is to use the backup server MVS only for backups, in which case GRS may not be needed. Do not share the spooling system; use a remote job entry system to transmit the backup jobs to the backup server MVS.

3.3.1.3 DFSMShsm or DFSMSdss Backup in a Second MVS

If you are using DFSMShsm or DFSMSdss for a data set backup procedure, with incremental backup, we recommend the following procedure:

First you have to do a volume snap with COPYVOLID(YES) for all the affected volumes, including catalog volumes. After resetting all the change bits on the production volumes you have just snapped, the production can continue. The real backups can now be done in parallel on another MVS, let′s call it the *backup server MVS*.

It is not necessary to change your currently running backup jobs; only the system where they run changes. An additional job is inserted that does the volume snaps and resets the change bits.

Here are the necessary steps:

1. On the production MVS, when you are ready for the backup, set the target volumes online.
2. Snap the production volumes with COPYVOLID(YES) (Figure 28). The target volumes will be offline after the snap.

```
//*
//STEP1   EXEC PGM=SIBBATCH
//SYSTERM  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
 SNAP VOLUME( -
       SOURCE(VOL(9RM01C))  -
       TARGET(VOL(9RM01D))  -
       COPYVOLID(YES) -
             )
/*
```

Figure 28. Snap Volume

3. Reset the change bits. Use DFSMSdss to reset the change bits without moving the data (Figure 29 on page 52). Resetting the bits is necessary because DFSMShsm acts on the snapped volumes rather than the original volumes.

```
//*
//STEP01 EXEC PGM=ADRDSSU,REGION=8M
//SYSPRINT DD SYSOUT=*
//INVOL1   DD VOL=SER=9RM01C,UNIT=3390,DISP=SHR
//OUTDD1   DD DUMMY
//SYSIN    DD    *
      DUMP DATASET (   -
      INCLUDE(**) -
      BY((DSCHA,EQ,1))) -
      LOGINDDNAME(INVOL1) -
      OUTDDNAME(OUTDD1) -
      RESET
```

Figure 29. Using DFSMSdss to Reset the Change Bits (DSCHA)

Note:
The logical dump with DS(INC(**)) and LOGINDD specified and the output set to DUMMY will not do the reads.

4. Restart production.
5. On the backup server MVS, set the snapped volumes online.
6. Run your incremental backups or full volume dumps on the backup server MVS against the snapped volumes.
7. Vary volumes offline.
8. Use IXFP to delete the functional devices (which contain the snapped volumes; see Figure 30).

   Alternatively you can initialize the snapped volumes and change the volsers. Only a minimal INIT is required (Figure 31). It should be followed by interval DDSR.

```
//STEP01   EXEC PGM=SIBBATCH
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSIN    DD *
 DELETE DEVICE(SUBSYS(rvasys) -
              UNIT(uuuu) -
              FORCE -
              VERIFY(FDID(ff)))
```

Figure 30. Deleting a Functional Device Using SIBBATCH

```
//INITSNAP EXEC PGM=ICKDSF
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 INIT UNITADDRESS(1010) VOLID(UNQ001) DEVTYPE(3390) -
      VTOC(0,3,87) INDEX(10,1,44) VERIFY(VOL001) NOVALIDATE PURGE
```

Figure 31. Example of Minimal INIT

3.4 VSAM KSDS Reorganization

VSAM KSDS reorganization is a common process. It becomes necessary after lots of inserts have occurred. The inserts lead to CI and CA splits. New records are physically added at the end of the data set. The requirement for reorganization for performance reasons is reduced in an RVA environment because fragmentation has no performance penalty due to the virtual disk architecture. Reorganization may be necessary to reclaim space within the cluster.

Several procedures are used to reorganize a KSDS. You can find examples, in addition to those presented in this section, in *DFSMS/MVS Using Data Sets* (SC26-4922) manual.

3.4.1.1 Traditional Reorganization

Many installations use IDCAMS REPRO to write all of the data of the affected KSDS to a sequential file on tape or on another DASD. Then the original KSDS is deleted and redefined, and the KSDS is reloaded from the sequential file (Figure 32).

```
//UNLOAD   EXEC PGM=IDCAMS
//IN1      DD   DSN=HLQ.VSAM1.KSDS,DISP=SHR
//OU1      DD   DSN=HLQ.VSAM1.UNLD,DISP=(NEW,CATLG),
//         SPACE=(TRK,(1,1)),
//         DCB=(,RECFM=VB,BLKSIZE=4000)
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
   REPRO  INFILE(IN1) OUTFILE(OU1)
/*
//DELDEF   EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
   DELETE      HLQ.VSAM1.KSDS
   DEFINE CLUSTER (NAME(HLQ.VSAM1.KSDS) -
            RECORDS(120,10)        -
            SPEED                  -
            INDEXED )              -
     DATA  (NAME(HLQ.VSAM1.KSDS.DATA) -
            CISZ(4096)             -
            KEYS(6 2)              -
            RECSZ(80 96) )         -
     INDEX (NAME(HLQ.VSAM1.KSDS.INDEX) )
/*
//RELOAD   EXEC PGM=IDCAMS
//IN2      DD   DSN=HLQ.VSAM1.UNLD,DISP=SHR
//OU2      DD   DSN=HLQ.VSAM1.KSDS,DISP=SHR
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
   REPRO  INFILE(IN2) OUTFILE(OU2)
```

Figure 32. Traditional VSAM KSDS Reorganization

3.4.1.2 Reorganization with SnapShot

Figure 33 shows VSAM KSDS reorganization with SnapShot. Use SnapShot for the VSAM KSDS reorganization on the large data sets first. You can then see if you really save time with it. If the REORG steps are not in the critical path of your batch jobs, the effect may be negligible.

Use the following procedure for KSDS reorganization with SnapShot.

1. Take a snap copy of the VSAM KSDS data set; use a new data set name.
2. DELETE DEFINE the original.
3. Use IDCAMS REPRO to reload it.
4. If you need a backup copy on tape as well, you can make it now, using IDCAMS REPRO directly to tape. Or you can make a snap copy of the reorganized data set and use DFSMSdss for backup.
5. Delete the snapped copy to reduce NCL.

```
//SNAP     EXEC PGM=SIBBATCH
//SYSTERM  DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
 SNAP DATASET(SOURCE(HLQ.VSAM1.KSDS) -
      TARGET(HLQ.VSAM1SN1.KSDS))
/*
//DELDEF   EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
   DELETE      HLQ.VSAM1.KSDS
   DEFINE CLUSTER (NAME(HLQ.VSAM1.KSDS) -
            RECORDS(120,10)        -
            SPEED                  -
            INDEXED )              -
     DATA  (NAME(HLQ.VSAM1.KSDS.DATA) -
            CISZ(4096)             -
            KEYS(6 2)              -
            RECSZ(80 96) )         -
     INDEX (NAME(HLQ.VSAM1.KSDS.INDEX) )
/*
//RELOAD   EXEC PGM=IDCAMS
//IN2      DD   DSN=HLQ.VSAM1SN1.KSDS,DISP=SHR
//OU2      DD   DSN=HLQ.VSAM1.KSDS,DISP=SHR
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
   REPRO  INFILE(IN2) OUTFILE(OU2)
```

Figure 33. VSAM KSDS Reorganization with SnapShot

Chapter 4. Using SnapShot with DB2

Most customers need a mixture of hardware and software to provide a cost-effective method of managing their enterprises. With its ability to replicate data almost instantaneously and maintain multiple copies of the same data, SnapShot offers DB2 users of the RAMAC Virtual Array subsystem a number of new functions for managing and using data more effectively.

Using SnapShot with DB2 enables customers to:

- Recover to a point of consistency (POC)
- Quickly back up DB2 when making software upgrades or recovering from a DB2 failure
- Replicate data for application test purposes.

With the availability of virtual concurrent copy, customers can now exploit SnapShot from the DB2 utilities. Using virtual concurrent copy with DB2 enables you to:

- Make a SHRLEVEL REFERENCE copy of a single table space, with a very short amount of time during which the source object cannot be updated.
- Make a SHRLEVEL REFERENCE copy of a group of table spaces, with a very short amount of time during which the source objects cannot be updated.

For a description of the key DB2 functions, data sets, and issues, see Appendix C, "DB2 Overview" on page 169.

4.1 Requirements for Using Virtual Concurrent Copy with DB2

The ease with which you can exploit DFSMSdss SnapShot and virtual concurrent copy with DB2 depends on the level of DB2 and the version of DFSMSdss you are using.

DB2 Version 3 is the minimum level that enables you to recover, using the DB2 RECOVER utility, from a copy taken outside DB2′s control. DB2 Version 4 is the minimum level that enables you to image copy and recover your DB2 data by utilizing DFSMSdss concurrent copy through DB2 utilities. With the availability of virtual concurrent copy, you have more of an opportunity to take advantage of concurrent copy in the DB2 utilities.

To minimize the outage when using concurrent copy to copy a list of DB2 objects within a single DB2 COPY command, ensure that APAR PN92578 (PTF UQ12896 - V4; PTF UQ12897 - V5) is applied to your DB2 subsystems.

APAR PN92578 allows the DB2 COPY utility to improve the availability of the data being copied when the CONCURRENT and FILTERDDN keywords are used. With this APAR on, DFSMSdss first logically copies all data. The logical copies are then available for updating while DFSMSdss completes the physical copies of all of the data.

Without APAR PN92578 applied, each object copied required its own output data set, and the last object copied would not be available for updating until all previous objects in the list of logical and physical copies had completed.

In addition to applying APAR PN92578 to your DB2 subsystems, you should also apply APAR PQ15353 (PTF numbers are unavailable at this time). APAR PQ15353 improves the performance of the DB2 RECOVER utility when the RECOVER process uses image copies made with the CONCURRENT and FILTERDDN keywords specified.

APAR PQ15353 changes the DB2 RECOVER utility to invoke DFSMSdss only once with a list of all the objects to be restored. Without this APAR applied, the DB2 RECOVER utility invokes DFSMSdss each time for every object being recovered.

4.2 Benefits of Virtual Concurrent Copy for DB2

Virtual concurrent copy support in DB2 means that you can use the DB2 COPY utility to back-up your DB2 data that resides on an RVA. Thus you can back up single or multiple objects, using DB2 backup and recovery processes, and the only time your data is unavailable for updating is the time required to complete the DFSMSdss SnapShot logical copy of the objects. If you are currently making multiple backups of your DB2 data, you can benefit from minimal I/O from virtual concurrent copy while the copies are logically completed.

For full details on how to implement virtual concurrent copy in a DB2 environment, refer to *Implementing DFSMSdss SnapShot and Virtual Concurrent Copy*, SG24-5268.

4.3 Benefits of SnapShot with DB2

The benefits you can obtain from using SnapShot without virtual concurrent copy, as a partial or full replacement for the DB2 online utilities largely depend on your installation′s requirements. In this chapter we give examples of how you can benefit from using SnapShot with DB2 and explain the factors to consider when you implement SnapShot.

4.3.1 Image Copy

SnapShot can be used with DB2 to create instant copies of data and provides significant value when a consistent copy is required.

DB2 Version 3 does not automatically invoke SnapShot through the DB2 utilities. You can use SnapShot to take image copies, but those copies are not invoked from the DB2 COPY utilities and so are not recorded in the DB2 catalog table. Recovery using those copies is therefore a manual operation.

Use of SnapShot for DB2 image copy assumes that you have SNAP DATASET support for VSAM data sets. The support is provided in SnapShot Release 1.2.

SNAP DATASET offers you a greater level of control than SNAP VOLUME by enabling you to be selective over the data sets that you want backed up for onsite and remote site recovery. SNAP DATASET simplifies your backup process if your DB2 data resides on DASD volumes shared with other applications or system data that require a common POC.

4.3.2 Disaster Recovery

SNAP VOLUME enables you to snap an entire DB2 subsystem at a functional volume level, allowing update processing to continue as soon as the copy has completed. This technique reduces the time the service is interrupted to the time needed to stop DB2, snap the volumes, and restart DB2. If you rely on DFSMSdss dumps for DB2 recovery, SnapShot can significantly reduce your backup and recovery time.

4.3.3 Creating Test Data and Systems

Volume SnapShot enables you to copy large amounts of data from one MVS system to another with relative ease. In addition, because use of back-end storage is minimal, you may be able to keep multiple versions of test data online to speed regression testing processes.

You can use SnapShot to back up DB2 system volumes or data sets.

You may want to test your DB2 production for Year 2000 readiness, using a test MVS image.

With its ability to clone data and maintain multiple versions, SNAP DATASET enables you to perform repeatable tests on new applications or program fixes that require regressing data to an earlier point in time, without having to constantly reload the data.

4.4 General DB2 Considerations

In this section we discuss those areas where you need to take special action when using SnapShot with DB2.

4.4.1 DB2 COPY

In this section we describe the DB2 COPY utilities and how they relate to SnapShot.

You can take a backup of a table space using the DB2 utilities. DB2 records information about the image copy in the SYSCOPY table of the DB2 catalog (SYSIBM.SYSCOPY). DB2 uses this information during recovery so that it can automatically use the most recent image copy and apply records from the log.

If you use SnapShot for taking copies, you are responsible for recording when you take the SnapShot copy and to which volume or data set you write the copy.

The output data set of a copy must be cataloged in order for the RECOVER utility to identify it. SnapShot generates data set names that are compatible with DB2-generated names.

You can take copies, using SHRLEVEL(REFERENCE), which allows other applications to read the data while it is being copied. You can recover to a specific previous copy, a specific point in time, or a previous quiesce point.

DB2 also provides the option to take image copies using SHRLEVEL(CHANGE) which allows the data to be updated during the DB2 copy. This creates a ″fuzzy″ copy that can be recovered to a point of data consistency by applying more log records than with SHRLEVEL(REFERENCE). If you are using SHRLEVEL(CHANGE), do not attempt to recover to a specific image copy, because uncommitted data may be copied. Recovering to a specific image copy is possible with SHRLEVEL(REFERENCE).

SHRLEVEL(CHANGE) provides the highest online availability, but recovery may take longer as you must recover from the ″fuzzy″ copy. SnapShot does not provide any benefits in availability when you use SHRLEVEL(CHANGE).

4.4.2 DB2 QUIESCE

The QUIESCE utility creates a POC for a table or group of tables. If you have a group of tables that are interrelated, you must ensure that they are all quiesced together, to preserve data integrity. When you quiesce DB2, there is a disruption to applications because access to tables is restricted for the duration of the quiesce.

QUIESCE optionally writes updated pages from buffers onto the disks. This is important when you are taking a snap of the data as DB2 holds changed data in buffers for some time before the data can be committed to the disks. For details on DB2 buffer management, see C.1.6, “DB2 Buffer Pools” on page 173.

To recover to a consistent point in time, you have to quiesce DB2 or make access to the data read-only. You can take your copy after or during the quiesce. If you take it after the quiesce, DB2 can recover to the quiesce point provided the copy is taken with the DB2 COPY utilities. The log point of the quiesce is recorded in the DB2 catalog.

For many customers availability requirements are such that they cannot quiesce their databases or change access to read-only mode. Using SHRLEVEL(CHANGE) creates a fuzzy copy from which recovery is to a current point in time only and requires that the log data sets are available.

4.4.3 DB2 RECOVER

You have several options for recovering DB2:

- LOGONLY

  Use LOGONLY recovery when you are recovering using an image copy made outside DB2. If you use SnapShot without the DFSMS/MVS concurrent copy support, you must use LOGONLY recover. It bypasses the restoration of the image copy and applies subsequent logs to the copy that you have restored outside DB2.

  You need to tell DB2 the time of the copy you made so that it can apply the correct logs. A sample recovery scenario for LOGONLY recovery is documented in 4.6.3.2, “To Recover from a SnapShot Backup Using LOGONLY and TORBA” on page 65.

- TOCURRRENT, TORBA, or TOCOPY

  You have several options as to where you want to recover. You can recover to the current point in time, when DB2 will apply all available log data sets to the present (TOCURRENT).

  You may want to recover to a previous point in time, perhaps for application recovery reasons. In this case you can recover to a specific relative byte address (RBA; the TORBA option) or to a specific image copy (the TOCOPY option). Both options are supported when you use the RECOVER utility on a copy made with SnapShot. TOCOPY is not supported when you use LOGONLY recovery.

4.4.4 DSN1COPY

DSN1COPY is the DB2 stand-alone copy utility. You can use it to take copies of table spaces when the table space is inactive or quiesced.

Use DSN1COPY when creating data to be moved between DB2 systems. It has some facilities to make it easy to move data between DB2s, for example, it can translate object IDs (OBIDs), and reset the log RBA. For further details on DB2 OBIDs, refer to 4.9, “Cloning DB2 Data” on page 70. DB2 internal identifiers are particularly important when you are creating test data or cloning a system for application testing.

4.4.5 DB2 Naming

DB2 also records internal identifiers within each DB2 object, such as table or database, that are specific to a DB2 subsystem. If you copy data from one subsystem to another, unless the DB2 catalog is identical, or the internal identifier has not been used, you have to reconcile them.

Changing internal identifiers is possible but requires involvement of a database administration. Changing the internal identifiers does not rewrite each page unless you are using type 2 indexes, which record an OBID on each page. If you rewrite every page, you will not realize the space saving benefits of SnapShot, although you will get the benefit of faster copies.

4.5 Technical Considerations

A number of DB2 procedures are relevant when you use SnapShot for DB2.

4.5.1 Serialization

In DB2 the integrity of your copy is ensured by the QUIESCE process. For example, when you implement SnapShot in an automated environment, using a job scheduler, always verify that the DB2 table space is in read-only mode before you issue the SNAP command. Otherwise, data integrity will be compromised.

The TOLENQF parameter enables you to snap a volume or data set when exclusive serialization control cannot be obtained. Integrity of the data cannot be assured. However, DB2 does not hold an exclusive enqueue on resources, and DB2 locking is unaffected by enqueue failure. When using SNAP DATASET, specify TOLENQF(YES).

4.5.2 Down Level Detection

When using a stand-alone or non-DB2 utility, it is possible to replace a DB2 data set with an incorrect or outdated copy. This *down level* copy may cause data integrity problems. Performing a cold start of DB2 can also result in a down level condition.

DB2 associates a level ID with every page set or partition. A down level condition can be detected by DB2 and will result in errors. Down level detection (DLD) is essentially a safety check to ensure that the data and log are consistent.

If you are manipulating data sets in DB2, for example, if you restore individual data sets into an application testing environment, you must take account of DLD. DSN1COPY can reset the level ID to a neutral value so that you can move data into another DB2.

4.5.3 ICF Catalogs

DB2 maintains a set of tables that store all information about the DB2 system. These tables reside in the catalog and directory data sets.

The VSAM catalog name is the ICF catalog entry or the HLQ used by DB2. It is also called the VCAT. The catalog and directory are a single recoverable entity and should be backed up and recovered together.

If you are cloning a DB2 system for application development or testing or plan to use DB2 remote site recovery, the catalog and directory must be identical. It is essential that you ensure that catalog and directory data sets for the test system do not become available to the production system by mistake.

4.5.4 Data Sets

SnapShot supports SNAP DATASET for multivolume data sets. If you are snapping DB2, using SNAP VOLUME, and you have multivolume data sets, it is a user responsibility to ensure that all parts of a multivolume data set are copied at the same time. Otherwise data integrity problems might occur.

4.5.5 Referential Integrity

Referential integrity allows DB2 to maintain data consistency when there are relationships between a set of tables. If an application has updates that span several tables, it is essential that all the tables in the *table set* are quiesced and copied together.

For more information about referential integrity, see C.1.4.1, "What Is Referential Integrity?" on page 172.

The REPORT online utility provides information about table spaces. You can obtain the names of all table spaces and tables in a referential structure by using this control statement:

```
REPORT TABLESPACESET tablespace-name  parms
```

The REPORT utility, when used with this RECOVERY control statement:

```
REPORT RECOVERY tablespace-name  parms
```

provides you with information about the image copy data sets and archive log data sets that might be needed during a recover. You can obtain other information from the report, such as the type of image copy taken (full, incremental, or concurrent), whether a utility was terminated, and when the table space was quiesced.

For more information about the REPORT utility, refer to Chapter 2-16, "REPORT," in the *DB2 for OS/390 V5 Application Programming and SQL Guide*, SC26-8958.

4.5.6 Data Sharing

The only real difference between a data sharing and non-data-sharing environment is that you have to deal with more than one set of DB2 boot strap data sets (BSDSs) and more than one set of logs in a data sharing environment.

You need image copies of all your data, including the catalog and directory, and you need a copy of the BSDS and logs from all members in the group at the disaster site. The data sharing group at the disaster site must be a mirror image of the group at the primary site.

Sometimes it is not possible to quiesce all data sharing members to obtain a single recovery point for their logs. There are some enhancements to DB2 Version 4 that facilitate usage of the archive logs without having to quiesce the members, and at the recovery site, specify the same conditional restart point to which all members can recover.

If you are planning to use SnapShot for recovery in a data sharing environment, make sure that you refer to the latest DB2 manuals for details as some of the commands and procedures that are documented in this redbook are not applicable to a data sharing environment.

4.6 Sample DB2 Backup and Recovery Procedures Using SnapShot

In this section we describe the general flow of the disaster recovery backup and recovery procedures, and explain how SnapShot can be integrated into your disaster recovery plan.

4.6.1 Backup Using SNAP VOLUME

You can use the procedure described in this section with SNAP VOLUME or DFSMSdss full volume dumps. The procedure does not require support for SnapShot in DB2 or DFSMS/MVS concurrent copy support. For more information, see C.1.10.2, "Backup Using Full Volume Dumps" on page 180.

The procedure is based on the availability of a pool of target functional volumes and recovery to RVA DASD at the recovery site. Here are the steps to follow:

1. Ensure that all DB2 volumes are included in the snap and DFSMSdss dump JCL.
2. Check for utilities.

   To display outstanding utilities, enter:

   ```
   -DIS UTILITY(*)
   ```

   You must take whatever actions are appropriate to complete the utility.
3. Check the status of all objects.

   To display objects in a restricted state, enter:

   ```
   -DIS DATABASE(*) SPACE(*) LIMIT(*) RESTRICT
   ```

   You must take whatever appropriate actions are needed to recover the DB2 object.
4. Check the active or in-doubt threads and resolve them.

   There must be no active threads. To display threads, enter:

   ```
   -DIS THREAD(*) TYPE(*)
   ```
5. Establish a systemwide POC for DB2 and other subsystems.
6. Issue ARCHIVE LOG MODE(QUIESCE) WAIT(YES) TIME(n)

   This command is optional, but it provides a clean log. Verify that the ARCHIVE command completes successfully.
7. Print a copy of the BSDS.

   Use the print log map utility, DSNJU004, to print the BSDS. Print a hardcopy for your fire store; the information will aid you in the recovery of DB2.
8. Issue -STOP DB2 MODE(QUIESCE).
9. Use SIBBATCH to snap the source volumes (see Figure 34).

```
//SNAPVOLS EXEC PGM=SIBBATCH
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSIN    DD *
 SNAP VOLUME(SOURCE(VOLUME(E5DB01)) -
            TARGET(VOLUME(SNDB01)) -
            TOLERATENQFAIL(NO) -
            REPLACE(YES) -
            COPYVOLID(NO))
    . . .
    . . .
    . . .
/*
```

Figure 34. SIBBATCH Volume Snap

10. Issue -START DB2.

11. Take a DFSMSdss full volume dump of the target volumes (Figure 35) if you want to take them offsite.

```
//DSSDUMP  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//TAPEBKUP DD UNIT=3480X,DISP=(NEW,CATLG),
//         DSN=DFDSS.DUMP.SNDB01.VE5DB01(+1)
//SYSIN    DD *
  DUMP FULL INDYNAM(SNDB01) -
       OUTDD(TAPEBKUP) ALLDATA(*) ALLEXCP
```

Figure 35. DFSMSdss Full Volume Dump of the Target Volume

4.6.2 Recovery Using SNAP VOLUME

The procedure described in this section allows you to recover at a disaster recovery site from backups taken using the procedure described in 4.6.1, “Backup Using SNAP VOLUME” on page 62. Here are the steps:

1. Set up the environment.

   Restore the DB2 system data sets.

2. Initialize the RVA DASD volumes to the target recovery volume names.

3. Restore the target recovery volumes.

4. Snap the target volume back to the original source volsers.

   Repeat this process until all DB2 data has been restored.

5. Modify the DSNZPARM module:

   - LUNAME

     If the LU name of the recovery machine differs from the name stored in the BSDS, use DSNJU003 to change it.

6. Print the BSDS.

   Find which active log DB2 was using.

7. Print the current active log.

   Run DSN1LOGP to verify that no transactions were processing at the end of the last active log and to determine whether utilities were in flight at the time.

8. Start DB2 with access MAINT:

   ```
   -START DB2 ACCESS(MAINT)
   ```

9. Health check the catalog and directory:

   ```
   -DIS DATABASE(DSNDB*) SPACE(*) LIMIT(*) RESTRICT
   ```

   Resolve any errors.

10. Stop and restart DB2 when you are satisfied that all problems have been resolved.

4.6.3 Recovering with a Data Set Copy Not Made by DB2

Use the sample procedure in this section to back up a table space for normal recovery purposes and as part of the data regression process for application testing.

This procedure requires SNAP DATASET support for VSAM data sets.

4.6.3.1 To Obtain a SnapShot Backup

To obtain a SnapShot backup:

1. Select the source data sets to be snapped.
2. Establish whether referential integrity is being used.

   Run the REPORT utility with the TABLESPACESET option. See 4.5.5, “Referential Integrity” on page 61.
3. Start the DB2 objects being backed up for read only access by issuing this command:

   ```
   -START DATABASE(database name) SPACENAM(tablespace-name) ACCESS(RO)
   ```

   This command ensures that no updates to data occur during the procedure.
4. Run QUIESCE with the WRITE(YES) option to quiesce all DB2 objects being backed up and flush any updated pages to disk (Figure 36).

```
//JSTEP01  EXEC DSNUPROC,UID='SNAP.QUIESCE',SYSTEM='DB2B',
//             UTPROC=''
//SYSIN    DD  *
  QUIESCE TABLESPACE SNADB001.SNATS001
/*
```

Figure 36. QUIESCE of a Table Space

5. Check that the QUIESCE utility completes successfully. Back up the data sets using SnapShot. The command in Figure 37 uses the SIBBATCH interface.

```
//SNAPVOLS EXEC PGM=SIBBATCH
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSIN    DD *
SNAP DATASET(SOURCE(DB2B.DSNDBC.SNADB001.SNATS001.I0001.A001)) -
     TARGET(DATASET(DB2B.DSNDBC.SNADB001.SNATS001.I0001.A001.CPY)) -
     TOLERATENQFAIL(NO) -
     REPLACE(YES))
/*
```

Figure 37. SIBBATCH Data Set Snap

6. Issue the following command to allow transactions to access the data:

   ```
   -START DATABASE(database name) SPACENAM(tablespace-name)
   ```

4.6.3.2 To Recover from a SnapShot Backup Using LOGONLY and TORBA

To recover from a SnapShot backup using the LOGONLY and TORBA control parameters of the RECOVERY utility:

1. Ensure that no other transactions can access DB2 objects between the time that a data set is restored and the time that RECOVER LOGONLY is run.

2. Stop the DB2 objects being recovered by issuing this command:

   ```
   -STOP DATABASE(database name) SPACENAM(tablespace-name)
   ```

3. Start the DB2 objects being recovered by issuing this command:

   ```
   -START DATABASE(database name) SPACENAM(tablespace-name) ACCESS(UT)
   ```

4. Use IDCAMS to delete all DB2 data sets that are being recovered (Figure 38).

```
/DELETE   EXEC PGM=IDCAMS
/SYSPRINT DD SYSOUT=*
/SYSIN    DD *
DEL 'DB2B.DSNDBC.SNADB001.SNATS001.I0001.A001'
```

Figure 38. Using IDCAMS to Delete a Source DB2 Data Set

5. Restore all DB2 data sets that are being recovered.

   Figure 39 uses SIBBATCH to snap a target DB2 data set to the source data set name.

   In this example, we have used the REPLACE(NO) option to ensure that we do not inadvertently snap over a similar named DB2 data set.

```
//SNAPVOLS EXEC PGM=SIBBATCH
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSIN    DD *
SNAP DATASET(SOURCE(DB2B.DSNDBC.SNADB001.SNATS001.I0001.A001.CPY)) -
     TARGET(DATASET(DB2B.DSNDBC.SNADB001.SNATS001.I0001.A001)) -
     TOLERATENQFAIL(NO) -
     REPLACE(NO))
/*
```

Figure 39. SIBBATCH Data Set Snap

6. Run the RECOVER TABLESPACE utility without the TORBA parameter and with the LOGONLY parameter to recover the DB2 data sets to the current point in time and to perform forward recovery using DB2 logs.

   Figure 40 on page 66 shows a table space recovery to the current point in time.

```
//recover EXEC PGM=DSNUTILB,REGION=32M,PARM='DB2B'
//STEPLIB  DD  DSN=DB2.LINKLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//UTPRINT  DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(50,30)),DISP=(,PASS)
//SYSUDUMP DD  SYSOUT=*,OUTLIM=30
//SYSIN    DD  *
 RECOVER TABLESPACE SNADB001.SNATS001 LOGONLY
/*
```

Figure 40. Using LOGONLY to Recover a DB2 Table Space

If you want to recover the DB2 data sets to a previous point in time, run the RECOVER TABLESPACE utility with both the TORBA and LOGONLY parameters (Figure 41).

```
//recover EXEC PGM=DSNUTILB,REGION=32M,PARM='DB2B'
//STEPLIB  DD  DSN=DB2.LINKLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//UTPRINT  DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(50,30)),DISP=(,PASS)
//SYSUDUMP DD  SYSOUT=*,OUTLIM=30
//SYSIN    DD  *
 RECOVER TABLESPACE SNADB001.SNATS001 LOGONLY TORBA (X'000007425468')
/*
```

Figure 41. Using TORBA and LOGONLY to Recover a DB2 Table Space

7. If you recover the DB2 objects to a previous point in time, recover all indexes on the recovered object (see Figure 42).

```
//recover EXEC PGM=DSNUTILB,REGION=32M,PARM='DB2B'
//STEPLIB  DD  DSN=DB2.LINKLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//UTPRINT  DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(50,30)),DISP=(,PASS)
//SYSUDUMP DD  SYSOUT=*,OUTLIM=30
//SYSIN    DD  *
 RECOVER INDEX (ALL) TABLESPACE SNADB001.SNATS001
/*
```

Figure 42. Recovering Table Space Indexes

8. Issue the following command to allow access to the recovered object if the recovery completes successfully:

```
-START DATABASE(database name) SPACENAM(tablespace-name) ACCESS(RW)
```

4.6.3.3 Considerations

There are a number of additional points to consider when you use SnapShot for DB2 backup and recovery:

- Avoid using the lower level qualifiers that describe the database, table space, and index names in the target data set name. You may have similar table spaces in different databases, and it may prove confusing when recovering data at a later time.

- Take care not to exceed the maximum number of characters when naming your target data set. Data set names cannot exceed 44 characters. For details on DB2 naming conventions, see C.1.8, “Data Set Naming Convention” on page 176.
- Consider taking another SnapShot copy of the DB2 data sets as a precautionary measure before embarking on recovery.
- With the LOGONLY option, when recovering a single piece of a multipiece linear page set, RECOVER opens the first piece of the page set. If the data set is migrated by DFSMShsm, it is recalled by DFSMShsm. Without LOGONLY, a data set recall is not requested.
- Backing up a single piece of a multipiece linear page set is not recommended. It can cause a data integrity problem if the backup is used to restore the data set at a later time.
- If you snap a data set at an arbitrary point in time, you must secure the same arbitrary point in time for the log.
- If you are snapping large amounts of data for backup purposes, we recommend that you back up the target data sets to tape and delete them afterward, to avoid NCL problems.

4.7 Application Testing

SnapShot support for VSAM data sets enables you to make multiple copies of the same data, giving you added flexibility when testing new applications or program modifications. If your test process involves frequently regressing DB2 test data to a previous point in time by using a load utility, SnapShot can help you reduce the time needed to restore data.

4.7.1 Recovering Application Test Data

Replace the load phase with the following procedure:

1. Select the source data sets to be snapped.
2. Establish whether referential integrity is being used.
3. Take a backup of the DB2 data sets, using the procedure given in 4.6.3.1, “To Obtain a SnapShot Backup” on page 64.
4. Perform your testing.
5. When you want to reset your test environment, recover the DB2 data sets, using the procedure described in 4.6.3.2, “To Recover from a SnapShot Backup Using LOGONLY and TORBA” on page 65.

   Execute the RECOVER TORBA LOGONLY, using the RBA for the quiesce taken at the previous point in time.

   Run the REPORT utility with the RECOVERY option to obtain the RBA.
6. Recover all indexes on the recovered objects.
7. Perform a second test on the same data.

The advantages of this method are:

- It is repeatable.

- When you add or delete data stored in the tables, you can make multiple backups at key points throughout testing to give you the flexibility to regress data to any given point.

4.7.2 Considerations

We recmmend:

- Using a load utility if the time taken to recover indexes proves slow.
- Recording the date and time when the backup was taken. They will help you to determine the RBA needed for the recovery when you run the REPORT utility.
- Taking a new backup if you undertake any action that would invalidate the SnapShot backup, such as making changes to a table structure.

4.8 Using SnapShot for DB2 System Changes and Error Recovery

In this section we discuss some possible uses of SnapShot to back up your DB2 subsystem.

You can use SnapShot to back up the DB2 catalog and directory when:

- Applying new levels of software to DB2
- Recovering from certain types of DB2 failure.

SnapShot can be used to take an almost instantaneous backup of your DB2 subsystem, giving you a fallback position should the recovery go wrong.

DB2 subsystem data can be allocated on either private volumes or a mixture of private and SMS volumes. The backup method that you choose will depend on several factors; for example, you may prefer to use SNAP DATASET because:

- DB2 data is on volumes shared with other system or application data sets that are outside DB2.
- An ICF catalog resides on the source volume.

When planning your recovery procedures, determine the best method of recovery first and base the backup procedure on it.

You can easily create complete copies of your DB2 subsystem, using either SNAP VOLUME or a combination of SNAP VOLUME and SNAP DATASET, eliminating the time needed to take a DFSMSdss dump.

We recommend that you prepare a general set of procedures to cover both software changes and error recovery situations.

4.8.1 DB2 Subsystem Backup

Here are the basic steps for backing up a DB2 subsystem:

1. Identify the source data sets to be snapped.
2. Disable any storage housekeeping, to prevent changes to the ICF catalogs.
3. Stop all transaction managers.
4. Perform any prerequisite check for the software change.
5. Check DB2 for active threads and utilities.

6. Stop all remote access to DB2.
7. Stop DB2.
8. Disable any automation that starts DB2.
9. Export all ICF catalogs with DB2 HLQs.
10. List the ICF catalogs of all DB2 HLQs.
11. SNAP all source DB2 volumes and data sets.
12. Make the software change.
13. Start DB2 in maintenance mode.
14. Perform installation verification checks.
15. If successful, restart DB2 for normal access.

