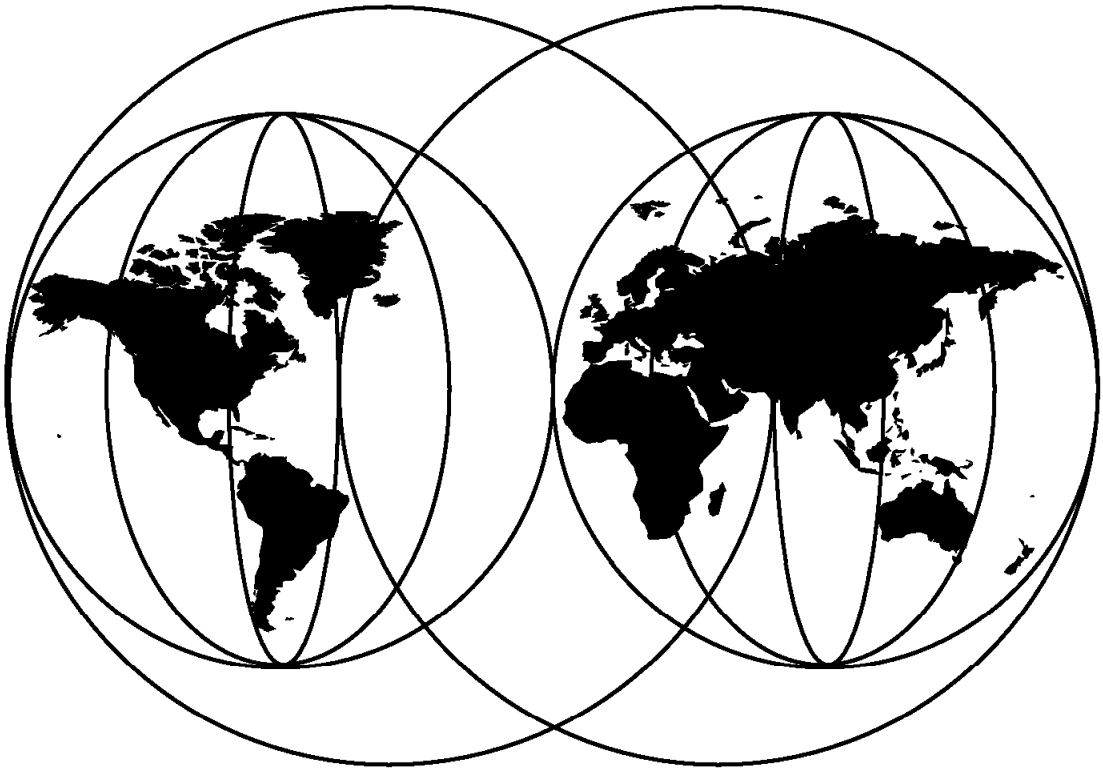
Implementing DFSMSdss SnapShot and Virtual Concurrent Copy

Alison Pate Clemente Vaia Jeff Todd Harald Aigner



International Technical Support Organization

http://www.redbooks.ibm.com

This book was printed at 240 dpi (dots per inch). The final production redbook with the RED cover will be printed at 1200 dpi and will provide superior graphics resolution. Please see "How to Get ITSO Redbooks" at the back of this book for ordering instructions.

SG24-5268-00

International Technical Support Organization

SG24-5268-00

Implementing DFSMSdss SnapShot and Virtual Concurrent Copy

June 1998

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix C, “Special Notices” on page 129.

First Edition (June 1998)

This edition applies to Version 1, Release 2 of IBM RAMAC SnapShot for MVS/ESA, and DFSMS/MVS Version 1, Releases 3 and 4.

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.

© **Copyright International Business Machines Corporation 1998. All rights reserved.**
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 1998

Preface

This redbook is a guide for implementing virtual concurrent copy and DFSMSdss SnapShot. These new functions are available with RAMAC SnapShot for MVS/ESA Version 1.2 in combination with DFSMSdss and SPE to DFSMS/MVS Version 1.3 and 1.4.

It is written for IBM and customer personnel who want detailed technical and operational guidance on implementation in a variety of environments.

Recommendations for implementing SnapShot in batch jobs, and in conjunction with DB2, IMS, DFSMShsm, and SAP, are included.

Implementing DFSMSdss SnapShot and Virtual Concurrent Copy updates and adds to the information presented in *Implementing SnapShot, SG24-2241*.

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 eight 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.

Clemente Vaia is an Advisory System SW Specialist in the Enterprise Systems Technical Support group in Italy. He joined IBM in 1981 and his areas of expertise include defect support on Storage products, DASD Storage Subsystem, Ramac, RVA.

Jeff Todd is an Advisory Software Engineer in the Santa Teresa Labroatory. He joined IBM in 1997 as a developer in the DB2 Utilities organization. He has twenty years of experience in data processing, with the last five years concentrated in DB2 backup and recovery solutions.

Harald Aigner is a Storage System Specialist in Germany. He joined IBM in 1974 as a large system operation specialist. He has ten years of experience in storage management. His areas of expertise include support of DASD, DFSMS/MVS, and related storage products for IBM internal and customers systems.

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

Scott Chen, 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
Maggie Cutler, International Technical Support Organization, San Jose Center

© Copyright IBM Corp. 1998

Janet Anglin, Storage Systems Division, San Jose
Laura Kunioka-Weis, Software Systems Division, Santa Theresa Laboratory
Terri Leamon, Storage Systems Division, San Jose
David Shackelford, DSS/SDM Development, Tucson
David Short, Storage Subsystems Division, San Jose
John Thompson, DSS/SDM Development, Tucson

Thanks to the following reviewers:

Eneo Baborsky, IBM Italy
Mike Downie, IBM Education and Training, Boulder
Ken Higgins, Storage Sales Specialist, Seattle
Daniel Leplaideur, ATSC, Mainz
Roger Miller, Software Systems Division, Santa Theresa Laboratory
Phil Norman, Storage Specialist, IBM UK
Karen Ranson, Software Systems Division, Santa Theresa Laboratory
David Sacks, Storage Sales Specialist, Chicago
Pete Sadler, IMS Specialist, IBM UK
Leif Schioler, Storage Specialist, IBM Denmark
Norbert Schlumberger, IBM Global Services, Germany

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 139 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

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

- Send us a note at the following address:

  `redbook@us.ibm.com`

Chapter 1. Introduction

In this chapter we introduce the new functions of virtual concurrent copy and DFSMSdss SnapShot. These functions combine the data duplication function of IBM SnapShot for MVS/ESA with the flexibility and operability of DFSMSdss.

As more and more installations move to 24 x 7 operation, they are faced with the challenges of increasing the availability of online applications without compromising recoverability in the event of failure. These dual pressures seem to conflict as time spent taking backups eats into online availability; reducing the number of backups reduces the ability to recover the application in the event of failure.

IBM SnapShot for MVS/ESA provides a solution as it enables you to replace traditional data copy techniques with almost instantaneous copies of data. With virtual concurrent copy and DFSMSdss SnapShot, ″traditional copy″ techniques benefit from the speed of SnapShot without changes to JCL.

1.1 DFSMSdss Support for SnapShot

New 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.

1.1.1 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 1 shows the DFSMSdss SnapShot operation.

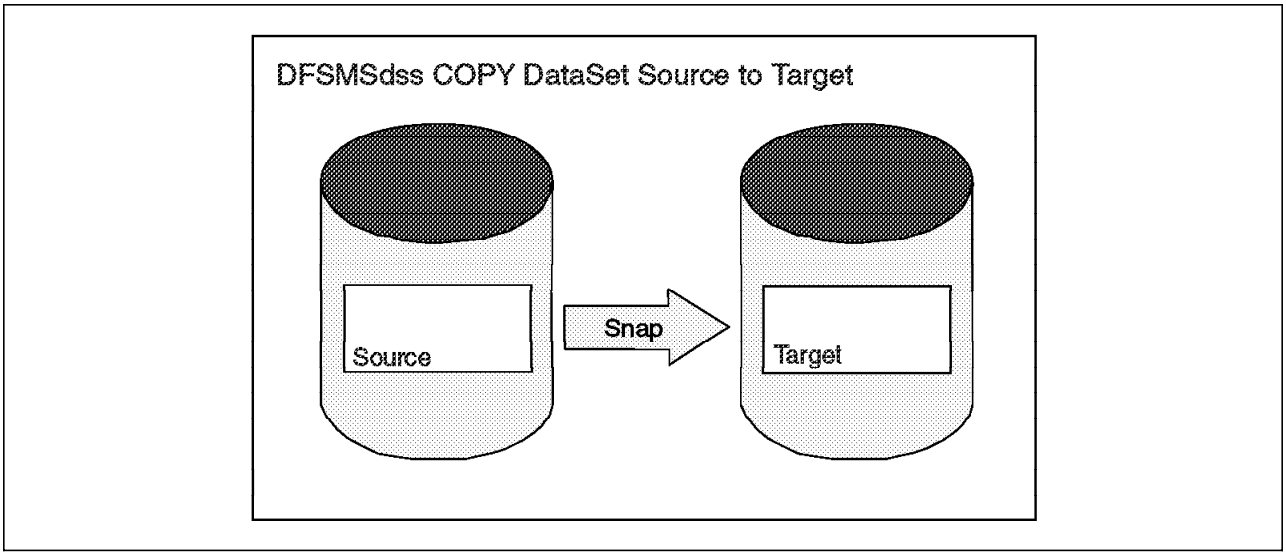


Figure 1. DFSMSdss SnapShot Operation

© Copyright IBM Corp. 1998

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.

1.1.2 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 2 shows the virtual concurrent copy operation.