If DB2 is in a failed state, begin recovery from the SNAP step (step 11), substituting change actions for recovery actions.

4.8.2 DB2 Subsystem Recovery Flow

The general flow of a DB2 subsystem recovery is:

1. Snap the target data sets over the source data sets.
2. For snapped volumes:
    a. Vary source volumes offline.
    b. Vary target volumes online.
    c. Import all ICF catalogs with DB2 HLQs.
3. List the ICF catalogs of all DB2 HLQs.
4. Start DB2 in maintenance mode.
5. Perform recovery verification checks.
6. If successful, restart DB2 for normal access.
7. Reformat the volsers of duplicate source volumes.

If DB2 is in a failed state, this procedure will restore the DB2 to the point of failure to enable you to recommence the recovery.

4.8.3 Considerations

We recommend:

- Backing up all data affected by the software change

  Make sure you include all data sets that will be updated by the software change.

- Not recovering to a snapped version of the DB2 catalog and directory if you have started DB2 for normal access and updates have been made to application data

4.9 Cloning DB2 Data

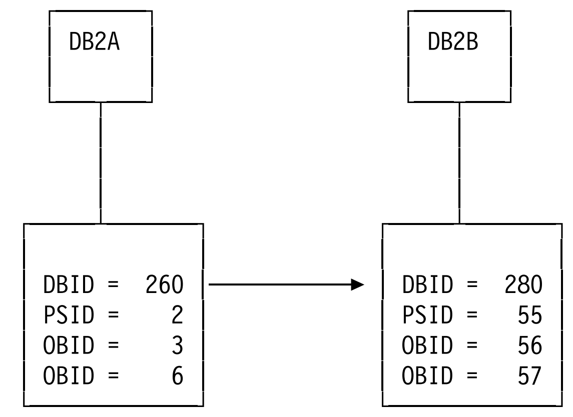
Copying DB2 data from one DB2 subsystem to another can be complex. If you are planning to selectively copy parts of DB2, you must have a deep and detailed understanding of both DB2 and your application. It is advisable to snap all of the required system and application data together. You may have some errors on restart, for example, when you write to some of the data sets following the restart, but the DB2 restart should be successful. In a test environment, complete data integrity may not present a problem.

4.9.1 DB2 Internal Identifiers

When DB2 creates an object, it assigns internal identifiers (like object descriptors or OBIDs) that are embedded within the data and are specific to each DB2. If you copy data from one DB2 subsystem to another, unless the DB2 catalog is identical or the internal identifier is unassigned, you will have to reconcile the difference between the identifiers. Changing DB2 internal identifiers is a database administrator (DBA) task.

Tools such as the DB2 DSN1COPY utility are available to copy data from one DB2 subsystem to another. DSN1COPY overcomes the problem of unique identifiers by translating them as the data is loaded into the table.

Figure 43 shows an example of how DSN1COPY translates the database identifier (DBID), page set identifier (PSID), and table identifiers (OBIDs).



```
//SYSXLAT  DD *
260,280
2,55
3,56
6,57
/*
```

Figure 43. DSN1COPY Translating DB2 Internal Identifiers

4.9.2 Cloning the DB2 Catalog and Directory

If you have a test MVS image that shares DASD with your production service, you might want to consider creating a clone copy of your production DB2 catalog and directory on your test MVS system. The benefit of doing this is that you can copy selective application data from the production DB2 to the test DB2, without the complexity of having to define objects to the test DB2 and translating their identifiers.

As long as no changes are made to the object′s structure, you can copy table spaces to the test system and recover them, using the procedure given in section 4.7.1, “Recovering Application Test Data” on page 67.

You need to reset the highest RBA of the test DB2 subsystem ahead of the production DB2′s highest RBA. We recommend that you perform a conditional restart of the test DB2, specifying an RBA well in advance of the production DB2, to avoid the problem of recovering a table space containing RBAs higher than the highest written RBA of the test DB2.

4.9.3 Copying a DB2 Subsystem

In the example that follows we assume there is no duplication of RACF, data set names, and so forth between the source and target MVS systems.

4.9.3.1 Preparation

Prepare the test system environment beforehand. Review the following items:

- System parameters
- Synchronization of SnapShot testing with other DBMSs
- Control of remote access during testing
- DB2 procedures
- RACF structure
- VTAM definitions
- ZPARM definitions
- DB2 data sets to be snapped

4.9.3.2 DB2 System Data Sets

You must include the following DB2 data sets:

- The catalog and directory data sets
- Both BSDSs
- All active log data sets
- All application data sets

You do not have to include the archive logs.

4.9.3.3 Migration Flow

Here is the procedure to follow when using SnapShot to duplicate a DB2 environment.

1. Ensure that all source data sets are on primary DASD.
2. Issue -ARCHIVE LOG MODE(QUIESCE) for a systemwide quiesce point.
3. Use -STO DB2 MODE(QUIESCE) to stop the source DB2.
4. Snap all DB2 volumes, using COPYVOLID(YES).

   Ensure that the snap is synchronized with your transaction manager, if the transaction manager is included in the move.

5. Snap the DB2 system data sets.
6. Export all ICF catalogs with DB2 HLQs.
7. Optionally, reformat the target volsers to protect against accidental usage.
8. Initialize an equivalent number of permanent volumes on the test system to receive the cloned data.
9. Vary the target volumes online on the test image.
10. Define identical empty ICF catalogs on the test volumes.
11. Import the source ICF Catalogs.
12. Define only the DB2 HLQs.
13. DFSMSdss logically copy the data from the snapped volumes to the permanent volumes.

    If the data is intended for test purposes only, consider copying the data to a single SMS storage group, to simplify the copy process and save on UCBs.
14. Snap the DB2 system data sets.
15. Modify ZPARM definitions.
16. Modify the BSDSs.
17. IPL to introduce APF authorization.
18. Start DB2 in maintenance mode, using the -STA DB2 ACCESS(MAINT) command.

4.9.3.4 Technical Considerations

When cloning a DB2 subsystem to another MVS image, consider:

- ZPARM definitions
- BSDS definitions

  Modify the BSDS.
- Removing active log records
- RACF
- The DB2 subsystem identifier (SSID)
- ICF catalog
  - Duplicate HLQs
  - Placement
- The DB2 special recognition character (SRC)
- Duplicate data set names.

4.9.4 Copying DB2 Databases

If you snap the DB2 subsystem and subsequently snap databases or tables, the log may be in advance of the data, and DB2 will not access the data sets. You can restart DB2 without some data sets, but that will result in a message indicating that the resource (data set) is not available. You can restore individual tables, using the procedure in 4.6.3.2, “To Recover from a SnapShot Backup Using LOGONLY and TORBA” on page 65.

When you clone DB2 applications, the additional back-end storage may become a consideration if you snap log data sets. In a typical DB2 system, the log data sets account for 20%—30% of the I/Os, so the update rate is high. The price in NCL may be sufficient that you prefer to restart an application without the log in the new DB2 environment.

Chapter 5. Using SnapShot with IMS/ESA

IMS provides several means of producing image copies, including static copies (no updates take place during the copy) and fuzzy copies (concurrent updates). IMS also provides user image copy (UIC) registration for Database Recovery Control (DBRC) when non-IMS utilities are used for backup and recovery. A UIC is any copy that is not directly controlled by the DBRC process. If you do not use one of the IMS utilities to make image copies, you must keep track of the copies because IMS does not track UICs.

Fuzzy copies enable an installation to extend the online availability of the IMS system. Static copies are more suitable for disaster recovery, or for recovering a point-in-time position, for example, following an application logic error. Fuzzy copies require concurrently created logs as input to the recovery process. The ability to take a static copy, which minimizes the unavailability of the database, is very attractive. This ability is provided by both SnapShot and concurrent copy.

IMS can support the fast data replication technique of SnapShot as a UIC. The UIC process also includes a copy made by concurrent copy before IMS/ESA Version 6. The user must restore the data set before invoking IMS recovery.

5.1 Using Virtual Concurrent Copy for IMS

IMS/ESA Version 6 introduces the new Image Copy 2 utility, DFSUDMT0.

Prior to IMS/ESA Version 6, concurrent copy was available but was not directly supported by DBRC. The image copy could be created and used, but the installation had to keep track of the copy. If the database was registered with DBRC, the installation indicated the concurrent copy to DBRC as a UIC. The concurrent copy had to be restored by the installation and registered as a user recovery with NOTIFY.RECOV before recovery could occur under DBRC control.

With DFSUDMT0, IMS/ESA Version 6 adds DBRC support for concurrent copy in both fuzzy and static modes. The database must be registered with DBRC. The DUMP format of the output data set differs from a regular IMS image copy format data set and cannot be processed directly by IMS utilities.

The recovery utility runs unchanged but must execute under DBRC control to enable use of the virtual concurrent copy made by DFSMSdss.

For full details on implementing virtual concurrent copy in an IMS environment refer to *Implementing DFSMSdss SnapShot and Virtual Concurrent Copy*, SG24-5268.

In this chapter we give an overview of IMS databases and backup procedures and explain how to use SnapShot to create image copies of IMS databases.

5.2 Overview of IMS Databases

IMS databases can consist of any of VSAM KSDSs, VSAM ESDSs, and OSAM data sets. OSAM is an IMS access method, and the data sets appear to the system as physical sequential data sets. The IMS control region logically links these database data sets to form databases and applications. An IMS database can consist of one or more database data sets, and any database can be accessed by one or more applications (Figure 44). For this reason, it is often difficult to find a POC for a single application or database.

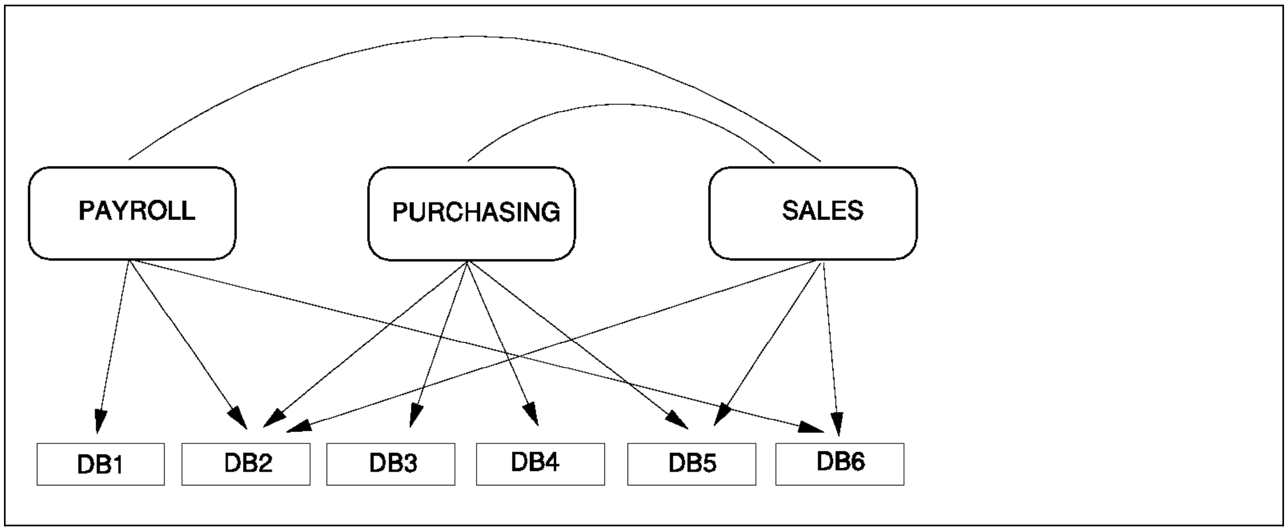


Figure 44. IMS Database Relationships

5.3 Overview of IMS Image Copy Procedures

Backups or image copies of IMS databases are necessary to provide insurance against the risk of data loss.

IMS provides utilities for creating and managing image copies. All image copy utilities work at the database data set level, and integrity is controlled if the database is registered with the database recovery control (DBRC) region.

Both static and fuzzy image copies are supported. A static image copy is a copy of a database taken while no updates occur. A static image copy is created by running the Database Image Copy utility (DFSUDMP0).

Due to the complex relationships between databases and applications, most installations have to stop access to all databases for the duration of the batch image copy if they want a static image copy. For many high availability installations, the extended outage required to create a static image copy using the Database Image Copy utility is unacceptable.

The Database Image Copy utility can be run with the CIC parameter to create a fuzzy image copy. IMS provides several other utilities for creating fuzzy image copies.

Fuzzy image copies can be created periodically while database updates occur. At regular intervals, for example, midnight each day, all databases are closed simultaneously to create a POC. Database access is unavailable only for the short period of time during which the databases are closed (see Figure 45 on page 77).

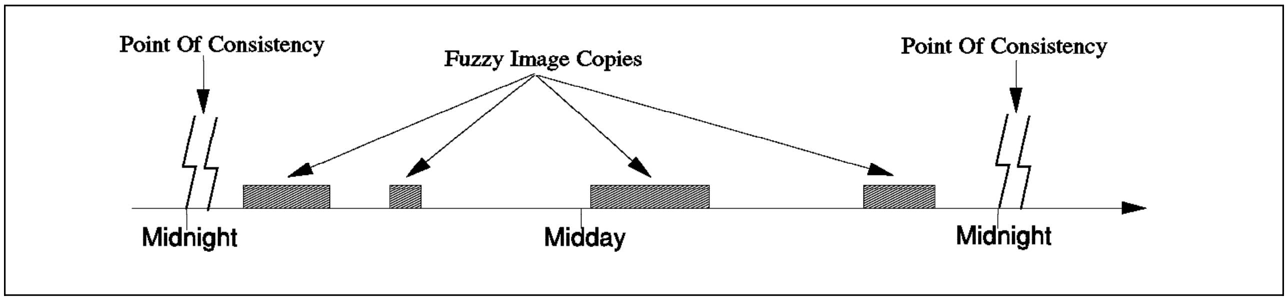


Figure 45. Creating Fuzzy Image Copies

5.3.1 IMS Version 6 Image Copy 2

IMS V6 introduces a new utility called Database Image Copy 2 (DFSUDMT0), which supports the DFSMS/MVS 3990 concurrent copy interface. Thus it is possible to create a static image copy with minimal interruption to online data availability.

The database must be registered with DBRC. Database Image Copy 2 creates an image copy in DFSMSdss dump format, which is registered with DBRC as a new image copy type. Up to four image copies can be created at once. Database updates can be resumed as soon as the concurrent copy session has begun.

5.3.2 Using SnapShot for IMS Image Copies

To use SnapShot to create a static image copy, all database updates must be stopped from all systems to achieve a POC. Updates must also be stopped for every related database. Due to the complexity of many IMS systems, the only way to achieve a POC may be to stop updates to *every* IMS database.

The database data sets are then snapped, and database updates can be restarted. The snap copies of the database data sets are then copied to tape by using DFSMSdss. Once the DFSMSdss copy to tape completes, the image copy must be registered with DBRC as a user image copy.

If a database has to be restored from a DFSMSdss copy, the database data sets must be restored from the DFSMSdss copy. All data belonging to the database must be restored, as well as all related databases, to avoid integrity problems. Once all required database data sets have been restored to the same point in time, the recoveries must be registered with the DBRC.

5.3.3 Benefits

SnapShot can provide these significant benefits in an IMS environment:

- Multiple database data sets can be snapped in a small period of time for consistency.
- Application outage can be greatly reduced because SnapShot is faster than Database Image Copy utility DFSUDMP0.
- More frequent image copies may be possible because of the reduced application outage required to take an image copy.

- SnapShot supports all database types.

5.3.4 Considerations

There are some additional factors to consider when using SnapShot for IMS backup and recovery:

- If SnapShot is used to back up databases at the data set level, a separate snap operation must be performed for each data set.
- A POC must be achieved that includes all related databases.
- A data set can be snapped only to another functional DASD device in the same RVA in which the source data set resides.
- In an SMS environment, the target of the snap must have a different data set name.
- Logical completion and physical completion of the image copy are separate.
- The image copy is created outside the control of IMS, so IMS must be notified when an image copy has been created.

5.3.5 Providing a Point Of Consistency

To ensure a clean point of recovery, you must ensure that changes to a database do not occur for the duration of the snap. All related database data sets must be snapped, to avoid integrity problems.

Updates to every related database must be disabled.

To avoid pseudoabends, any transactions or programs which can update the databases must be stopped. Use IMS command /STO TRAN xxxx.

Note

Some transactions can detect an unavailable database and defer processing of the message until the database is available again. These transactions do not have to be stopped.

Once all transactions have been stopped, every related database must be disabled for update. This can be done by issuing IMS command /DBD xxxx for each database, which will place the databases in a read-only state. Alternatively, IMS command /DBR xxxx can be issued for each database which will disable both read and write access.

IMS Version 5: IMS Version 5 implements *update sets*. An update set begins when any IMS system has update authority to the database and ends when every IMS system releases update authority to the database. An update set removes the need to perform an online log data set (OLDS) log swap to create a recovery point.

Before IMS Version 5: For IMS Version 4 and earlier, to prevent excessive OLDS log swaps, issue the /DBR or /DBD commands with the NOFEOV parameter. Issue the *last* /DBR or /DBD command without the NOFEOV parameter to cause an OLDS log swap and provide a clean recovery point.

A POC is also achieved when IMS is not active.

5.3.6 Snapping the Databases

IXFP performs the snaps of the database data sets. The commands can be issued from a batch job by calling the SIBBATCH program, from TSO by using the SIBADMIN command, or by operator command.

Database data sets can be snapped at the data set or volume level. Data set and volume snap are discussed extensively in Chapter 2, "SnapShot Technical Considerations" on page 9.

5.3.6.1 Snapping Database Data Sets

Database data sets can be snapped at the data set level. A separate snap command is required for each database data set. In an SMS environment, the target data sets must have a unique name.

IMS's GENJCL utility can be used to generate JCL statements and the necessary SNAP DATASET commands. Refer to *IMS/ESA Utilities Reference: System*, SC26-8035 for information on GENJCL.

Figure 46 shows a SIBBATCH batch step to snap three database data sets. All snap commands can be issued from one job step, and the snaps occur serially.

SnapShot defines and allocates the target data sets. The target data set names have SNAP appended to them in this example.

```
//SNAPDBDS EXEC PGM=SIBBATCH
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSIN    DD *
SNAP DATASET(SOURCE(DATASET(HLQ.DATABASE.PROD1)) -
             TARGET(DATASET(HLQ.DATABASE.PROD1.SNAP)) -
             TOLERATENQFAIL(NO) -
             REPLACE(NO))
SNAP DATASET(SOURCE(DATASET(HLQ.DATABASE.PROD2)) -
             TARGET(DATASET(HLQ.DATABASE.PROD2.SNAP)) -
             TOLERATENQFAIL(NO) -
             REPLACE(NO))
SNAP DATASET(SOURCE(DATASET(HLQ.DATABASE.PROD3)) -
             TARGET(DATASET(HLQ.DATABASE.PROD3.SNAP)) -
             TOLERATENQFAIL(NO) -
             REPLACE(NO))
```

Figure 46. Using SIBBATCH to Snap Three Data Sets

5.3.6.2 Snapping Database Volumes

We recommend snapping and backing up databases at the data set level. However, database snaps *can* be performed at the volume level. You must ensure that the volumes for every appropriate database are snapped. If multivolume database data sets are snapped, each volume on which the multivolume data sets reside must be snapped.

The target volumes must be defined as the same device type and model as the source volumes and reside within the same RVA subsystem.

Figure 47 on page 80 shows a batch job to snap a virtual 3390-03 IMS database volume with volser IMSVL1. Unit address 1010, which is a virtual 3390-03 within the same RVA subsystem, is the target of the snap. Unit 1010 was previously used as the target of another snap, and the volume has since been dumped to tape, so unit 1010 can now be overwritten.

The snap is run with COPYVOLID(NO) and CONDVOL(LBL), so the target′s volser (SNPVL1) is not modified by the snap. The VVDS and VTOC are copied from the source and do not match the volser of the target volume. Therefore the target volume has a unique volser and can remain online to be dumped to tape. If the volume later needs to be restored, volume IMSVL1 must be initialized before the restore starts, and the restore should not contain the DFSMSdss COPYVOLID parameter. After the restore completes, the VVDS and VTOC match the original volser (IMSVL1).

```
//SNPIMSVL EXEC PGM=SIBBATCH
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSIN    DD *
SNAP VOLUME(SOURCE(VOLUME(IMSVL1)) -
            TARGET(UNIT(1010)) -
            TOLERATENQFAIL(NO) -
            REPLACE(YES) -
            COPYVOLID(NO) -
            CONDVOL(LBL))
```

Figure 47. Using SIBBATCH to Snap a Volume

5.3.7 Restarting the Databases

Once all databases have been snapped, updates can be resumed to the databases. You can restart any databases that were stopped, using the IMS command /STA DB xxxx. You can restart any transactions that were stopped, using IMS command /STA TRAN xxxx.

5.3.8 Copying the Snap Copies to Tape

Now you must copy the snap copies of the databases to tape, using DFSMSdss. The method you use depends on whether the database data sets were snapped at the volume level or the data set level.

5.3.8.1 Copying Snapped Database Data Sets to Tape

Figure 48 on page 81 shows a DFSMSdss job to copy a set of snap target data sets to tape. In this example, the snap copies are deleted after they have been successfully copied to tape. You may want to leave these copies on virtual DASD after they have been copied to tape, to prevent having to wait for physical tape mounts if you need to restore the data sets.

```
//DSSCOPY  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//TAPEBKUP DD UNIT=3480X,DISP=(NEW,CATLG),
//         DSN=HLQ.DATABASE.BACKUP
//SYSIN    DD *
  COPY DATASET(INCLUDE(HLQ.DATABASE.**.SNAP) -
       TOLERATENQFAILURE(NO) -
       OUTDDNAME(TAPEBKUP) -
       DELETE
```

Figure 48. Using DFSMSdss to Copy Snap Target Data Sets to Tape

5.3.8.2 DFSMSdss Physical Dump of Snapped Volumes

The entire snap target volume can be physically dumped to tape (Figure 49). Every track on the volume will be dumped, regardless of the data type or catalog status. The data sets on the volume do not need to be cataloged.

```
//DSSDUMP  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//TAPEBKUP DD UNIT=3480X,DISP=(NEW,CATLG),
//         DSN=DFDSS.DUMP.IMSVL1
//SYSIN    DD *
  DUMP FULL INDYNAM(SNPVL1) -
       OUTDD(TAPEBKUP) ALLDATA(*) ALLEXCP
```

Figure 49. DFSMSdss Physical Volume Dump

5.3.8.3 DFSMSdss Logical Dump of Snapped Volumes

Logical dump requires that each VSAM data set on the volume being dumped be cataloged. The database data sets on the snap target volume will be uncataloged. Therefore each VSAM database data set on the snap target volume must be recataloged if a DFSMSdss logical dump is to be performed (Figure 50).

The database data sets on the target volume have the same names as the database data sets on the source volume, so either the source VSAM database data sets must be renamed, which is not practical for IMS database data sets, or the target volume must be varied online to another system that has an independent catalog structure. Every VSAM database data set on the target volume can then be cataloged on the second MVS system.

```
//RECAT     EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//VOLDD     DD  VOL=SER=IMSVL1,UNIT=3390,DISP=SHR
//SYSIN     DD   *
   DEFINE CLUSTER(NAME(HLQ.DATABASEDB.PROD1) -
          VOLUMES(IMSVL1) -
          FILE(VOLDD) RECATALOG)
```

Figure 50. Using IDCAMS to Recatalog a VSAM Cluster

The snap target volume can then be logically dumped to tape from the second MVS system (Figure 51 on page 82).

```
//DSSDUMP  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//TAPEBKUP DD UNIT=3480X,DISP=(NEW,CATLG),
//         DSN=DFDSS.DUMP.IMSVL1
//SNAPVOL  DD DISP=SHR,UNIT=3390,VOL=SER=SNPVL1
//SYSIN    DD *
  DUMP DATASET(INCLUDE(**)) -
       INDDNAM(SNAPVOL) -
       OUTDD(TAPEBKUP)
```

Figure 51. DFSMSdss Logical Volume Dump

5.3.9 Registering the Image Copy with DBRC

Once the snap target database data sets or volumes have been copied to tape, you must register the image copy with the DBRC. Use the /NOTIFY.UIC command.

If for any reason the DFSMSdss copy job did not complete successfully, determine the cause of the failure and rerun the DFSMSdss copy job. The snap target database data sets or volumes should still exist, so the DFSMSdss copy job can be rerun without causing any further database outage.

5.3.9.1 Logical and Physical Completion

You need to consider the implications of *logical completion* of the image copy, which is when the SnapShot is complete, and database access can be resumed, and *physical completion*, which is when the snap copy has been written to tape (Figure 52 on page 83).

You must not register the image copy with DBRC until *physical completion*. The /NOTIFY.UIC command must include a parameter to indicate the time stamp of the image copy.

The recovery time stamp can be any point between when all databases updates were stopped and before any database was reopened for update. The time of *logical completion* of the snaps may be a good time to use.

Before IMS Version 5: Before IMS Version 5, if an OLDS log swap occurred with the final /DBR or /DBD command, the OLDS can be displayed and the time of the log swap used as the recovery time stamp.

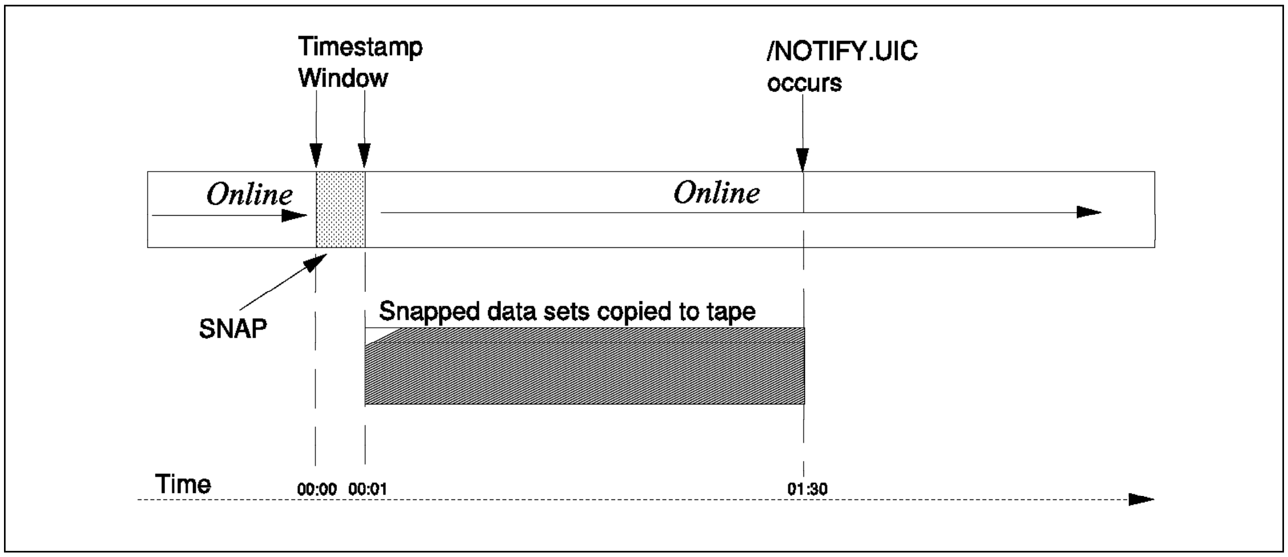


Figure 52. Flow of IMS Image Copy Using SnapShot

Figure 52 shows:

- 00:00 - Databases are deallocated, and the database data sets are snapped.
- 00:01 - The snap operation completes, and the databases are restarted. Online or batch processing is resumed. This is *logical completion* of the image copy. From this point, the snapped copies of the database data sets can be copied to tape.
- 01:30 - The copy to tape completes. This is *physical completion* of the image copy. After physical completion, DBRC must be informed that an image copy has been created.

  In this example, the recovery time stamp can be any time between 00:00 and 00:01.

Please note that these times do not bear any relation to any real tests or figures. They are provided only to demonstrate the difference between logical and physical completion.

Figure 53 shows the format of the /NOTIFY.UIC command.

```
/NOTIFY.UIC DBD(name) DDN(name)  -
            RUNTIME(yydddhhmmsst)  -
            UDATA(string)
```

Figure 53. Format of IMS /NOTIFY.UIC Command

DBD(name) indicates the name of the database, DB1.

DDN(name) indicates the DD name of the database data set.

RUNTIME(yydddhhmmsst) indicates the time stamp of the image copy.

UDATA is an optional field used to indicate the name of the DFSMSdss backup data set. GENJCL can use the variable field of this parameter to generate restore JCL.

IMS Version 6: For IMS Version 6, an extra parameter is implemented. *USID(number)* indicates the value of the update set identifier of the database when the image copy was created.***:*** For a full description of the /NOTIFY.UIC command and all of its parameters, refer to *IMS/ESA Utilities Reference:System.*

5.3.10 Restore Considerations

To restore an IMS database backed up by using DFSMSdss, all related database data sets must be recovered to the same point in time. Updates for each database first need to be disabled.

If the DFSMSdss data set name was recorded in the comment field when you issued /NOTIFY.UIC, use GENJCL to retrieve this data set name and generate the appropriate DFSMSdss restore commands and JCL.

All related databases must be restored from the same image copy position to avoid integrity problems.

Once all required database data sets have been restored, use IMS command /NOTIFY.RECOV to register the database recovery with DBRC. Figure 54 shows the format of the /NOTIFY.RECOV command.

```
/NOTIFY.RECOV   DBD(name) DDN(name) -
                          RUNTIME(yydddhhmmsst)  -
                          RCVTIME(yydddhhmmsst)
```

Figure 54. Format of IMS /NOTIFY.RECOV Command

The /NOTIFY.RECOV command contains the run time of the image copy (RUNTIME) and the time stamp of the recovery (RCVTIME).

5.3.11 Performance Considerations

If the snapped database data sets are to be copied to tape after the snap operation, and the tape copy jobs run concurrently with the application, you may notice a degradation in response times to IMS. The severity of this degradation depends on the number of tape copy jobs that are run concurrently and the amount of activity that IMS generates on the RVA.

Refer to Chapter 2, "SnapShot Technical Considerations" on page 9 for general performance considerations and guidelines.

5.3.12 Net Capacity Load Considerations

When a data set or volume is snapped, initially the copy uses no extra back-end storage. Over time, as tracks in either copy of the data are updated, an increase in the amount of NCL occurs. The actual amount of NCL used depends on the amount of data snapped and the number of updates occurring to either copy.

If the amount of NCL available is limited, delete the snap copies of the databases as soon as they have been successfully dumped to tape.

You may have sufficient NCL available to keep the copy of the database on DASD indefinitely, allowing fast recovery of the database from the most recent backup if application recovery is required.

5.3.13 Automation Considerations

You can use tools such as OPC/ESA and AOC/MVS to automate your IMS image copy procedures. Review and modify existing automation procedures to allow database access when the snap completes.

Here is an example of the steps AOC/MVS and OPC/ESA could perform to automate the image copy process:

1. On completion of critical batch, OPC/ESA event triggered tracking (ETT) triggers the execution of an AOC/MVS REXX EXEC.
2. AOC/MVS issues IMS commands to stop the appropriate databases and transactions.
3. When the required databases have been stopped, AOC/MVS displays the IMS OLDS and saves the time of the last log switch in a global variable.
4. An AOC/MVS REXX EXEC issues the SNAP DATASET commands for the database data sets.
5. AOC/MVS starts the appropriate IMS databases and transactions.
6. AOC/MVS triggers OPC/ESA to resume IMS batch processing.
7. OPC/ESA submits a job to copy the snapped database data sets to tape.
8. On completion of the copy job, OPC/ESA triggers AOC/MVS to notify DBRC of the image copy, retrieving the time stamp stored in a global variable.

5.4 Using SnapShot to Duplicate IMS Databases

You can use SnapShot to quickly duplicate some or all of your IMS databases. First you must define the new databases, using standard IMS procedures. These new databases can be used in the same IMS region as the original databases, or in a separate region in the same LPAR, or in a separate LPAR that shares access to the RVA. Then use SnapShot to create the new database data sets, which will be identical to the original database data sets.

The main advantage of using SnapShot is that the new database initially uses no extra back-end storage. Only as tracks of either copy are updated will NCL usage grow. This is advantageous for databases that are periodically refreshed from the production copy.

Another advantage of using SnapShot is that the copy operation is much faster than traditional data duplication methods.

To duplicate IMS databases, first identify every required database, including all related databases. To ensure the integrity of the new databases, a POC must be achieved on the original databases before they are snapped. The considerations are the same as for when using SnapShot to create an IMS image copy as described in 5.3.2, “Using SnapShot for IMS Image Copies” on page 77.

We recommend snapping databases at the data set level, using a separate SNAP DATASET for each individual database data set after a point of consistency has been achieved. Use the GENJCL utility to generate the SNAP DATASET batch job and commands.

The new database data sets must have a unique name. The new databases must be defined to IMS with the new database data set names.

When a VSAM cluster is snapped, AIs are not included with the snap copy and must be rebuilt.

5.4.1 Using Volume Snap

we recommend snapping databases at the data set level. However, you can snap the databases at the volume level. You must ensure that all appropriate volumes are snapped so that each database is snapped. If you have multivolume database data sets, you must ensure that each volume is snapped.

5.4.1.1 COPYVOLID(YES)

If the volume snaps are run with the COPYVOLID(YES) parameter, the source volume′s volser is copied to the target volume, and the target volume is automatically taken offline. It cannot be varied back online to the same system unless the original volume is first taken offline, because of the duplicate volsers.

When the system is IPLed, the duplicate volsers are detected, and the operator has to manually determine which volume should be taken offline. If the wrong volume is left online, both copies of the database could be corrupted. The target volume can be accessed only from another system that shares access to the RVA but does not share the catalog structure of the original system.

The new database data sets have to be manually recataloged on the second system. They have the same names as the original data sets. A sample IDCAMS job is shown in Figure 50 on page 81.

5.4.1.2 COPYVOLID(NO)

If the volume snaps are run with COPYVOLID(NO), the target volume′s volser is retained. The VVDS and VTOC are copied from the source volume, but they are updated to reflect the target volume′s volser. The target volume can be left online to the same system as the source volume because it has a different volser.

The new database data sets have to be manually cataloged before they can be accessed by IMS. If they are on the same MVS as the original data sets or on a system that shares the catalog with the original system, you have to rename the original database data sets before the new database data sets can be cataloged or the new versions can be renamed and cataloged. If you change the names of the data sets, you must update the database definition to reflect the new data set names.

Chapter 6. Using SnapShot for Disaster Recovery

Development and implementation of a disaster recovery plan can be a complex task and is beyond the scope of this book. For guidance in developing your plan, refer to *Disaster Recovery Design Concepts*, SG24-4211.

An effective disaster recovery procedure often involves taking regular backups of critical data and storing those backups at a physically remote location. Usually all applications have to be stopped for the duration of the backup, to ensure a consistent point of recovery. You can use SnapShot to reduce this application outage.

In this chapter we explain how you can modify your current disaster recovery procedures to exploit the features of SnapShot and discuss some implications to consider.

6.1.1 Catalogs

SnapShot cannot be used to back up ICF catalogs at the data set level. SnapShot can be used to snap volumes on which ICF catalogs reside.

If a volume is restored at your recovery site, you must consider the catalog implications. If the catalog structure has been restored, and each data set on the restored volume has an entry in the restored catalog, you should not need to recatalog the data sets if the restored volser matches that reflected for the data sets in the restored catalog.

If the volser of the restored volume is different from the volser reflected in the catalog, you have to recatalog each of the data sets on the volume (see Figure 55).

If you do not restore your ICF catalogs, you have to define catalogs on your recovery system and manually recatalog each of the data sets on the restored volumes.

```
DEFINE CLUSTER(NAME(PROD.VSAM.CLUSTER) INDEXED -
      VOLUME(VOL001) RECATALOG
```

Figure 55. IDCAMS Command to Recatalog a VSAM Cluster on Volume VOL001

If you snap a volume with an ICF catalog and on the restore explicitly restore the catalog by name, you must restore it to the same volser and device type from which it was dumped. Alternatively you can restore the entire volume.

6.1.2 SMS Recovery

On an SMS managed volume, all data sets must be cataloged. When you restore your volumes at the recovery site, you may have to recatalog the data sets manually.

If you are using VOLUME SNAP to copy your entire environment, we suggest that you address the considerations detailed in 6.2, “Using SnapShot for Data Set Backups” on page 90.

6.1.3 Data Currency

SnapShot can improve the currency of your backup data by enabling you take backups more often. Currently you can take a daily backup after the completion of overnight batch, before application access is required during business hours. Using SnapShot, you may be able to take backups more often, such as four times a day, and incur only a small or no extra outage to the application.

6.1.4 Remote Automated Tape Libraries

Users of remote automated tape libraries (ATLs) must regulate the number of deferred backups that run concurrently, dependent on other ATL activity at the time, and the amount of bandwidth available.

6.1.5 Data Safety

Actual tape backup of your data set may not occur until well after the SnapShot operation has complete. You must therefore determine when can you consider your data safely backed up:

- When the SnapShot operation has completed?
- When the snapped data set has been copied to tape?
- When the tapes have been physically moved to a secure location?

You may need to change your disaster recovery procedures when you use SnapShot to ensure that your data is safe according to your service level agreement.

6.1.6 NCL Considerations

Initially, snapping a volume or data set does not use any back-end storage. As tracks of either copy are updated, an increase in NCL will become evident. If applications or batch are expected to perform a lot of updates to the snapped data set or volume, deleting the snapped copies as soon as they have been successfully dumped to tape can help reduce the NCL level. Refer to 2.10, “Net Capacity Load Considerations” on page 35.

6.1.6.1 Deleting Snap Data Sets

Deleting a data set using conventional methods that call DADSM processing will ensure that any back-end storage allocated to the data set is released immediately by dynamic DDSR. You can use IDCAMS DELETE (Figure 56) or JCL DD cards with a disposition of DELETE.

```
//STEP01   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DEL 'SNAP.DATASET.ONE'
 DEL 'SNAP.DATASET.TWO'
```

Figure 56. Using IDCAMS to Delete Snap Copies of Data Sets

6.1.6.2 Deleting Snap Volumes

You can delete volumes through IXFP, using SIBBATCH, to ensure that any back-end storage allocated to the volume is released immediately (see Figure 57). In a multisystem environment, this may be operationally complex.

In Figure 57, the FORCE parameter is required as it allows the device to be deleted even though it still contains user data. The FDID parameter provides a means of verifying that the correct device is being deleted, by cross checking the FDID and the unit address.

```
//STEP01   EXEC PGM=SIBBATCH
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSIN    DD *
 DELETE DEVICE(SUBSYS(rvasys) -
              UNIT(uuuu) -
              FORCE -
              VERIFY(FDID(ff)))
```

Figure 57. Using SIBBATCH to Delete a Functional Device

The functional device must be redefined before it is required again (Figure 58).

```
//STEP01   EXEC PGM=SIBBATCH
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSIN    DD *
 DEFINE DEVICE(SUBSYS(rvasys) -
       FDID(ff) -
       DEVTYPE(ddddd)
```

Figure 58. Using SIBBATCH to Define a Functional Device

You can release the back-end storage allocated to the device by performing a minimal INIT of the functional device. However, this method does not call DADSM, so IXFP is not aware that the space has been released. The back-end storage allocated to the volume will be released the next time interval DDSR executes. The recommended approach is to prepare a job that performs the INIT and issues an interval DDSR command, to ensure that the NCL is freed immediately.

6.1.7 Remote Copy

| Peer-to-peer remote copy (PPRC) allows the synchronous copying of data from a primary RVA T82 to one or more secondary RVA T82s at the same or a different location up to 43km distant.

| All RVA models can be used as a secondary subsystem in an XRC configuration.

| The combination of SnapShot and remote copy can, however, greatly enhance your data availability and time to recover. One of the potential problems of using remote copy for disaster recovery is that of resynchronization. If the links between the primary and secondary site are broken, they must be resynchronized before the primary site is again protected against failure. Once resynchronization starts, the secondary site is not available for recovery until the

process completes and both sites are in synchronization, thereby exposing the primary site if there is then a disaster.

The links between the primary and secondary site may be lost because of a failure of the links, an incipient disaster, or in the event of a planned outage at the primary or secondary site.

SnapShot, in combination with XRC and PPRC, can greatly improve this availability, for both planned and unplanned events.

With the RVA (or multiple RVAs) at the secondary site, you can use SnapShot to create a consistent point-in-time copy of the entire system. In the event of an outage, planned or unplanned, before resynchronization begins, take a complete SnapShot of the entire system, using VOLUME SNAP.

The snapped volumes can then be varied offline, and resynchronization can begin. If there is subsequently a problem with the resynchronization, or a disaster ensues, the production workload can be restarted from the snapped volumes. The system would be at an earlier POC, but recovery could begin immediately. The combination of remote copy and SnapShot can provide a comprehensive availability solution for customers who have the highest requirements for availability.

In addition to enhancing disaster recovery, SnapShot can also be used in conjunction with remote copy to improve systems availability. Using SnapShot a consistent point-in-time copy can be taken of the entire subsystem, without any interruption to production data availability, by suspending remote copy momentarily while volume snaps are taken, then reestablishing the remote copy pairs, or XRC session. The snapped volumes can then be backed up to tape, or made available to another system′s image for processing.

For full details on using SnapShot with PPRC, see *RAMAC Virtual Array: Implementing Peer-to-Peer Remote Copy*, SG24-5338.

6.2 Using SnapShot for Data Set Backups

The process of identifying candidates for SnapShot and using SnapShot to back up data at the data set level is discussed extensively in 3.1, “Identifying SnapShot Candidates” on page 42.

6.3 Using SnapShot for Full Volume Dumps

In addition to taking incremental backups, many installations rely on taking periodic full volume dumps of some or all of their DASD volumes. In this section we explain how to use SnapShot to perform physical or logical volume dumps.

IXFP performs the SnapShot operation. A separate command is required for each data set or volume to be snapped.

For volume snaps, all of the target volumes must reside within the same RVA subsystem as the source volumes. Each target volume must be online and defined as the same device type and model as the source volume.

Generally, you would perform the volume snap and dump procedure in the following sequence:

1. Stop application access to the volumes.
2. Snap the required volumes.
3. Enable application access to the volumes.
4. Dump the snapped copies of the volumes to tape.

6.3.1 COPYVOLID(NO)

If you run the volume snap with COPYVOLID(NO) and the default CONDVOL(ALL) parameter, the operation does not modify the target volume′s volser. The VVDS and VTOC are copied from the source volume, but they are modified to reflect the volser of the target volume. The target volume is left online after the snap, and a physical dump to tape can be run from the same system.

The data sets on the target volume are uncataloged, and you can perform a DFSMSdss physical dump of the volume. To perform a DFSMSdss logical volume dump, the VSAM clusters on the volume must be recataloged.

If a volume is to be restored from a physical DFSMSdss dump, it has to be restored to the volser it had after the snap operation. In Figure 59, the volume would have to be restored to volser SPARE1. Alternatively, you can restore all data sets from the dump with an INCLUDE(**) statement.

If your recovery system is capable of using SnapShot, you can easily *snap back* this volume to its original volser, PRD001. This operation updates the volser, VVDS, and VTOC to PRD001.

If your recovery system is not capable of using SnapShot, there is no simple way of performing the snapback. You would have to recatalog each data set on the volume and update SMS storage groups with the new volser.

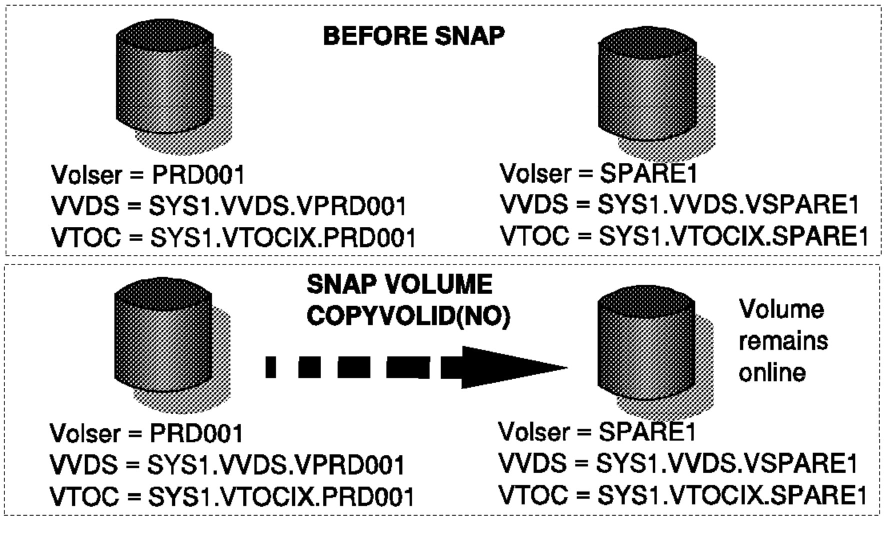


Figure 59. Effects of COPYVOLID(NO)

6.3.2 CONDVOL(LBL)

Use the CONDVOL(LBL) parameter with COPYVOLID(NO) to snap a volume and give the target volume a different volser, allowing it to remain online after the snap operation is complete. *The VVDS and VTOC are not updated to reflect the new volser.*Figure 60 shows the effect of using the CONDVOL(LBL) parameter. The target volume can then be physically dumped to tape without being varied online to another system.

To recover a volume snapped with CONDVOL(LBL) from a DFSMSdss physical dump, you must initialize it with the correct volser before you start the restore. Then run the DFSMSdss restore without the COPYVOLID parameter.

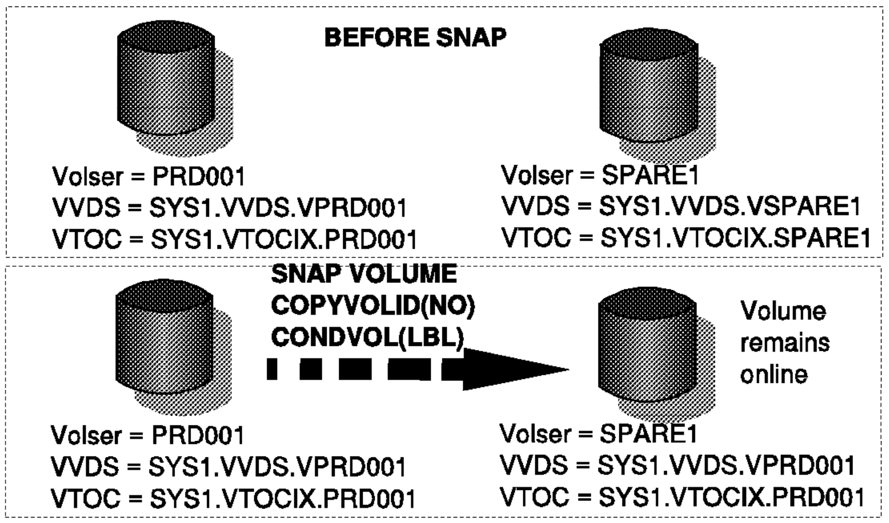


Figure 60. Effects of COPYVOLID (NO) and CONDVOL (LBL)

6.3.3 COPYVOLID(YES)

If you run SNAP VOLUME with the COPYVOLID(YES) parameter, as shown in Figure 61 on page 93, the target volume′s volser is changed to be the same as the source volume. The VVDS and VTOC are copied with the rest of the volume and reflect the correct volser. On completion of the snap, the target volume and source volume have duplicate volsers. The target volume is automatically taken offline by IXFP.

If you have a second MVS system that shares the RVA subsystem, you can vary the volume online to the second system, as long as there are no other volumes online to the second system with the same volser, and perform either a physical or logical DFSMSdss dump to tape from the second system.

Recovering the volume from a physical dump would be a straight restore, with no volser modifications required.

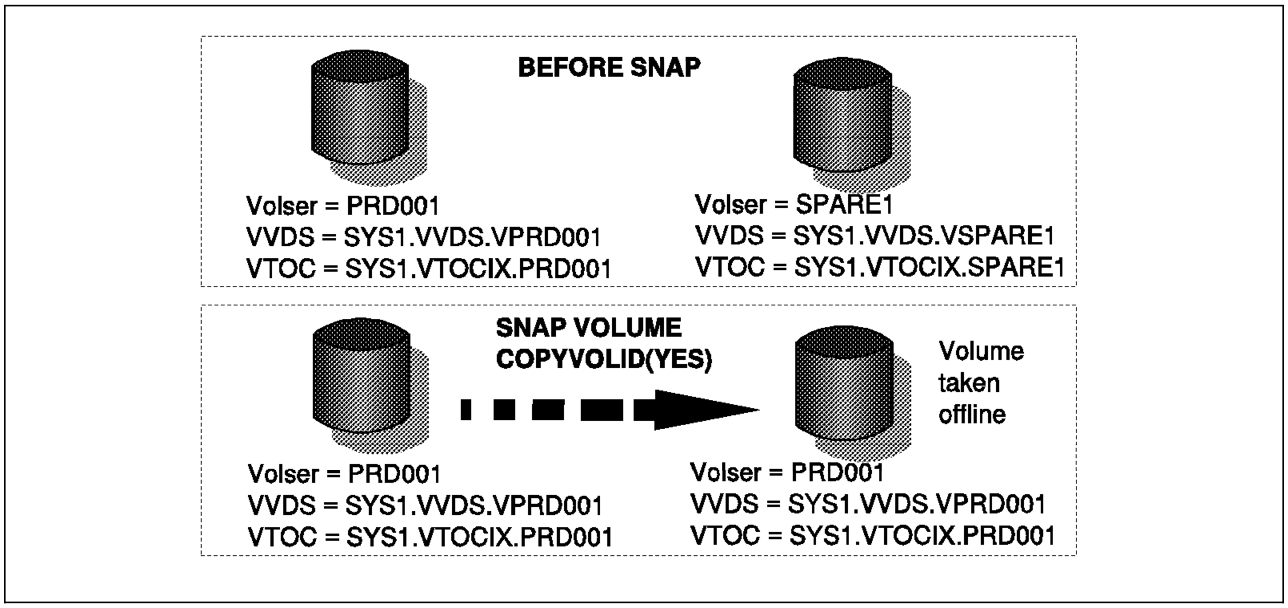


Figure 61. Effects of COPYVOLID(YES)

Use Table 7 to decide which COPYVOLID parameter to use and to plan your recovery procedure.

Table 7. COPYVOLID Options

	Backup Procedure	**Recovery Procedure**
SnapShot available for recovery		
Shared RVA during backups	Snap with COPYVOLID(YES). Dump to tape from second system.	Restore straight from tape.
RVA not shared during backups	Snap with COPYVOLID(NO). Dump to tape.	Restore from tape. Snap back to original volser.
SnapShot not available for recovery		
Shared RVA during backups	Snap with COPYVOLID(YES). Dump to tape from second system.	Restore straight from tape.
RVA not shared during backups	Snap with CONDVOL(LBL) Dump to tape.	Initialize target volser. Restore from tape without DFSMSdss COPYVOLID.

6.3.4 Managing Target Volumes

Many installations that plan to use volume snap will find that they are constrained by the number of spare functional devices available for the targets of volume snaps.

A procedure for managing the target volumes must be developed, to ensure that target volumes are not overwritten or deleted before they have been dumped to tape. The procedure you develop will depend on the number of spare functional devices you have and the number of volumes to be snapped. This may be a simple matter of designing your batch schedule for the volume snap and dump jobs such that the number of jobs being run concurrently cannot exceed the number of target volumes available.

If you can perform your volume snap and dump to tape in one multistep job, you can set up a JES2 or JES3 batch job class that will be used by only these snap and dump jobs. The number of JES2 or JES3 initiators that execute jobs in this job class can then be modified to match the number of spare functional devices. You may need to ensure that operators or other personnel are aware that jobs in this job class should not be modified to run in another job class.

A permanent catalog entry can be allocated for each target volume. The snap and dump jobs can exclusively enqueue this catalog entry, ensuring that only one job can run concurrently for each target volume.

An OPC/ESA special resource can be set up for each available target volume. The volume snap jobs can be set up in OPC/ESA with a dependency that one of these special resources is available. After a volume has been snapped and dumped to tape, the OPC/ESA special resource can be reset from the dump batch job.

If you are running multiple parallel dumps to tape it is essential that you ensure there are enough paths to the tape drives.

6.3.5 Logical and Physical Volume Dumps

Most of the discussion in this section deals with using SnapShot to snap volumes and performing *physical* volume dumps. In the past, physical volume dumps were often preferred over logical volume dumps because logical dumps took much longer to complete. With recent enhancements to DFSMSdss and I/O subsystems, the time difference between logical and physical dumps is not as great.

To perform a DFSMSdss logical volume dump, all VSAM data sets on the source volume must be cataloged. In most cases, the data sets on snapped volumes are not cataloged, so only a DFSMSdss physical dump can be performed.

Snapped VSAM volumes can be logically dumped if there are two MVS systems sharing access to the RVA subsystem. The volume snap is run with COPYVOLID(YES), and the target of the snap operation is varied online to the second MVS system after the snap has completed.

The appropriate user catalogs for each data set also have to be copied from the first system to the second system. Shared user catalogs cannot be used, as volume updates on the first system will be reflected in the shared catalog, but the updates have not occurred on volumes that are online to the second system.

6.3.6 Performance Considerations

Running dump jobs concurrently with the application affects performance of the entire RVA subsystem. Here are some guidelines on the number of dump jobs to run concurrently to minimize the impact on the application:

- On an RVA2 subsystem, run two dumps jobs concurrently with the application.
- On an RVA Turbo 4-path subsystem, you can run three or four dump jobs concurrently with the application, depending on the load on the RVA.
- On an RVA Turbo 8-path subsystem, you can run six dump jobs concurrently with the application.

  Depending on the type of RVA and tape hardware, the importance of application response times, and the urgency of getting the dump tapes offsite, adjust the number of concurrent DFSMSdss jobs to suit your requirements accordingly.

6.3.7 Sample Backup Procedure with One MVS System

This sample backup procedure is used by an installation that performs weekly full volume dumps of 10 of its DASD volumes.

The volumes to be snapped and dumped are 3390-03s, labeled VOL001 through VOL010, and each volume contains VSAM data sets.

The installation has only one MVS system available. In the event of a disaster, it will restore to a traditional DASD subsystem, incapable of performing SnapShot functions. The DASD volumes are snapped with the COPYVOLID(NO) with the CONDVOL(LBL) parameter.

The RVA subsystem must have sufficient spare functional devices, of the same device type and model, support each of the volume snaps. These will be the targets of the volume snaps. They must be online to the system at the time of the volume snap. In this case, 10 spare 3390-03 functional devices are available, which have been labeled SNP001 through SNP010.

Here is the procedure:

1. Disable application access to the volumes.
2. Snap the ten 3390-03s to target volumes.

   You can use one SIBBATCH job performing 10 SNAP VOLUME commands (see Figure 62 on page 96), or 10 separate SIBBATCH jobs, each performing a single SNAP VOLUME command.

```
//SNAPVOLS EXEC PGM=SIBBATCH
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSIN    DD *
 SNAP VOLUME(SOURCE(VOLUME(VOL001)) -
            TARGET(VOLUME(SNP001)) -
            TOLERATENQFAIL(NO) -
            REPLACE(YES) -
            COPYVOLID(NO) -
            CONDVOL(LBL))
 SNAP VOLUME(SOURCE(VOLUME(VOL002)) -
            TARGET(VOLUME(SNP002)) -
            TOLERATENQFAIL(NO) -
            REPLACE(YES) -
            COPYVOLID(NO) -
            CONDVOL(LBL))
   ...
   ...
   ...
 SNAP VOLUME(SOURCE(VOLUME(VOL010)) -
            TARGET(VOLUME(SNP010)) -
            TOLERATENQFAIL(NO) -
            REPLACE(YES) -
            COPYVOLID(NO) -
            CONDVOL(LBL))
```

Figure 62. SIBBATCH Volume Snaps

3. Restart the applications.
4. Dump the target volumes to tape, using DFSMSdss. Figure 63 shows the DFSMSdss jobstep to dump the first snap volume, SNP001.

```
//DSSDUMP  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//TAPEBKUP DD UNIT=3480X,DISP=(NEW,CATLG),
//         DSN=DFDSS.DUMP.VOL001(+1)
//SYSIN    DD *
  DUMP FULL INDYNAM(SNP001) -
       OUTDD(TAPEBKUP) ALLDATA(*) ALLEXCP
```

Figure 63. DFSMSdss Used to Dump One Snap Volume to Tape

5. To reduce NCL usage, vary the target volumes offline and delete them as soon as they have been successfully dumped to tape (see Figure 64 on page 97).

```
//DELSNAP  EXEC PGM=SIBBATCH
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSIN    DD *
 DELETE DEVICE(SUBSYS(RVA0001) -
              UNIT(1010) -
              FORCE -
              VERIFY(FDID(10)))
```

Figure 64. Deleting the Target Volume at Unit Address 1010

6.3.8 Sample Backup Procedure with Two MVS Systems

This procedure is similar to that described in 6.3.7, “Sample Backup Procedure with One MVS System” on page 95; however, the installation has a second MVS system that has access to the RVA subsystem.

The volume snaps are run with the COPYVOLID(YES) parameter and physically dumped to tape from the second MVS image, with their original volsers.

The second MVS system must have no volumes online with volsers identical to the snap target volumes.

After the target volumes have been dumped to tape, they are relabeled to a unique volser so that they can be put online to the first MVS system next time the snaps are run.

Here is the procedure:

1. Disable application access to the volumes.
2. Snap the ten 3390-03s to target volumes (Figure 65).

```
//SNAPVOLS EXEC PGM=SIBBATCH
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSIN    DD *
 SNAP VOLUME(SOURCE(VOLUME(VOL001)) -
            TARGET(VOLUME(UNQ001)) -
            TOLERATENQFAIL(NO) -
            REPLACE(YES) -
            COPYVOLID(YES))
 SNAP VOLUME(SOURCE(VOLUME(VOL002)) -
            TARGET(VOLUME(UNQ002)) -
            TOLERATENQFAIL(NO) -
            REPLACE(YES) -
            COPYVOLID(YES))
   ...
   ...
   ...
 SNAP VOLUME(SOURCE(VOLUME(VOL010)) -
            TARGET(VOLUME(UNQ001)) -
            TOLERATENQFAIL(NO) -
            REPLACE(YES) -
            COPYVOLID(YES))
```

Figure 65. SIBBATCH Volume Snaps:

When these snaps complete, the target volumes will have duplicate volsers and will be automatically varied offline.

3. Restart the applications.
4. Vary online the 10 target volumes to the second MVS system.
5. Dump the 10 snapped volumes to tape, using DFSMSdss on the second MVS system. Figure 66 shows the DFSMSdss jobstep to dump the first snap volume, VOL001.

```
//DSSDUMP  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//TAPEBKUP DD UNIT=3480X,DISP=(NEW,CATLG),
//          DSN=DFDSS.DUMP.VOL001(+1)
//SYSIN    DD *
  DUMP FULL INDYNAM(VOL001) -
       OUTDD(TAPEBKUP)
```

Figure 66. DFSMSdss Used to Dump One Snap Volume to Tape

6. Once the snapped volumes are successfully dumped to tape, run a minimal INIT against them to ensure that they have a unique volser and that any back-end storage allocated to the volumes is released the next time interval DDSR is run (Figure 67).

```
//INITSNAP EXEC PGM=ICKDSF
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 INIT UNITADDRESS(1010) VOLID(UNQ001) DEVTYPE(3390) -
      VTOC(0,3,87) INDEX(10,1,44) VERIFY(VOL001) NOVALIDATE PURGE
```

Figure 67. Minimal INIT to Change Volser and Release Back-end Storage

7. Vary online the target volumes to the first MVS system, so that they are ready for the next time the volume snap and dump process is scheduled to run.

6.4 Using SnapShot with Physical Full Volume Restores

As with all disaster recovery procedures, the first step is to get a system running at your recovery site to start the recovery process.

Many installations set up a small *one-pack* system capable of running any software required during the restore process. Depending on your recovery plan, the one-pack system should include DFSMSdss and IXFP.

A DFSMSdss dump of the one-pack system must be available at the recovery site. In the event of a disaster, use stand-alone DFSMSdss to restore the one-pack system. You can then IPL the one-pack system and run the subsequent volume restores from it.

You can also run any volser clipping or volume snapbacks from the one-pack system.

6.4.1 Sample Recovery Procedure with an RVA

In this sample recovery procedure, the installation has one MVS system available at its usual production site and an RVA subsystem available at the recovery site.

The regular backup procedure is to snap ten 3390-3s (VOLxxx) with the COPYVOLID(NO) parameter. After the snap, the target volumes retain their unique volsers (UNQxxx) and so remain online. DFSMSdss is used to physically dump the target volumes to tape.

A one-pack system has been set up with DFSMSdss and IXFP.

Here are the steps to recover the system:

1. Use stand-alone DFSMSdss to restore the one-pack system and then IPL it.
2. Define twenty 3390-03 functional devices to IXFP (Figure 68).

```
//DEFVOLS  EXEC PGM=SIBBATCH
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSIN    DD *
 DEFINE DEVICE(SUBSYSTEM(BRSRVA) FDID(D'1':D'20') DEVTYP(33903))
```

Figure 68. Define Twenty 3390-03s

3. Use ICKDSF to initialize 10 volumes with volsers VOLxxx and 10 volumes with volsers UNQxxx (Figure 69).

```
//INITVOL  EXEC PGM=ICKDSF
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 INIT UNITADDRESS(1010) VOLID(VOL001) DEVTYPE(3390) -
      VTOC(0,3,87) INDEX(10,1,44) VERIFY(VOL001) NOVALIDATE PURGE
```

Figure 69. Minimal Initialization of One Volume

4. Restore the 10 snapped 3390-03s (UNQxxx) from tape (Figure 70).

```
//DSSREST  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//TAPEBKUP DD UNIT=3480X,DISP=(NEW,CATLG),
//         DSN=DFDSS.DUMP.UNQ001(0)
//TARGET   DD DISP=SHR,VOL=SER=UNQ001,UNIT=3390
//SYSIN    DD *
  RESTORE FULL OUTDD(TARGET) -
       INDD(TAPEBKUP) COPYVOLID
```

Figure 70. DFSMSdss Used to Restore One Snap Volume from Tape: COPYVOLID

5. Snap back the 10 volumes to their original volsers (Figure 71 on page 100).

```
//SNAPVOLS EXEC PGM=SIBBATCH
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSIN    DD *
 SNAP VOLUME(SOURCE(VOLUME(UNQ001)) -
            TARGET(VOLUME(VOL001)) -
            TOLERATENQFAIL(NO) -
            REPLACE(YES) -
            COPYVOLID(NO))
 SNAP VOLUME(SOURCE(VOLUME(UNQ002)) -
            TARGET(VOLUME(VOL002)) -
            TOLERATENQFAIL(NO) -
            REPLACE(YES) -
            COPYVOLID(NO))
   ...
   ...
   ...
 SNAP VOLUME(SOURCE(VOLUME(UNQ010)) -
            TARGET(VOLUME(VOL001)) -
            TOLERATENQFAIL(NO) -
            REPLACE(YES) -
            COPYVOLID(NO))
```

Figure 71. SIBBATCH Volume Snaps

6.4.2 Sample Recovery Procedure without an RVA

This sample recovery procedure is used when an RVA subsystem is not available at the recovery site. The system will be restored to a traditional subsystem, such as an IBM 3990.

The regular backup procedure is to snap ten 3390-3s (VOLxxx) with COPYVOLID(NO) with the VOLCOND(LBL) parameter. After the snap, DFSMSdss is used to physically dump the target volumes to tape.

A one-pack system has been set up with DFSMSdss.

Here are the steps to recover the system:

1. Use stand-alone DFSMSdss to restore the one-pack system and then IPL it.
2. Use ICKDSF to initialize 10 volumes (Figure 72).

```
//INITVOL  EXEC PGM=ICKDSF,PARM='NOREPLYU'
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 INIT UNITADDRESS(1010) VOLID(VOL001) DEVTYPE(3390) -
      VTOC(0,3,87) INDEX(10,1,44) NOVERIFY PURGE
```

Figure 72. Minimal Initialization of One Volume

3. Restore the 10 snapped 3390-03s (SNPxxx) from tape, without the DFSMSdss COPYVOLID parameter (Figure 73 on page 101).

```
//DSSREST  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//TAPEBKUP DD UNIT=3480X,DISP=(NEW,CATLG),
//         DSN=DFDSS.DUMP.VOL001(0)
//TARGET   DD DISP=SHR,VOL=SER=VOL001,UNIT=3390
//SYSIN    DD *
  RESTORE FULL OUTDD(TARGET) -
       INDD(TAPEBKUP)
```

Figure 73. DFSMSdss Restore of One Volume from Tape: Without COPYVOLID

4. At this point the volume label (VOLSER) does not match the VTOC or VVDS, so you must CLIP the label back to its original value (VOLxxx).

6.5 Using SnapShot with DFSMShsm ABARS

You can use SnapShot in conjunction with DFSMShsm Aggregate Backup And Recovery Support (ABARS) to produce consistent backups of groups of snapped data sets.

If you have installed the software support for DFSMSdss SnapShot and virtual concurrent copy, DFSMShsm can take advantage of concurrent copy when performing aggregate backup.

For details on how to exploit virtual concurrent copy from DFSMShsm, see *Implementing DFSMSdss SnapShot and Virtual Concurrent Copy*, SG24-5268.

This chapter discusses how to implement ABARS using SnapShot commands.

An aggregate group for the snap target data sets must be defined. Use a naming standard to easily identify the snap targets and write the aggregate group filter list to capture all target data sets.

Take the data set snaps at a time of low application activity, for a more consistent backup. After all data set snaps have completed, create the aggregate backup.

You can delay creating the aggregate backup until a later time if, for example, tape drive availability is constrained at the time the snap copies are taken.

6.5.1 Example

In this example, the installation has five large data sets that need to be backed up with ABARS. The backup must run during the critical batch path. The backup must be consistent, so some batch update jobs cannot begin until after the backup has been created. The backup is often delayed due to a shortage of available tape drives, consequently delaying the batch run.

Snap is used to snap the production data sets, and critical batch processing can be resumed. The aggregate backup of the snap copies is run at a later time when more tape drives are available.

An aggregate group is defined through the Interactive Storage Management Facility (ISMF). The aggregate group is called SNAPAG and contains the following filter list:

```
INCLUDE(HLQ.PROD.**.SNAP)
```

An instruction data set is also defined to accompany the aggregate backup. This data set contains text which explains that, in the event of recovery, each data set must be renamed or snapped back to its original name.

A new job is implemented and scheduled to run at the point in the batch stream where the backup is required. This job snaps the production data sets as shown in Figure 74.

```
//SNAPAGG  EXEC PGM=SIBBATCH
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSIN    DD *
 SNAP DATASET(SOURCE(DATASET(HLQ.PROD.DATASET1)) -
             TARGET(DATASET(HLQ.PROD.DATASET1.SNAP))
 SNAP DATASET(SOURCE(DATASET(HLQ.PROD.DATASET2)) -
             TARGET(DATASET(HLQ.PROD.DATASET2.SNAP))
 SNAP DATASET(SOURCE(DATASET(HLQ.PROD.DATASET3)) -
             TARGET(DATASET(HLQ.PROD.DATASET3.SNAP))
 SNAP DATASET(SOURCE(DATASET(HLQ.PROD.DATASET4)) -
             TARGET(DATASET(HLQ.PROD.DATASET4.SNAP))
 SNAP DATASET(SOURCE(DATASET(HLQ.PROD.DATASET5)) -
             TARGET(DATASET(HLQ.PROD.DATASET5.SNAP))
```

Figure 74. Snap Data Sets in Preparation for Aggregate Backup

On completion of this job, the critical batch stream is resumed.

Another job is scheduled to run later when more tape drives are available (Figure 75). This job initiates the aggregate backup. Once the aggregate backup has been successfully created, the snap target data sets are deleted.

```
//AGBKUP   EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 HSEND WAIT ABACKUP SNAPAG EXECUTE
//*
//DELSNAP  EXEC PGM=IKJEFT01,COND=(0,NE)
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DEL 'DATASET.HLQ.PROD.DATASET1.SNAP'
 DEL 'DATASET.HLQ.PROD.DATASET2.SNAP'
 DEL 'DATASET.HLQ.PROD.DATASET3.SNAP'
 DEL 'DATASET.HLQ.PROD.DATASET4.SNAP'
 DEL 'DATASET.HLQ.PROD.DATASET5.SNAP'
```

Figure 75. Initiating Aggregate Backup and Deleting Snap Targets

6.5.2 Converting Aggregate Backups to Use SnapShot

You may want to modify your currently existing aggregate backup procedures so that you can use SnapShot to create a more consistent backup.

SnapShot does not support wildcards or filter lists, therefore aggregate group selection lists must be expanded so that a separate SNAP command can be issued for each data set. If you have installed the support for DFSMSdss SnapShot, you can use DFSMSdss COPY commands to backup your aggregate. DFSMSdss COPY will automatically use SnapShot where possible, with the benefit that DFSMSdss COPY support filter lists and wildcards.

You may not be able to snap many of the data sets in the aggregate group. You cannot snap tape data sets, data sets that have been migrated by DFSMShsm, or DASD data sets that do not reside on RVA DASD. Data sets whose names change frequently, such as generation data sets, or data sets that contain a time stamp in their name, add further complications as the SNAP commands must be changed each time a data set name changes.