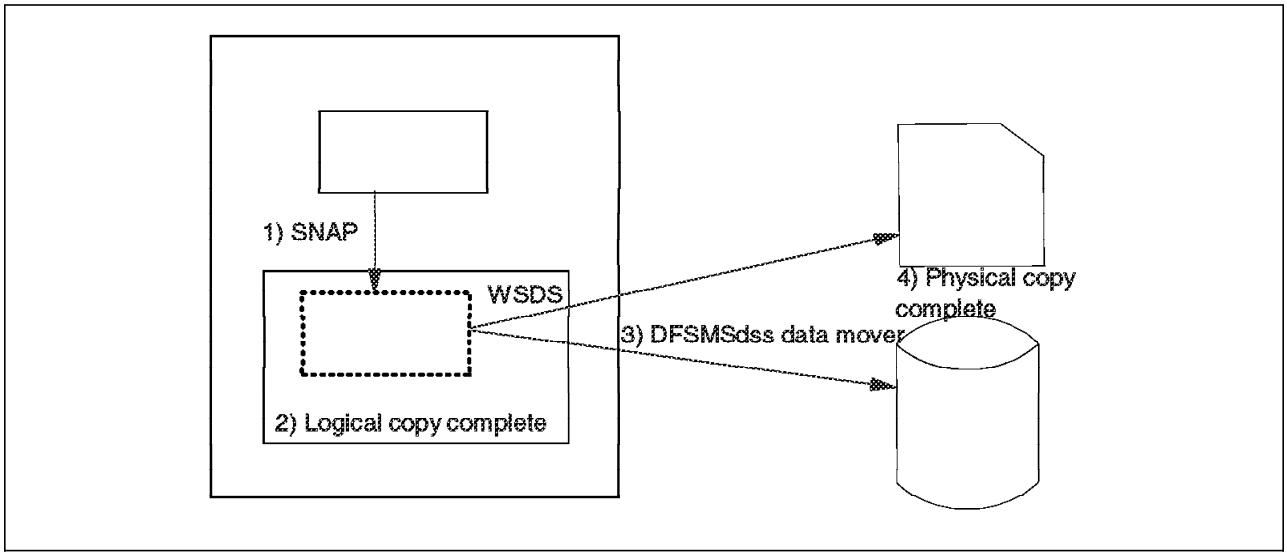


Figure 2. Virtual Concurrent Copy Operation

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.

1.2 Data Copy Techniques

Virtual concurrent copy and DFSMSdss SnapShot enhance the data copy capabilities of DFSMSdss, combining the benefits of DFSMSdss and of SnapShot.

1.2.1 IBM SnapShot for MVS/ESA

SnapShot is a virtual data duplicator that exploits the architecture of the RVA to create copies of data almost instantaneously. SnapShot produces copies without data movement because it logically manipulates pointers within the RVA. Because there is no actual movement of data, snapping can take seconds rather than minutes or hours, and host processor and channels are not involved in the data transfer. 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, the snap is a virtual copy of the data.

For a description of the RVA architecture and SnapShot operation, see Appendix A, “Overview of Ramac Virtual Array and SnapShot” on page 121.

1.2.2 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 Prerequisites

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:
   - OW29883
   - OW30320
   - OW29881

- OW26926
- OW32466
- OW32415

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. Technical Overview

In this chapter we discuss the new DFSMSdss functions. There are two flavors of DFSMSdss SnapShot support: DFSMSdss SnapShot is used for DFSMSdss COPY when the source and target data sets are located on compatible volumes in the same RVA subsystem.

Virtual concurrent copy is used for DUMP and COPY. It enables you to take a consistent copy or dump of your data while applications update the data.

2.1 DFSMSdss SnapShot

Functions have been added to DFSMSdss to support SnapShot for the COPY function. DFSMSdss and SnapShot now work together, both taking advantage of each other's characteristics without changes to existing procedures to benefit from this support.

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. It is much faster than traditional copy methods and is especially beneficial for moving large amounts of data.

2.1.1 Characteristics

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 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.

DFSMSdss can be run either as a batch job or through the application programming interfaces (APIs).

When SPHERE is specified in the DFSMSdss command, for a VSAM cluster, all associated alternate indexes, clusters, and paths are copied, if all SPHERE parts reside in the same RVA subsystem. Through DFSMSdss support, SnapShot is now capable of handling VSAM alternate indexes.

The new DFSMSdss SnapShot COPY support is designed to be transparent to the user.

© Copyright IBM Corp. 1998

2.1.1.1 DFSMSdss Filtering

Now that DFSMSdss can take advantage of such SnapShot characteristics as fast data movement, without any additional space required for the copy, filtering and partially qualified data set names are also available for the data sets that SnapShot handles.

When using SnapShot directly, you can copy only one data set with a single SNAP DATASET command; furthermore, only one volume is copied with a single SNAP VOLUME command.

In the SNAP DATASET command, both the data set and volume names must be fully qualified, so if one of these names changes, you must modify your SnapShot JCL commands to reflect the change.

Now that SnapShot can take advantage of DFSMSdss characteristics, multiple data sets or multiple volumes can be handled by a single DFSMSdss command. By using the DFSMSdss wildcard character function, it is no longer necessary to modify JCL every time the data set name changes.

2.1.1.2 DFSMSdss COPY

The DFSMSdss COPY and DUMP commands can be performed on a full volume, a range of tracks, or at the data set level.

If you specify the DATASET keyword with the COPY command, DFSMSdss performs a *logical* copy. A data set copy is always a logical operation, regardless of how or whether you specify input volumes.

If you specify the FULL or TRACKS keyword with the COPY command, DFSMSdss performs a *physical* copy.

For COPY, DFSMSdss automatically uses SnapShot if possible, resulting in a DFSMSdss SnapShot copy. If DFSMSdss SnapShot is not possible, DFSMSdss tries to use a virtual concurrent copy snap if the CONCURRENT COPY keyword is specified.

2.1.1.3 Serialization

Now with DFSMSdss support, you can serialize multiple data sets or multiple volumes at the same time. This serialization provides an effective point of consistency among all data sets involved in the copy operation and enables you to make a consistent point-in-time copy of data sets that are logically related.

2.1.2 Requirements

Only DFSMSdss COPY is supported by DFSMSdss SnapShot. 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 1 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 1. 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.

2.1.3 How It Works

When DFSMSdss SnapShot is issued and all requirements for a SnapShot are met, the system data mover (SDM) API, ANTRQST, is automatically called and a snap is issued. This is done completely transparently to your applications.

When you let DFSMSdss SnapShot dynamically allocate the target, it prefers the same RVA where the source resides, just as SnapShot prefers.

If the target is SMS managed, the DFSMSdss volume preferencing technique is used.

For details of the API functions, see Appendix B, “System Data Mover Application Program Interface Functions” on page 127.

2.1.3.1 DFSMSdss Copy Hierarchy

When a DFSMSdss COPY operation is executed, DFSMSdss chooses the most appropriate technique in the following sequence:

1. SnapShot
2. Track image copy
3. Track packing (block movement)
4. Utilities

SnapShot is the first preference, but other copy techniques are used when appropriate.

2.1.3.2 Target Dynamically Allocated

Target data sets can be dynamically allocated or preallocated.

Dynamic allocation works differently in different situations:

Non-SMS	DFSMSdss requires a list of volumes, so it calls ANTRQST to obtain the list of all SnapShot-capable volumes and compares it with the list of volumes provided in the DFSMSdss COPY command. If there is a match between the user lists, a DFSMSdss SnapShot is executed; if there is no match, SnapShot is not attempted and a traditional DFSMSdss copy is done instead.
SMS	A user may or not specify a volume list, but SMS overrides the selection based on ACS routines.
Multivolume	All source and target components must be snappable and in the same RVA, otherwise a traditional DFSMSdss copy is executed.
VSAM sphere	All components must be on the same RVA.
VSAM base cluster	Data and index parts must be on the same RVA although alternate indexes can be somewhere else.
Utilities	Any time DFSMSdss invokes a utility to copy the data, a traditional DFSMSdss copy is done.

2.1.4 Impact on Net Capacity Load

Despite the stated advantages of DFSMSdss SnapShot, its impact on net capacity load (NCL) remains the same as data set SnapShot.

When a snap is taken, it does not consume any back-end storage because the source and target share the same tracks. However, as updates occur on the source or target, they are written to a new location, so new tracks are used in the disk arrays. These new tracks occupy more back-end storage, so the potential effect on NCL must be considered.

When you use DFSMSdss SnapShot, you can automatically take advantage of SnapShot capability, without any changes to existing applications, so, until you have updated every track, the NCL is always less than it is with traditional copy.

If NCL is a concern, consider the length of time that snapped data sets or volumes are kept on the RVA and eventually delete some of the snapped data to lower the overall RVA NCL. Alternatively you can use DFSMShsm to migrate the snapped data sets to tape.

You can monitor NCL from the host through the IBM Extended Facility Product (IXFP), or you can use the Subsystem Configuration panel from the Local Operator Panel (LOP).

2.1.5 Performance

DFSMSdss SnapShot dramatically reduces the time required for data duplication when compared to traditional copying, thus minimizing the downtime for applications using the data. There is a tangible reduction of CPU utilization and I/O activity as well.

DFSMSdss can call SnapShot internally and transparently to the user and, if all the requirements are met, it executes a DFSMSdss SnapShot instead of a traditional DFSMSdss COPY, greatly reducing the job execution time. The most meaningful result of using SnapShot is the increase in application availability.

The time that DFSMSdss spends filtering data sets is still the same, even in a DFSMSdss SnapShot scenario. Therefore the more precisely you specify the data sets to be processed, the sooner SnapShot can be invoked.

This performance improvement is achieved by combining the benefits of DFSMSdss and SnapShot, without modifying existing JCL. For more details on performance, see Chapter 5, " DFSMSdss SnapShot Performance Considerations" on page 47.

2.1.6 Multivolume Data Sets

For data set COPY, DFSMSdss SnapShot can handle multiple source volumes, and DFSMSdss processes multivolume data sets in their entirety.

A data set in this context could be a non-VSAM data set, a VSAM cluster (with one or more components), or a VSAM sphere (with one or more VSAM clusters).

When processing a multivolume data set, before invoking SDM, DFSMSdss determines whether the entire data set resides in the same partition of the same RVA subsystem. For further details on how DFSMSdss SnapShot handles multivolume data sets, see 3.3, "Multivolume Data Sets" on page 22.

2.1.7 VSAM Considerations

DFSMSdss SnapShot provides enhanced usability for VSAM data sets through its support of alternate indexes.

From a SnapShot point of view, all considerations related to VSAM data sets are still valid, that is, by using the name of the VSAM cluster you can issue a SNAP DATASET to snap VSAM data sets. In the case of a key-sequenced data set (KSDS) cluster, both the data and the index are copied. The target data set name must be different from the source data set name.

The SNAP DATASET command does not support sphere processing, but when SPHERE is specified in the DFSMSdss command, together with VSAM cluster, all associated alternate indexes, clusters, and paths are copied, if all SPHERE parts reside in the same RVA subsystem.

If indexed 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. If data and index are preallocated so that the source and target component are in the same RVA (data may be on one RVA, index on another), DFSMSdss uses traditional I/O to copy the data. In the same case the SNAP DATASET command snaps both components.

If the base cluster components are on an RVA and there is an alternate index on a different subsystem, DFSMSdss uses DFSMSdss SnapShot when you specify the base cluster. If you use the SPHERE keyword, DFSMSdss is used as the data mover because of the "all or nothing" rule for processing data sets.