A REXX EXEC can be developed to generate SNAP commands and a new aggregate group filter list dynamically each time the backup is to be created.

You can leave your current aggregate group filter list as it is and set up a second aggregate group that the REXX EXEC will modify. The REXX EXEC can execute a backup of the first aggregate group, using the VERIFY parameter. This will not actually create an aggregate backup but will produce a list of each data set that would be included in the aggregate group.

The REXX EXEC can process the output of the ABACKUP VERIFY command and optionally attempt to snap each data set. For successful snaps, the REXX EXEC includes the snap target data set in the second aggregate group filter list. For unsuccessful snaps, the REXX EXEC includes the original data set name in the second aggregate group filter list. The REXX EXEC can also build a list of snapback commands that can be included with the second aggregate group as an instruction file. The REXX EXEC can then execute a full backup of the second aggregate group.

This method can support regularly changing data set names, tape data sets, and migrated data sets and still take advantage of SnapShot to provide a more consistent aggregate group.

For a sample REXX EXEC, see Appendix D, “Sample ABARS Procedure Using SnapShot and REXX” on page 183.

Chapter 7. Using SnapShot for Year 2000 Testing

January 1, 2000 is less than a year away, you need to plan for and address the 1999 to 2000 date change well in advance of that date. Even before January 1, 2000, your computing environment might not work as expected because your systems and/or application programs might not properly process dates beyond December 31, 1999.

The potential exposure comes primarily from, but is not limited to, the use of a two-digit-year (YY) format, instead of a four-digit-year (YYYY) format, for year representation within programs, files, databases, and processes. For example, the year 1997 is represented as 97, the year 1999 as 99, and so on. Thus, January 1, 2000 would be represented as 01/01/00 (if you use the MMDDYY format in your data) and might be interpreted as January 1, 1900, rather than January 1, 2000. Thus programs that perform arithmetic operations, comparisons, and sorting of data fields are likely to yield incorrect results when manipulating year-date data of 2000 and beyond. The potential exposures that might be encountered, and their variations, listed in the *The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation*, GC28-1251 are referred to as the *Year 2000 challenge*.

The scope of the Year 2000 challenge spans the entire information technology industry. A data mismatch can exist in any level of hardware or software, from microcode to applications, in files and databases, and can occur on any information system platform. Several scenarios in the literature describe the implications for your business of applications that are not Year 2000 compliant.

Although the Year 2000 challenge is complicated and far-reaching, it is not technically difficult to resolve when viewed on a module-by-module or routine-by-routine basis. The degree of complexity is directly related to the interrelationships among routines and programs and the data passed among them. It is not a trivial programming exercise.

Your realization of a Year 2000 ready system demands immediate management commitment, dedicated resources (personnel and hardware), strategic planning, rapid development, acceptance testing, a well-planned migration, and an adequate budget.

IBM defines a product to be Year 2000 ready as follows: When used in accordance with its associated documentation, it is capable of correctly processing, providing, and receiving date data within and between the twentieth and twenty-first centuries, provided all other products (for example, software, hardware, and firmware) used with the product properly exchange date data with it.

On October 31, 1995, IBM announced Year 2000 support. It stated that, for any S/390 platform running MVS or OS/390 to be considered Year 2000 ready, all data sets must use ICF catalogs. It will not be possible to access data through a VSAM catalog or OS/VS control volume (CVOL) catalog when the system date changes past December 31, 1999. Thus MVS customers who still have data sets cataloged in CVOLs or in the old VSAM catalogs must migrate them to ICF catalogs before the end of 1999.

In this chapter we discuss the use of SnapShot for preparing an OS/390 Year 2000 test system. Please refer to the *The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation*, GC28-1251 and the IBM Year 2000 Web site, *www.ibm.com/year2000*, for more information about the Year 2000 challenge.

All RVA subsystems are Year 2000 compliant with the latest level of LIC available in March 1997. IXFP and SnapShot are Year 2000 compliant.

7.1 Year 2000 Test Environment

When performing a Year 2000 test, it is critical that you provide a separate, isolated test system. Such a system requires extra hardware. You will need extra storage for the OS/390 test operating system and your test data and extra CPU capacity to run the test system. Consultants estimate the extra storage to be an additional 20% to 30% of your installed storage.

For maximum security, IBM recommends that you use a dedicated RVA for the Year 2000 testing. This is to make absolutely sure that there is no path or link available to mix up the data generated for testing with your production data. Volumes from the test system and production system can coexist on the same RVA, but then you must handle them with the utmost caution to prevent any pollution of the production data.

If you share an RVA between a Y2K system and a production system there is no requirement to change the time and date on the RVA, as this is used only by IXFP for reporting purposes.

The Year 2000 test system can run on a separate S/390 processor or in an LPAR in an S/390 processor. In PR/SM, each LPAR contains its own logical time-of-day (TOD) clock that operates independently of other logical TODs in other LPARs, even if the S/390 processor gets the physical TOD clock value from an IBM 9037 Sysplex Timer. Therefore in the same central electronic complex (CEC), with power-on-reset (POR) in LPAR mode, you can run LPARs set with different times and dates.

Time Machine

A system running with a future date set is called a *time machine*. It is of utmost importance that this system not access any files that are used in the daily production system or other systems with today′s date set. If it can, there is a chance that VTOCs, labels, catalogs, and other types of control data sets may be corrupted with future dates set by the operating system in the time machine. When the daily production system with today′s date accesses such files, unpredictable results may occur.

Therefore you must establish a separate operating system environment and separate test data. Data files must only be brought in to the time machine; nothing can be taken out, with the exception of files on unlabeled tapes.

The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation, GC28-1251 lists products that support dates between the twentieth and twenty-first centuries. OS/390 Release 2 is Year 2000 ready. It supports four-digit dates beyond January 1, 2000.

7.2 Year 2000 and SnapShot

The RVA subsystem is the best DASD device to use for your Year 2000 testing. When the source is snapped, the snapped volume does not consume physical back-end storage, because the target shares the same tracks as the source. Extra space is not needed to accommodate the new target volume at this point. As updates are made, new tracks are written to the back-end disk arrays. Because most of the data sets that make up OS/390 are program libraries and are mostly read-biased with very few updates, not much extra physical disk space is needed to make a clone copy of the operating system.

To make a clone copy of your production system to use in a time machine, follow these steps:

1. Ensure that you have functional volumes available.

   These new target volumes must be in the same RVA as your source OS/390 system volumes. It is not necessary to make a copy of your DLIB volume.

   You must also ensure that you have other necessary hardware available to your time machine, such as operating system consoles and tapes.

2. Perform a SNAP VOLUME of your production system.

   Use COPYVOLID(YES) to get an exact duplicate of your system. After the snap, the volumes go offline from the production system.

3. If you use a separate RVA for the tests you can use a datamover or PPRC to copy the volumes, created by SnapShot for the test system, to your Year 2000 RVA. Continue with the next steps when the complete test database is copied to the Year 2000 RVA.

4. Define an LPAR or have a separate S/390 processor available.

   As the LPAR or processor will become the time machine, it must not be able to access the files of the production system. Isolated unshared DASD on the time machine is critical. Simply by IPLing a time machine, you could experience problems with VTOCs on shared DASD.

   In an LPAR environment you can restrict LPAR access for devices, through the definitions entered for your I/O configuration in the hardware configuration definition (HCD) dialogs. Here you can restrict LPAR access to an I/O device on a shared channel path by using the explicit device candidate list to select which LPARs can access that I/O device. On the HCD Define Device / Processor panel, enter YES or NO in the Explicit Device Candidate list field to specify whether you want to restrict LPAR access to an I/O device. YES specifies that only your selected LPARs can access this I/O device. Note that the LPAR must also be in the channel path access or candidate list to access the device. On the Define Device Candidate List panel, place a forward slash (/) to the left of each selected LPAR name. You can also specify through HCD that the devices in the time machine should be offline at IPL to a particular operating system configuration.

   When you are done with your Year 2000 testing, you must initialize all functional volumes or other disks using ICKDSF that you have used for your Year 2000 testing before placing them back into your production system. Alternatively, if you have a template volume, you can snap that volume over the date-contaminated functional volume. Use the SNAP VOLUME command with the REPLACE(YES) option. You must perform this action to destroy any data sets, including the VTOC, catalogs, control data sets, PDS directories, and your data, that may have gotten Year 2000 dates into them in some way. The writing of a new VTOC on the disk is the easiest way to ensure that any previous data on the disk is erased.

5. Verify your configuration and IPL the time machine.

   Before you can IPL, you may have to update the SYS1.PARMLIB CLOCKxx member. From your source system, run ICKDSF REFORMAT to change the label on the new target SYSRES disk. Bring it online and do the PARMLIB updates. Then relabel it back to its original name. At this point you should also ensure that the data you need for your testing is brought in to the time machine.

   In *The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation*, GC28-1251 there are lists of things to remember. Probably one of the most important things to remember is to check that your TSO userid will not be expired by your security system as it has not been used for some years. You will also probably want to disable DFSMShsm, and any tape management system that has date-related expiration of data sets; otherwise you may see many data sets go. Make sure your time machine configuration now is in an isolated nonshared DASD environment. Perform the IPL from your new functional volumes.

6. Set the clock.

   Set the TOD clock to some future date. *The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation*, GC28-1251 discusses some dates to test, and for how long. Basically, set the clock to some time at the end of December 1999 and let it run a few days past the turn of the century. Then you also want to test the end of February in the Year 2000, to check your leap year routines. When changing the clock, be aware of the different clocks used in OS/390, particularly Greenwich mean time (GMT; or coordinated universal time (UTC)) and the local time. Resetting GMT causes the TOD clock to be reset. It also affects the local time. Respecifying GMT causes the local time value to be recalculated with the new GMT and the system time zone constant. Resetting local time does not affect GMT or the TOD clock. However, it causes the system time zone constant, which is initialized at IPL from the CLOCKxx member of SYS1.PARMLIB or from the Sysplex Timer if attached, to be recalculated.

   The way the TOD clock is set depends on how you are running your time machine:

   - Dedicated S/390 processor PORed in basic mode

     The system automatically issues message IEA888A at IPL time if OPERATOR PROMPT is included in the active CLOCKxx member of SYS1.PARMLIB. You can set the TOD clock by answering the IEA888A message during IPL. The answer must contain, *GMT*, as this is the parameter that sets the TOD clock. Otherwise only the local time offset will be set. Figure 76 on page 109 shows the IEA888A message and the reply.

   - Dedicated S/390 processor PORed in LPAR mode

     In LPAR mode the TOD clock is set only during the POR process. It is set from the battery-operated clock in the processor control element on ES/9000 machines, or from the service element clock in Parallel Enterprise Servers (9672).

- In an LPAR

  Each LPAR operates its own logical TOD. To set a new value in the logical TOD in OS/390 running in an LPAR, that LPAR cannot be a member of a sysplex. The ETRMODE parameter in the active CLOCKxx member of SYS1.PARMLIB specifies whether MVS is using a Sysplex Timer (YES) or not (NO). If your processor is connected to an active Sysplex Timer, make sure that ETRMODE NO is included in the active CLOCKxx member of SYS1.PARMLIB for the OS/390 image you are IPLing in the time machine. ETRMODE NO specifies that MVS is not to use the Sysplex Timer.

- In the RVA (only if a seperate Year 2000 RVA is used)

  In this cases where a separate RVA is available you can set the TOD of the RVA to the same Year 2000 date as in your operating system. This can be done on the RVA operator panel or with the IXFP in the Subsystem Configuration. This allows you to test the IXFP reporting functions.

For the sake of convenience during your testing, we suggest that you also offset the clock some hours, so that the midnight moment will be at a time more convenient for you, rather than in the middle of the night.

```
OO IEA888A GMT DATE=yyyy.ddd,CLOCK=hh.mm.ss,LOCAL DATE=yyyy.ddd,
CLOCK=hh.mm.ss
REPLY U, OR GMT/LOCAL TIME

R OO,'DATE=yyyy.ddd,CLOCK=hh.mm.ss,GMT'
```

Figure 76. Message IEA888A

See the *OS/390 MVS Initialization and Tuning Reference*, SC28-1752 for details on SYS1.PARMLIB and the CLOCKxx member.

7.3 Using SnapShot to Duplicate Database Environments

You can use SnapShot to duplicate some or all of a set of databases. This may be useful for Year 2000 testing where the test databases are to be periodically refreshed from a consistent or static copy of the production databases.

You may want to snap your production databases once to create a second static, copy that is only rarely refreshed. With a dedicated Year 2000 RVA available this static copy should be moved to the test system for further steps. An additional copy of the database can be created by snapping the static databases. This third copy of the databases can be used for application testing, and it can be quickly refreshed from the static copy as required, without causing an outage to the production database (see Figure 77 on page 110 and Figure 78 on page 110)

Using SnapShot for this refresh process can significantly reduce delays to the Year 2000 testing schedule, as compared to using standard data duplication methods to refresh the data.

To ensure the integrity of the test database, a POC must be achieved on the production databases from which the static copy of the databases will be created. Follow these steps:

1. Identify all required databases.
2. Stop all batch or online updates to the production databases.
3. Snap the production databases to create the static copy. Figure 77 shows an example of sharing an RVA between production and test LPARS.

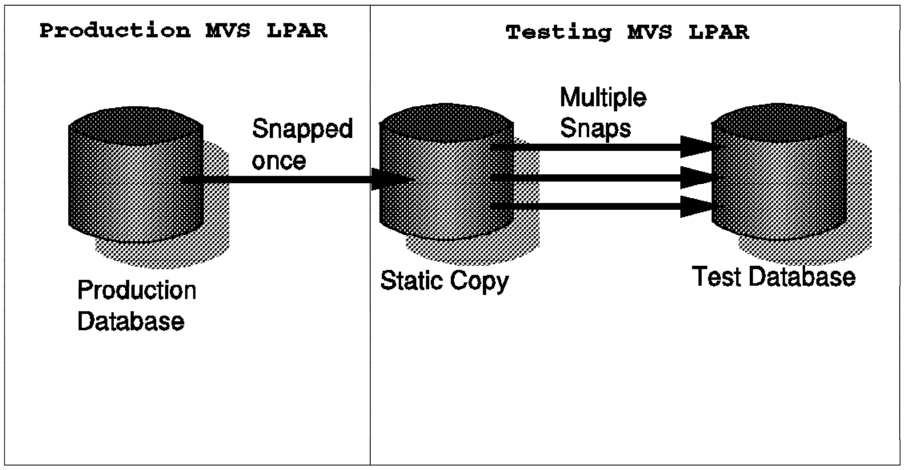


Figure 77. Production, Static and Test Databases w/o dedicated RVA

4. Move this static copy to your dedicated Year 2000 RVA if one is available.

   Figure 78 shows how you can use a dedicated RVA for Y2K testing. In this example the datamover might be PPRC or DFSMSdss dumps and restores.

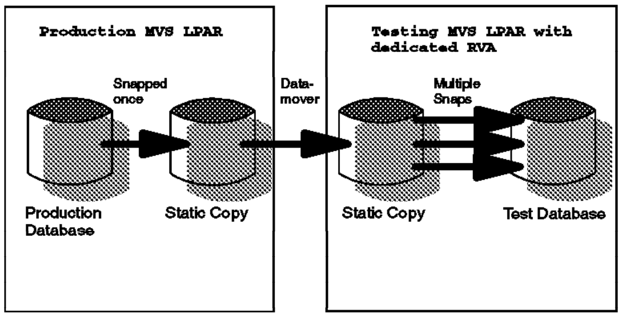


Figure 78. Production, Static and Test Databases with dedicated RVA

5. Resume update processing to the production databases.
6. Snap the static copy to create the Year 2000 test databases.
7. Refresh the Year 2000 test databases as required by snapping from the static copy.

Additionally, you may be able to create multiple copies of the test database. Testing can be performed by individuals or teams on their own copy of the databases, and the test databases can be refreshed without affecting the other team.

7.3.1 Snapping Database Data Sets

The production database data sets can be snapped at the data set level. A separate SNAP command is required for each database data set. In a DFSMS/MVS environment, duplicate data set names are not permitted, so each snap target data set name must be unique.

If testing is to be done on the same system as the production system, or the test and production systems have shared catalogs, the production, static, and test copies of the databases must all have unique data set names. Not only is this operationally complex, but it also exposes the installation to the risks discussed in 7.1, "Year 2000 Test Environment" on page 106.

If testing can be performed on a separate system that does not share the production system′s catalog structure but does share access to the RVA subsystem, the test database data sets can have identical names to the production database data set names.

To achieve a separated system, you have to identify a pool of volumes where the static database data sets will be placed. The production database data sets are then snapped to this pool. In a DFSMS/MVS environment, a Guaranteed Space storage class can be assigned with the SNAP DATASET command to direct the target data sets to a specific volume.

The target volumes can then be varied offline to the production system, moved to the Year 2000 RVA (when available) and varied online to the test system. Each of the static database data sets must be manually recataloged on the test system (Figure 79).

```
//RECAT    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DEFINE CLUSTER(NAME(HLQ.DATABASE.PROD1) -
        RECATALOG VOLUME(SNPVL1))
```

Figure 79. Recataloging a VSAM Cluster

Once the static database data sets have been cataloged on the test system, they can be snapped as often as necessary to create and refresh the test database. The test database data sets can have the same name as the original production database data sets if the test and production systems do not share a common catalog environment.

Here are the points to keep in mind when using SNAP DATASET to duplicate databases.

- Sufficient *functional* VTOC space must be available on the virtual DASD volumes to support the target database data sets.
- In a shared catalog environment, every data set name should be unique.
- All necessary database data sets must be included to create a complete and consistent test environment.

7.4 Snapping Database Volumes

You can use SnapShot to snap the volumes on which your production databases reside. The SNAP VOLUME command is discussed in more detail in 2.5, “Snapping Volumes” on page 27. If you are testing on a separate system that has access to the RVA subsystem, you can snap each of the database volumes, vary the target volumes offline from the production system, move them to the dedicated Year 2000 RVA (when available), and vary them online to the test system. This is the static copy of your database, from which you can refresh your test database.

7.4.1.1 COPYVOLID(YES) or COPYVOLID(NO)

If you run the volume snaps with COPYVOLID(YES), the target volumes will be relabeled with the same volser as the source volumes. When either the production or test system is IPLed, MVS will detect the duplicate volsers, and the operator will be asked which of the volumes are to be put offline. If the wrong volume is left online, the production databases could be corrupted.

The use of COPYVOLID(NO) eliminates this problem; however, the test environment is not an exact replica of the production environment, and more work may be required in some environments to ready the system for database testing use.

7.4.1.2 Catalog Implications

If you use SNAP VOLUME, the database data sets residing on the static database volumes do not have to be cataloged. These data sets will never be directly referenced by anything other than SNAP VOLUME, which does not require that the data sets be cataloged.

The database data sets on the test volumes must be cataloged on the test system. Issue IDCAMS DEFINE commands for each data set. Alternatively, you can copy the catalog from the production system to the test system. See Chapter 8, “Using SnapShot for System Changes” on page 115.

7.4.1.3 Functional Devices

A sufficient number of functional devices must be available and defined as the same device type as the production database volumes. The number of devices must be sufficient to support the static as well as test database volumes.

7.4.2 Net Capacity Load Considerations

There will be three or more copies of the databases:

- The production databases
- A static copy of the production databases
- The test databases, which are periodically refreshed from the static copy
- Any additional copies of the test databases

As time passes, track images of the production databases will be modified as a result of normal daily processing. Each track update causes an increase in the amount of NCL. Because the static copy of the databases is rarely refreshed from the production databases, the amount of NCL will continue to grow as the two copies diverge.

Additionally, a third copy of the databases will be used for testing. As tracks in the testing databases are changed, further NCL use will occur. However, every time the test databases are refreshed from the static databases, any extra NCL used by the test databases is released.

The more often the copies of the databases are refreshed, the less the impact on NCL usage.

7.5 Using SnapShot and XRC for Year 2000 Testing

SnapShot can be used in combination with XRC for Year 2000 testing. If you require that your Year 2000 systems be maintained at the same level of update as your production systems, you need the updates propagated to your Year 2000 system. However, it is essential to avoid polluting the production system with Year 2000 "stuff," in which case, you could use XRC to continuously copy the production systems to a set of RVAs. The XRC data mover system would be the only system with connections to both the production and Year 2000 test systems, and it would write only to the Year 2000 disks, thus avoiding contamination. The Year 2000 systems use the RVAs for testing.

Figure 80 shows a sample configuration for Year 2000 testing with a combination of RVA and XRC.

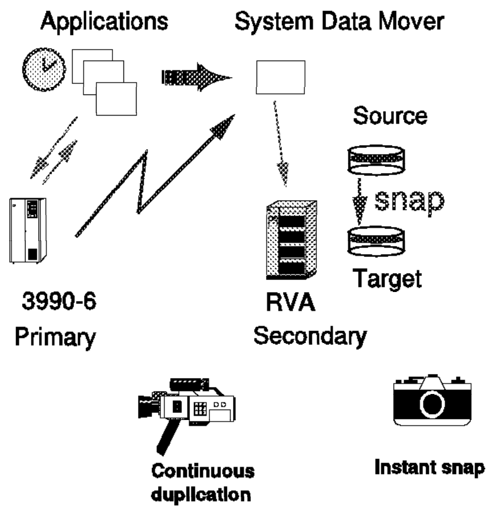


Figure 80. RVA and XRC for Year 2000 Testing

It is also possible to use SnapShot on the RVAs to take a point-in-time copy of the production system. Then, for testing, you could take further copies of the production data as needed. And those copies could always reset to the ″original″ copy by just snapping again from the ″frozen″ snap copy on the RVAs. So the RVAs contain three copies of production data: the ″live″ copy being updated by XRC; the ″master″ copy, which was a snap from the XRC copy; and the ″work″ copy (or copies) snapped from the ″master″ copy.

7.6 Using SnapShot for DB2 Year 2000 Testing

You can use SnapShot in a Year 2000 environment as an aid in the testing of applications for Year 2000 readiness. Using the example given in 4.7, "Application Testing" on page 67, you can use SnapShot to take multiple copies of DB2 tables spaces and test application programs from any given piont in the text cycle that you choose. On a wider scale, you can use SnapShot to copy entire DB2 test subsystems, using the backup and recovery method described in 4.9, "Cloning DB2 Data" on page 70.

The following DB2 products provide Year 2000 support:

- DB2 for MVS/ESA Version 3 (program product 5685-DB2)
- DB2 for MVS/ESA Version 4 (program product 5695-DB2)
- DB2 for OS/390 Version 5 (program product 5655-DB2)

We recommend migrating to a current release of DB2 and applying current service to be ready for the transition. Apply PTFs for Year 2000 support. APAR PN85320 provides the needed corrections for Version 3 and Version 4. Year 2000 support is incorporated in Version 5.

All previous releases of DB2 went out of service as of December 31, 1996, but the APAR for DB2 Version 2 Release 3 is PN76298.

DB2 Version 3 is planned to be out of service by December 1998. Several DB2 catalog tables do not have TIMESTAMP columns. Those columns have been added in Version 5, but DB2 operation is not affected, other than in customer interpretation of the date.

IBM′s Year 2000 Technical Support Center provides a Web page, *www.software.ibm.com/year2000/papers/db2y2k.html* that details DB2′s support for dates, times, and time stamps—called *datetime values*. The web page explains how to find the programs that depend on tables that contain these fields, by querying the DB2 catalog; shows how application plans, packages, and SQL statements for static SQL can be found quickly, and offers guidance on date and time stamp arithmetic and datetime comparisons.

Chapter 8. Using SnapShot for System Changes

In this chapter we discuss some uses of SnapShot for operating system changes and maintenance.

8.1 System Backup

You can use SnapShot to make backup and additional copies of your operating system. Many installations keep a separate set of operating system packs for test, maintenance, and recovery reasons. These operating system packs are backed up to tape and kept with the stand-alone restore version of DFSMSdss on a IPL-able tape. In addition, many installations keep a backup operating system on a single, self-contained, IPL-able system disk. They use the disk for emergency IPL to perform quick emergency repair actions on the production operating system. In this way installations do not have to restore a complete operating system pack or run some stand-alone program to perform repairs. This single-disk operating system is often kept on an IPL-able backup tape.

8.1.1 ServerPac

The OS/390 ServerPac is an operating system replacement method. The idea is to install your pregenerated and tested OS/390 ServerPac on a set of selected disks. All additional products you select from the OS/390 ServerPac checklist are integrated into the same set of distribution (DLIB), target, and SMP/E libraries and the OS/390 product. You use ISPF dialogs to customize and load the target, DLIB, and SMP/E data sets. If you keep this recommended structure for the OS/390 system and keep other program libraries and data sets that are not delivered with OS/390 on separate disks and SMP/E zones, you have a single replaceable entity for your operating system.

8.1.2 Creating a Single SYSRES with SnapShot

With the concept of a shared SYSRES volume and the introduction of system variables in MVS/ESA Version 5 and further enhancement in OS/390, the capability of IPL-ing many MVS/ESA or OS/390 images from a single set of operating system packs has greatly improved. As a consequence, the necessity to keep many separate system disks for several images in LPARs or CECs is greatly reduced.

If, however, you will be taking backups or building new operating systems from an existing operating system, you can use SnapShot. As the operating system disks mostly contain read-only data sets and program libraries, the RVA is an excellent device to maintain several identical operating systems, as no extra back-end store is required. Please remember that each volume requires an UCB.

You can use SnapShot to make a backup of your operating system, build a new system from an existing operating system, or clone an existing operating system. You can use the SNAP VOLUME command to make complete copies of the operating system disks, or you can use the SNAP DATASET command to make individual snap copies of most of the data sets that make up your operating system.

You can easily create complete copies of your production operating systems by using the SNAP VOLUME command to duplicate the system volumes. If you use the COPYVOLID(YES) parameter, you have an exact copy of your operating system. The newly copied disks will go offline to the system where you took the snap. You can keep them as backup disks, or you can use them to IPL another MVS image.

When you have duplicate volsers in a system, only one of the copies can be online to a single system. During IPL of the system, the operator receives messages about the duplicate volsers and queries about which device numbers to take offline. This may create some confusion for the operating staff, and which device numbers should be online to which systems must be well documented. An alternative is to use HCD. HCD lets you define the data about device parameters and features that is required by the operating system configuration. In the Define Device to Operating System Configuration panel, you can specify that the device numbers containing these new duplicate volumes should be considered online or offline to an operating system at IPL time.

The RVA is an excellent device for keeping a single-disk IPL-able operating system. You can make a such a system on a functional volume, using the SNAP DATASET command. Because the SNAP DATASET command does not support all data set types, you must run your operating system from a single functional volume. You have to build the data sets, using traditional utilities like IDCAMS.

Select a functional volume with a unique name so it can be online to your source system and put IPL-text on it. Because of the catalog implications and the smaller sizes you need, build the necessary catalog structures, page data sets, and other needed logs and other output data sets, like SMF, spool space, and jobqueue checkpoint. Then snap the operating system libraries to your new volume. Make the necessary updates to SYS1.PARMLIB, SYS1.PROCLIB, and SYS1.VTAMLST, and you should be able to IPL this single-disk operating system with a minimum TSO/VTAM configuration.

8.2 Recovery

If you keep a snapped version of your operating system disks, you may be able to IPL your backup system from those disks. This implies that you should keep only the master catalog on the operating system disks. There are no specific changes to system restore procedures with SnapShot. Once you restore from tape, you may have some data integrity exposures. However, the advantage with SnapShot is that you can keep a separate set of your operating system disks on functional volumes and have them ready for IPL in case of an operating system failure.

When you are installing service on your system with SMP/E, you might consider taking a SnapShot of your SMP/E environment and the target libraries. If testing shows that the application of service was unsuccessful, you can snap back to your original environment without running the time-consuming SMP/E restore jobs.

8.3 Integrity

SnapShot command processing requires serialization to provide data integrity and to prevent lock-out situations. SnapShot basically has the same serialization methods as DFSMSdss. If you are in a shared DASD environment, you will most probably be using a systemswide serialization tool like GRS in OS/390 or another serialization tool. Make sure that the enqueue requests from SnapShot are propagated to the other systems in the GRS complex by updating the Include Resource Name List (RNL) with the major names used. SnapShot uses the standard IXFP nonauthorized enqueue major name, SIBIXFP, with different minor names to serialize SnapShot processing. The minor name *volser* with scope SYSTEM is used for source and target allocations, and the *dsname* with scope STEP is used during source and target processing.

To serialize a SNAP DATASET command and its associated catalog and VTOC operation, SnapShot issues enqueues. Major name SYSDSN, minor name *dsname*, scope SYSTEM and SYSDSCB, and minor name *volser*S*dsname* scope SYSTEMS are used. During VTOC updates a device reserve is issued to serialize the source and target devices. The major name is SYSVTOC, with the minor name *volser* and the scope SYSTEMS. Operating system commands that are directly or indirectly invoked by SnapShot may also issue other enqueues and reserves.

When you issue a SNAP VOLUME command in a shared DASD environment, the target functional must be offline to other systems sharing the DASD, except the one performing the SNAP VOLUME command. When the snap is complete, the target volume is identical to the source. If you specify COPYVOLID(YES), the label from the source is retained on the target, and the target is varied offline. If you specify COPYVOLID(NO), the original label of the target is restored, and the target remains online.

If you specify TOLERATEENQFAILURE (NO), the data set will not be snapped if it is unavailable for exclusive serialization. If not open, the data set will be serialized for snap and released when complete. If you specify TOLERATEENQFAILURE (YES), SnapShot first tries to get exclusive serialization. If that fails, it performs the snap anyway. If you use TOLERATEENQFAILURE(YES) with DFSMSdss, you run the risk of creating a fuzzy backup. With SnapShot, because the snap operation is completed in fewer I/Os, there is less exposure to ″fuzziness″ for single-volume, single-extent data sets. However, it is your responsibility to ensure that there is consistency between the data sets that you snap.

8.4 Catalogs

In an MVS/ESA system, three kinds of catalogs are supported:

- OS CVOL catalogs introduced with the OS/360 operating systems in the 1960s
- VSAM catalogs
- ICF catalogs

The preferred ICF catalog is a repository for data set information and replaces VSAM and OS CVOL catalogs. If you have not already done so, convert your catalogs to ICF catalogs. Aside from all of the benefits of ICF catalogs, a major reason to convert now is that only ICF catalogs support dates after December 31, 1999.

Snapshot accesses the catalogs only through the formal DFSMS/MVS catalog services; the catalog structures are never directly accessed. The catalog functions used are query, update, and define.

The catalog data sets are not supported by SNAP DATASET. If you want to use SnapShot to back up your catalogs, you can take a volume snap of a volume that contains a user catalog.

You can use SnapShot to achieve a more consistent point in time backup of your catalog structure, by snapping the volumes that contain the catalogs. You can deallocate catalogs with the modify command:

```
F CATALOG,CLOSE(UCAT.JOSE)
F CATALOG,UNALLOCATE(UCAT.JOSE)
```

To prevent a catalog from being reopened, we recommend that you lock the catalog before the snap and unlock it after the snap. This requires IDCAMS ALTER IGG.CATLOCK authority in RACF. It is a good idea to submit the unlock job on hold, ahead of time. This approach may be disruptive to any jobs requiring access, but it prevents an intervening request from reopening the catalog before the snap is completed.

If you use indirect cataloging for SYSRES data sets, catalog entries have ****** as the volser name. No recataloging is needed when you use the snapped volume as the data sets on the active SYSRES are located during a catalog search. Indirect cataloging simplifies use of a SnapShot volume as an IPL volume.

Chapter 9. Using SnapShot in a VM/ESA Environment

IBM RAMAC SnapShot for VM/ESA Version 1 Release 1, IBM program product 5654-A03, provides a high-speed data duplication solution in the VM/ESA environment. Users of the RVA subsystem can exploit SnapShot for VM/ESA to dramatically reduce elapsed time and the computing resources consumed with traditional copying methods.

SnapShot for VM/ESA can be used for:

- Quick copying of volumes and minidisks
- Some traditional DASD dump restore (DDR) functions
- Quick formatting of minidisk space
- Quick formatting of temporary disk space
- Quick setup of new users
- VM backups
- Database backups

With SnapShot for VM/ESA you can make quick, duplicate views, or copies, of your data on a functional volume in the RVA subsystem, saving much of the time spent with traditional data copying methods. The time and resources required to format a minidisk can also be greatly reduced.

SnapShot for VM/ESA works on the volume and minidisk level. Therefore system administrators can copy single volumes and general users can make quick copies of their individual minidisks. If the target disk is not within the same RVA, you can invoke a data mover program to physically copy the data to the target.

You can also use SnapShot for VM/ESA as an alternative way of creating minidisks. Instead of the time consuming traditional formatting of a minidisk, you can use the RVA technology for instant formatting.

If you keep a preformatted minidisk with a PROFILE EXEC and PROFILE XEDIT and other standard distributions, you can snap this minidisk every time you set up a new user.

SnapShot for VM/ESA can also be used for the VM backups, database backups, and replicating test data for database testing.

The minimum prerequisite software requirements for SnapShot for VM/ESA are:

- 5684-112, VM/ESA Version 1 Release 2.2
- 5648-A17, IXFP Version 2 Release 1, plus PTF L170471

The appropriate microcode updates to support SnapShot for VM/ESA must also be installed on the RVA subsystem, see Chapter 1, “Introduction to RVA and SnapShot” on page 1 for details.

9.1 Installing SnapShot for VM/ESA

You must have installed IXFP Version 2 Release 1 before you install SnapShot for VM/ESA. The SnapShot for VM/ESA software is packaged in VMFPLC2 format and is installed with the VMFPLC2 load functions. The IXFP run disks are used during installation, and need some additional space. See *IBM RAMAC SnapShot for VM/ESA Installation Notes*, SC27-7218 for details.

9.2 Running SnapShot for VM/ESA

SnapShot for VM/ESA is run as subcommands under the SIBADMIN utility of IXFP. To invoke the utility, you issue the SIBADMIN command, which creates the environment where you can run the SnapShot for VM/ESA subcommands and the other IXFP configuration subcommands. When you issue the SIBADMIN command, you can enter the subcommands at the SIB: prompt. The subcommands can also be included in the SIBADMIN command.

The RVA services utility, SIBVMRVA, enables you to issue the SnapShot for VM/ESA subcommands from a CMS REXX EXEC or as a command from the CMS Ready prompt.

9.3 Snapping Volumes

Snapping volumes provides VM/ESA users with a means of quickly making a duplicate view of a single volume or a contiguous range of cylinders within a volume in an RVA subsystem. The function is invoked through the SNAP VOLUME command, which is a subcommand of the IXFP SIBADMIN command.

This command duplicates a single volume or a range of cylinders on a volume. The source and target volumes must reside within the same RVA subsystem and be of the same device type.