Note

For VSAM KSDSs and relative record data sets (RRDSs), where the target data set is multivolume, DFSMSdss does not invoke DFSMSdss SnapShot or virtual concurrent copy. If the target is single volume, a multivolume source can be copied with DFSMSdss SnapShot or virtual concurrent copy COPY. This restriction is expected to be temporary; details are documented in APAR OW32415.

SnapShot does not support the use of the IDCAMS DEFINE command parameters, IMBED and REPLICATE. Data sets defined with these parameters are also not supported by DFSMSdss SnapShot; DFSMSdss uses traditional I/O to copy these data sets.

IMBED indicates that the sequence set, the lowest level of the index, is placed with the data component, writing the sequence set record for each control area as many times as it fits on the first track adjacent to the control area.

REPLICATE indicates how many times each index record is to be written on a track. Each index record is written on a track as many times as it will fit.

Both the IMBED and REPLICATE parameters reduce rotational delay and improve performance in a traditional DASD environment, but in a virtual disk architecture these concerns are no longer valid. The two parameters do not provide any performance advantage in an RVA subsystem and should be discontinued.

2.2 Virtual Concurrent Copy

DFSMSdss uses virtual concurrent copy for DUMP and COPY. A virtual concurrent copy enables you to take a consistent copy or dump of your data while applications update the data. You invoke virtual concurrent copy by specifying the CONCURRENT keyword on a DFSMSdss COPY or DUMP statement.

2.2.1 Characteristics

Concurrent copy is a function of DFSMSdss and the IBM 3990-6 control unit 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.

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 2.1, “DFSMSdss SnapShot” on page 5. 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 a 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 an IBM 3990-6 control unit, 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.

2.2.2 Requirements

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 1 on page 7 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 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.

2.2.3 How It Works

Virtual concurrent copy can be used for DFSMSdss COPY and DFSMSdss DUMP. For COPY, if DFSMSdss SnapShot is not possible, DFSMSdss will use virtual concurrent copy if the concurrent copy (CC) keyword is present. For DUMP DFSMSdss will use virtual concurrent copy if the concurrent copy (CC) keyword is present.

To perform a virtual concurrent copy, a preallocated WSDS is required on one or more volumes in the same RVA subsystem as the source data set. See Chapter 4, “Working Space Data Sets” on page 37 for a complete description of WSDSs and how to allocate them.

After the source is snapped to the WSDS, the copy is logically complete and the data can be made available to the applications for update. The data in the WSDS can be copied or dumped to the target specified, using DFSMSdss as the data mover. Once the data has been physically copied to the target, the copy is physically complete. Finally the tracks associated with the WSDS are freed through DDSR. This process is illustrated in Figure 3 on page 12

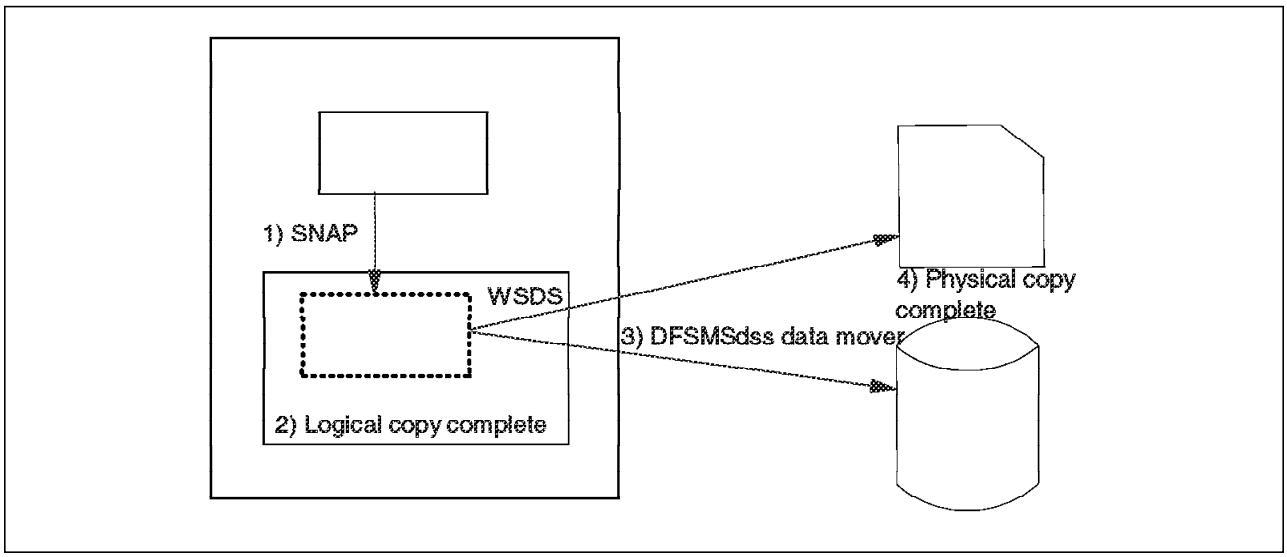


Figure 3. Virtual Concurrent Copy Operation

2.2.3.1 Virtual Concurrent Copy Flow

Here is a summary of the virtual concurrent copy flow:

1. Register session (this is an internal operation in the SDM)
2. Get serialization
3. Establish concurrent copy initialization
   Here the SDM recognizes that the device is an RVA and issues a SNAP to the WSDS instead of performing a concurrent copy initialization.
4. Release serialization (logical copy complete)
5. Move data (physical copy complete)
6. Terminate session

Once the snap has been taken, the serialization is released and the copy is said to be *logically complete*. DFSMSdss issues the ADR767I message, if you specify the NOTIFYCONCURRENT option, for each data set as it is logically copied. Once all of the data sets have been logically copied, the ADR734I message is issued and the data can be made available to the applications for update. Once the data is physically copied from the WSDS to the target data set, the ADR013I message is issued indicating that the physical copy or dump is complete.

Unlike concurrent copy there is no limit on the number of virtual concurrent copy sessions.

2.2.4 Impact on Net Capacity Load

The virtual concurrent copy function does not use space in the storage subsystem or the host. SnapShot uses the WSDS, which is purely virtual disk space.

The impact on NCL for WSDS space can vary from 0% for a data set that has no updates during the virtual concurrent copy operation, to 100% if the entire data set is updated before the operation is physically completed, that is, when DFSMSdss moves the data to the target tape or disk data set.

For further details on how to estimate the impact on NCL, see 4.4, “Size Requirements” on page 43.

2.2.5 Virtual Concurrent Copy Restrictions

The usage restrictions for virtual concurrent copy are the same as for concurrent copy:

- Virtual concurrent copy cannot be used if a point-in-time copy is absolutely required at a specific point in time, for example, at the end of the online day. If a virtual concurrent copy operation fails after DFMSdss signals that the virtual concurrent copy initialization is complete with this message:

  ```
  ADR734I CONCURRENT COPY INITIALIZATION SUCCESSFUL
  ```

  it not possible to recover the data at the point in time at which the virtual concurrent copy operation was started. The data may have been updated while the concurrent copy operation was in progress; in fact, after the ADR734I message is issued, the update activities on the data are resumed. The copy must be repeated at a new point in time. Snapped data in the WSDS is automatically cleaned up.
- If DFSMSdss invokes a utility such as IDCAMS REPRO or IEBCOPY for a data set COPY operation, virtual concurrent copy is not carried out for those data sets.
- The CONCURRENT keyword applies to all of the data being dumped or copied by the function under which it is specified. It cannot be applied to a subset of the data being processed.
- VM minivolumes are supported on RVA devices to the extent that they are supported by IXFP device reporting. See Chapter 12, “ Using DFSMSdss SnapShot in a VM/ESA Environment” on page 119.

2.2.6 DFSMSdss COPY Command

The DFSMSdss COPY command includes three functions:

- Copy a full volume
- Copy ranges of tracks
- Copy data sets

The COPY command supports virtual concurrent copy only when CONCURRENT is specified. The CONCURRENT keyword is optional on all types of COPY operations.

2.2.6.1 Full Volume and Tracks COPY

If the data that is being copied does not reside on hardware that supports concurrent copy or SnapShot, even if CONCURRENT is specified, the COPY operation will proceed using normal I/O methods, as if CONCURRENT is not specified. Figure 4 on page 14 shows the result of the operation.

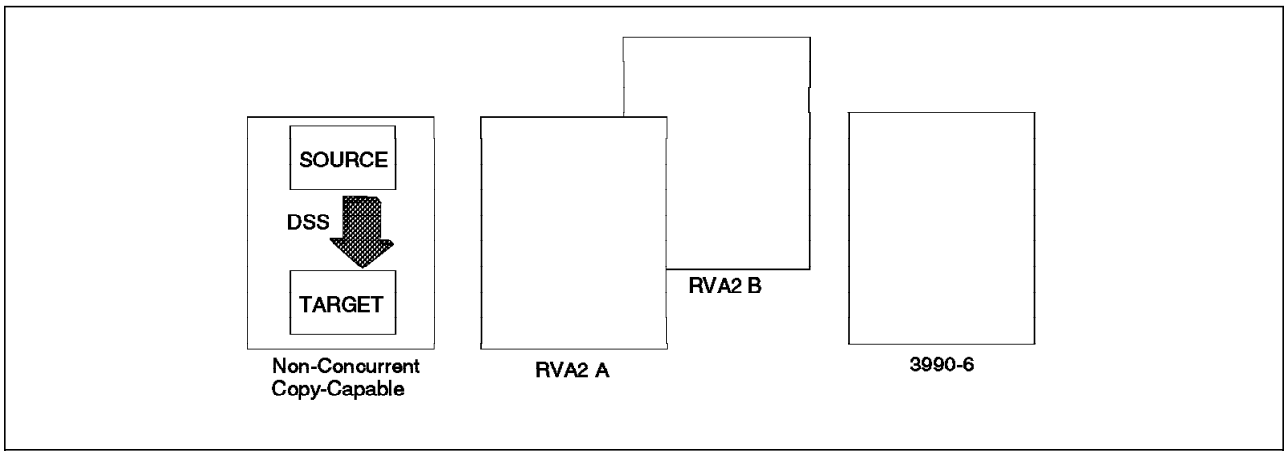


Figure 4. Full Volume Copy: Source Does Not Support Concurrent Copy

Figure 5 shows that if all of the requirements are met, DFSMSdss SnapShot is used to copy the source to the target, even if CONCURRENT is specified.