Privilege class A and B must be assigned to users who issue the command, usually systems programmers or systems administrators. Users of the SNAP VOLUME command must be able to issue the DEFINE MDISK and ATTACH CP commands, as SnapShot for VM/ESA uses these commands to obtain access to the source volume. The volume to be snapped, the source volume, and the target volume must be either free or attached to the system; they cannot be dedicated to any guest virtual machine or attached to any user. Both the source and target unit addresses must be real addresses.

The SNAP VOLUME command supports a data mover. A data mover is required when the source and target volumes to be snapped are not in the same RVA subsystem or partition in the RVA. The DATAMOVERNAME parameter describes a macro that SnapShot for VM/ESA executes to dynamically invoke the program that physically copies the volume. SIBDMDDR is a sample macro provided by IXFP that copies cylinder extents to a target device. The data mover program completes the copy operation before returning control to the user. If a data mover program is not specified in the command but is needed, or if a data mover cannot be located, the operation is terminated.

The RVA subsystem emulates 3380 and 3390 disks, which are count-key-data (CKD) format devices. On a physical disk with CKD format, the user records on a track are typically numbered in sequence. The layout of a physical record on the track is the count area, the optional key area, and the data area, all separated by gaps. The count area contains the ID of the data area that follows and has three fields. The first field describes the address of the record expressed as 5 bytes, CCHHR, which is the real cylinder, track number, and record number on the track, starting from cylinder zero on the physical disk. The two other fields describe the length of the key, which can be zero if there is no key area, and the length of the data area that contains the user data. On a VM minidisk, the count area ID fields are relative to the start of the minidisk, not to the real cylinder zero on a physical disk.

If you are snapping a volume, or portions of a volume, that contains minidisks, the count area ID fields in the records must not be relocated to the real cylinder zero of the target volume to indicate the new physical location of the minidisk on the target volume. If such relocation occurs, the user of the minidisk will experience I/O errors when trying to read from the record that had its count area ID field relocated. The RELOCATE(YES|NO) parameter controls relocation. The default value is NO; relocation of the count area ID fields will not take place, and the RVA will use the source count area ID fields on the target.

If you want to snap a portion of a volume to another disk, and it does not contain any minidisks, it may be appropriate to relocate the count fields. The RELOCATE(YES|NO) parameter is left for you to use, with a warning that you should know what you are doing.

Figure 81 shows the format of the SNAP VOLUME command.

```
SNAP VOLUME (
  SOUrce (VOLUME(volser) | (UNIT(device) FROM(cyl) )
  TaRGet (VOLUME(volser) | (UNIT(device) FROM(cyl) )
  FOR(ncyls) DataMoverNaMe(MACRO(macname)|NONE)
  RELOCate(Yes|No) DEBUG(ON|OFF) PROMPT(NO|YES))
```

Figure 81. SNAP VOLUME Command

Figure 82 shows an example of how to use SnapShot for VM/ESA to snap volumes 100 through 110 within the same RVA subsystem.

```
sibadmin
SIB0711I  Enter END to exit prompt mode.
sib:
snap volume (source(unit(100) target(unit(110))
SIB4791D Snap 3339 cylinders from VMVOL1 100 0-3338 to VMSNAP 110 0-3338?
SIB4791D Reply YES to continue or NO to cancel the subcommand.
sibadmin:
yes
SIB4617I 15:56:00 Snapshot completed, rc=0.
sib:
```

Figure 82. Using SnapShot to Snap a Volume

9.4 Snapping Minidisks

Snapping minidisks provides general VM/ESA users with a means of quickly making a duplicate view, or copy, of an individual minidisk. The function is invoked through the SNAP MINIDISK command, which is a subcommand of the IXFP SIBADMIN command.

The command duplicates one minidisk. The source and target minidisks must be on like device types and reside in the same RVA subsystem. If the minidisks do not reside in the same RVA subsystem, you have the option of using a data mover program to physically copy the minidisk to the target.

You can snap other users′ minidisks. You can use parameters to specify the VM userid and password for the owner of the minidisk. This requires that the minidisk be defined in the directory. Otherwise you can only snap existing virtual devices in your own configuration. You can also specify the link mode of the source and the target minidisks as specified in the CP LINK command.

See 9.3, “Snapping Volumes” on page 120 for a brief description of CKD format on physical disks. There is an option to specify whether to relocate the count area ID fields in the records of the snapped minidisk to match the real cylinder on the target minidisk or to use the source count area on the target disk. When you snap a minidisk, the new address of the tracks should not be reflected in the count area ID fields of the records. You can use the RELOCATE(NO) parameter, which is the default, to specify snapping the minidisk without relocating the count areas, so cylinder zero of the minidisk continues to be cylinder zero of the target minidisk, no matter where the minidisk is located on the physical target functional volume.

The SNAP MINIDISK command supports a data mover. A data mover is required when the source and target minidisks to be snapped not are in the same RVA subsystem. The DATAMOVERNAME parameter describes a macro that SnapShot for VM/ESA executes to dynamically invoke the program that physically copies the data. The data mover program completes the copy operation before returning control to the user. If a data mover program is not specified in the command but is needed, the operation is terminated.

Figure 83 shows the format of the SNAP MINIDISK command.

```
SNAP MiniDISK (
  SOUrce ( DEVice(device) FROM(cyl|0)
  USERid(userid) LinkMode(mode|RR) PassWord(password) )
  TaRGet ( DEVice(device) FROM(cyl|0)
  USERid(userid) LinkMode(mode|RR) PassWord(password) )
  FOR(ncyls) DataMoverNaMe(MACRO(macname)|NONE)
  RELOCate(Yes|No) DEBUG(ON|OFF)  )
```

Figure 83. SNAP MINIDISK Command

Figure 84 on page 123 shows an example of using SnapShot for VM/ESA to snap minidisks 191 through 291 within the same RVA subsystem.

```
sibadmin
SIB0711I  Enter END to exit prompt mode.
sib:
snap mdisk ( source(dev(191)) target(dev(291)) )
SIB4791D Snap 100 cylinders from USER 191 0-99 to USER 291 0-99?
SIB4791D Reply YES to continue or NO to cancel the subcommand.
sibadmin:
yes
SIB4617I 15:56:00 Snapshot completed, rc=0.
sib:
end
Ready; T=0.13/0.16 20:08:36
```

Figure 84. Using the SNAP MINIDISK Command

9.5 Instant Format

Before you can use any new minidisk, it must be formatted. This applies to your newly assigned minidisks, and to temporary minidisks. The CMS FORMAT command is used to initialize a VM minidisk for use with CMS files. Because the FORMAT command takes a long time and is a heavy user of system resources such as CPU and I/O, system performance may be affected.

SnapShot for VM/ESA provides a function that allows you to replace the traditional CMS minidisk FORMAT command with an instant formatting of CMS minidisks. This replacement is transparent to users of the FORMAT command. The Instant Format function is implemented through the SIBFMTSS utility, which snaps an already formatted minidisk to the target minidisk. The target cannot be larger than the source, but it can be smaller.

The basic principle of the SIBFMTSS utility is to intercept the CMS FORMAT command and use SnapShot for VM/ESA instead. If a snap cannot be done, the CMS FORMAT command is executed. The SIBFMTSS utility requires preformatted CMS minidisks as templates and intercept routines to intercept the CMS FORMAT command.

The minidisk is then formatted by snapping the source minidisk to the new target minidisk. This process is so fast that users may not realize that the format has completed.

9.5.1 Implementation

The system programmer must make available one or more preformatted CMS minidisks and the intercept routines. See *IBM RAMAC SnapShot Installing and Using SnapShot*, SC27-7217 for details.

You must format a CMS minidisk for every device type and block size you will be using, and the size of each minidisk must be the largest number of cylinders you expect users to format. The preformatted CMS minidisks must be available to users of the Instant Format function, and they must be specified as environment variables to SIBFMTSS. If you have multiple RVA subsystems, you need these minidisks in every RVA subsystem.

Use the SIBTRAP utility to intercept the CMS FORMAT command. This utility sets up a nucleus extension with a specified name and allows you to trap a command or function and have a REXX EXEC executed in its place. The SIBTRAP utility is designed to be executed as a CMS command; it is not an IXFP subcommand.

We recommend testing the Instant Format function before making it available. You should also test it from a general userid to make sure all required software is accessible before you update the SYSPROF EXEC to make it generally available.

If you issue the NUCXDROP CMS command, be aware that you will lose all nucleus extensions, including SIBTRAP. In order for instant format to continue working, you have to reinstate the SIBTRAP after the NUCXDROP.

9.5.2 Use

The IXFP files used for the Instant Format function must be on a minidisk available to all users of the CMS FORMAT command. The SIBFMTSS REXX EXEC receives control when a CMS FORMAT command is issued. Several environment variables can be passed to SIBFMTSS, either during installation of SIBTRAP or on the CMS FORMAT command. These environment variables include the source preformatted CMS minidisks, a low limit on cylinders to format before SnapShot should be used, tracing, error messages, and statistics options.

9.6 Guest Support

In this section we discuss the use of SnapShot for VM/ESA from guest operating systems in a virtual machine. A volume snap issued from VM/ESA is not supported if the volumes to be snapped are dedicated to a guest virtual machine. The volumes must be free or attached to the system.

9.6.1 VM/ESA

Several installations are running VM/ESA as a guest under VM/ESA for testing purposes. There are no special considerations for the second-level VM system; we recommend that you run SnapShot from the primary level VM/ESA system. You can use SnapShot to snap the VM/ESA system volumes before applying any fixes or maintenance.

9.6.2 VSE/ESA

The VM/ESA versions of IXFP and SnapShot for VM/ESA are not required to use an RVA subsystem from a VSE/ESA guest under VM/ESA. All SnapShot functions can be performed on dedicated devices by the VSE/ESA guest using IXFP/SnapShot for VSE/ESA which is described in Chapter 10, "Using IXFP/SnapShot in a VSE/ESA Environment" on page 129.

9.6.3 OS/390

The VM/ESA versions of IXFP and SnapShot for VM/ESA are not required to use an RVA subsystem from an OS/390 guest under VM/ESA. All SnapShot functions can be performed on dedicated devices by the OS/390 guest using the SnapShot for MVS/ESA program product.

To perform snaps on any VM/ESA files or volumes in this environment, SnapShot for VM/ESA must be installed.

9.7 Integrity

In this section we discuss some integrity issues related to the use of SnapShot for VM/ESA.

9.7.1 Host Data Caching

SnapShot for VM/ESA makes virtual copies of data, stored in the RVA subsystem. VM/ESA's minidisk cache or other vendors in-memory cache programs use processor storage to cache data normally stored on a DASD subsystem. If host data caching programs are used for the disks in the RVA subsystem, the data cached in the host can lose synchronization with the data on the RVA. When you use SnapShot and host caching programs, we recommend that you turn off the host caching program before invoking SnapShot for VM/ESA. To flush the data buffers out of processor storage to disk, quiesce running applications that are using the disks that will be involved in the snap. When the snap is finished, you can turn the caching program back on again and restart the application you stopped.

9.7.2 Volumes and Minidisks

The SNAP VOLUME command uses real addresses for both the source and target volumes and is intended to copy a volume that is not dedicated to another virtual machine or attached to a user. The source and target volumes must be attached to the system or be free.

The SNAP MINIDISK command uses virtual addresses for both the source and target volumes and is intended to copy minidisks, including full pack minidisks.

9.8 Recovery

In a recovery situation where a minidisk or volume is lost, you can use SnapShot for VM/ESA to snap back the snapped copies you previously took. Figure 85 shows an example of using SnapShot for VM/ESA to snap back minidisk 291 to minidisk 191 within the same RVA subsystem.

```
sibadmin
SIB0711I  Enter END to exit prompt mode.
sib:
snap mdisk (source(dev(291)) target(dev(191)))
SIB4791D Snap 100 cylinders from nnnn 291 0-99 to nnnn 191 0-99?
SIB4791D Reply YES to continue or NO to cancel the subcommand.
sibadmin:
yes
SIB4617I 15:56:00 Snapshot completed, rc=0.
sib:
end
Ready;
```

Figure 85. Using the SNAP MINIDISK Command to Snap Back

9.9 RVA Services Utility SIBVMRVA

SIBVMRVA is a utility that enables VM/ESA system administrators to directly invoke advanced functions of the RVA, such as SnapShot. The SnapShot services provided are minidisk snapping and Instant Format. These services can be invoked from a CMS REXX environment or as a command from the CMS prompt. SIBVMRVA is not an IXFP subcommand, so it does not require the SIBADMIN utility. It is designed to be executed as a REXX function. Figure 86 shows the format of the SIBVMRVA command for invoking SnapShot for VM/ESA from a REXX EXEC. If you are using the utility as a CMS command, omit the parentheses and separate the parameters by spaces.

```
SIBVMRVA (SnapShot,source,target,scchh,ecchh|End,tcchh,Cfr|Nocfr)
```

Figure 86. SIBVMRVA SnapShot REXX Function

In the scchh, ecchh, and tcchh parameters, the disk extent parameters are given in the form cc(.hh), where cc is the decimal cylinder number, and hh is the decimal track number. A period separates the cylinder and track number. The .hh is optional. If .hh is omitted, it defaults to zero for the start extent and to the last track on a cylinder for the end extent. For example, if you specify 123 for the start extent, it is cylinder 123 track 00. If you specify 123 for the end extent, it is cylinder 123 track 14. Both 3380 and 3390, the supported functional volumes, have 15 tracks per cylinder. The Cfr|Nocfr parameter specifies whether count field relocation should be performed or not. See 9.4, "Snapping Minidisks" on page 122 for more information about count field relocation. Figure 87 shows the format of the SIBVMRVA command for invoking Instant Format from a REXX EXEC. If you are using the utility as a CMS command, omit the parentheses and separate the parameters by spaces.

```
SIBVMRVA (CMSFmt,device,cyls|End,VLabel,blksize,source)
```

Figure 87. SIBVMRVA Instant Format REXX Function

Figure 88 shows how to snap a range of tracks with the SIBVMRVA utility used as a CMS command. The range of tracks can, for example, be a VSE/ESA file occupying the extent from cylinder 123 track 4 to cylinder 124 track 14. All cylinder and track values are given in decimal values. Count field relocation is not to be performed, which is the default.

```
SIBVMRVA Snapshot 120 140 123.04 124
```

Figure 88. SIBVMRVA Instant Format REXX Function

9.10 SIBTRAP

The SIBTRAP utility allows you to intercept a command or function and have a REXX EXEC that is executed in its place. It creates a nucleus extension with a unique name that executes the specified REXX EXEC as a function. This utility is run as a CMS command, it does not require the SIBADMIN environment. Figure 89 on page 127 shows the format of the SIBTRAP command.

```
SIBTRAP cmdname execname parms
```

Figure 89. SIBTRAP Command

Chapter 10. Using IXFP/SnapShot in a VSE/ESA Environment

IXFP/SnapShot for VSE/ESA provides a high-speed data duplication solution in the VSE/ESA environment. Users of the RVA will benefit from considerable improvements for system management, application development, data availability, and an enormous saving of time for copy operations and batch windows. See *RAMAC Virtual Array, Peer-to-Peer Remote Copy, and IXFP/SnapShot for VSE/ESA*, SG24-5360 for detailed information on IXFP/SnapShot for VSE/ESA.

The following are minimum requirements for use of IXFP/SnapShot for VSE/ESA:

- VSE/ESA 2.3.0 (5690-VSE) installed with VSE/AF Supervisor with PTF DY44820 or higher.
- RAMAC Virtual Array Licensed Internal Code (LIC) 04.03.00 or higher is needed. (See Note.) An equivalent microcode level is required for StorageTek Iceberg 9200)
- SnapShot requires Feature 6001 for 9393 systems or RPQ 8S0421 for StorageTek Icebergs.
- If IXFP/SnapShot for VSE/ESA is used in a VM/VSE environment, VM APAR VM61486 is required for mini-disk cache (MDC) support.

Note: Although LIC level 03.02.00 contains support for SnapShot, LIC level 04.03.00 introduced non-disruptive code load for the RVA′s microcode and is the recommended minimum level.

Before you use IXFP/SnapShot in the VSE/ESA environment, we recommend contacting your local IBM support center to get the latest information about levels of VSE/ESA, 9393 microcode and APARs.

10.1 Installing IXFP/SnapShot for VSE/ESA

To install IXFP/SnapShot you receive a tape with the product, which is stacked as any other Optional Product available with VSE/ESA. The procedure on how to install an optional product within VSE/ESA is described in *VSE/ESA Installation*, SC33-6604.

During the installation you must consider the following requirements to run the product:

- The product has to be installed in a VSE sublibrary (′lib.sublib′). IBM recommends using sublibrary PRD2.PROD.
- The phase $IJBIXFP has to be loaded into shared virtual area (SVA)

The phase $IJBIXFP requires only about 50 KB in the VSE sublibrary and therefore only about 50 KB in the SVA-24.

The following job stream Figure 90 on page 130 can be used to put the phase into SVA.

```
// LIBDEF *,SEARCH=(lib.sublib)
SET SDL
$IJBIXFP,SVA
/&
```

Figure 90. Job to put $IJBIXFP in the SVA

Note: You can include the job in the standard startup of your VSE system to have the phase included in the SVA always after an IPL.

Shipped with the product, a member IXFPOME.Z is installed in sublibrary ′lib.sublib′, which enables you to include the IXFP/SnapShot for VSE/ESA error messages into the VSE/ESA Online Messages Explanation (OME) file. ′lib.sublib′ is the library where IXFP/SnapShot is installed.

To make this happen, you must execute the following sample load job to get job IXFPOME.Z into the POWER RDR queue, (see Figure 91).

```
* $$ JOB JNM=JOBLOAD,CLASS=0,DISP=D
* $$ LST CLASS=A,DISP=H
// JOB JOBLOAD    LOAD IXFPOME.Z POWER RDR QUEUE
// EXEC DTRIINIT
ACCESS lib.sublibLOAD IXFPOME.Z
LOAD IXFPOME.Z
/*
/&
* $$ EOJ
```

Figure 91. Job to put LOAD IXFPOME.Z in the POWER RDR Queue

After putting the job IXFPOME into the VSE/ESA RDR queue, you must release it from there to put the IXFP/SnapShot messages into the OME file.

The IXFP/SnapShot for VSE/ESA messages included in this job are the same as listed in the current IXFP/SnapShot Memo to Current Licensees (GI10-0487-00).

Note: IXFP/SnapShot messages cover the message prefix IXF and state that you should use the job carefully to avoid any message overlay in the OME file with other IBM or vendor products.

The job IXFPOME.Z looks as shown in Figure 92 on page 131.

```
* $$ JOB JNM=IXFPOME,CLASS=0,DISP=D
* $$ LST CLASS=A,DISP=H
// JOB IXFPOME// OPTION PARTDUMP,NOSYSDMP
// OPTION PARTDUMP,NOSYSDMP
* LOADING IXFP/SnapShot MESSAGES INTO THE VSE MESSAGES EXPLANATION FILE
// PAUSE     PRESS ENTER TO CONTINUE
// EXEC DTRIATTN,PARM='EXPLAIN OFF'
// DLBL NEWMSGS,'VSE.MESSAGES.ONLINE',,VSAM,CAT=IJSYSCT
// EXEC IESMSGS,SIZE=60K,PARM='EU'
X
X IXFP Messages
X
/*
// EXEC DTRIATTN,PARM='EXPLAIN ON'
* IXFP/SnapShot MESSAGES HAVE BEEN LOADED
/&
* $$ EOJ
```

Figure 92. Job to load IXFP/SnapShot Messages

Note: The sample job is provided only in mixed-case English for installation on English Language systems. In case you want to use the job on non-English language systems, please change the PARM='EU' as follows:

- for German : PARM='GU'
- for Spanish : PARM='SU'
- for Japanese : PARM='JU'

For display in uppercase English please ensure that member IJBxDEF.Z is assembled with the following setting : IJBDEF DEFAULT,CASE,UPPER. For details on the use of IJBDEF please refer to the VSE/ESA Administration manual (SC33-6605).

10.2 IXFP/SnapShot for VSE/ESA

IXFP/SnapShot for VSE/ESA is a combination of software and RVA microcode functions. It has three main functions, namely : SnapShot, DDSR and Report. SnapShot is the data duplication utility which exploits the RVA's virtual disk architecture to achieve instantaneous copy without actually using resources. DDSR releases space occupied by a data set, a volume and a CYLINDER range and a specified data set. Report provides information about the space utilization and subsystem utilization summary of the RVA.

You invoke IXFP/SnapShot for VSE/ESA functions through the attention routine in three basic ways. These are:

1. From a console:

   You enter IXFP/SnapShot for VSE/ESA function on the command line of the VSE operator console or through the ICCF console operator interface.

   Note: Please make sure that you are authorized to enter operator commands when using the ICCF console operator interface.

2. From a batch job:

   You can code the function you want to invoke in the PARM field of the VSE-provided DTRIATTN module and submit the batch job for execution.

3. From a REXX clist:

   Similar to DTRIATTN, you can code the IXFP function in the SENDCMD of REXX. Use of REXX clist offers more flexibility and allows use of logic for automated procedures.

   Note: More information on the REXX/VSE Console Automation Capability can be found in the *REXX/VSE V6R1.2 Reference Manual*, SC33-6642-01, Chapter 14.

10.2.1 IXFP/SnapShot for VSE/ESA General Syntax

Figure 93 is the general syntax of the IXFP command.

```
                   ┌─,──────────────────────────────────────────────────────────────┐
►►──IXFP───┬─SNAP,─┬─▼source──┬────────────────────────┬─:target──┬────────┬──┬──────────────┬──┴─┬──┬─────────────┬──┬────►◄
           │       │          └─(scyl──┬─-scyl─┬─)─────┘           └─(tcyl)─┘  └─,VOL1=volid──┘    │  └─,NOPROMPT───┘  │
           │       │                   └─.ncyl─┘                                                   │                   │
           │       └─source(DSN='data-set-name'):target──┬────────┬───────────────────────────────┘                   │
           │                                              └─(tcyl)─┘                                                   │
           ├─DDSR──┬─────────────────────────────────────┬──┬─────────────┬────────────────────────────────────────────┤
           │       │  ┌──────────────────────────────┐   │  └─,NOPROMPT───┘                                            │
           │       ├──▼,unit──┬──────────────────────┬─┴─┤                                                             │
           │       │          └─(dcyl──┬─-dcyl─┬─)───┘   │                                                             │
           │       │                   └─.ncyl─┘         │                                                             │
           │       └─,unit(DSN='data-set-name')──────────┘                                                             │
           │        ┌─────────┐                                                                                        │
           └─REPORT─▼─┬─────┬─┴────────────────────────────────────────────────────────────────────────────────────────┘
                      └─,id─┘
```

Figure 93. General IXFP Command Syntax

10.2.2 IXFP SNAP

SNAP identifies this command as a SNAP function. Figure 94 is the syntax of the IXFP SNAP function.

```
                   ┌─,──────────────────────────────────────────────────────────────┐
►►──IXFP─────SNAP,─┬─▼source──┬────────────────────────┬─:target──┬────────┬──┬──────────────┬──┴─┬──┬─────────────┬───────►◄
                   │          └─(scyl──┬─-scyl─┬─)─────┘           └─(tcyl)─┘  └─,VOL1=volid──┘    │  └─,NOPROMPT───┘
                   │                   └─.ncyl─┘                                                   │
                   └─source(DSN='data-set-name'):target──┬────────┬───────────────────────────────┘
                                                          └─(tcyl)─┘
```

Figure 94. IXFP SNAP Command Syntax

There are three ways to do a SNAPSHOT. You can take a copy of:

1. A Total VOLUME

   You copy a full volume by identifying the device address (cuu) or VOLID label of the source and target in the IXFP SNAP command. Intermixing of the device address and VOLID label to specify the source and target in the IXFP command is allowed. Use the VOL1= parameter to specify the volume label that the target will assume after the copy has completed. If the VOL1= parameter is omitted, then the source and target volumes will have the same VOLID label when the copy is completed.

   The NOPROMPT parameter is included to prevent the decision-type messages from being issued. Otherwise, the decision-type messages will be issued for the operator to verify and confirm.

VSE/VSAM

VSE/VSAM support is not currently included in IXFP/SnapShot for VSE/ESA. This does not mean, however, that you cannot take advantage of IXFP/SnapShot for VSE/ESA with VSE/VSAM data sets. Using volume snap, VSAM data sets can be indirectly copied. This is possible when the VSAM Catalog, Space and Cluster are all in the same volume. In VSE, there is a way to reference the two USER CATALOGs individually through the use of the ASSGN JCL statement; however, the VSAM share option restrictions still apply. The ideal scenario is to assign the target volume into another LPAR or copy to tape using IDCAMS REPRO in order not to encounter the restrictions. If the intention of the volume snap is for backup purposes using FASTCOPY or DDR (VM), then the VSAM share option restrictions will not be encountered.

2. A range of CYLINDERS

   You copy a range of CYLINDERs by identifying the device address or VOL1 label of the source and target in the IXFP SNAP command. In addition, the decimal start-cylinder (scyl) and end-cylinder (-scyl) or number of cylinders (,ncyl) are specified in parentheses and appended to the source specification. Optionally, the target start-cylinder (tcyl) may be included to specify where copying is to start on the target device.

   You specify the target start-cylinder (tcyl) to relocate the range of cylinders on the target volume. Indicate target start-cylinder 1000 to relocate the range of cylinders on the target volume.

   You may or may not include the NOPROMPT parameter to prevent the decision-type messages from being issued, or allow them to be issued.

3. A non-VSAM FILE

   You copy a file by indicating the data set name on the source device using the DSN= parameter, aside from identifying the source and target device address or VOLID labels. The file specified must be non-VSAM. Optionally, the target start-cylinder (tcyl) may be included to specify where copying is to start on the target device if relocation is required.

Notes:

If source is identified by its VOLID, it must either be the only volume with that VOLID, or it must be the only VOLUME with that VOLID which is up (DVCUP), otherwise an error message will be issued.

The target device must be down (DVCDN) prior to initiating the volume snap function.

If the target device is identified by its VOLID, it must either be the only volume with that VOLID, or it must be the only volume with that VOLID which is down (DVCDN), otherwise an error message will be issued.

Only when doing full VOLUME snaps can you use the VOL1 parameter.

Volume copying will be done unconditionally within the specified or assumed boundaries. VSE does not perform any VTOC checking on the specified target device and thus will not provide any warning messages regarding overlapping extents, secured or unexpired files.

The highest (end) cylinder number must not exceed 32767 or the maximum number of cylinders of the devices. The start cylinder number must not be greater than the end cylinder number.

Cylinder range copying will be done unconditionally within the specified or assumed boundaries. VSE does not perform any VTOC checking on the specified target device and thus will not provide any warning messages regarding overlapping extents, secured or unexpired files.

The highest (end) cylinder number must not exceed 32767 or the maximum number of cylinder of the devices. The start cylinder number must not be greater than the end cylinder number.

When snapping files, VSE will perform VTOC checking on the specified target device and will provide any warning messages regarding overlapping extents, secured or unexpired files.

The source and the target device must be of the same type and must be within the same RVA subsystem. (If you are using test partitions, the source and the target must also be in the same partition.) SnapShot commands can be set up to be invoked by end users as well as by database administrators and system programmers.

10.2.3 IXFP DDSR

DDSR identifies this command as a DDSR function. Deleted Data Space Release (DDSR) causes the releasing of the physical storage space associated with:

- Expired FILES
- A total VOLUME,
- A range of CYLINDERS
- A specified FILE

This function ensures that the all space releases done on the VSE system (normally through deletes) results in space releases in the RVA subsystem. Figure 95 is the syntax of the IXFP DDSR command.

```
►►──IXFP────DDSR─┬──────────────────────────────────┬──┬────────────┬──────────────────────────────────►◄
                 │  ┌──────────────────────────────┐│  └─,NOPROMPT─┘
                 ├──▼,unit─┬──────────────────────┬┴┤
                 │         └─(dcyl─┬─-dcyl─┬─)─────┘ │
                 │                 └─.ncyl─┘         │
                 └─,unit(DSN='data-set-name')────────┘
```

Figure 95. IXFP DDSR Command Syntax

1. Expired files

   DDSR, when specified without any additional parameters, deletes the VTOC entries and all physical space for all expired non-secured files residing on VSE managed RVA devices. These extents are freed up and become part of free space.

   The NOPROMPT parameter is included to prevent the decision-type messages to be issued. Otherwise, the decision-type messages will be issued for the operator to verify and confirm.

Notes:

When doing DDSR for expired files, VSE will perform checking on the online (up) units.

DDSR will check only those RVA devices managed by the VSE system.

DDSR will consider only the files created by VSE.

2. A total VOLUME

   Deleting and releasing space for an entire volume is possible when you include the device address or VOLID label with no other operands.

   Notes:

   The device should belong to the RVA subsystem managed by the VSE system.

   The device should be in the off-line (DVCDN) condition.

   The device should be re-initialized before using it as a regular volume again. However, if it going to be used as a target for a volume snap, then you can leave it as is.

3. A range of CYLINDERS

   This command is similar to the DDSR volume, except that additional specifications are added to signify the decimal start cylinder and end cylinder (dcyl-dcyl) or the decimal start cylinder and the number of cylinders (dcyl,ncyl) to delete.

   Notes:

   The device should belong to the RVA subsystem managed by the VSE system.

   The device should be in the off-line (DVCDN) condition.

   The highest (end) cylinder number must not exceed 32767 or the maximum number of cylinder of the devices. The start cylinder number must not be greater than the end cylinder number.

4. A specified FILE

   This command is similar to doing DDSR for a specific range of cylinders. The difference is that a DSN= (data-set-name) specification is coded in place of the decimal start cylinder and decimal end cylinder.

   Notes:

   The device should belong to the RVA subsystem managed by the VSE system.

   The device should be in the online (DVCUP) condition, otherwise the command is rejected and an error message is displayed.

   The file must be non-VSAM.

   The file will be deleted unconditionally and the space returned to the RVA free-space.

   When processing multi-volume-files, DDSR should be repeated for all the volumes containing the file extents.

10.2.4 IXFP REPORT

Provides information about the space utilization of a single or a range of RVA devices, and/or provides information about the space utilization of all devices (that had been added during IPL) of an RVA subsystem, as well as important subsystem utilization summary information.

Figure 96 is the syntax of the IXFP REPORT command.

```
                        ┌─────┐
►►──IXFP────REPORT──▼     │──────────────────────────────────────────────────────────►◄
                     └─,id─┘
```

Figure 96. IXFP REPORT Command Syntax

When invoked without the ′id′ parameter, all information about every RVA subsystem, including all the devices known to the VSE system, is displayed.

If information for a specific device, a specific range of devices, or a specific RVA subsystem is needed, then the ′id′ parameter is included to limit the scope of the report output.

To get all space information for the entire VSE system, issue the command without any additional parameter.

Notes:

The device should belong to the RVA subsystem managed by the VSE system.

The REPORT function, if used under VM, only works for full-pack mini-disks or dedicated devices.

NOPROMPT is not a valid parameter for IXFP REPORT.

Please refer to Figure 97 on page 138 which shows a sample IXFP Report output.

```
ixfp report
AR 0015 SUBSYSTEM 1321117
AR 0015 *** DEVICE    DETAIL  REPORT ***
AR 0015     <---FUNC. CAPACITY (MB)--->  <---CAPACITY (%)--->   PHYS.   COMP.
AR 0015 CUU   DEF   ALLOC STORED UNUSED  ALLOC  STORED UNUSED USED(MB)  RATIO
AR 0015 80E 2838.0  N/A    369.2 2468.7   N/A    13.01  86.99   213.6    1.72
AR 0015 80F 2838.0  N/A      0.8 2837.1   N/A     0.03  99.97     0.0    9.65
AR 0015
AR 0015 *** DEVICE    SUMMARY REPORT
AR 0015 CAPACITY         <-----TOTAL----->  <------TOTALS %------>     COMP.
AR 0015    DEFINED            5676.032 MB  100.00                      RATIO
AR 0015    STORED              370.129 MB           6.52
AR 0015      PHYS.USED         213.688 MB                   3.76       1.73
AR 0015    UNUSED             5305.903 MB          93.48
AR 0015
AR 0015 *** SUBSYSTEM SUMMARY REPORT ***
AR 0015 SYSTEM    DEFINED-CAPACITY     DISK-ARRAY-CAP  FREE-DISK-ARRAY-CAP
AR 0015   PROD       726532.208 MB      117880.209 MB        57752.311 MB
AR 0015
AR 0015  NET-CAPACITY-LOAD(%)   COLL.-FREE-SPACE(%)   UNCOLL-FREE-SPACE(%)
AR 0015  TEST   PROD  OVERALL   TEST   PROD  OVERALL  TEST   PROD  OVERALL
AR 0015   0.00  51.01   51.01    0.00  47.55   47.55   0.00   1.44    1.44
AR 0015 1I40I  READY
```

Figure 97. IXFP REPORT Example

The IXFP REPORT function produces three types of report, namely:

1. Device Detail Report
2. Device Summary Report
3. System Summary Report

The Device Detail Report is always displayed whatever type IXFP REPORT is invoked. The Device Summary and System Summary Reports are displayed when a report for a device range or the entire subsystem is requested.

10.3 Differences Between IXFP/SnapShot and IXFP/SnapShot for VSE/ESA

- IXFP/SnapShot for VSE/ESA combines the most important functions of IXFP and SnapShot in one product.
- The usage of the ECAM interface by IXFP/SnapShot for VSE/ESA is transparent to the user.
- VSAM datasets are indirectly copied.
- There is no Data Mover Support.

Appendix A. IXFP SMF Record Subtype 8

IXFP SMF Record Subtype 8 now records the data for the new space utilization fields, SHARED and UNIQUE, in the PHYSICAL CAPACITY USED report.

The format of the space utilization performance record has been updated to include the new variables for SHARED and UNIQUE physical capacity used. The size of the record has increased by approximately 116 bytes.

The format of the record is:

OFFSET DEC HEX	FIELD NAME	LENGTH	TYPE	DESCRIPTION
524 20C	NOTRMAPU	4	INTEGER	(***) NUMBER OF UNIQUE TRACKS MAPPED.
528 210	BEBYTUNQ	8	INTEGER	(***) NUMBER OF UNIQUE BACK END BYTES.
536 218	BEBYTSHR	8	INTEGER	(***) NUMBER OF SHARED BACK END BYTES.
544 220	PHCAPUSS	8	INTEGER	SHARED PHYSICAL CAP USED (IN MEGABYTES)
552 228	PHCAPUSU	8	INTEGER	UNIQUE PHYSICAL CAP USED (IN MEGABYTES)
560 230	TPHCPSRP	8	INTEGER	(PRODUCTION PARTITION:) TOTAL SHARED PHYSICAL CAPACITY USED - PRODUCTION PARTITION
568 238	TPHCPSRT	8	INTEGER	(TEST PARTITION:) TOTAL SHARED PHYSICAL CAPACITY USED - TEST PARTITION
576 240	TPHCPSRB	8	INTEGER	(TOTALS:) TOTAL SHARED PHYSICAL CAPACITY USED - BOTH PARTITIONS
584 248	TPHCPUNP	8	INTEGER	(PRODUCTION PARTITION:) TOTAL UNIQUE PHYSICAL CAPACITY USED - PRODUCTION PARTITION
592 250	TPHCPUNT	8	INTEGER	(TEST PARTITION:) TOTAL UNIQUE PHYSICAL CAPACITY USED - TEST PARTITION

```
600 258   TPHCPUNB     8     INTEGER  (TOTALS:) TOTAL UNIQUE
                              PHYSICAL CAPACITY USED -
                              BOTH PARTITIONS

608 260   RES7        32    CHARACTER (RESERVED)
```

Appendix B. Sample SNAPDSSU Procedure

This procedure and EXEC are available in softcopy through the Internet. The file is *SNAPDSSU.TXT*.

Point your Web browser to:

`ftp://www.redbooks.ibm.com/redbooks/SG242241`

Alternatively you can go to:

`www.redbooks.ibm.com`

and select SEARCH and then Additional Redbook Materials.

The SNAPDSSU procedure consists of three steps:

- Step 1 - performs a delete of SNAPDSSU data sets
- Step 2 - executes the DFSMSdss DUMP command in simulation mode, using the PARM=′TYPRUN=NORUN′ parameter, to obtain a list of data set names to be processed by the SNAPDSSU EXEC
- Step 3 - executes SNAPDSSU

  During execution, SNAPDSSU performs the following:

  – Checks the catalog status of source and target data sets
  – Generates a target data set name, based on parameters passed to SNAPDSSU
  – Creates a job to snap all source data sets selected by the DFSMSdss step, which we refer to as the SNAPTO JCL or job throughout this appendix.
  – Creates matching JCL to snap back data sets, which we refer to as the SNAPBACK JCL or job throughout this appendix.

B.1 SNAPDSSU Command Syntax

Figure 98 on page 144 shows the command format and parameters you can pass to SNAPDSSU.

```
SNAPDSSU DATASET( -
                 PREFIX(blank|8 char)                         -
                 TARGETVOL(volser) -                          -
                 SECLVL(blank|8 char|UNIQUE)                  -
                 SUFFIX(blank|8 char|UNIQUE)                  -
                 CATCHECK(blank|SOURCE|SRC|TARGET|TGT|BOTH)   -
                 VOLUMECOUNT(blank|interger)                  -
                 ESOTERIC(blank|8 char)                       -
                 DATAMOVERNAME(blank|NONE|DSS|DFDSS|ADRDSSU)  -
                 DATACLASS(blank|8 char)                      -
                 STORCLASS(blank|8 char)                      -
                 MANAGEMENTCLASS(blank|8 char)                -
                 HOSTCOPYMODE(blank|EXCLUSIVE|EXCL|SHARED|SHR) -
                 TOLERATEENQFAILURE(blank|YES|NO)             -
                 TOLERATETRUNCATION(blank|YES|NO)             -
                 CATALOG(blank|YES|NO)                        -
                 REPLACE(blank|YES|NO)                        -
                 SNAPBACK(blank|YES|NO|REPLACE))
```

Figure 98. SNAPDSSU Command Format

B.1.1 SNAPDSSU Options

You should customize the SNAPDSSU procedure to meet your requirements.

The SNAPDSSU options that can be modified are:

PREFIX (blank|8 char)

This must be a valid HLQ. SNAPDSSU checks that it exists and terminates if not found.

When you specify a PREFIX, it replaces the HLQ of every data set in the filter list.

SECLVL (blank|8 char|UNIQUE)

You can specify any valid second level qualifier in SECLVL, as long as it meets with normal data set naming standards.

If you specify UNIQUE, the second level qualifier is replaced with the step number of each SNAP command generated.

SUFFIX (blank|8 char|UNIQUE)

You can specify any valid low level qualifier in SUFFIX, as long as it meets with normal data set naming standards.

If you specify UNIQUE, two low level qualifiers in date and time format are added to the target data set for each SNAP command generated.

CATCHECK (blank|SOURCE|SRC|TARGET|TGT|BOTH)

If you specify CATCHECK() or omit the option, SNAPDSSU does not perform any check on source and target data sets.

You can instruct SNAPDSSU to check both source and target data sets with the BOTH parameter, or to select the SOURCE and TARGET only.

If SOURCE catalog checking is requested and no ICF catalog entry can be found during processing, the data set is deemed uncataloged, and SNAPDSSU does not generate a SNAP command for it. SNAPDSSU issues a warning message, and the step completes with a condition code of 4.

If you want to generate a SNAP command for the data set, either catalog the data set in question or rerun SNAPDSSU without the CATCHECK option.

SNAPBACK (blank|YES|NO|REPLACE)

If you specify SNAPBACK() or omit the option, SNAPDSSU does not generate the SNAPBACK JCL.

The YES parameter generates SNAPBACK JCL for every source data set selected.

When the REPLACE parameter is specified, SNAPDSSU includes the REPLACE(YES) option for every SNAP command generated.

B.1.2 SNAP Command Options

You can customize most of the SNAP command options. When you pass an option to SNAPDSSU, it is applied to every data set for which SNAPDSSU creates a SNAP command.

If you do not specify an option, or pass a blank parameter, that is:

```
PREFIX()
```

the option is not generated for the SNAP command, and the SNAP command default is used.

Note: Make sure you fully understand the SNAP command before using the SNAPDSSU procedure. Be aware of the defaults and the effects of overriding them.

The SNAP options that can be modified are:

TARGETVOL	(volser)
VOLUMECOUNT	(blank|integer)
ESOTERIC	(blank|8 char)
DATAMOVERNAME	(blank|NONE|DSS|DFDSS|ADRDSSU)
DATACLASS	(blank|8 char)
STORCLASS	(blank|8 char)
MANAGEMENTCLASS	(blank|8 char)
HOSTCOPYMODE	(blank|EXCLUSIVE|EXCL|SHARED|SHR)
TOLERATEENQFAILURE	(blank|YES|NO)

TOLERATETRUNCATION
(blank|YES|NO)

CATALOG (blank|YES|NO)

REPLACE (blank|YES|NO)
The REPLACE option is for the source data set only.

For more information about the SNAP DATASET command, refer to *IBM RAMAC SnapShot for MVS/ESA Using SnapShot*, SC27-7178all

Figure 99 performs a delete of the SYSPRINT and SNAPSHOT JCL data set.

```
//SSMSCSN JOB (GB86687),'SNAP DSET JOB',
// CLASS=H,MSGCLASS=H,NOTIFY=&SYSUID
//**************************************************************
//*** COMMENTS:                                              ***
//**************************************************************
/*JOBPARM SYSAFF=*
//STEP01    EXEC PGM=IDCAMS,REGION=1028K
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
       DELETE 'SNAPSHOT.DB2DSET.SYSPRINT'
       DELETE 'SNAPSHOT.DB2DSET.SNAPTO.CNTL'
       DELETE 'SNAPSHOT.DB2DSET.SNAPBACK.CNTL'
       SET MAXCC = 0
/*
```

Figure 99. SNAPDSSU Procedure: Step 1

Figure 100 uses DFSMSdss to generate a list of data sets to be processed by the SNAPDSSU EXEC.

```
//STEP02    EXEC PGM=ADRDSSU,REGION=4096K,PARM='TYPRUN=NORUN'
//SYSPRINT  DD DSN=SNAPSHOT.DB2DSET.SYSPRINT,DISP=(NEW,CATLG),
//             DCB=(RECFM=FB,LRECL=00132,BLKSIZE=),
//             SPACE=(TRK,(0020,0010),RLSE),UNIT=SYSDA
//DASD  DD DUMMY
//SYSIN     DD DSN=SNAPSHOT.DB2DSET.FILTERS(DB2DSET),DISP=SHR
/*
```

Figure 100. DFSMSdss Dump Using TYPRUN='NORUN'

Figure 101 show the sample DFSMSdss DUMP command syntax used in our example.

```
           DUMP OUTDD(DASD) -
           TOL(ENQF) -
           DATASET(INCLUDE(DB2B.**) -
                   EXCLUDE(DB2B.BSDS*.**,  -
                           DB2B.*.DSNDB*.**))
```

Figure 101. Sample DFSMSdss DUMP Command

We chose to execute the DFSMSdss SYSIN from a library rather than including it in the actual SNAPDSSU job, to simplify the process of tailoring JCL. SNAPDSSU places no special requirements on the way the DFSMSdss step is coded.

Figure 102 show an example of the command format for the SNAPDSSU EXEC.

```
//STEP03    EXEC PGM=IKJEFT1A,DYNAMNBR=1000,REGION=0M,COND=(7,LE)
//SYSPROC   DD  DSN=SNAPSHOT.PROD.CLIST,DISP=SHR
//INPUT     DD  DSN=SNAPSHOT.DB2DSET.SYSPRINT,DISP=OLD
//OUTPUT    DD DSN=SNAPSHOT.DB2DSET.SNAPTO.CNTL,DISP=(NEW,CATLG),
//             DCB=(RECFM=FB,LRECL=00132,BLKSIZE=),
//             SPACE=(TRK,(0020,0010),RLSE),UNIT=SYSDA
//OUTPUT2   DD DSN=SNAPSHOT.DB2DSET.SNAPBACK.CNTL,DISP=(NEW,CATLG),
//             DCB=(RECFM=FB,LRECL=00132,BLKSIZE=),
//             SPACE=(TRK,(0020,0010),RLSE),UNIT=SYSDA
//*
//*  COMMENT OUT THE NEXT TWO DD CARD IF YOU DO NOT WISH TO
//*  ADD A JOBCARD/JCL BEFORE THE SNAP STEPS
//*
//TOCARD    DD DSN=SNAPSHOT.TOCARD.CNTL(DB2DSET),DISP=SHR
//BACKCARD  DD DSN=SNAPSHOT.BACKCARD.CNTL(DB2DSET),DISP=SHR
//*
//*  COMMENT OUT THE NEXT TWO DD CARD IF YOU DO NOT WISH TO
//*  APPEND JCL AFTER THE SNAP STEPS
//*
//*  FAILURE TO DO SO WILL RESULT IN IEC141I 013-18 ERROR MESSAGES
//*
//*
//ADDTOJOB  DD DSN=SNAPSHOT.ADDTOJOB.CNTL(DB2DSET),DISP=SHR
//ADDBAJOB  DD DSN=SNAPSHOT.ADDBAJOB.CNTL(DB2DSET),DISP=SHR
//SYSTSPRT  DD  SYSOUT=*
//SYSTSIN   DD  *
    SNAPDSSU DATASET( -
             PREFIX(SNAPSHOT) -
             TARGETVOL(9RMDA2) -
             CATCHECK(BOTH) -
             ESTERIC() -
             DATAMOVERNAME(NONE) -
             DATACLASS(DATC001) -
             TOLERATEENQFAILURE(YES) -
             TOLERATETRUNCATION(NO) -
             CATALOG(YES) -
             REPLACE(NO) -
             SNAPBACK(REPLACE))

//
```

Figure 102. Sample SNAPDSSU JCL

You can specify optional DD statements in the SNAPDSSU JCL to include your own job card for the SNAPTO and SNAPBACK jobs generated and append additional JCL.

The following DD statements are supported:

TOCARD	Use the TOCARD DD statement to override the SNAPDSSU default job card for the SNAPTO generated job. As well as job card information, you can include additional JCL steps in the TOCARD data set to process ahead of the generated SNAP commands. SNAPDSSU copies the content of the data set or PDS member that you specify.
BACKCARD	Use the BACKCARD DD statement to override the SNAPDSSU default job card for the SNAPBACK generated job. As well as job card information, you can include additional JCL steps in the BACKCARD member to process after the generated SNAP commands. SNAPDSSU copies the content of the data set or PDS member that you specify.
ADDTOJOB	Use the ADDTOJOB DD statement to append JCL to the SNAPTO job. If you do not define the DD statement in the SNAPDSSU step, no action is taken.
ADDBAJOB	Use the ADDBAJOB DD statement to append JCL to the SNAPBACK job. If you do not define the DD statement in the SNAPDSSU step, no action is taken.

B.2 Hints and Tips

After SNAPDSSU has been installed, set up a number of DFSMSdss filters to use in step 2 of the SNAPDSSU procedure.

Tailor the SNAPDSSU DATASET options. SNAPDSSU generates the SNAP command JCL for you. SNAPDSSU does not submit the job or execute the SNAP commands. Therefore you can safely run the procedure time and time again, until you are satisfied with the JCL you have generated.

B.2.1 Considerations

- Take care when using the PREFIX, SECLVL, and SUFFIX options. Consider the effect of the values you put in these options and how they will change the length of the target data set names SNAPDSSU generates.

  When SNAPDSSU constructs the target data set name, it checks the length of the new data set. If the target data set length exceeds 44 characters SNAPDSSU generates an alternative *unique* target data set name, to avoid a failure when you submit the SNAPTO job. SNAPDSSU ssues a warning message, and the step completes with a condition code of 4.

- Avoid being too general with the data set name filters you include in the DFSMSdss DUMP Include list.

If you select data sets with only two level qualifiers and change the prefix, each data set acquires the same target data set name.

For example:

```
   DFSMSdss Filter             Source Data           Target Data
                              sets selected         sets selected
                                                    PREFIX(SNAPSHOT)

                        ====>   STSGMM.ISPPROF  ====>   SNAPSHOT.ISPPROF
                              STSGAH.ISPPROF        SNAPSHOT.ISPPROF
 ...(INCLUDE(STSG*.**)
```

A target data set of SNAPSHOT.ISPPROF is created for data sets STSGMM.ISPPROF and STSGAH.ISPPROF.

B.2.2 Recommendations

- You can use SNAPDSSU to generate SNAP commands from the output of a DFSMSdss physical or logical DFSMSdss data set dump.
- Provided that you keep the SYSPRINT from the DFSMSdss step, you can run SNAPDSSU, repeatedly specifying different options.
- You can edit the SYSPRINT to include or exclude data sets.

  This process can be of useful if you are consolidating data to fewer volumes.
- Define aliases for SnapShot purposes only. This will help you to identify target data set versions easily. It also provides an effective way of moving data sets.
- You can use SNAPDSSU to move data to volumes, whether the source and target volumes are SMS or private volumes. Generate the SNAP commands, using SNAPDSSU to process source data set names, passing the SNAP HLQ as the new PREFIX. Snap the data sets. Then, run SNAPDSET a second time against the target data sets to re-create the source data set, specifying the original data set HLQ in the PREFIX option.

B.3 Sample SNAPDSSU Output

Output from SNAPDSSU is in the following format:

1. A summary of the options selected
2. A list of the data sets selected for processing
3. A summary of the data sets procesed by SNAPDSSU

SNAPDSSU provides informational (SIBIxxxx), warning (SIBWxxxx), and error (SIBExxxx) messages.

Generally, if a warning message is issued, the SNAPDSSU step completes with a condition code of 0004. If an error is encountered, SNAPDSSU terminates with a condition code of 0008.

SNAPDSSU output:

```
1READY
 The Target data set SNAPBACK(REPLACE) parameter was specified
 The Target data set REPLACE parameter was specified
 The Data set catalog  CATALOG(YES) was specified
 Tolerate truncation TOLTRUNC(NO) was specified
 Tolerate Enqueue failure TOLENQF(YES) was specified
 Host copy mode EXCL was specified
 A Management class of  MANC001 was specified
 No data mover was specified
 A volume count of  2  was specified
 Catalog checking was specified for BOTH data sets
 Prefix   : SNAPSHOT was specified
 TARGETVOL: 9RMDA2  was specified
 SIBI0011 - The following datasets were selected for processing:
 0
 0     NONVSAM  DB2B.DSNDB01.DBD01.ICF1L000.G00RRV00
 SIBW0021   -   Dataset:   DB2B.DSNDB01.DBD01.ICF1L000.G00RRV00   is   not
catalogued
 0     NONVSAM  DB2B.DSNDB01.DBD01.ICF1L000.G00TTV00
 SIBW0021   -   Dataset:   DB2B.DSNDB01.DBD01.ICF1L000.G00TTV00   is   not
catalogued
 0     NONVSAM  DB2B.DSNDB01.DBD01.ICF1L000.G00PPV00
 SIBW0021   -   Dataset:   DB2B.DSNDB01.DBD01.ICF1L000.G00PPV00   is   not
catalogued
 0     NONVSAM  DB2B.DSNDB01.DBD01.ICF1L000.G0035V00
 0     NONVSAM  DB2B.DSNDB01.DBD01.ICF1L000.G0036V00
 0     NONVSAM  DB2B.DSNDB01.SCT02.ICF1L000.G0032V00
 0        VSAM  DB2B.LANESRA.TROPPUS.G0001V00
 0        VSAM  DB2B.DSNDBC.EZKDB023.EZKTS231.I0001.A001
 0        VSAM  DB2B.DSNDBC.EZKDB023.EZKTS232.I0001.A001
 0        VSAM  DB2B.DSNDBC.EZKDB023.EZKTS233.I0001.A001
 0        VSAM  DB2B.LOGCOPY2.DS05
 0        VSAM  DB2B.LOGCOPY2.DS06
 0        VSAM  DB2B.LOGCOPY2.DS07
 0        VSAM  DB2B.LOGCOPY2.DS08
 0        VSAM  DB2B.LOGCOPY2.DS09
 0        VSAM  DB2B.LOGCOPY2.DS10
 0        ........
 0        ........
 0        ........
 0
 SIBI0012 - SNAP commands have been created for the following datasets:
 0
 0   Source data set name:  DB2B.LANESRA.TROPPUS.G0001V00
 0   Target data set name:  SNAPSHOT.LANESRA.TROPPUS.D97284.T70269
 0   Source data set name:  DB2B.LOGCOPY2.DS10
 0   Target data set name:  SNAPSHOT.LOGCOPY2.DS10.D97284.T70269
 0   Source data set name:  DB2B.LOGCOPY2.DS09
 0   Target data set name:  SNAPSHOT.LOGCOPY2.DS09.D97284.T70269
 0   Source data set name:  DB2B.LOGCOPY2.DS08
 0   Target data set name:  SNAPSHOT.LOGCOPY2.DS08.D97284.T70269
 0   Source data set name:  DB2B.LOGCOPY2.DS07
 0   Target data set name:  SNAPSHOT.LOGCOPY2.DS07.D97284.T70269
 0   Source data set name:  DB2B.LOGCOPY2.DS06
 0   Target data set name:  SNAPSHOT.LOGCOPY2.DS06.D97284.T70269
 0   Source data set name:  DB2B.LOGCOPY2.DS05
 0   Target data set name:  SNAPSHOT.LOGCOPY2.DS05.D97284.T70269
 0   Source data set name:  DB2B.LOGCOPY2.DS04
 SIBW0022 - Invalid data set length. Dataset name will be generate by
SNAPDSSU
          Review the generated JCL.
 0   Source data set name:  DB2B.DSNDBC.EZKDB023.EZKTS233.I0001.A001
 0   Target data set name:  SNAPSHOT.STEP20.D97284.T70269
 SIBW0022 - Invalid data set length. Dataset name will be generate by
SNAPDSSU
          Review the generated JCL.
 0   Source data set name:  DB2B.DSNDBC.EZKDB023.EZKTS232.I0001.A001
 0   Target data set name:  SNAPSHOT.STEP21.D97284.T70269
 SIBW0022 - Invalid data set length. Dataset name will be generate by
SNAPDSSU
          Review the generated JCL.
 0   Source data set name:  DB2B.DSNDBC.EZKDB023.EZKTS231.I0001.A001
 0   Target data set name:  SNAPSHOT.STEP22.D97284.T70269
 SIBW0022 - Invalid data set length. Dataset name will be generate by
SNAPDSSU
          Review the generated JCL.
 0   Source data set name:  DB2B.DSNDB01.SCT02.ICF1L000.G0032V00
 0   Target data set name:  SNAPSHOT.STEP336.D97284.T70269
 SIBW0022 - Invalid data set length. Dataset name will be generate by
SNAPDSSU
          Review the generated JCL.
 0   Source data set name:  DB2B.DSNDB01.DBD01.ICF1L000.G0036V00
 0   Target data set name:  SNAPSHOT.STEP337.D97284.T70269
 SIBW0022 - Invalid data set length. Dataset name will be generate by
SNAPDSSU
          Review the generated JCL.
 0   Source data set name:  DB2B.DSNDB01.DBD01.ICF1L000.G0035V00
 0   Target data set name:  SNAPSHOT.STEP338.D97284.T70269
 0   ........
 0   ........
 0   ........
 0
 SIBI0013 - Data set filtering is complete. 338  of 341  data sets were processed
```

```
0
0
SIBI0010 - SNAP back commands have been created for the datasets selected
0
```

B.4 SNAPDSSU REXX EXEC

```
/*--REXX exec -----------------------------------------------------\
| Exec       : SNAPDSSU                                             |
|              Creates Snap commands datasets                       |
|-------------------------------------------------------------------|
|COPYRIGHT: LICENSED MATERIALS - PROPERTY OF IBM                    |
|                                                                   |
|           THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM"     |
|                                                                   |
|           xxxx-xxx (C) COPYRIGHT IBM CORPORATION 1997, 1997       |
|           ALL RIGHTS RESERVED                                     |
|                                                                   |
|           US GOVERNMENT USERS RESTRICTED RIGHTS -                 |
|           USE, DUPLICATION OR DISCLOSURE RESTRICTED               |
|           BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.             |
|                                                                   |
|           SEE IBM COPYRIGHT INSTRUCTIONS.                         |
|-------------------------------------------------------------------|
| Author     : Richard Moore                                        |
|-------------------------------------------------------------------|
| Version    : 1.0                                                  |
|-------------------------------------------------------------------|
| Invocation : By batch                                             |
|-------------------------------------------------------------------|
| Overview   : This program processes the SYSPRINT of a ADRDSSU     |
|              job run with the TYPRUN='NORUN' parameter.           |
|              SNAPDSSU generates two sequential datasets:          |
|                                                                   |
|              Dataset 1: Contains the source to target SNAP        |
|                                                                   |
|                         The program provides optional parameter   |
|                         to:                                       |
|                             a) Tailor the Target data set name    |
|                             b) Override the default options       |
|                                of the SNAP command.               |
|                                                                   |
|                                Note: If no option is passed to    |
|                                      SNAPDSSU or a blank is       |
|                                      passed in the parameter      |
|                                      eg, REPLACE() the option     |
|                                      will be included in the      |
|                                      SNAP command generated.      |
|                                                                   |
|                                                                   |
|              Dataset 2: Contains the SNAP-back data set commands  |
|                         with the (Optional) REPLACE parameter     |
|                                                                   |
|-------------------------------------------------------------------|
| Parameters :                                                      |
|                                                                   |
|  blank = () :  Enables you to retain the SNAPDSSU option in the   |
|                JCL.                                               |
|  8 char     :  Free format.                                       |
|                                                                   |
|  PREFIX(blank|8 char)                                             |
|  TARGETVOL(volser) -                                              |
|  SECLVL(blank|8 char|UNIQUE)                                      |
|  SUFFIX(blank|8 char|UNIQUE)                                      |
  |  CATCHECK(blank|SOURCE|SRC|TARGET|TGT|BOTH)                      |
  |  VOLUMECOUNT(blank|integer)                                      |
  |  ESOTERIC(blank|8 char)                                          |
  |  DATAMOVERNAME(blank|NONE|DSS|DFDSS|ADRDSSU)                     |
  |  DATACLASS(blank|8 char)                                         |
  |  STORCLASS(blank|8 char)                                         |
  |  MANAGEMENTCLASS(blank|8 char)                                   |
  |  HOSTCOPYMODE(blank|EXCLUSIVE|EXCL|SHARED|SHR)                   |
  |  TOLERATEENQFAILURE(blank|YES|NO)                                |
  |  TOLERATETRUNCATION(blank|YES|NO)                                |
  |  CATALOG(blank|YES|NO)                                           |
  |  REPLACE(blank|YES|NO)                                           |
  |  SNAPBACK(blank|YES|NO|REPLACE)                                  |
  |                                                                  |
  |------------------------------------------------------------------|
  | GlobalVs   : None                                                |
  |------------------------------------------------------------------|
  | Change Log : None                                                |
  \-----------------------------------------------------------------*/

 /*    Trace ?r  */
Parse UPPER Arg vars

/*******************************************************************/
/*  Initialize variables                                           */
/*******************************************************************/
rrc = 0
finrrc = 0
total = 0
I = 0
prefix = ''
suffix = ''
tarvol = ''
seclvl = ''
volcnt = ''
dsvar = ''
eso = ''
dmo = ''
dat = ''
sto = ''
manc = ''
hos = ''
tole = ''
tolt = ''
catl = ''
repl = ''
tname = ''
dsname = ''
dstshd = ''
dttm = 'D'DATE(j)'.T'time(s)


/*******************************************************************/
/*  Check option variables                                         */
/*******************************************************************/
varctr = WORDS(vars)
If varctr = 0 then
Do
   say 'SIBE0030 - SNAPNVSM was invoked without parameters'
   Exit 8
end
 c = varctr
 Do until c = 0
  Parse var vars
      prmvar = WORD(vars,c)
      prmvar = STRIP(prmvar,b)
      lenparm = LENGTH(prmvar)
      If c = varctr then
      Do
        If substr(prmvar,lenparm,lenparm) ¬= ')' then
        do
              say 'SIBE0031 - Unmatched parentheses or missing' ||,
                  ' continuation character.'
              say 'Correct syntax'
          Exit 8
        end
        else
          If lenparm > 6 then
          do
           If substr(prmvar,lenparm - 1,lenparm) = '))' then
           do
           prmvar = substr(prmvar,1,lenparm - 1)
           end
          end
         If c = varctr & prmvar = ')' |,
            c = varctr & prmvar = ' ' then
         do
            prmvar = 'endsix'
            lenparm = LENGTH(prmvar)
         end
      end
      If c = 2 & prmvar = '(' then parf = 'fnd'
      If c = 1 then
      Do
        If substr(prmvar,1,7) = 'DATASET' & parf = 'fnd'
        then prmvar = 'DATASET('
        If substr(prmvar,1,8) ¬= 'DATASET(' then
        do
              say 'SIBE0031 - Unmatched parentheses or missing' ||,
                  ' continuation character.'
              say 'Correct syntax'
          Exit 8
        end
        else  prmvar = substr(prmvar,9,lenparm)
         If c = 1 & prmvar = ' ' then
         do
            prmvar = 'sixsix'
            lenparm = LENGTH(prmvar)
         end
      end
    If lenparm < 6 then
    Do
      If prmvar = '(' & c = 2 then NOP
      else call errmsg
    end
    call varint
 c = c - 1
end
If prefix = '' & suffix = '' & seclvl = ''  then
   Do
      say 'SIBE0032 - SNAPDSSU must be invoked with either the' ||,
          ' PREFIX, SECLVL or'
      say ' SUFFIX option specified. '
      rrc = 8
      exit rrc
   end

/*******************************************************************/
/*  Process the DSS SYSPRINT for data set names                    */
/*******************************************************************/

   call openfile

/*******************************************************************/
/*  Create the SNAPTO JCL                                          */
/*******************************************************************/
   call process

    If stpnum ¬= 0 then stpnum =  stpnum - 1
    say 'SIBI0013 - Data set filtering is complete.' stpnum ' of ' ||,
        total ' data sets were processed'
     say '0          '


/*******************************************************************/
/*  Create SNAPBACK JCL                                            */
/*******************************************************************/

If snab = 'YES' | snab = 'REPLACE' then
Do
     say '0          '
    say 'SIBI0010 - SNAP back commands have been created for the' ||,
        ' datasets selected'
     say '0          '
end

/*******************************************************************/
/*  Add additional JCL to SNAPTO job deck                          */
/*******************************************************************/

ddinfo = LISTDSI("addtojob" "file")

If sysreason = 0 then
Do
   " EXECIO * DISKR ADDTOJOB (STEM addto_LINES. FINIS) "
     rrc = RC
     If rrc = 0 then
     Do h = 1 TO addto_LINES.0
         Parse var addto_lines.h tojcl
         output = tojcl
         PUSH output
         "EXECIO 1 DISKW OUTPUT "
     end
     Else Exit sysreason
end
```

```
/*******************************************************************/
/*  Add additional JCL to SNAPBACK job deck                        */
/*******************************************************************/

ddinfo = LISTDSI("addbajob" "file")

If sysreason = 0 then
Do
    " EXECIO * DISKR ADDBAJOB (STEM addba_LINES. FINIS) "
      rrc = RC
      If rrc = 0 then
      Do q = 1 TO addba_LINES.0
          Parse var addba_lines.q bajcl
          output = bajcl
          PUSH output
          "EXECIO 1 DISKW OUTPUT2 "
      end
     Else Exit sysreason
end

/*******************************************************************/
/*  The Exit                                                       */
/*******************************************************************/

EXIT finrrc


/*******************************************************************/
/*  Interpret parameters passed by options                         */
/*******************************************************************/
varint:
If prmvar = 'sixsix' | prmvar = 'endsix' then
Do
/*If prmvar = 'endsix' then c = c - 1 */
  parmvar = ''
  return rrc
end
   pchk = VERIFY(')',prmvar,m)
   qchk = VERIFY('(',prmvar,m)
   If pchk ¬= 1 | qchk ¬= 1 then call errmsg
   prmvar = STRIP(prmvar,t)
   evar = LASTPOS(')',prmvar)
   svar = POS('(',prmvar)
   If svar + 1 = evar then return rrc
Select
    When SUBSTR(prmvar,1,6) = 'PREFIX' then
    Do
       prefix = SUBSTR(prmvar,svar +1,evar - svar - 1)
       ssuf = LENGTH(prefix)
       If ssuf > 8 then call errmsg
      /* check to see if alias exists   */
       x = OUTTRAP('LINEOUT1.')
      " LISTCAT ENT('"prefix"') "
      if rc ¬= 0 then
      Do
          say 'SIBE0033 - prefix' prefix  'does not exits'
          exit 8
      end
   y = OUTTRAP('OFF')
   say 'Prefix   :' prefix 'was specified'
end
When SUBSTR(prmvar,1,6) = 'SUFFIX' then
Do
   suffix = SUBSTR(prmvar,svar +1,evar - svar - 1)
   lsuf = LENGTH(suffix)
   If lsuf > 8 then call errmsg
   If suffix = 'UNIQUE' then
   Do
       suffix = 'D'DATE(j)'.T'time(s)
   end
   say 'Suffix   :' suffix  'was specified'
end
When SUBSTR(prmvar,1,6) = 'SECLVL' then
Do
   seclvl = SUBSTR(prmvar,svar +1,evar - svar - 1)
   lsec = LENGTH(seclvl)
   If lsec > 8 then call errmsg
   If seclvl = 'UNIQUE' then
   Do
   say 'The second level qualifier for all data sets will' ||,
       ' be replaced with the job step number'
   end
   Else
        say 'The second level qualifier for all data sets will' ||,
        ' be replaced with:  ' seclvl
end
When SUBSTR(prmvar,1,6) = 'TARGET' then
Do
   tarvol = SUBSTR(prmvar,svar +1,evar - svar - 1)
   lvol = LENGTH(tarvol)
   If lvol > 6 then call errmsg
   say 'TARGETVOL:' tarvol ' was specified'
end
When SUBSTR(prmvar,1,6) = 'CATCHE' then
Do
   catcheck = SUBSTR(prmvar,svar +1,evar - svar - 1)
   If catcheck = 'SOURCE' | catcheck = 'SRC' |,
   catcheck = 'TARGET' | catcheck = 'TGT' |,
   catcheck = 'BOTH' then
   Do
     say 'Catalog checking was specified for 'catcheck 'Data sets'
   end
   Else
      If catcheck = 'NO' | catcheck = '' then
      say 'No catalog checking was specified'
        Else call errmsg
end
When SUBSTR(prmvar,1,6) = 'VOLUME' then
Do
    volcnt = SUBSTR(prmvar,svar +1,evar - svar - 1)
    If DATATYPE(volcnt,N) = 1 then
    Do
       say 'A volume count of ' volcnt ' was specified'
       vc = 'y'
    end
    Else
        Do
```

```
           If DATATYPE(volcnt,N) ¬= 1 then call errmsg
        end
  If vc = '' then say 'No volume count was specified'
end
When SUBSTR(prmvar,1,6) = 'ESOTER' then
Do
   eso = SUBSTR(prmvar,svar +1,evar - svar - 1)
    If DATATYPE(eso,n) ¬= 1 then
   Do
     say 'Esoteric ' eso ' was specified'
   end
   Else call errmsg
end
When SUBSTR(prmvar,1,6) = 'DATAMO' then
Do
   dmo = SUBSTR(prmvar,svar +1,evar - svar - 1)
   If dmo = 'DSS' | dmo = 'DFDSS' | dmo = 'ADRDSSU' then
   Do
     say 'Data mover ' dmo 'was specified'
   end
   Else
       If dmo = 'NONE' then
       Do
          say 'No data mover was specified'
       end
       Else call errmsg
end
When SUBSTR(prmvar,1,6) = 'DATACL' then
Do
   dat = SUBSTR(prmvar,svar +1,evar - svar - 1)
    If DATATYPE(dat,n) ¬= 1 then
   Do
     say 'A data class of ' dat ' was specified'
   end
   Else call errmsg
end
When SUBSTR(prmvar,1,6) = 'STORCL' then
Do
   sto = SUBSTR(prmvar,svar +1,evar - svar - 1)
    If DATATYPE(sto,n) ¬= 1 then
   Do
     say 'A Storage class of ' sto ' was specified'
   end
   Else call errmsg
end
When SUBSTR(prmvar,1,6) = 'MANAGE' then
Do
   manc = SUBSTR(prmvar,svar +1,evar - svar - 1)
    If DATATYPE(manc,n) ¬= 1 then
   Do
     say 'A Management class of ' manc ' was specified'
   end
   Else call errmsg
end
When SUBSTR(prmvar,1,6) = 'HOSTCO' then
Do
   hos = SUBSTR(prmvar,svar +1,evar - svar - 1)
   If hos = 'SHARED' |  hos = 'SHR' |  hos = 'EXCLUSIVE' |,
      hos = 'EXCL' then
   Do
     If hos = 'SHR' then hos = 'SHARED'
     If hos = 'EXCL' then hos = 'EXCLUSIVE'
     say 'Host copy mode' hos 'was specified'
   end
   Else call errmsg
end
When SUBSTR(prmvar,1,4) = 'TOLE' then
Do
    If SUBSTR(prmvar,1,18) = 'TOLERATEENQFAILURE' |,
       SUBSTR(prmvar,1,7) = 'TOLENQF' then
    Do
       tole = SUBSTR(prmvar,svar +1,evar - svar - 1)
       If tole = 'YES' | tole = 'NO' then
       Do
         say 'Tolerate enqueue failure TOLENQF('tole')' ||,
             ' was specified'
       end
       Else call errmsg
    end
    If SUBSTR(prmvar,1,18) = 'TOLERATETRUNCATION' |,
         SUBSTR(prmvar,1,6) = 'TOLTRUNC' then
    Do
       tolt = SUBSTR(prmvar,svar +1,evar - svar - 1)
       If tolt = 'YES' | tolt = 'NO' then
       Do
         say 'Tolerate truncation TOLTRUNC('tolt')' ||,
             ' was specified'
       end
       Else call errmsg
    end
end
When SUBSTR(prmvar,1,6) = 'CATALO' then
Do
   catl = SUBSTR(prmvar,svar +1,evar - svar - 1)
   If catl = 'YES' | catl = 'NO' then
   Do
     say 'The Data set catalog  CATALOG('catl')' ||,
         ' was specified'
   end
   Else call errmsg
end
When SUBSTR(prmvar,1,6) = 'REPLAC' then
Do
   repl = SUBSTR(prmvar,svar +1,evar - svar - 1)
   If repl = 'YES' | repl = 'NO' then
   Do
     say 'The Target data set REPLACE parameter' ||,
         ' was specified'
   end
   Else call errmsg
end
When SUBSTR(prmvar,1,6) = 'SNAPBA' then
Do
   snab = SUBSTR(prmvar,svar +1,evar - svar - 1)
   If snab = 'YES' | snab = 'REPLACE' | snab = 'NO' then
   Do
     say 'The Target data set SNAPBACK('snab') parameter' ||,
         ' was specified'
```

```
        end
        Else call errmsg
     end
     otherwise call errmsg
end
return rrc

/*******************************************************************/
/*  Procedure to interpret DSS SYSPRINT                            */
/*******************************************************************/
Openfile:
/*    retrieve records from SYSPRINT       */

" EXECIO * DISKR INPUT (STEM MST_LINES. FINIS) "
If RC ¬= 0 then
 Do
   say 'SIBE0034 - error retrieving SYSPRINT output'
 end

/*    Process the SYSPRINT record            */

Do p = 1 TO mst_LINES.0
 cluster = ''
 Parse var mst_lines.p file_msg file_rest
   If p = 2 then
   Do
     If substr(file_msg,2,8) ¬= 'ADR031I' then
     Do
         say 'SIBE0035 - No valid DFSMSdss output was detected'
         EXIT 8
     end
   end
   If substr(file_msg,1,7) = 'ADR129E' then
   Do
       say 'SIBE0036 - Invalid KEYWORD specified in DFSMSdss parms'
       EXIT 8
   end
   If substr(file_msg,2,8) = 'ADR415W' then
   Do
       say 'SIBW0020 - No datasets were selected for processing'
       EXIT 4
   end
   If substr(file_msg,2,8) = 'ADR475I' then
   Do
       say 'SIBI0011 - The following datasets were selected for' ||,
          ' processing:'
       say '0          '
   end

/*  Strip the leading and trailing blanks from second stem variables */

      file_rest = STRIP(file_rest,b)

/*    Set variables                     */
lnepass = ''
/*    Obtain the ds names               */

   If WORD(file_msg,1) = 0 | WORD(file_msg,1) = '-' then
   Do
```

```
        If WORD(file_rest,1) = 'CLUSTER' then
        Do
           total = total + 1
           dsname =  WORD(file_rest,3)
           say '0         VSAM '   dsname
           chk = 'yes'
        end
        else
           If WORD(file_rest,1) = 'CATALOG'  |,
              WORD(file_rest,1) = 'COMPONENT'  then lnepass = pass
           else
           Do
                total = total + 1
                dsname  =  WORD(file_rest,1)
                say '0      NONVSAM '   dsname
                chk = 'yes'
           end

           /* check to see if dataset exists   */

           catex = ''

           If catcheck = 'SOURCE' & chk = 'yes' |,
              catcheck = 'BOTH' & chk = 'yes' then
              Do
                dsvar = 's'
                call catchk dsname dsvar
                dsvar = ''
              end
           If catex = "" & lnepass = '' then
           Do
                i = i + 1
                record.i = dsname
           end
   end
   chk = ''
end
return rrc

/*********************************************************************/
/*  General parm error message                                       */
/*********************************************************************/
Errmsg:
Do
   say 'SIBE0037 - Invalid parameter specified' prmvar
   Exit 8
end
return rrc

/*********************************************************************/
/*  Check catalog status                                             */
/*********************************************************************/
catchk:
If dsvar = 's' then
Do
x = OUTTRAP('LINEOUT1.')
   " LISTCAT ENT('"dsname"') "
   rrc = rc
   If rrc ¬= 0 then
   Do
     catex = 'no'
     say 'SIBW0021 - Dataset:' dsname  'is not catalogued'
     dsname = ''
   end
   y = OUTTRAP('OFF')
End
If dsvar = 't' then
Do
x = OUTTRAP('LINEOUT1.')
   " LISTCAT ENT('"tname"') "
   rrc = rc
   If rrc = 0 then
   Do
     say 'SIBW0023 - Data set:' tname  '.  A data set of the same' ||,
         ' name already exists'
     tname = ''
     rrc = 4
   end
   Else rrc = 0
   y = OUTTRAP('OFF')
End
   if rrc > 0 then finrrc = rrc
return rrc

/*****************************************************************/
/*  Process datasets passed from SYSPRINT                        */
/*****************************************************************/
process:
If dstshd = ''  then
Do
     say '0          '
    say 'SIBI0012 - SNAP commands have been created for the' ||,
        ' following datasets:'
     say '0          '
   dstshd = 'OK'
   end
 j = I
 stpnum = 1

 If stpnum = 1 then
 Do
     ddinfo = LISTDSI("tocard" "file")

     If sysreason = 0 then
     Do
     " EXECIO * DISKR TOCARD (STEM CRD_LINES. FINIS) "
        rrc = RC
        If rrc = 0 then
        Do s = 1 TO crd_LINES.0
           Parse var crd_lines.s card
            output = card
            PUSH output
            "EXECIO 1 DISKW OUTPUT "
        end
     end
     Else
         Do
            ddname = 'output'
           call snapcard ddname
        end
end

Do until j = 0
 Parse var record.j dsname
  ddname = 'output'
  outvar =  '//STEP'stpnum'   EXEC PGM=SIBBATCH'
  call io
  outvar =  '//SYSTERM  DD  SYSOUT=*'
  call io
  outvar =  '//SYSPRINT DD  SYSOUT=*'
  call io
  outvar =  '//SYSIN    DD  *'
  call io
  outvar =  ' SNAP DATASET( -'
  call io
  outvar =  '      SOURCE('dsname')  -'
  call io
  call cgname
  outvar =  '      TARGET('tname')  -'
  tname = ''
  call io

     If tarvol ¬= '' then
     Do
       outvar = '      VOLUME('tarvol')  - '
       call io
     end

     If volcnt ¬= '' then
     Do
       outvar = '      VOLUMECOUNT('volcnt')  - '
       call io
     end

     If eso ¬= '' then
     Do
       outvar = '      ESOTERIC('eso')  - '
       call io
     end

     If dmo ¬= '' then
     Do
       outvar = '      DATAMOVERNAME('dmo')  - '
       call io
     end

     If dat ¬= '' then
     Do
       outvar = '      DATACLASS('dat')  - '
       call io
     end

     If sto ¬= '' then
     Do
       outvar = '      STORCLASS('sto')  - '
       call io
     end
```

```
      If manc ¬= '' then
      Do
        outvar = '      MANAGEMENTCLASS('manc')  - '
        call io
      end

      If hos ¬= '' then
      Do
        outvar = '      HOSTCOPYMODE('hos')  - '
        call io
      end

      If tole ¬= '' then
      Do
        outvar = '      TOLERATEENQFAILURE('tole')  - '
        call io
      end

      If tolt ¬= '' then
      Do
        outvar = '      TOLERATETRUNCATION('tolt')  - '
        call io
      end

      If catl ¬= '' then
      Do
        outvar = '      CATALOG('catl')  - '
        call io
      end

      If repl ¬= '' then
      Do
        outvar = '      REPLACE('repl')  - '
        call io
      end

   outvar =  '                             )'
   call io
   outvar =  '/*'
   call io
  j = j - 1
  stpnum = stpnum + 1
 end

return rrc

/*******************************************************************/
/*  Write to output files                                          */
/*******************************************************************/
io:
  PUSH outvar
  "EXECIO 1 DISKW" ddname
return rrc

/*******************************************************************/
/*  Reconstruct the Target data set name                           */
/*******************************************************************/
cgname:
 qual = dsname
 qual = TRANSLATE(qual,' ','.')
 qnum = WORDS(qual)
 qctr = qnum
 Do while qctr ¬= 0
   qualf = WORD(qual,qctr)
   q.qctr = qualf
   qctr = qctr - 1
 end
 z = qnum
 p = 0
 tname = ''
 Do until z = 0
  Parse var q.z qualf
  if prefix ¬= '' & z = 1 then  qualf = prefix
  if z = 1 & prefix = ''  then hlq = qualf
  else hlq = prefix
  if seclvl ¬= '' & z = 2 then qualf = seclvl
  if seclvl = 'UNIQUE' & z = 2 then qualf = 'STEP'stpnum
  tname = qualf||' '||tname
  z = z - 1
 end
  if suffix ¬= '' then
  tname = tname||''||suffix
  tname = STRIP(tname,b)
 tname = TRANSLATE(tname,'.',' ')

    ntname = LENGTH(tname)
    If ntname > 44 then
    Do
        tname = hlq||'.STEP'stpnum'.'dttm
        say 'SIBW0022 - Invalid data set length. Dataset name' ||,
            ' will be generate by SNAPDSSU'
        say '           Review the generated JCL.'
        rrc = 4
        if rrc > 0 then finrrc = rrc
    end
           say '0    Source data set name: '   dsname
           say '0    Target data set name: '   tname

If catcheck = 'TARGET' | catcheck = 'BOTH' then
Do
    dsvar = 't'
    call catchk tname dsvar
    dsvar = ''
end

If snab = 'YES' | snab = 'REPLACE' then call snapback

return rrc

/*********************************************************************/
/*  Build SNAPBACK JCL                                               */
/*********************************************************************/
snapback:
 If stpnum = 1 then
 Do
     ddinfo = LISTDSI("backcard" "file")
```

```
      If sysreason = 0 then
      Do
      " EXECIO * DISKR BACKCARD (STEM CRD2_LINES. FINIS) "
         rrc = RC
         If rrc = 0 then
         Do h = 1 TO CRD2_LINES.0
            Parse var crd2_lines.h card2
             output2 = card2
             PUSH output2
             "EXECIO 1 DISKW OUTPUT2 "
         end
      end
      Else
          Do
             ddname = 'output2'
             call snapcard ddname
          end
  end

    ddname = 'output2'
    outvar = '//STEP'stpnum'   EXEC PGM=SIBBATCH'
    call io
    outvar = '//SYSTERM  DD  SYSOUT=*'
    call io
    outvar = '//SYSPRINT DD  SYSOUT=*'
    call io
    outvar = '//SYSIN    DD  *'
    call io
    outvar = ' SNAP DATASET( -'
    call io
    outvar = '      SOURCE('tname')  -'
    call io
    outvar = '      TARGET('dsname')  -'
    call io

       If snab = 'REPLACE' then
       Do
         outvar = '      REPLACE('YES')  - '
         call io
       end

    outvar = '                            )'
    call io
    outvar = '/*'
    call io

    ddname = 'output'

 return rrc

 /*****************************************************************/
 /*  Add jobcard to JCL                                           */
 /*****************************************************************/
 snapcard:
  parse var ddname
  If ddname = 'output' then
  outvar =   "//SNAPTO JOB (,),'SNAP TO JCL',"
  else if ddname = 'output2' then
       outvar =   "//SNAPBACK JOB (,),'DATASET SNAPBACK JCL',"
 PUSH outvar
 "EXECIO 1 DISKW "ddname
 outvar =   "//     CLASS= ,MSGCLASS= ,NOTIFY=&SYSUID "
 PUSH outvar
 "EXECIO 1 DISKW "ddname
return rrc
```