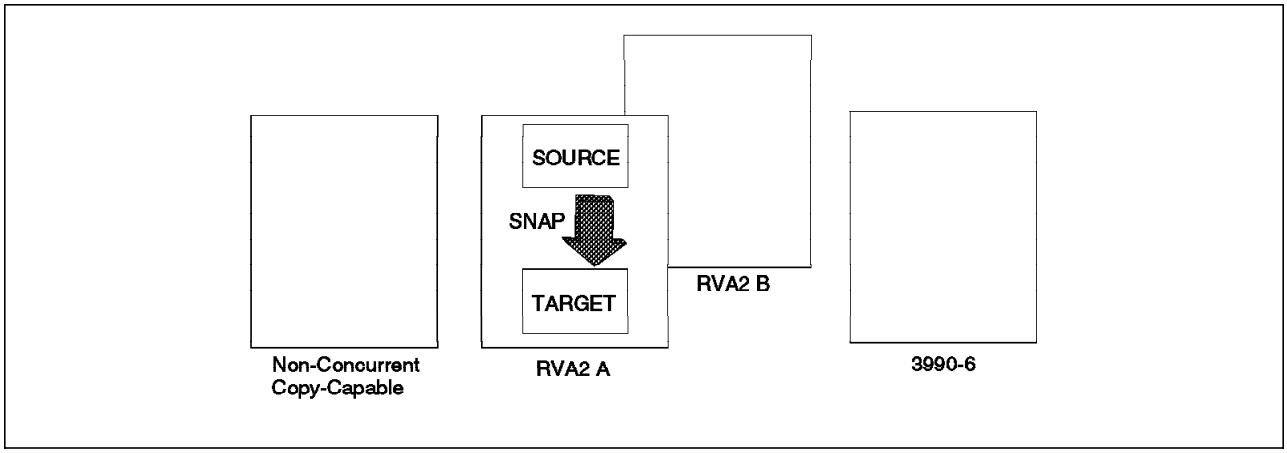


Figure 5. Full Volume Copy: DFSMSdss SnapShot

If the source volume is in an RVA, the target does not meet all of the requirements needed to execute a SnapShot function, and the CONCURRENT keyword is specified, DFSMSdss attempts to use virtual concurrent copy to copy the data. Figure 6 on page 15 shows the result of the operation.

If the source data set is in an RVA, DFSMSdss attempts to allocate the target data set on the same device type in the same partition of the same RVA, thus increasing the probability of being able to use DFSMSdss SnapShot to copy the data.

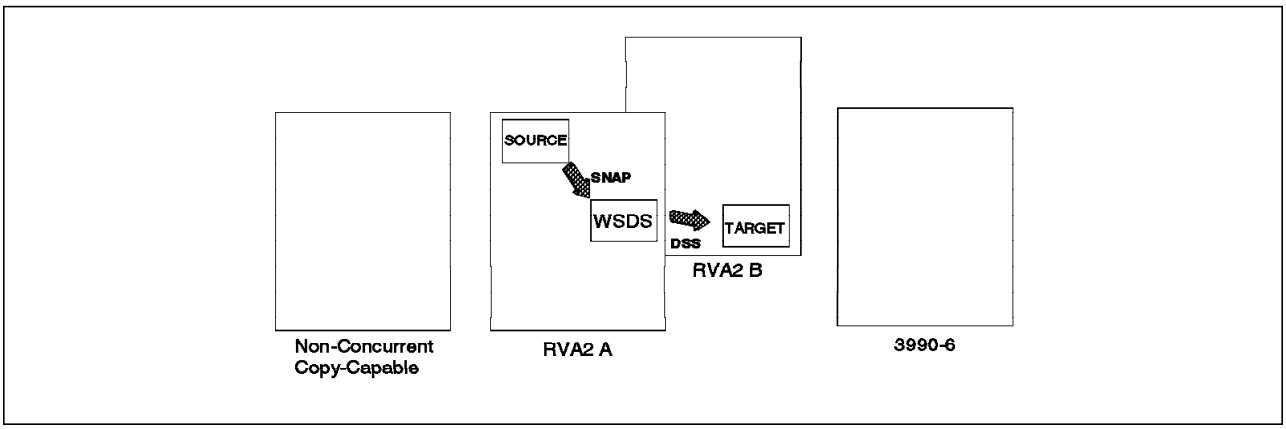


Figure 6. Full Volume Copy: Virtual Concurrent Copy

2.2.6.2 Logical Data Set COPY

Provided that the source data set is in an RVA and the target data set has the correct number of tracks on the same device type in the same partition of the same RVA and that data transformations (for example, reblocking) are not required, DFSMSdss uses DFSMSdss SnapShot, whether or not CONCURRENT is specified, to copy the data set.

If the data that is being copied does not reside on hardware that supports concurrent copy or SnapShot, even if CONCURRENT is specified, the COPY operations will proceed using normal I/O methods as if CONCURRENT is not specified. Figure 7 shows the result of the operation.

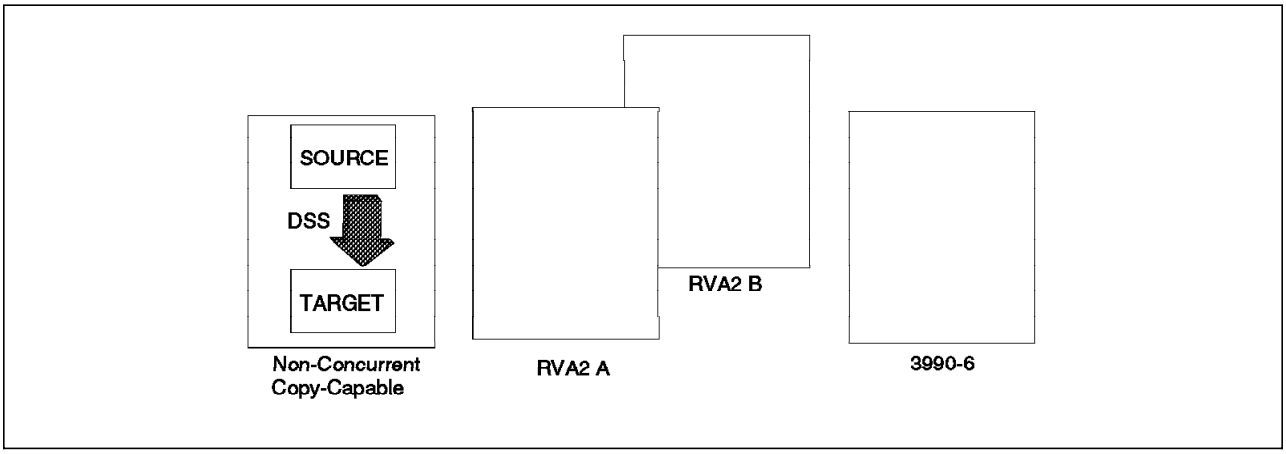


Figure 7. Logical Data Set Copy: Source Does Not Support Concurrent Copy or SnapShot

If the source data set is in an RVA and CONCURRENT is specified, but DFSMSdss must use a utility (for example, IDCAMS REPRO) to copy the data set, the copy of the data set will proceed using the utility as if CONCURRENT is not specified. Figure 8 on page 16 shows the result of the operation.

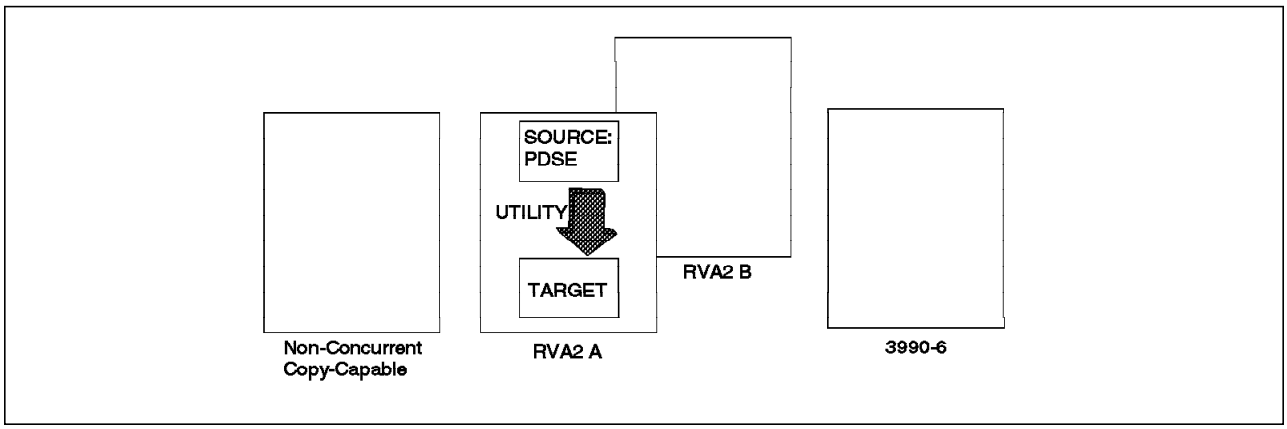


Figure 8. Logical Data Set Copy: DFSMSdss Uses a Utility for PDSE

If the source data set is in an RVA and DFSMSdss SnapShot cannot be used, DFSMSdss will use virtual concurrent copy to copy the data set, provided that CONCURRENT is specified. Figure 9 shows the result of the operation.

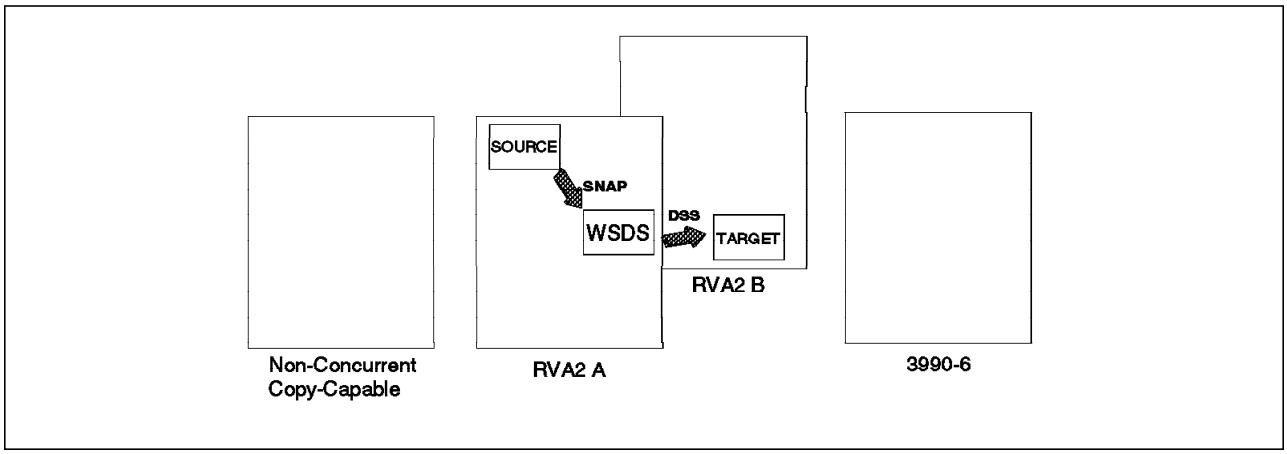


Figure 9. Logical Data Set Copy: Virtual Concurrent Copy

2.2.7 DFSMSdss DUMP Command

The DFSMSdss DUMP command includes four functions:

- Dump a full volume
- Dump ranges of tracks
- Physically dump data sets
- Logically dump data sets

The DUMP operation invokes virtual concurrent copy only when the CONCURRENT keyword is specified on the DUMP command.

2.2.7.1 Full Volume, Tracks, and Physical Data Set DUMP

When dumping a full volume or ranges of tracks, you must specify the source volume. Only one source volume is allowed. When physically dumping data sets, you must specify one or more source volumes.

If the source volume does not reside on hardware that supports concurrent copy or SnapShot, the dump operation proceeds using normal I/O methods as if CONCURRENT is not specified. Figure 10 shows the result of the operation.

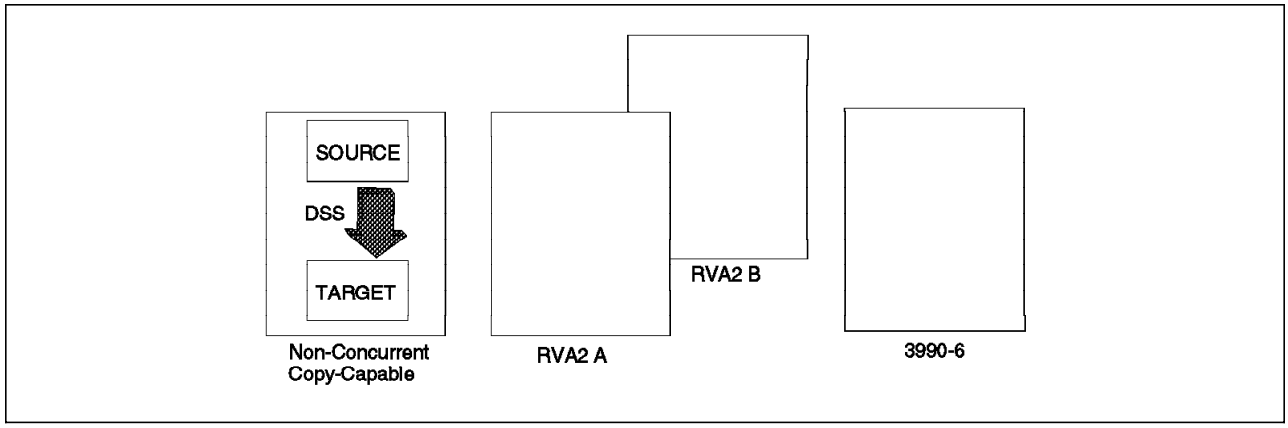


Figure 10. Physical Data Set Dump: Source Does Not Support Concurrent Copy or SnapShot

If the source volume is in an RVA, DFSMSdss attempts to use virtual concurrent copy to dump the data, as shown in Figure 11.

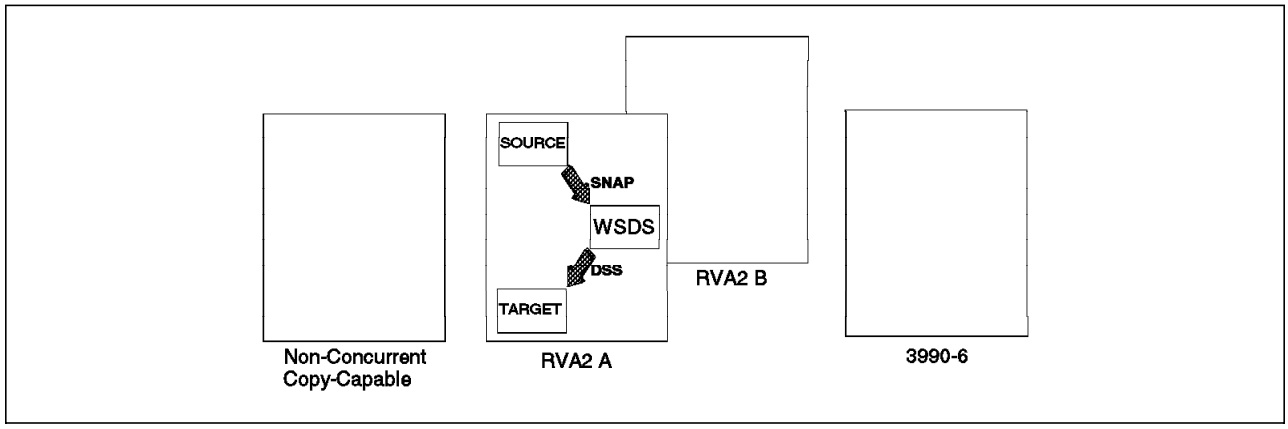


Figure 11. Physical Data Set Dump: Virtual Concurrent Copy

If virtual concurrent copy fails before the logical completion, the dump proceeds using normal I/O methods as if CONCURRENT is not specified.

If virtual concurrent copy fails after the logical completion, the dump fails.

2.2.7.2 Logical Data Set DUMP with CONCURRENT

When logically dumping data sets, you can optionally specify one or more source volumes. If the source volume does not reside on hardware that supports concurrent copy or SnapShot, the dump operation proceeds using normal I/O methods as if CONCURRENT is not specified (see Figure 10).

If the source data set is in an RVA, DFSMSdss uses virtual concurrent copy to dump the data set, as shown in Figure 11.

If virtual concurrent copy fails before the logical completion, the dump proceeds using normal I/O methods as if CONCURRENT is not specified.

If virtual concurrent copy fails after the logical completion, the DUMP ends for all data sets being processed with concurrent copy.

2.2.8 DFSMSdss COPYDUMP Command

The DFSMSdss COPYDUMP command enables you to make multiple copies of DFSMSdss-produced dump data.

The data to be copied, a sequential data set, can be on tape or a DASD volume, and copies can be written to a tape or a DASD volume.

There is no support for SnapShot or concurrent copy for the COPYDUMP command.

2.2.9 Summary

No new external commands are required to exploit the new functions of virtual concurrent copy and DFSMSdss SnapShot. Virtual concurrent copy and concurrent copy are compatible; anything that works with concurrent copy on the 3990 storage control works with virtual concurrent copy on the RVA.

For copy operations, DFSMSdss has a new data movement option. DFSMSdss always attempts to perform a DFSMSdss SnapShot, provided that the requirements are met. If DFSMSdss SnapShot is not possible, and the CONCURRENT COPY keyword is present, DFSMSdss tries to use a virtual concurrent copy snap.

If neither DFSMSdss SnapShot not virtual concurrent copy is possible, DFSMSdss copies the data using traditional data movement techniques.

For dump operations, virtual concurrent copy will be performed if the CONCURRENT keyword is specified and the requirements are met. There is no DFSMSdss SnapShot support for dump operations. If virtual concurrent copy is not possible, DFSMSdss dumps the data, using traditional data movement techniques.

Chapter 3. Implementing the New Functions

In this chapter we discuss how to take advantage of the new DFSMSdss functions.

One of the major benefits of DFSMSdss SnapShot is its ability to use DFSMSdss filtering. Thus you can:

- Back up related data sets together, providing the benefits of DFSMSdss serialization
- Avoid changing JCL when the data set names change

3.1 DFSMSdss Filtering

Functions such as filtering and partially qualified data set names are now available for data handled by SnapShot.

DFSMSdss filters data sets by:

- Specifying inclusion (INCLUDE) and/or exclusion (EXCLUDE) criteria for fully or partially qualified data set names
- Applying data set characteristics (BY) criteria to the data sets selected for inclusion and exclusion filtering.

A partially qualified data set name is one that uses asterisks (*) or a percent sign (%) to represent qualifiers or parts of qualifiers. A single asterisk (*) represents one qualifier. Additionally, it can be used to indicate that only a part of a qualifier has been specified; it can represent the first, last, middle, or first and last parts. A double asterisk (**) indicates that leading, trailing, or middle qualifiers either do not exist or play no role in the selection process. The percent sign (%) acts as a placeholder for a single character during data set name filtering.

After DFSMSdss has selected the data sets by applying INCLUDE and/or EXCLUDE criteria, you can further filter the data sets by applying BY criteria.

Using the BY parameter, you can filter on a lot of data set characteristics, among them:

ALLOC	Allocation type (cylinder, track)
CATLG	Whether a data set is cataloged or not
DATACLAS	SMS data class
DSORG	Data set organization
FSIZE	Data set size
MGMTCLAS	SMS management class
STORCLAS	SMS storage class

Other data set characteristics can be used; for a complete list and more information about setting up filtering criteria, see the *DFSMS/MVS V1R4 DFSMSdss Storage Administration Guide*.

© Copyright IBM Corp. 1998

Using meaningful naming conventions enables you to take full advantage of DFSMSdss filtering by data set name function. If the naming conventions reflect the actual use of your data sets, they can be used to identify groups of data sets that can be treated in the same way. With such conventions, you can run DFSMSdss functions against filtered data set names to select a large group of data sets with a single DFSMSdss COPY command.

When using the SNAP DATASET command, you have to code a separate command for each data set you need to snap.

The SNAP DATA SET command requires that the data set names be fully qualified so that, if one of the names changes, you must modify your SnapShot JCL commands to reflect the change. Figure 12 shows a traditional SIBBATCH used to snap three different data sets.

```
//SNAP     EXEC PGM=SIBBATCH
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSIN    DD *
 SNAP DATASET(SOURCE(DATASET(APP.DATABASE.DS1)) -
              TARGET(APP.DATABASE.DS1.SNAPPED) -
              TOLERATEENQFAIL(NO) -
              REPLACE(YES))
 SNAP DATASET(SOURCE(DATASET(APP.DATABASE.DS2)) -
              TARGET(APP.DATABASE.DS2.SNAPPED) -
              TOLERATEENQFAIL(NO) -
              REPLACE(YES))
 SNAP DATASET(SOURCE(DATASET(APP.DATABASE.DS3)) -
              TARGET(APP.DATABASE.DS3.SNAPPED) -
              TOLERATEENQFAIL(NO) -
              REPLACE(YES))
```

Figure 12. SIBBATCH Used to Snap Three Data Sets

Now that SnapShot can take advantage of DFSMSdss characteristics, multiple data sets or multiple volumes can be handled by a single DFSMSdss COPY command.