Appendix C. DB2 Overview

The information in this chapter has been extracted from the DB2 manual set to provide storage administrators and non database users with an overview of DB2. We make references to relevant topics in the DB2 manuals throughout the overview to assist those readers who want to research DB2 further.

C.1.1 The DB2 Subsystem

A DB2 subsystem comprises the following data sets:

C.1.1.1 Catalog and Directory Data Sets

DB2 maintains a set of tables called *DB2 catalog and directory tables* that store information about the DB2 system. The tables reside in the DB2 catalog and directory data sets. The DB2 catalog tables describe tables, columns, indexes, programs, authorizations, and other DB2 objects. DB2 automatically updates the DB2 catalog tables during normal operation and in response to certain Structured Query Language (SQL) statements. The system programmer creates the catalog and directory tables at installation time.

For more information about the DB2 catalog and directory tables, refer to Appendix D, “DB2 Catalog Tables,” in the *DB2 for OS/390 V5 Application Programming and SQL Guide*, SC26-8958.

C.1.1.2 Boot Strap Data Sets

The BSDS is a VSAM key-sequenced data set (KSDS) DB2 uses to record critical information. The BSDS contains, among other information:

- An inventory of all active and archive log data sets known to DB2
- A wrap-around inventory of all recent DB2 checkpoint activity

  DB2 records a relative byte address (RBA) when it begins and ends a checkpoint. If a request to stop DB2 has been made, DB2 records the checkpoints as a shutdown checkpoint.
- The distributed data facility (DDF) communication record, which contains the DB2 LOCATION name and virtual telecommunications access method (VTAM) logical unit (LU) name
- The highest RBA written

  During its operations, DB2 periodically records in the BSDS the RBA of the last log record written. This record can be interpreted as an approximation of the end of the log because the BSDS field is frequently being updated by DB2.
- Conditional restart control records

  The conditional restart control record is used to restart DB2, bypassing its normal recovery operations. A conditional restart can be performed in a development environment where application programs are being tested and data consistency is not critical. It is also used for disaster recovery and to recover DB2 from certain types of failures.
- VSAM catalog name

  The VSAM catalog name is the ICF catalog entry or the data set high level qualifier (HLQ) used by DB2.

DB2 maintains two copies of the BSDS. Typically, each data set is placed on a separate volume for maximum availability.

C.1.1.3 Active Log Data Sets

DB2 records all data changes and significant events in the active log data sets. The system programmer defines the active logs to the BSDS at installation time. To add to or reduce the number of active logs, you update the BSDS, using the change log inventory program, DSNJU003.

You can customize DB2 to process in single logging or dual logging mode. We recommend dual logging, as it provides added security when you recover DB2 data. DB2 will switch to an alternate log when it encounters errors on the first log.

DB2 continually writes log records to the active log data set until it becomes full and switches recording to the next log or pair of logs. DB2 then copies the contents of the active log to a DASD or magnetic tape data set called the *archive log*.

C.1.1.4 Archive Log Data Sets

The archive log can consist of up to 1000 data sets, each of which is a sequential data set (physical sequential) that resides on a DASD or magnetic tape volume. An archive log data set is created during the log off-load process (when an active log data set is copied to an archive log data set). During this process a copy of the BSDS is also written to a different sequential data set. The data sets are created with a date and time to indicate when the archive took place.

The format of an archive log data set is:

dsnhlq.ARCHLOG**x**.D**date**.T**time**.**ynnnnnnn**

where:

dsnhlq	The data set prefix specified at DB2 installation time
x	1 for the first copy of the logs and 2 for the second copy
date	System generated date indicating when the log was created (optional)
time	System generated time indicating when the log was created (optional)
y	**A** (for an active log) or **B** (for a BSDS archive log)
nnnnnnn	Represents the series of low-level qualifiers DB2 generates for archive log data set names, beginning with A0000001 and incrementing to A0000002, A0000003, and so forth.

In the following example DB2 archived on Monday 4th August at 01:01.362 in the morning. It is the first copy of the logs and the 972nd version to be created.

```
DB2B.ARCHLOG1.D97216.T0101362.A0000972
```

C.1.2 Table Spaces

You create table spaces using the SQL CREATE TABLESPACE statement.

A table space is one or more data sets in which one or more tables are stored. A table space can consist of from 1 to 64 VSAM data sets, which can together contain up to 64 gigabytes (GB) of data. A LARGE table space can consist of up to 254 data sets, or 1 terabyte of data. The data sets that DB2 manages (contained in DB2 storage groups) are linear data sets (LDSs). Table spaces are divided into equal-sized units, called *pages*, which are written to or read from DASD in one operation.

DB2 supports the following table space types:

Simple	A table space with more than one table. The space consists of pages, and each page can contain rows from many tables.
Segmented	A table space with more than one table. The space consists of groups of pages called segments. Each segment is dedicated to holding rows of a single table.
Partitioned	A table space with only a single table. The space is subdivided into partitions based on the key range of a nominated partitioning index. Each partition can be processed separately by utilities, possibly allowing concurrent access by other utilities and SQL statements.

When you create a table space, you can specify the database to which the table space belongs and the storage group it uses. If you do not specify the database and storage group, DB2 assigns the table space to the default database and the default storage group. You also control whether the table space is simple, segmented, or partitioned.

DB2 also provides support for multivolume table spaces and indexes.

For more information about table types, refer to Chapter 2-6, “Deciding What Type of Table Space and How Many,” in the *DB2 for OS/390 V5 Administration Guide*, SC26-8957.

C.1.3 Indexes

You create indexes using the SQL CREATE INDEX statement.

An index is an ordered set of pointers to the data in a DB2 table. The index is stored separately from the table. Each index is based on the values of data in one or more columns of a table. After you create an index, DB2 maintains the index, but you can check, repair, or reorganize it.

You can create an index on a table any time after you create the table, either before or after you load data into the table. Except for changes in performance, users of the table are unaware that an index is being used. DB2 decides whether or not to use the index to access the table.

Each index occupies its own index space, which can contain from 1 to 64 VSAM LDSs. When you create an index, an index space is automatically defined in the same database as the table.

Use indexes to:

- Improve performance. Indexed table spaces provide faster access to data than nonindexed table spaces.
- Ensure uniqueness. A table with a unique index cannot have two rows with the same values in the column or columns that form the index key.

DB2 supports the following index types:

Partitioned Indexes	An index created on a table in a partitioned table space is a partitioned index and is divided into multiple index spaces.
Clustering Indexes	A clustering index determines the approximate order in which records of the base table are stored. Therefore, DB2 can access the entire table in the sequence of the clustering key faster than in any other sequence. Each table can have only one clustering index.

C.1.4 Data Integrity

DB2 data sets are related, with values and pointers from indexes to data. DB2 has pointers inside the data set, in the DB2 catalog and directory data sets, and in the logs. Data integrity is of paramount importance to DB2 and must always be considered when working on DB2 data.

C.1.4.1 What Is Referential Integrity?

Referential integrity is the condition of a set of tables in which all references from one table to another are valid; referential integrity is the enforcement of all referential constraints.

A referential constraint can be a rule that a value of a foreign key must appear as a value of the primary key of some specific table. Foreign keys and primary keys can be defined as part of the SQL CREATE TABLE statement.

Having referential integrity does not mean that the data is necessarily correct. Undertaking such actions as loading or recovering DB2 tables can have an effect on data consistency, so any change you make must be carefully planned beforehand.

Recovery considerations are more complex for related tables such as those with referential constraints and require special handling. DB2 automatically enforces referential integrity during SQL insert, update, and delete operations and optionally enforces referential integrity during load operations. But it does not absolutely prevent you from recovering part of a set of interrelated tables to a previous point in time, making it inconsistent with the rest of the set. Such an operation produces abundant warning messages, but administration of the total recovery operation is still the user′s responsibility.

C.1.4.2 Recovering Tables Involved in a Referential Constraint

Point in time recovery can cause table spaces to be placed in check pending status if they have table check constraints or referential constraints defined on them. When recovering tables involved in a referential constraint, you should recover all the table spaces involved in a constraint. This is the table space set. To avoid setting check pending, you must do both of the following:

- Recover the table space or the table space set to a quiesce point or to an image copy made with SHRLEVEL REFERENCE.

  If you do not recover each table space of the table space set to the same quiesce point, and if any of the table spaces are part of a referential integrity structure:

  – All dependent table spaces that are recovered are placed in check pending status with the scope of the whole table space.

  – All dependent table spaces of the above recovered table spaces are placed in check pending status with the scope of the specific dependent tables.

- Do not add table check constraints or referential constraints after the quiesce point or image copy.

If you recover each table space of a table space set to the same quiesce point, but referential constraints were defined after the quiesce point, then the check pending status is set for the table space containing the table with the referential constraint.

Use the DB2 REPORT TABLESPACESET utility to list the names of all table spaces in the set; see 4.5.3, “ICF Catalogs” on page 60.

For more information about referential integrity, refer to Chapter 2-3, “Defining Referential Constraints,” in the *DB2 for OS/390 V5 Administration Guide*, SC26-8957.

C.1.5 Data Set Placement

Application table spaces and indexes are created in databases associated with a DB2 storage group, otherwise known as a *STOGROUP*.

Note: For the purposes of clarity, we refer to a DB2 storage group as a *STOGROUP* and a Storage Management Subsystem (SMS) storage group as an *SMS storage group* throughout this chapter.

You create the database and STOGROUP by using the CREATE DATABASE and CREATE STOGROUP SQL statements. The STOGROUP is used to support DASD space requirements for table spaces and indexes within the database. The STOGROUP identifies a list of DASD volumes that can be used for allocation of table spaces and indexes. STOGROUPs can also be set up to pass data placement control to SMS.

The VCAT parameter of the CREATE STOGROUP statement identifies the ICF catalog entry for the STOGROUP. Storage administrators usually refer to the VCAT as a catalog alias or HLQ. Any table space or index you create by using the STOGROUP will have the same VCAT.

C.1.6 DB2 Buffer Pools

Figure 103 on page 174 illustrates the types of storage used by DB2.

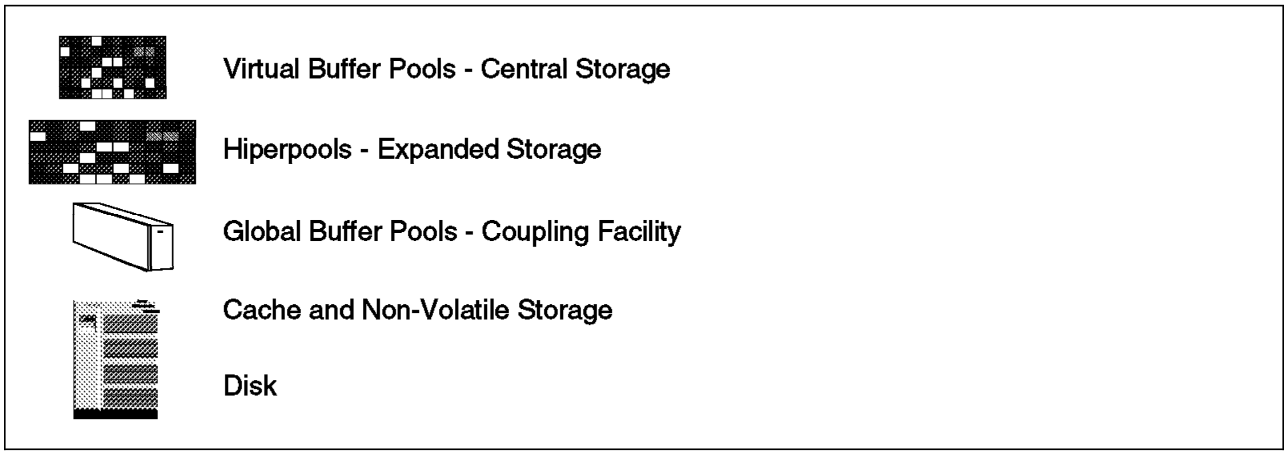

Virtual Buffer Pools - Central Storage
Hiperpools - Expanded Storage
Global Buffer Pools - Coupling Facility
Cache and Non-Volatile Storage
Disk

Figure 103. Storage Hierarchy

Buffer pools, also known as *virtual buffer pools*, are areas of virtual storage used temporarily to store pages of table spaces or indexes. When an application program needs to access a row of a table, DB2 retrieves the page containing that row and places the page in a buffer. If the row is changed, the buffer must be written back to the table space. If the needed data is already in a buffer, the application program will not have to wait for it to be retrieved from DASD.

DB2 supports a second level of storage, the *hiperpool*, for each virtual buffer pool. A hiperpool is an extension to the virtual buffer pool. Virtual buffer pools hold the most frequently accessed data. Data in virtual buffer pools that is not accessed frequently can be moved to its corresponding hiperpool.

DB2′s use of a virtual buffer pool or hiperpool is governed by several preset values called *thresholds*. Each threshold is a level of use which, when exceeded, causes DB2 to take some action. When you reach some thresholds, it indicates a problem, while reaching other thresholds merely indicates normal buffer management. The level of use is usually expressed as a percentage of the total size of the virtual buffer pool or hiperpool. For example, the *immediate write threshold* of a virtual buffer pool is set at 97.5%; when the percentage of unavailable pages in a virtual buffer pool exceeds that value, DB2 writes pages to DASD when updates are completed.

As DB2 is used, there are a number of thresholds for data in the buffers. If the percentage of changed pages for a single data set passes a threshold, that data is written. If the percentage of pages holding changed data in the buffer pool passes another threshold, the changed pages are written. If a DB2 subsystem checkpoint occurs, all of the changed pages are written. If a table space is closed, that data is written to DASD.

DB2 can use large buffers to avoid writing data to DASD and to make the writing more efficient. The objectives are to process multiple updates on a page before writing out the page and to accumulate many pages so that they can be written together in one sweep across the disk, rather than in separate I/Os. The writes are limited to a range of 150 blocks, so that the I/O response time is reasonable. Up to 32 blocks can be written with a single write. Most of the data can be written asynchronously from the applications. This technique makes data recovery more complex but is a benefit for performance.

When dealing with DB2 data it is important to understand that as DB2 is used, updated pages can remain in storage. If you take a DFSMSdss backup or

SnapShot of the DB2 data sets, any updates in the buffer pools will not be included in the backup. If used in a recovery, DB2 would encounter integrity problems with data restored from such a copy.

Another level of buffer pools is the group buffer pool, associated with the DB2 data sharing environment.

The sizes of buffer pools and the thresholds can be adjusted dynamically while DB2 is running.

For more information about buffer pools, refer to *Data Sharing: Planning and Administration*, SC26-8961 and the *DB2 for OS/390 V5 Administration Guide*, SC26-8957.

C.1.7 DB2 Utilities

There are two types of DB2 utilities: *online utilities* and *stand-alone utilities*. In this section we describe some of the utilities you will use to manage DB2 data.

C.1.7.1 DB2 Copy Utilities

You can take a backup of a table space using the DB2 utilities. DB2 records information about the image copy in the SYSCOPY table of the DB2 catalog. DB2 uses this information during recovery so that it can automatically use the most recent image copy and apply records from the log.

The DB2 image copy capabilities are:

COPY The COPY online utility runs only when DB2 is active and can create consistent (SHARELEVEL REFERENCE) and fuzzy (SHRLEVEL CHANGE) copies of individual DB2 table spaces. You can create a full image copy that takes a copy of all data in a table space or create incremental copies which contain only updates made since the last image copy. DB2 stores information in the catalog about image copies created with the COPY utility and uses those copies during recovery.

You can invoke concurrent copy to make a full image copy using the CONCURRENT control statement of the COPY utility.

DSN1COPY The DSN1COPY utility is a stand-alone utility that takes a consistent copy of a DB2 VSAM data set. DSN1COPY runs as a separate job outside the control of DB2 and can run whether or not DB2 is active. However, if DB2 is active, you must start the database concerned as read only for the duration of the copy. DB2 does not record information in the catalog about copies taken with DSN1COPY, and consequently the RECOVER utility does not automatically use copies taken with DSN1COPY.

C.1.7.2 DB2 QUIESCE Utility

You can take a quiesce point for a table space, partition, or list of table spaces with the DB2 QUIESCE utility. DB2 writes changed pages from the table spaces to DASD and records the current RBA and the timestamp of the quiesce point in DB2 catalog table SYSCOPY. For example, you use the QUIESCE utility to obtain a common quiesce point for a set of referentially related table spaces to give you a point in time where the data is consistent. You use this RBA to recover a table space to the point of consistency using the RECOVER utility and the TORBA option.

C.1.7.3 DB2 RECOVER Utility

The RECOVER utility can be used to recover data to the current time or a previous point-in-time and to apply only the logs to a previously restored version of data.

With:

RECOVER TABLESPACE	You can recover an entire table space, a partition or data set, pages within an error range, or a single page.
RECOVER INDEX	You can recover indexes by recreating them from the tables on which they are based.

For more information about the DB2 online utilities, refer to the respective chapters in the *DB2 for OS/390 V5 Utility Guide and Reference*, SC26-8967.

C.1.8 Data Set Naming Convention

DB2 data sets use a specific naming convention. The following example is the naming convention for DB2 Version 4:

catname.DSNDB**x.dbname.psname.**I0001.A**nnn**

where:

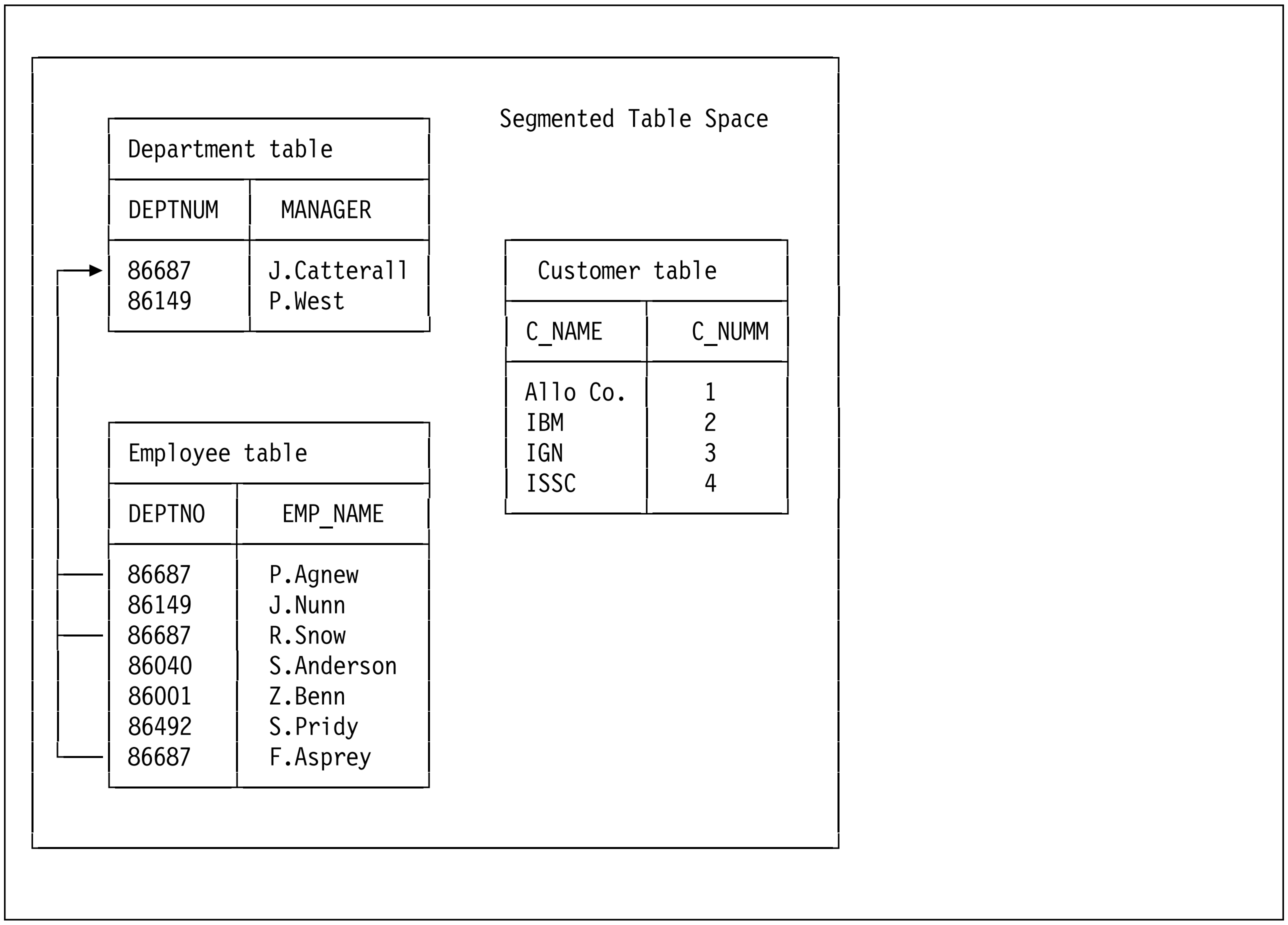
catname	VCAT used by DB2
x	Used by DB2 to differentiate between the cluster and data portions of the LDS. **C** for VSAM clusters or **D** for VSAM data components.
dbname	The database name. If the data set is for a table space, dbname must be the name given in the CREATE TABLESPACE statement. If the data set is for an index, dbname must be the name of the database containing the base table.
psname	The table space or index name. This name must be unique within the database. You will use this name on the CREATE TABLESPACE or CREATE INDEX statement. (You can use a name longer than eight characters on the CREATE INDEX statement, but the first eight characters of that name must be the same as in the data set′s psname.)
nnn	The data set number. For partitioned table spaces, the number is 001 for the first partition, 002 for the second, and so forth, up to the maximum of 64 partitions. For simple or segmented table spaces, the number is 001 for the first data set.

C.1.9 Recovery Example

C.1.9.1 Recovery Scenario

Figure 104 shows a segmented table space that we will use to illustrate one way in which DB2 data sets are related. We use a badly designed database to highlight the DB2 activity and demonstrate what problems you face if you choose to ignore data recovery during the design phase.

Details of the actions DB2 undertakes are not exhaustive; we merely want to bring to the attention of nondatabase users the complex relationship DB2 has with its data.



Department table

DEPTNUM	MANAGER
86687	J.Catterall
86149	P.West

Employee table

DEPTNO	EMP_NAME
86687	P.Agnew
86149	J.Nunn
86687	R.Snow
86040	S.Anderson
86001	Z.Benn
86492	S.Pridy
86687	F.Asprey

Customer table

C_NAME	C_NUMM
Allo Co.	1
IBM	2
IGN	3
ISSC	4

Figure 104. DB2 Data Integrity: Example of a Segmented Table Space

A database has been set up using a segmented table space. It contains the following tables:

- Employee
- Department
- Customers

The employee and department tables have been designed with a many-to-one relationship, so that many employees on the employee table can have the same value in the DEPTNO column. The department has an identical DEPTNUM, listing all the departments in the organization. Each department must be unique, so each row in the table must have a value in DEPTNUM that is also unique; and, if a department is deleted, all rows on the employee table relating to that DEPTNUM will be deleted also. The customer table is a list of all the company′s customers. Other than sharing the same table space, it is independent of the other two tables.