With the DFSMSdss wildcard filtering capability, there is no need to modify the JCL every time the data set name changes. Figure 13 shows how, even with partial qualified data set names, you can easily substitute the SIBBATCH listed in Figure 12 with a single DFSMSdss COPY command.

```
//DSS      EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//DASD1    DD UNIT=3390,VOL=(PRIVATE,SER=RVA001),DISP=OLD
//DASD2    DD UNIT=3390,VOL=(PRIVATE,SER=RVA002),DISP=OLD
//SYSIN    DD *
  COPY DATASET(INCLUDE(APP.DATABASE.DS%)) -
       INDDNAME(DASD1) -
       OUTDDNAME(DASD2) REPLACE
```

Figure 13. DFSMSdss Used to Copy Data Sets to Preallocated Target Data Sets

DFSMSdss does not serialize all of the data sets being considered during filter processing. It is possible that, between the time when DFSMSdss does the

filtering and builds the list of data sets to process and the time when DFSMSdss actually processes them, some or all of the data sets can be moved, deleted, or migrated, and the DFSMSdss operation may fail.

When DFSMSdss SnapShot is used, it is unlikely that the status of the selected data sets will have changed by the time they are processed because SnapShot is a faster process than DFSMSdss.

3.2 Serialization

DFSMSdss uses volume serialization and data set serialization. Volume serialization is accomplished by using the RESERVE macro. Data set serialization is accomplished by using the ENQ macro and the DFSMSdss DYNALLOC function.

For volume serialization, DFSMSdss issues the RESERVE macro to prevent changing the VTOC entries during the following operations:

- Physical dump
- Physical copy
- Restore (except for logical and physical data set)
- Print (except for data set)
- Defrag

The RESERVE macro is issued before processing begins on a volume and is released only when processing is completed.

For data set serialization, enqueues are taken on the data set name for the data set copy, data set dump, data set restore, defrag, print, compress, and release operations to prevent multiple, simultaneous updates to the same data set.

In either case, when DFSMSdss uses SnapShot rather than traditional I/O, the time to execute the copy functions is definitively less, so the time during which RESERVE and ENQ are held is greatly reduced, thus allowing the locked resource to be released sooner.

What happens if you are doing a DFSMSdss COPY with CC specified and the DFSMSdss filtering selects five data sets, two of them SnapShot capable and three ineligible to be snapped? When is the serialization for the five data sets released?

DFSMSdss can optimize serialization. In fact, for those data sets that are successfully established (snapped), the serialization is released as soon as the initialization of the virtual concurrent copy session is complete, that is, as soon as the logically completed message:

```
ADR734I CONCURRENT COPY INITIALIZATION SUCCESSFUL
```

is issued. For the other three data sets, ineligible to be snapped, serialization is held until the data has been moved for them, so the serialization is not held any longer than necessary.

Another concern you may have is related to sequence, that is, what happens, from a serialization point of view, if the DFSMSdss filtering returns five data sets in the following sequence:

1. SnapShot capable
2. DFSMSdss
3. DFSMSdss
4. DFSMSdss
5. SnapShot capable

The answer is that the order of the data sets does not really matter. For each data set, DFSMSdss does the following:

1. Gets the serialization
2. Attempts concurrent copy initialization
3. If concurrent copy initialization worked, releases the serialization; if not, holds it.

After DFSMSdss has completed the above steps for all data sets, it issues the ADR734I message, goes through the list of data sets again, and moves the data for each one. So, by the time the ADR734I message is issued, the serialization has been released for all data sets that have been successfully initialized for concurrent copy processing. So, even in this case, the serialization is not held any longer than necessary.

In summary, combining DFSMSdss and SnapShot characteristics results in:

- Consistency across related data sets
- No JCL changes required for DFSMSdss COPY
- Simpler maintenance of data set lists in JCL due to DFSMSdss filtering capability
- DFSMSdss releasing the locked resources quickly because of fast SnapShot processing.

3.3 Multivolume Data Sets

SnapShot can snap from a single volume source data set to a multivolume target data set and vice versa.

With SnapShot the source and target must be in the same RVA. Figure 14 on page 23 shows the result of a SnapShot operation for a non-VSAM data set.

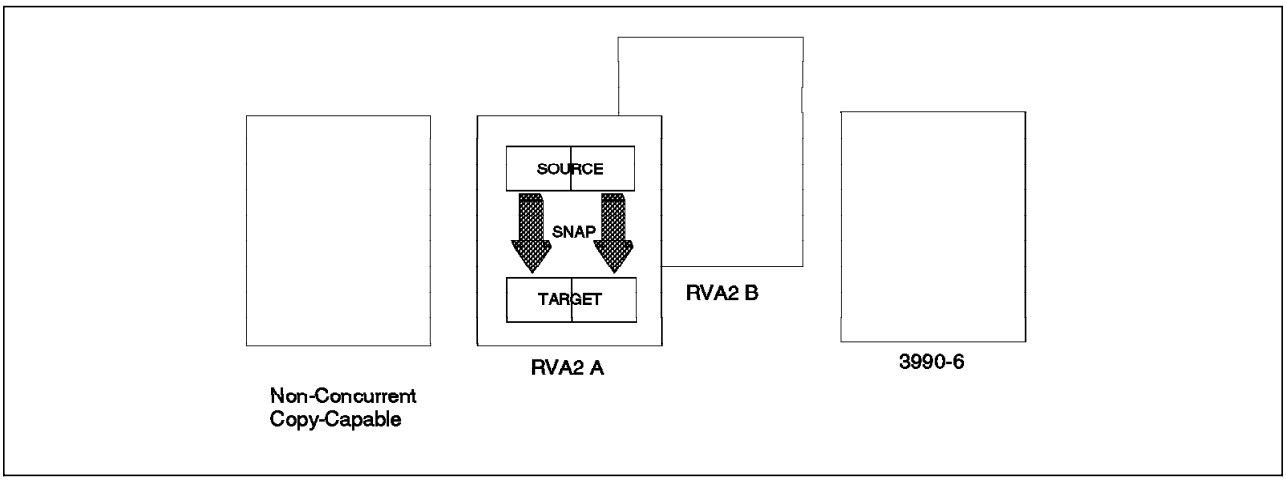


Figure 14. Multivolume Data Set in a Single RVA2

Note that in Figure 14, the target data set is graphically represented as a multivolume data set, but remember that SnapShot can consolidate, so a multivolume data set can be “snapped,” and the target could become a single volume data set.

Figure 15 shows the same SnapShot operation, this time for a VSAM KSDS, which has DATA and INDEX components on two different volumes, but both volumes belong to the same RVA. However, SnapShot can handle mulitvolume data sets that span RVAs as a datamover operation.
The target data set can be dynamically allocated or preallocated.

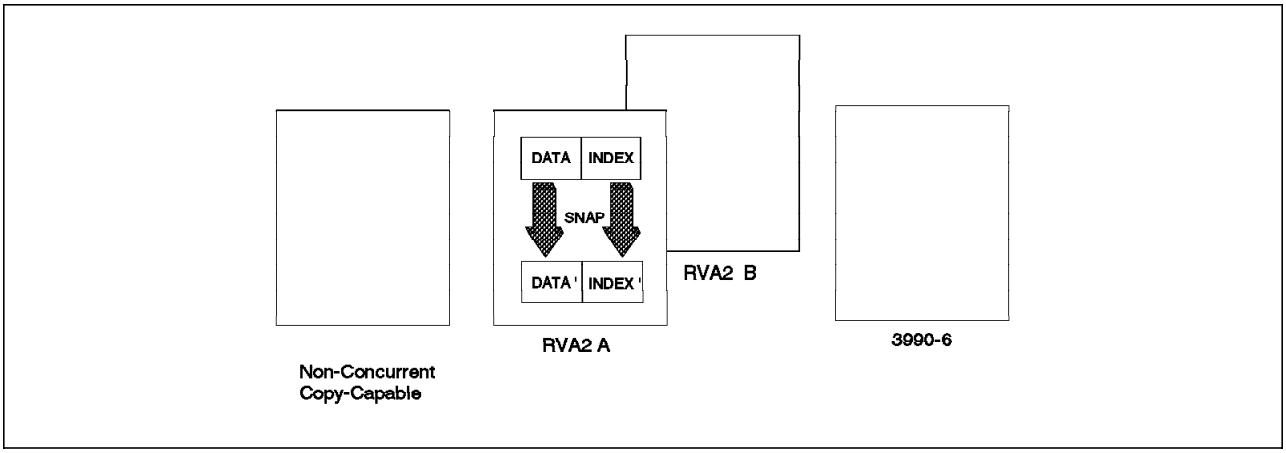


Figure 15. Multivolume KSDS in a Single RVA2

3.3.1 DFSMSdss SnapShot

When you try to perform a DFSMSdss SnapShot and you choose dynamic allocation, DFSMSdss attempts consolidation. The target data set is dynamically allocated onto the RVA that has the largest part of the source component. A data mover is invoked for components moved between different RVAs; that is, if part of the multivolume data set resides on a non-SnapShot-capable device, the entire data set is moved by using traditional DFSMSdss copy. If the SNAP command cannot resolve the source and the target to the same subsystem, a data mover, if it is specified, is called to complete the snap request. If a data mover is not specified, and, as in this case, 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.

When you choose to preallocate the target, obviously consolidation can be avoided. For example, a VSAM data set's DATA component may be on one RVA and the INDEX component may be on a different RVA. Each component is separately copied by traditional DFSMSdss copy onto the same RVA or toward two separate RVAs, as shown in Figure 16.

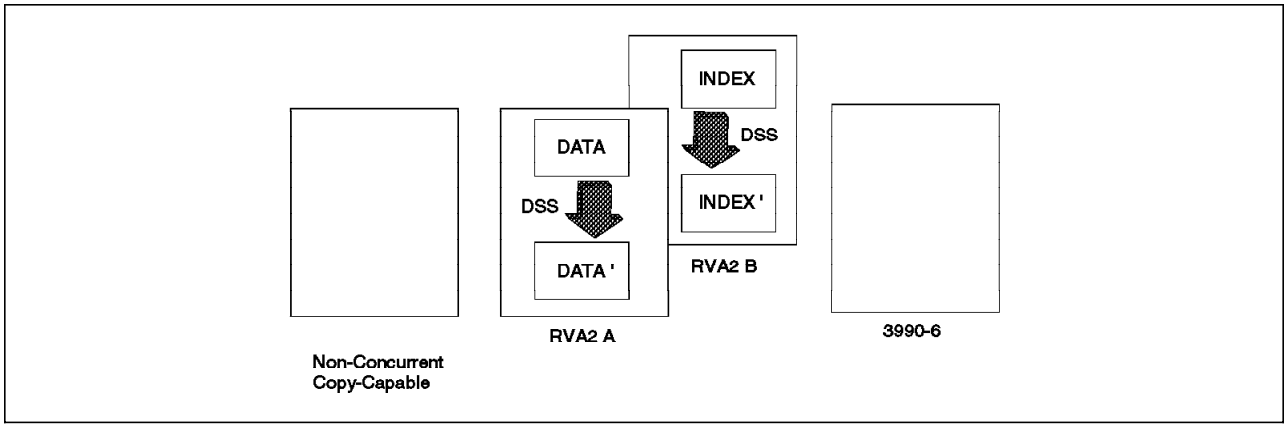


Figure 16. Multivolume KSDS Spanning Two RVA2s

So we can say that, to use DFSMSdss SnapShot, all parts of the source and target data sets must be in the same RVA; otherwise traditional DFSMSdss copy is executed to move all the parts, even if they are SnapShot capable. This is the All or Nothing rule (see Figure 17).

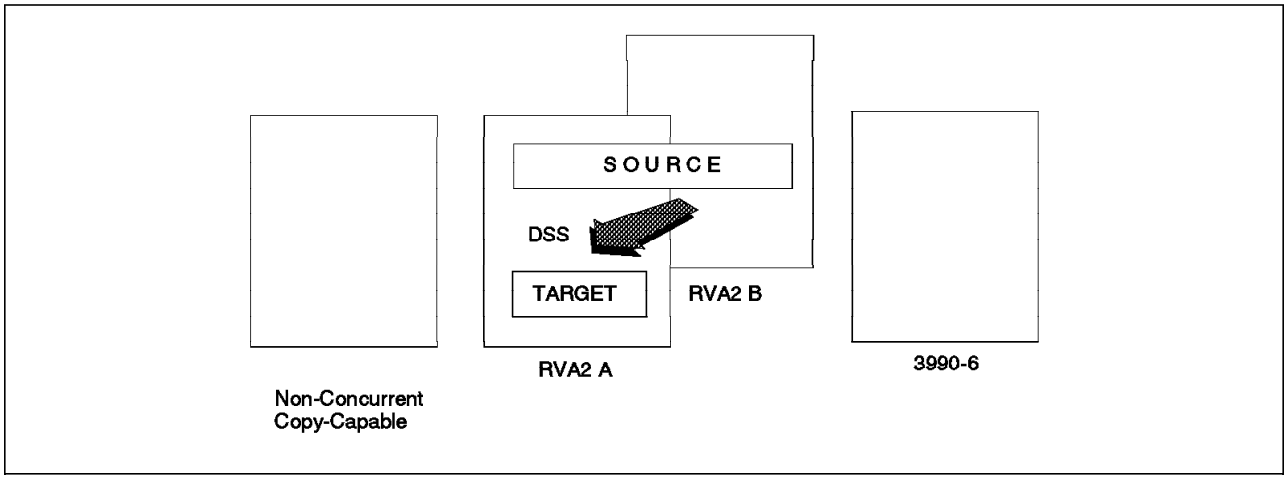


Figure 17. Multivolume Data Set Spanning Two RVA2s

With DFSMSdss the target data sets are not always placed on the same number of volumes as the original data set. Instead DFSMSdss tries to allocate the new data set on as few volumes as possible, so consolidation from multivolume to single volume may result.

With DFSMSdss, to logically copy entire multivolume data sets, you do not have to specify all of the volumes in the LOGINDDNAME or LOGINDYNAM volume list if you also specify the SELECTMULTI keyword.

Table 2 on page 25 presents the criteria that determine the results of a SnapShot or DFSMSdss COPY job.

Table 2. Criteria Determining the Result of a SnapShot or DFSMSdss Job

Job Submitted	Multivolume	Source Entirely SnapShot capable	Source and Target SnapShot Friendly (1)	Result
SnapShot	No		No	Traditional DFSMSdss copy
SnapShot	No		Yes	Traditional SnapShot
SnapShot	Yes		No	Traditional DFSMSdss copy
SnapShot	Yes	All	Yes	Traditional SnapShot
SnapShot	Yes	Part	Yes	SnapShot + DFSMSdss data mover
DFSMSdss Copy	No		No	Traditional DFSMSdss copy
DFSMSdss Copy	No		Yes	DFSMSdss SnapShot
DFSMSdss Copy	Yes		No	Traditional DFSMSdss copy
DFSMSdss Copy	Yes	All	Yes	DFSMSdss SnapShot
DFSMSdss Copy	Yes	Part	Yes	All multivolume parts moved by traditional DFSMSdss copy

Notes: 1: "SnapShot friendly" means same device type, RVA, partition. No changes to data set format. Utilities not required.

Table 2 is merely a simple summary. If the source and target data sets exist on device types other than the 3380 or 3390, the SnapShot command fails and the data mover is not invoked.

3.3.2 Virtual Concurrent Copy

With 3990 concurrent copy, if you process a multivolume data set for which some volumes are on a concurrent-copy-capable subsystem and some on a non-concurrent-copy-capable subsystem, DFSMSdss retains serialization on the data set until processing is complete.

If you process multiple data sets or volumes, DFSMSdss releases serialization immediately for those data sets or volumes that are on a concurrent-copy-capable subsystem. DFSMSdss retains serialization for those data sets or volumes that are on a non-concurrent-copy-capable subsystem, in exactly the same way as if you had not used the CONCURRENT keyword.

Virtual concurrent copy offers much more flexibility for handling multivolume data sets than either SnapShot or DFSMSdss SnapShot.

Figure 18 on page 26 shows the result of a DFSMSdss COPY or DFSMSdss DUMP command, with the CONCURRENT keyword specified for a multivolume KSDS spanning two RVAs. As you can see, each component of the cluster is first separately snapped to a different WSDS, and then virtual concurrent copy DUMP moves the KSDS to tape. The same applies for non-VSAM multivolume data sets; that is, each part of the multivolume data set is first snapped to a different WSDS, and then DFSMSdss moves all the parts to tape.

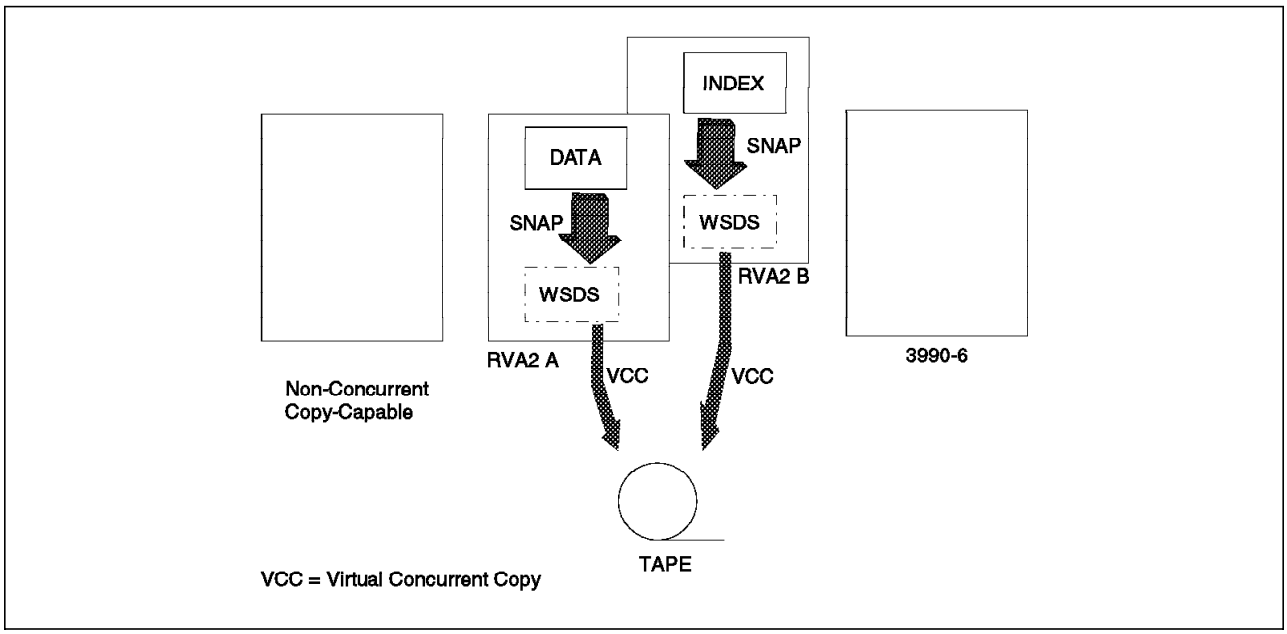


Figure 18. Multivolume Data Set Spanning Two RVA2s: Virtual Concurrent Copy

Figure 19 shows the result of a DFSMSdss COPY or DFSMSdss DUMP command, with the CC keyword specified for a multivolume data set spanning an RVA and a CC-capable 3990. As you can see, the part of the multivolume data set that resides on a SnapShot capable device is snapped to a WSDS, and then the virtual concurrent copy DUMP moves it to tape. The part of the multivolume data set that resides on a concurrent-copy-capable device is moved directly to tape by a 3990 concurrent copy DUMP operation.

The same applies for VSAM multivolume data sets; that is, the component of the multivolume KSDS that resides on a SnapShot capable device is snapped to a WSDS, and then the virtual concurrent copy DUMP moves it to tape. The component of the cluster that resides on a concurrent-copy-capable device is moved directly to tape by a 3990 concurrent copy DUMP operation.