In DB2 terms, we have introduced a referential constraint to manage the data, where the rule is that every department number shown in the sample employee table must appear in the department table. In our example, the department table is a parent table, and its column of unique identifiers (DEPTNUM) is a primary key. The employee table is a dependent table, and the column of employee department assignments (DEPTNO) is a foreign key. DB2 will automatically enforce the constraint as the primary and foreign keys are explicitly defined.

C.1.9.2 Recovery from User Error

During a week′s processing the following events occur:

1. On Sunday the database is started in read only (RO) mode with the -STA DATABASE command, to ensure that no updates are made.

2. The table space is quiesced using the DB2 QUIESCE utility with the WRITE YES option.

   DB2 establishes a quiesce point and writes all changed pages (held in the buffer pools) from the tables spaces to DASD.

3. A full image copy (FIC) is taken using the COPY tablespace utility and the SHRLEVEL REFERENCE control statement.

   Share level reference allows read only access to the data while it is copied. The data is consistent at the moment the copying starts and remains consistent until the copying ends, giving you a point of consistency when recovering.

   DB2 records the FIC and quiesce RBA in the SYSCOPY catalog table storing the current RBA and the timestamp of the quiesce point. At that point, neither table space contains any uncommitted data. A row with ICTYPE Q is inserted into SYSCOPY for each quiesce, and a row with ICTYPE F for a full image copy.

4. The table space is restarted in read/write (RW) mode, to allow normal access to continue.

5. On Monday the employee table is updated with two new employees reporting to department 86687.

   DB2 records the updates in the log. DB2 also registers the log RBA ranges used during each period of time when the table space was open for update in SYSLGRNX, a directory table.

6. On Tuesday a second full image copy is taken. This time the SHRLEVEL CHANGE option is specified. A quiesce is taken to provide a point of consistency for recovery. The RBA for the second quiesce is X′0000001D709F′.

   Share level change allows read/write access to the data while it is copied. The data is not consistent, as uncommitted data might be copied.

7. On Thursday department 86687 is deleted by accident by a user of the online application used to update the tables. As a result of this action the delete cascades to the employee table and all rows containing 86687 in the DEPTNO column are deleted.

Again, this activity is recorded in the logs.

8. On Wednesday the customer table is updated with two new customers.

9. On Friday the database administrator (DBA) discovers the problem. After discussion with the application owner, the decision is made to recover the table space to the quiesce point just after the second full image copy. The DBA also discovers at this time that recovery to a prior point in time will regress the customer table. In this situation the application owner is instructed to unload the data from the customer table, as the new updates will be lost in the recovery, and advised to move the table to its own table space, to avoid making recovery more complex than necessary. The DBA then recovers the table space back to the full image copy, using the RECOVER TABLESPACE and TORBA option, specifying X′000001D709F′ as the recovery point. After recovering the table space, the DBA recovers all related indexes, using the RECOVER INDEX utility, as the TORBA has put all related indexes into a recover pending (RECP) state.

   Typically, RECOVER restores an object to its current state by applying all image copies and log records. In our example RECOVER uses the full image copy taken on the Wednesday and the logs (active or archive) up until the quiesce point. Had the table space been physically corrupted, the recovery requirement would have been to restore the data to the point of failure. In this case, DB2 would use the full image copy and apply all log records. DB2 would use SYSLGRNX to limit the log information that must be scanned to apply changes to a table space or partition being recovered by using only the logs where the table space was open for update. Recovery time is reduced, and unnecessary DFSMShsm recall activity for migrated archive logs is eliminated.

For more information about backup and recovery, refer to Chapter 4-6, “Backing Up and Recovering Databases,” in the *DB2 for OS/390 V5 Administration Guide*, SC26-8957.

C.1.10 DB2 Backup and Recovery Concepts

In this section we describe the disaster backup and recovery considerations for DB2 in a non-data-sharing environment.

C.1.10.1 Backup and Recovery

DB2 provides means for recovering data to its current state or to an earlier state. The units of data that can be recovered are table spaces, partitions, and data sets.

Backup of DB2 data outside its control, known as *offline*, normally takes the form of a physical volume dump using DFSMSdss. Within DB2 a full, concurrent, or incremental image copy can be taken using the DB2 COPY utility.

The choice of recovery approach is selected based on the needs of the business. Recovery time, or the time taken to resume service, are key factors when determining a suitable recovery point of consistency (POC). Recovery can be to a point in time one week ago or up to the point of failure, achieving as near 100% data currency with minimal processing loss.

After DB2 is installed, backup and recovery procedures are developed for onsite and remote site disaster recovery, to avoid costly and time-consuming losses of data. These procedures will take into account both DB2 subsystem and application data, to include:

- Creating a point of consistency
- Backing up the DB2 catalog and directory and your data
- Recovering the DB2 catalog and directory and your data

More general recovery procedures will cover:

- Recovering from out-of-space conditions
- Restoring system and data objects to a point of consistency
- Recovering from an MVS component failure
- Recovering from a hardware or power failure

These topics are discussed further in Chapter 4.6, “Backing Up and Recovering Databases,” in the *DB2 for OS/390 V5 Administration Guide*, SC26-8957.

There are many ways you can manage DB2 data. In this section we describe two approaches to backup and recover of DB2 data to demonstrate how DB2 can benefit from using SnapShot, namely:

- Offline recovery approach

  Using DFSMSdss volume dumps for backup and recovery.

- DB2 recovery approach using DFSMSdss and the DB2 online utilities for backup and recovery.

C.1.10.2 Backup Using Full Volume Dumps

The offline recovery approach is based on taking a physical volume dump of the entire DB2 subsystem. This requires a POC, which is normally achieved by stopping DB2. This approach achieves a certain level of data currency up to the backup POC but does not provide the ability to recover to the point of failure. It requires a common POC for all applications, and a lengthy outage to system and application resources while the data is backed up.

Today our requirement for higher availability with no loss of data means this approach may no longer be affordable, as it involves a lengthy service outage to recover lost data processed in the interim period between your backups and the disaster. However, if you use this method you can achieve a significant reduction in the length of your backup window by using SnapShot to copy DB2 data.

Backup for disaster recovery using DFSMSdss full volume backups principally involves the following steps:

1. Establish a POC for DB2 and other subsystems, such as Information Management System (IMS) or Customer Information Control System (CICS).
2. Check for utilities.
3. Check the status of all objects.
4. Take image copies of application data and the DB2 catalog and directory.
5. Issue ARCHIVE LOG MODE(QUIESCE).
6. Issue -STOP DB2 MODE(QUIESCE).
7. Snap the target volumes.

8. Restart DB2.
9. Take a DFSMSdss full volume backup of the target volumes and system data sets.

C.1.10.3 Backup and Recovery Using DB2 Utilities

The DB2 recovery approach is based on image copies and archive logs.

Backup Using DB2 Utilities: Backup for disaster recovery principally involves the following steps:

1. Establish a POC for DB2 and other subsystems, such as IMS or CICS.
2. Check for utilities.
3. Check the status of all objects.
4. Take image copies of application data and the DB2 catalog and directory.
5. Issue ARCHIVE LOG MODE(QUIESCE).
6. Issue -STOP DB2 MODE(QUIESCE).
7. Take a backup of the BSDS, logs, image copies, and system data sets, using DFSMSdss
8. Restart DB2.

The nondisruptive alternative to this approach is to recover DB2 with a conditional restart record using the logs.

Remote Site Recovery of DB2 Using DB2 Archive Logs: Recover of DB2 using the logs involves the following steps:

1. Restore the BSDS, archive logs, image copies, and system data sets, using DFSMSdss.
2. Run DSN1LOGP to analyze which transactions were in process at the end of the last archive log and whether utilities were in flight at that moment.
3. Create a conditional restart control record, using the change log inventory utility, DSNJU003, to modify the BSDS.
4. Conditionally start DB2 in ACCESS(MAINT) mode.
5. Resolve any in-doubt units of work.
6. Recover the catalog and directory.
7. Recover related DB2 products, such as the DB2 automated utility generator (DB2AUG), to aid application recovery.
8. Restart DB2, to allow routine access.
9. Recover application data to a POC.

C.1.10.4 Operational Backup

Backup for operational recovery principally involves the following steps:

1. Run the REPORT utility to determine whether the table space is a part of a referential structure.
2. Start the DB2 object being backed up for read-only access.
3. Quiesce the table spaces.
4. Take an image copy of the table space.

5. Start the DB2 object being backed up for read/write access.

C.1.10.5 Operational Recovery

Operational recovery principally involves the following steps:

1. Run the REPORT utility to determine whether the table space is a part of a referential structure.
2. Use STOP TABLESPACE to stop general access to the table space and index.
3. Use START TABLESPACE ACCESS(UT) to start the table space or index for utility only access.
4. Recover the table spaces and indexes, using the RECOVER utility, specifying parameters based on the type of recovery required.
5. Check for data inconsistency.
6. Start the table space for general access, when recovered.

As SnapShot lends itself more toward point of consistency recovery and replicating data for application testing, we have concentrated on these specific areas in our examples. DB2 high availability and recovery are covered in a lot of detail in the standard product books and in various IBM Redbooks.

Appendix D. Sample ABARS Procedure Using SnapShot and REXX

This procedure and EXEC are available in softcopy through the Internet. The file is *ABARS.TXT*.

Point your Web browser to:

`ftp://www.redbooks.ibm.com/redbooks/SG242241`

Alternatively you can go to:

`www.redbooks.ibm.com`

and select SEARCH and then Additional Redbook Materials.

This procedure is divided into three separate TSO batch jobs to avoid data set contention problems:

- *Job 1* - performs an AGGREGATE BACKUP VERIFY and writes the output report to a data set (Figure 105 on page 184)
- *Job 2* - executes the REXX EXEC that processes the output from the aggregate report, snaps the data sets, and builds a new aggregate group filter list (Figure 106 on page 184). Update processing can resume to the original data sets after job 2 completes.
- *Job 3* - performs an AGGREGATE BACKUP EXECUTE of the new aggregate group (Figure 107 on page 184). Once the backup has completed, the snap target data sets are deleted.

Two aggregate groups have been defined. The first aggregate group contains a filter list that defines which data sets need to be included with the aggregate group. This aggregate group is named AGGONE.

The filter list for the second aggregate group is generated by the REXX EXEC. This aggregate group is named AGGSNAP. The aggregate group was defined with a text instruction data set that will contain the snapback commands which need to be issued if the aggregate group is to be restored from the backup.

The REXX EXEC performs the following steps:

1. Reads the output from ABACKUP AGGONE VERIFY to determine the names of the data sets in the aggregate group
2. Excludes certain data sets on the basis of their volser, such as tape data sets, migrated data sets, and data sets on non-RVA DASD
3. Snaps each data set that has not been excluded
4. Builds a new filter list. This includes the data set names of the snap target data sets. For any data sets that were not snapped, it includes the original data set name. This list is written to aggregate group AGGSNAP′s control data set.
5. Builds a list of snapback commands and writes it to the instruction data set to be included with AGGSNAP
6. Builds a list of DELETE commands to delete the snap target data sets. These DELETE commands are issued after the aggregate backup has been completed in job 3.

You should customize the EXEC to your requirements. Some parameters that can be modified are:

- ***excludevolpfx*** - You can set this to the prefix of your tape volumes. The EXEC will not attempt to snap data sets on volumes with this prefix.
- ***target_agroup_name*** - This is the name of the second aggregate group, AGGSNAPP. The selection list for this aggregate group will be built from the output from the snap job.
- ***snapfail*** - If this value is set to anything other than OK, the EXEC terminates with return code 16 if an attempt to snap a data set fails for any reason.

The REXX EXEC initializes the SIB environment for every data set it tries to snap. Depending on the number and size of data sets in the aggregate group, this REXX EXEC could take longer to execute than performing a normal ABARS backup. Try this procedure and compare it to a normal ABARS backup before changing your procedures.

```
//STEP01   EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
DEL 'PROD.ABARS.REPORT'
HSEND WAIT ABACKUP AGGONE FODS('PROD.TEMP.ABARS.REPORT') VERIFY
SUBMIT 'PROD.JOB.CNTL(AGJOB2)'
```

Figure 105. ABARS Procedure: Job 1

```
//STEP01   EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*
//AGREPORT DD DSN=PROD.TEMP.ABARS.REPORT,DISP=(OLD,DELETE)
//POSTFILE DD DSN=PROD.ABARS.CNTL.AGGSNAP,DISP=(OLD,KEEP)
//SNAPBACK DD DSN=PROD.ABARS.AGGSNAP.INSTRUCT,DISP=(OLD,KEEP)
//CLEANUP  DD DSN=PROD.TEMP.POSTSNAP.COMMANDS,
//          DISP=(NEW,CATLG),UNIT=SYSALLDA,
//          DCB=(RECFM=FB,LRECL=80),SPACE=(TRK,(2,2),RLSE)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
EXEC 'PROD.JOB.CNTL(AGREXX)'
SUBMIT 'PROD.JOB.CNTL(AGJOB3)'
```

Figure 106. ABARS Procedure: Job 2

```
//STEP01   EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DSN=PROD.TEMP.POSTSNAP.COMMMANDS,DISP=(OLD,DELETE)
```

Figure 107. ABARS Procedure: Job 3

```
/** Rexx ***************************************************************/
/*                                                                      */
/* Sample Rexx exec to snap Aggregate Group.                            */
/*                                                                      */
/* - Reads output from previously issued ABACKUP with VERIFY            */
/*   parameter to get a list of dsns in the Aggregate Group.            */
/*                                                                      */
/* - Attempt to SNAP each DASD data set in the Aggregate Group.         */
/*                                                                      */
/* - Build new Aggregate Group including each succesfully created       */
/*   snap target, and original data sets which could not be snapped.    */
/*                                                                      */
/* - Build command list to perform snapbacks and write to ABARS         */
/*   instruction data set.                                              */
/*                                                                      */
/* - Build command list to delete snap copies after backup.             */
/*                                                                      */
/************************************************************************/

excludevolpfx = 'TAP'
target_agroup_name = 'AGGSNAP'
snapfail = 'OK'

/*--------------------------------------------------------------------*/
/* Read output from ABACKUP VERIFY and attempt to snap DASD data sets */
/*--------------------------------------------------------------------*/
exvol_length = length(excludevolpfx)
"EXECIO * DISKR AGREPORT (STEM REPLINE. FINIS"
nosnap = 0
snap = 0
do x = 5 to repline.0
  parse value repline.x with tempdsn 'VOLSER=' tempvol
  tempdsn = strip(tempdsn)
  tempvol = strip(tempvol)
  if length(tempdsn)=0 then nop
  else if (tempvol = 'MIGRAT' | ,
   left(tempvol,exvol_length) = excludevolpfx) then call no_snap
  else if tempvol/='' then call snap_it
end
call build_filtlist
call build_snapback
call build_cleanup
exit

/*--------------------------------------------------------------------*/
/* Volume excluded, or Snap failed                                    */
/*--------------------------------------------------------------------*/
no_snap:
     say tempdsn" not snapped. Volume = "tempvol
     nosnap = nosnap + 1
     nosnapdsn.nosnap = tempdsn
return

/*--------------------------------------------------------------------*/
/* Attempt to Snap data set                                           */
/*--------------------------------------------------------------------*/
snap_it:
    "SIBADMIN SNAP DATASET(SOURCE('"tempdsn"') ",
    "TARGET('"tempdsn".SNAP') REPLACE(YES))"
    if rc=0 then do
      snap = snap + 1
      snapdsn.snap = tempdsn'.SNAP'
      presnapdsn.snap = tempdsn
      say tempdsn" snapped to "snapdsn.snap
    end
    else do
     if snapfail /= OK then exit 16
     say "Non zero return code snapping "tempdsn".SNAP"
     call no_snap
    end
return

/*--------------------------------------------------------------------*/
/* Build the new Aggregate Group filter list */
/*--------------------------------------------------------------------*/
build_filtlist:
"newstack"
queue "INCLUDE( -"
do y = 1 to nosnap
  queue " "nosnapdsn.y", -"
end
do z = 1 to (snap - 1)
  queue " "snapdsn.z", -"
end
queue " "snapdsn.snap")"
queue ""
"execio * diskw postfile"
return

/*--------------------------------------------------------------------*/
/* Build SnapBack commands                                            */
/*--------------------------------------------------------------------*/
build_snapback:
"newstack"
queue "/* The following SIBADMIN commands should be issued after the */"
queue "/* Aggregate Group has been ARECOVERed to SNAPBACK the data   */"
queue "/* sets to their original names.                              */"
queue " "
do z = 1 to snap
  queue "SNAP DATASET(SOURCE('"snapdsn.z"') ,"
  queue "TARGET('"presnapdsn.z"') REPLACE(YES))"
end
"EXECIO * DISKW SNAPBACK"
return

/*--------------------------------------------------------------------*/
/* Build ABARS EXECUTE command and DELETE snap target commands        */
/*--------------------------------------------------------------------*/
build_cleanup:
"newstack"
queue "HSEND WAIT ABACKUP "target_agroup_name" EXECUTE"
do z = 1 to snap
  queue "DELETE '"snapdsn.z"'"
end
"EXECIO * DISKW CLEANUP"
return
```

Appendix E. Special Notices

This publication is intended to help storage administrators, system programmers, database adminstrators, and technical operations managers and staff in the planning and implementation of IBM RAMAC SnapShot for MVS/ESA and IBM RAMAC SnapShot for VM/ESA. It is not intended as a specification of any programming interfaces that are provided by SnapShot. See the PUBLICATIONS section of the IBM Programming Announcement for IBM RAMAC SnapShot for MVS/ESA, and IBM RAMAC SnapShot for VM/ESA, and IXFP/SnapShot for VSE/ESA 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, North Castle Drive, Armonk, NY 10504-1785.

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 information about non-IBM (″vendor″) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. 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.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

DATABASE 2	DB2
DFSMS	DFSMS/MVS
DFSMS/VM	DFSMSdfp
DFSMSdss	DFSMShsm
DFSMSrmm	DFSORT
ECKD	Enterprise Systems Connection Architecture
ES/3090	ES/9000
ESCON	FFST
IBM	IMS
IMS/ESA	MVS/DFP
MVS/ESA	NetView
OS/390	Parallel Sysplex
PR/SM	Predictive Failure Analysis
RACF	RAMAC
RETAIN	RMF
S/390	Service Director
VM/ESA	VSE/ESA

SET and the SET logo are trademarks owned by SET Secure Electronic Transaction LLC.

Microsoft, Windows, and the Windows 95 logo
are trademarks or registered trademarks of Microsoft Corporation.

Java and HotJava are trademarks of Sun Microsystems, Inc.

IXFP	Storage Technology Corporation
Extended Storage Architecture	Storage Technology Corporation
Iceberg	Storage Technology Corporation
XSA	Storage Technology Corporation
XSA/Reporter	Storage Technology Corporation
SnapShot	Storage Technology Corporation
StorageTek	Storage Technology Corporation
SAS	SAS Institute, Incorporated
SAS/C	SAS Institute, Incorporated
SAS/GRAPH	SAS Institute, Incorporated

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

Appendix F. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

F.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see "How to Get ITSO Redbooks" on page 193.

- *IBM RAMAC Virtual Array*, SG24-4951
- *Implementing Concurrent Copy*, GG24-3990
- *Implementing DFSMSdss SnapShot and Virtual Concurrent Copy*, SG24-5268
- *RAMAC Virtual Array, Peer-to-Peer Remote Copy, and IXFP/SnapShot for VSE/ESA*, SG24-5360
- *Disaster Recovery Design Concepts*, SG24-4211
- *DB2 for MVS/ESA Version 4 Data Sharing Implementation*, SG24-4791
- *RAMAC Virtual Array: Implementing Peer-to-Peer Remote Copy*, SG24-5338

F.2 Redbooks on CD-ROMs

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at `http://www.redbooks.ibm.com/` for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr Format)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037

F.3 Other Publications

These publications are also relevant as further information sources:

- *IBM Extended Facilities Product Configuration and Administration*, SC26-7178
- *IBM RAMAC SnapShot for MVS/ESA Using SnapShot*, SC26-7173
- *IBM RAMAC SnapShot for MVS/ESA Installing SnapShot*, SC26-7174
- *IBM RAMAC SnapShot for VM/ESA Installing and Using SnapShot*, SC26-7217
- *IBM RAMAC SnapShot for VM/ESA Installation Notes*, SC26-7218
- *OS/390 MVS Initialization and Tuning Reference*, SC28-1752
- *DFSMS/MVS Using Data Sets*, SC26-4922
- *DFSMSdfp Storage Administration Reference*, SC26-4920

- *IMS/ESA Utilities Reference: System*, SC26-8035
- *DB2 for OS/390 V5 Application Programming and SQL Guide*, SC26-8958
- *DB2 for OS/390 V5 Command Reference*, SC26-8960
- *DB2 for OS/390 V5 Utility Guide and Reference*, SC26-8967
- *DB2 for OS/390 V5 Administration Guide*, SC26-8957
- *The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation*, GC28-1251

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** `http://www.redbooks.ibm.com/`

  Search for, view, download, or order hardcopy/CD-ROMs redbooks from the redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this redbooks site.

  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.

- **E-mail Orders**

  Send orders by e-mail including information from the redbook fax order form to:

In United States:	e-mail address: usib6fpl@ibmmail.com
Outside North America:	Contact information is in the ″How to Order″ section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the ″How to Order″ section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the ″How to Order″ section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and redbooks by accessing the IBM Intranet Web site at `http://w3.itso.ibm.com/` and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access `MyNews` at `http://w3.ibm.com/` for redbook, residency, and workshop announcements.

IBM Redbook Fax 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.

Index

A

ABARS
  aggregate 101
  naming convention 101
  procedures 103
  sample 103
  VERIFY 103
Aggregate Backup and Restore Support
  See ABARS
AI 9
allocation
  extents 14
  primary allocation 14
alternate index
  See AI
AOC/MVS 85
application testing
  DB2 67

B

backup procedures
  second MVS 50
batch
  analysis 44
  application processing 41
  backup procedure 45, 49
  backup window 44
  data set reorganization 41
  incremental backup 41
  interim backup 41
  operational backup 49
  Optimizer 44
  procedures 41
  recovery procedure 47
  report generation 41
  SNAP DATASET 44
  SNAPAID 44
  throughput 41
boot strap data set
  See BSDS
BSDS 61

C

CA 53
catalog
  See also ICF catalog
  backup 87
  COPYVOLID 42
  DB2 71
  deallocate 118
  EXPORT 29
  ICF 117

catalog *(continued)*
  implications 3
  IMPORT 30
  indirect cataloging 118
  logical dump 30
  operating system 116
  physical dump 49
  recovery 87
  second MVS 50, 51
  SMS 28
  system data sets 117
  user catalog 29, 32
  VOLUME SNAP 28, 87
  VSAM 117
  Year 2000 105, 118
change bit 51
  See also DSCHA
CI 53
CKD 1
CLIP 32
CMS FORMAT 123
cold start
  DB2 60
compaction 2
compression 2
concurrent copy 33
  description 7
CONDVOL
  COPYVOLID 15
control area
  See CA
control interval
  See CI
COPY
  catalog 21
  device type 21
  extended format VSAM 21
  multivolume 21
  PDSE 21
  utility 21
COPY utility 57
COPYVOLID
  CONDVOL 15
  VOLCOND 32
count key data
  See CKD

D

DA 23
DADSM 88
data consistency 57
data currency 88
data in memory
  DB2 58, 173

data mover 10, 34
data safety 88
data set naming
  DB2 57
database
  Year 2000 testing 109
database identifier
  See DBID
database recovery control
  See DBRC
DB2
  active log 170
  application testing 67
  archive log 170
  backup 179
  BSDS 169
  buffers 58, 173
  catalog 57, 60, 69, 71, 169
  cloning data 57, 60, 70
  cold start 60
  conditional restart 71, 169
  COPY 56, 57, 175
  data consistency 57
  data integrity 172
  data set placement 173
  data sets 169
  data sharing 61
  DFSMSdss dumps 57
  directory 69, 71, 169
  disaster recovery 57
  DLD 60
  DSN1COPY 59, 70, 175
  function 55
  image copy 56
  indexes 171
  internal identifiers 70
  locking 60
  LOGONLY 65
  LOGONLY recovery 58
  migration flow 71
  naming convention 10, 57, 67, 176
  OBID 59, 70
  overview 169
  point of consistency 56
  QUIESCE 58, 176
  RECOVER 57, 176
  RECOVER TABLESPACE 65
  recovery 58, 62, 65, 179
  referential integrity 61, 172
  REPORT 61
  requirements 55
  serialization 60
  SHRLEVEL 57, 178
  SnapShot benefits 55
  STOGROUP 173
  system changes 68
  system data sets 71
  system volumes 57
DB2 *(continued)*
  table set 61
  table space 171
  test system 71
  TOCOPY 58
  TORBA 58
  Type 2 index 59
  utilities 56, 175
  VCAT 60, 169
  Year 2000 57
  Year 2000 Web page 114
DBID 70
DBRC
  description 76
  notification 77, 78, 82
  registration 77, 82
  UIC 77
DDSR 19, 88
device reserve 117
DFSMS/MVS
  ACS routines 13, 26, 32
  allocation 13, 32
  DATACLAS 13
  DFSMShsm 33
  Optimizer 43
  storage group 25
  striping 26
  volume preferencing 13, 25
DFSMSdss
  CONCURRENT 33
  incremental backup 49, 52
  logical dump 30, 31, 52, 81, 94
  performance 95
  physical dump 30, 31, 49, 91, 94
  shared MVS 51
  stand-alone restore 98, 115
  VSAM 30, 31, 94
DFSMSdss SnapShot
  benefits 21
  description 6
  prerequisites 7
  requirements 21
DFSMShsm
  ABARS 33, 101
  backups 49
  CDS 50
  change bit 52
  disaster recovery 33
  HRECOVER 51
  shared MVS 51
DFSMSrmm
  CDS 50
DFSUDMP0 76
DFSUDMT0 77
direct access data set
  See DA
disaster recovery
  COPYVOLID 31

disaster recovery *(continued)*
  DB2 57
  principles 87
  procedures 95, 97
  recovery procedures 98
  secure location 88
  system 98
DLD 60
down level detection
  See DLD
DSCHA 52
DSN1COPY 59
dump scheduling

E

entry sequenced data set
  See ESDS
entry-sequenced data set
  See ESDS
ESDS 9, 23
extended format data set 13, 23
  striped 23
  VSAM 23
extended remote copy
  See XRC

F

FBA 1
FDT 1
fixed block architecture
  See FBA
freespace collection 1
FTD 1
FTT 1
functional device
  deletion 38, 46, 52, 89
  description 1
  ICKDSF 47
  initialization 38
  management 94, 98, 107
functional device table
  See FDT
functional track directory
  See FTD
functional track table
  See FTT

G

GDG 23
generation data group
  See GDG
global resource serialization
  See GRS
GRS 51, 117

H

hardware configuration definition
  See HCD
hardware requirements 5
HCD 107, 116
HFS 13, 23
hierachical file system
  See HFS
Hierarchical File System
  See HFS

I

IBM Extended Facilities Program
  See IXFP
IBM global services 44
Iceberg 5
ICF catalog 23, 87
ICKDSF
  functional device 47
  REFORMAT 32
IDCAMS
  AI 10
  recatalog 87
image copy
  definition 76
  IMS 76
  logical completion 82
  physical completion 82
  RUNTIME 83
IMS
  3990 concurrent copy 77
  access method 76
  alternate index 86
  applications 76
  authority 78
  automation 85
  control region 76
  data sets 76
  database cloning 85
  database types 78
  DBRC 75, 82
  GENJCL 84
  GENJCL utility 79
  image copies 75
  image copy 76, 78, 82
  image copy 2 77
  integrity 84
  multivolume database 79
  NOTIFY command 82
  OLDS swap 78, 82
  performance considerations 84
  point of consistency 76, 77, 78
  procedures 85
  pseudoabends 78
  recovery time stamp 83
  restarting database 80
  restore considerations 84

IMS *(continued)*
  tape backup 80
  transactions 78
  UIC 75, 77
  utilities 75
  Version 5 78, 82
  Version 6 77
  Version 6: 84
  volume snap 79
incremental backup 90
indexed sequential data set
  See ISAM
instant format 119
integrated catalog facility
  See ICF
interactive storage management facility
  See ISMF
IOSQ 39
ISAM 23
ISMF 101
IXFP
  DDSR 38
  function 1
  installation 25
  SMF 17, 18
  space utilization 16
  volume preferencing 13, 24

K
key sequenced data set
  See KSDS
key-sequenced data set
  See KSDS
KSDS 9, 23

L
LDS 9, 23
linear data set
  See LDS
logical partition
  See LPAR
LOGONLY recovery 58
LPAR 106

M
minidisk cache 125
multivolume data set
  consolidation 10
  data mover 10
  VOLUMECOUNT 11, 14

N
NCL
  compression 35
  description 1
NCL *(continued)*
  elastic capacity 37
  management 89, 112
  monitoring 37, 46, 88
  predicting NCL 35
  prediction 84
net capacity load
  See NCL
NUCXDROP 124

O
OBID 59
object ID
  See OBID
OLDS swap
  NOFEOV 78
OPC/ESA 85, 94
operating system
  cloning 107
  data integrity 116
  IPL 116
  maintenance 115
  recovery 116
  replication 115
Optimizer
  report 43
OSAM 76

P
page data sets 23
page set identifier
  See PSID
partitioned
  See PO
partitioned data set
  See PO
partitioned data set extended
  See PDSE
PDSE 13, 23
peer-to-peer remote copy
  See PPRC
performance
  analysis 36
  batch throughput 41
  DFSMSdss elapsed time 36
  factors 36
  guidelines 37
  tests 36
physical copy 2
PO 23
point of consistency 55
PPRC 89
preformatting 39
PSID 70

Q

QUIESCE
  DB2 58

R

RAID DASD 49
RBA 169
RECOVER utility 57
referential integrity
  DB2 172
  description 61
regression testing 57
relative byte address
  See RBA
relative record data set
  See RRDS
remote copy 89
REPORT utility 61
RRDS 9, 23
RVA
  Year 2000 compliance 106

S

SAF 39
second MVS
  backup 92
  catalog 32, 94
  disaster recovery 97
security
  facility class 39
  procedures 39
  resource names 39
serialization
  DB2 60
  DFSMSdss 38
  SnapShot 38
ServerPac 115
shared DASD
  serialization 51
SIBBATCH 41
SIBDMDDR 120
SIBFMTSS 119, 123
SIBTRAP 123, 126
SIBVMRVA 120, 126
SMF
  format 17, 18
SMP/E
  SnapShot 116
SMS
  catalog 28
  recovery 87
SNAP DATASET
  ACS routines 20
  allocation 10, 13, 20, 24
  backup procedure 45
  catalog 118
SNAP DATASET *(continued)*
  data mover 20
  data set types 20
  DATACLAS 13, 14
  extended format data set 13
  format 19
  multiple extents 24
  multivolume data set 10, 14
  operation 41
  recovery procedure 47
  serialization 38
  space allocation 19
  supported types 9, 13
  system data sets 116
  TOLERATETRUNCATION 14
  unsupported data set 20
  VOLUME keyword 13
  VOLUMECOUNT 11, 14
SNAP MINIDISK
  directory 122
  format 122
  RELOCATE 121
  virtual addresses 125
SNAP VOLUME
  COPYVOLID 15, 32, 41
  range cylinders 120
  VM/ESA 120
  volume conditioning 15
  volume dumps 90
SNAPAID
  analysis 42
  benefits 42
  report 42
SnapShot
  ABARS 101
  architecture 2
  catalog services 118
  command environment 4
  commands 79
  concepts 1
  enhancements 9
  enqueue processing 117
  identifying candidates 42
  IMS benefits 77
  incremental backup 90
  initialization 89
  logical copy 2
  NCL 35, 88
  operation 3
  performance 36, 84, 95
  physical dump 91
  processing 37, 38
  Release 1.2 9
  remote copy 89
  resources 3
  restrictions 4
  security 39
  serialization 38, 117

SnapShot *(continued)*
  services 44
  shared DASD 117
  SMP/E 116
  SNAPAID 42
  snapback 91, 99
  SYSRES 115
  system backup 115
  Year 2000 compliance 106
software requirements 5
software support 7
space utilization
  physical capacity 16
  SHARED/UNIQUE 16
STOGROUP
  DB2 173
storage hierarchy 174
Storagetek 9200 5
supported data sets 23
SYSCOPY 57
Sysplex Timer 106
SYSRES
  cloning 115
  data types 116
  disaster recovery 98
  indirect cataloging 118
  SnapShot 115
System Authorization Facility
  See SAF
system data sets
  catalog 117
SYSXLAT 70

T
tape
  dump scheduling 95
  remote vaulting 88
time machine 106
time of day clock
  See TOD
TNT
  description 2
  reference counter 2, 3
TOD 106
track number table
  See TNT
Type 2 index 59

U
update in place 1
utilities 175

V
variable relative record data set
  See VRRDS
VCAT
  DB2 60
virtual concurrent copy
  benefits 22
  characteristics 22
  database support 22
  description 6
  logical completion 6
  physical completion 6
  prerequisites 7
  requirements 22
  utility 22
virtual disk architecture 1, 53
VM minidisks 4
VM/ESA
  data mover 120
  guest support 124
  host data caching 125
  instant format 123, 126
  minidisk formatting 119
  minidisks 119
  privilege class 120
  recovery 125
  RELOCATE 121
  REXX EXEC 124
  SIBADMIN 120
  SIBVMRVA 120
  SNAP MINIDISK 121
  software requirements 119
  utilities 126
  volumes 119
volume preferencing
  ACS routines 25
  data mover 25
  description 13
VOLUME SNAP
  catalog implications 28
  COPYVOLID 27
  data mover 27
  format 27
  REPLACE 27
  serialization 38
  TARGET 28
  volume selection 28
VRRDS 9, 24
VSAM
  allocation 25
  alternate index 9, 23, 86
  catalog 117
  cluster 10, 25
  considerations 25
  control area 53
  control interval 53
  data mover 10, 26
  extended format 23
  IDCAMS REPRO 53
  IMBED 20, 22, 25
  index 25

VSAM *(continued)*
  logical dump 91
  multiple extents 25
  performance 36
  recatalog 29
  reorganization 53
  REPLICATE 20, 22, 25
  SHAREOPTION 9
  sphere processing 25
  supported organizations 9
  target names 10
VSAM variable record data set
  See VRRDS
VSE/ESA 124
VTOC 23

X
XRC 89

Y
Year 2000
  catalog 105
  challenge 105
  database testing 109
  RVA compliance 106
  security 106
  shared DASD 107
  SNAP VOLUME 107, 112
  storage requirement 106
  test system 105
  time machine 106
  time setting 109
  Web page 106

ITSO Redbook Evaluation

Implementing SnapShot
SG24-2241-01

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.ibm.com/`
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to `redbook@us.ibm.com`

Which of the following best describes you?
__Customer __Business Partner __Solution Developer __IBM employee
__None of the above

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!)**

____

____

____

____

____

SG24-2241-01
Printed in the U.S.A.

Implementing SnapShot

SG24-2241-01