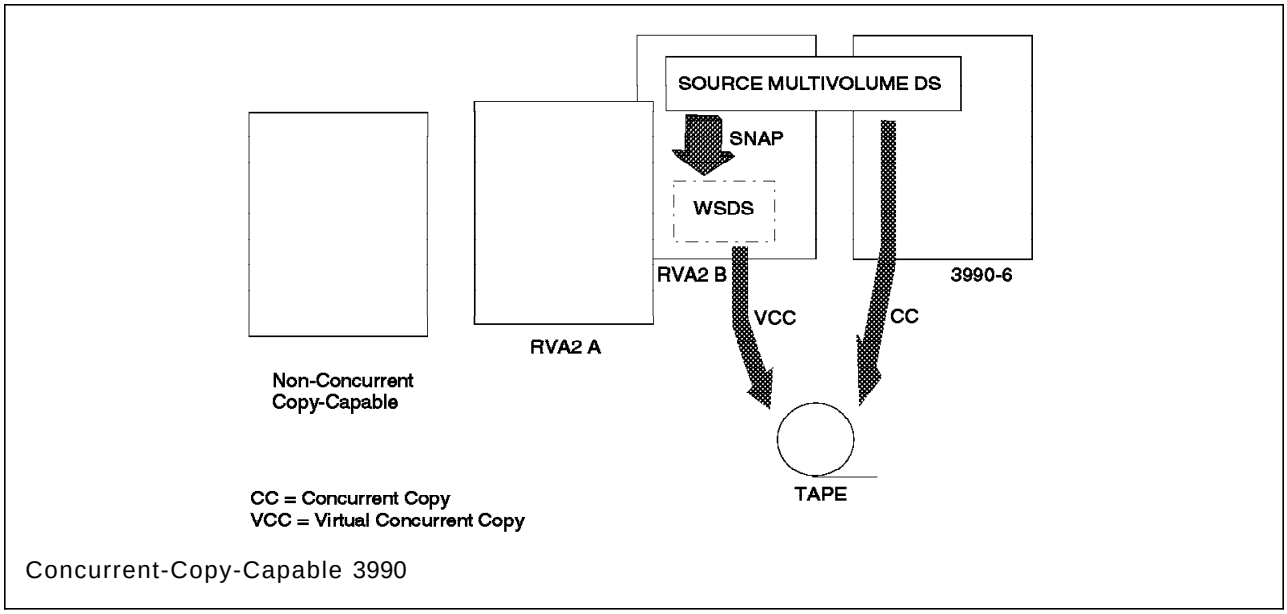


Figure 19. Multivolume Data Set Spanning an RVA and

3.4 Supported Data Sets

Table 3 lists the data sets that are supported by SnapShot, DFSMSdss COPY (with or without the CONCURRENT keyword), and DFSMSdss SnapShot.

DFSMSdss DUMP can be used for all data set formats.

It is clear from Table 3 that DFSMSdss and DFSMSdss SnapShot are superior copy techniques that you can use without modifying your applications.

Table 3. DFSMSdss COPY: Supported Data Sets

	SnapShot 1.2 SNAP DATASET	**DFSMSdss Using Traditonal 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	**N**	Y	**N**
VSAM data sets with IMBED or REPLICATE	**N**	Y	**N**
Alternate indexes	**N**	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 (1)	Y
Linear data sets	Y	Y	Y
Relative record data sets (RRDSs)	Y	Y	Y
VSAM variable record RRDSs (VRRDSs)	Y	Y	Y

Legend: **N=** No **Y=** Yes
Notes:1: Valid only in the same RVA, see 3.3, “Multivolume Data Sets” on page 22 for a complete discussion. Currently for VSAM KSDSs and RRDSs where the target is multivolume, DFSMSdss does not invoke DFSMSdss SnapShot or virtual concurrent copy. This restriction is expected to be temporary; details are documented in APAR OW32415.

As you can see from Table 3, DFSMSdss COPY supports most data sets, with these few exceptions:

- The logical data set copy function does not support HFS data sets.
- The copy function is not supported for VSAM compressed extended format data sets copied to a non-SMS-managed target.
- When you perform a logical copy operation of a VSAM compressed data set, the target data set allocation must be consistent with the source data set allocation.

3.5 Virtual Concurrent Copy

Virtual concurrent copy works in the same way for both to DFSMSdss COPY and DUMP. The source must be SnapShot capable; however, the target can be a non-RVA. Virtual concurrent copy also allows the data set format to change as part of the copy or dump operation. This is because virtual concurrent copy uses WSDSs, which act as placeholders for some tracks, used as the target of a snap. The data movement to the target data set takes place from the WSDS, while the source data is available to the application for update.

The CC keyword must be specified and WSDS space must be available. For a full description of WSDSs, see Chapter 4, “Working Space Data Sets” on page 37.

3.5.1 Examples

Applications that use 3990 concurrent copy today can now operate, without modification, with data that is stored on an RVA subsystem.

Figure 20 on page 29 shows an example of a job that can be used to test the new DFSMSdss SnapShot support for DSS COPY with concurrent copy and DSS FULL VOLUME PHYSICAL DUMP FULL.

DSS FULL VOLUME PHYSICAL RESTORE restores your data to the time when the physical full volume dump was taken.

```
//************************
//*   DSS COPY WITH CC   *
//************************
//COPY    EXEC PGM=ADRDSSU,REGION=4096K
//SYSPRINT DD SYSOUT=*
//SYSMDUMP DD DSN=SVF.DUMP01.SYSMDUMP,DISP=OLD
//DDIN     DD DISP=SHR,VOL=SER=RVA001
//DDOUT    DD DISP=SHR,VOL=SER=RVA002
//SYSIN    DD *
  COPY DS(INCLUDE(**)EXCLUDE(SYS1.**)) -
       INDD(DDIN) -
       ALLDATA(*) -
       ALLEXCP -
       CANCELERROR -
       CC -
       SPHERE -
       TOL(ENQF) -
       RENAMEU(*.*.P3.**,*.*.CV2.**) -
       ODD(DDOUT)
//*
//****************************************
//* DSS FULL VOLUME PHYSICAL DUMP WITH CC *
//****************************************
//DUMP    EXEC PGM=ADRDSSU,REGION=4096K
//SYSPRINT DD SYSOUT=*
//SYSMDUMP DD DSN=SVF.DUMP01.SYSMDUMP,DISP=OLD
//LBGS03    DD DISP=OLD,VOL=SER=RVA002,UNIT=3390
//TAPE     DD DSN=F407.ISVF.CLAIMRV2,DISP=(,PASS),
//       LABEL=(1,SL),UNIT=TAPE,VOL=(,,,100)
//SYSIN    DD *
  DUMP FUL -
       INDD(LBGS03) -
       ALLDATA(*) -
       ALLEXCP -
       CC -
       OUTDD(TAPE)
//*
//************************************
//* DSS FULL VOLUME PHYSICAL RESTORE *
//************************************
//REST    EXEC PGM=ADRDSSU,REGION=4096K
//SYSMDUMP DD DSN=SVF.DUMP01.SYSMDUMP,DISP=OLD
//SYSPRINT DD SYSOUT=*
//TAPE     DD DSN=F407.ISVF.CLAIMRV2,DISP=(OLD,DELETE),
//       LABEL=(1,SL),UNIT=TAPE,VOL=(,,,100)
//SYSIN    DD *
  RESTORE FUL -
       IDD(TAPE) -
       CPYV -
       ODY(39GS03)
/*
```

Figure 20. Virtual Concurrent Copy for DFSMSdss COPY and DUMP

3.5.2 Virtual Concurrent Copy Performance

The performance of a concurrent copy operation is not usually a problem, because applications can still access the data, but for a matter of seconds while the SDM initializes the session. There are some impacts that you must consider, however.

Virtual concurrent copy increases data availablility and enables you to take the time taken to actually move the data out of the critical path.

In a concurrent copy environment, if you send the output from the concurrent copy operations directly to tape, the duration of the tape copy operations may be elongated to protect overall system performance. In many installations, utilization of tape drives during overnight processing is very high, and the increase in the elapsed time of the tape operations could delay otherwise unrelated work.

Virtual concurrent copy performs in a similar way to concurrent copy; however, there are some benefits to virtual concurrent copy. Because the data is copied from the WSDS to the target data set, there is no contention on the source data set between the DFSMSdss job and the applications. If the source data has not changed, the source and WSDS target actually reside on the same physical tracks on the array, but the architecture of the RVA ensures that this does not present a performance problem. Frequently referenced data resides in cache, so there is a strong likelihood that the application will realize a cache hit. Because data is spread across many disks in the array, there is little physical contention on the back-end disks.

In addition, the RVAT82 provides a total of eight host paths, which significantly increases the number of concurrent DFSMSdss jobs that can run without affecting the performance of the other applications.

Virtual concurrent copy also has the advantage that neither cache space nor expanded storage data space is consumed for sidefile use.

3.6 DFSMSdss Considerations

DFSMSdss processing differs from SnapShot processing in several respects, which may be significant from an operational perspective.

3.6.1 Extending the Target Data Set

If the target is too small to contain the source, SnapShot extends the target data set. SnapShot uses normal extend processing functions. An SMS multivolume data set extends to the current primary volume if the space is available, or to the next candidate volume.

When DFSMSdss COPY is used to move data to a preallocated data set, you must specify the REPLACE keyword, and the preallocated data set name must be identical to the source data set name. In an SMS environment, you cannot copy to preallocated data sets because SMS does not support duplicate data set names.

With DFSMSdss there are some cases where you need to preallocate the target to restore data sets that contain device-dependent information. If you restore indexed sequential, unmovable, direct, or absolute track data sets by

preallocating them, the size and the location of the extents on the dump volume and on the restore volume should match. If the preallocated data set is too small, DFSMSdss deletes it and reallocates a new data set. The allocation may fail, or it may result in a new data set with a different configuration.

3.6.2 Number of Extents

With SnapShot you have to reduce the number of extents of all source data sets because the time required to snap a data set does not depend on the size of the source data set but on the number of extents it has.

This is no longer valid for DFSMSdss SnapShot support, so there is no need to reduce the number of extents of the source data set. Table 4 shows how the time to copy a VSAM source data set changes when the number of extents changes.

Table 4. Time to COPY with Different Number of Extents

Number of VSAM Source Data Set Extents	Direct SnapShot (sec)	DFSMSdss SnapShot (sec)
1	4	2
21	9	2
91	22	2

The time is independent of the number of the extents for COPY with DFSMSdss SnapShot because DFSMSdss can process multiple extents in a SnapShot operation.

3.6.3 COPYVOLID

With the SNAP VOLUME command, COPYVOLID(YES) indicates that the volser of the source is also copied to the target, and the target is varied offline. The target becomes a complete identical copy of the source. Therefore you cannot access the target from the system that has made the copy because the volser will be a duplicate, but you can vary the target online and use it with another MVS that shares access to the RVA.

When you choose the COPYVOLID(NO) option, the volume is still completely copied, but the target volser is not to be 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.

Along with COPYVOLID(NO) you can choose to code the CONDVOL(LBL) option, which enables you to snap a volume and give the target volume a different volser, thus allowing it to remain online. Table 5 shows the results on the target volume of a VOLUME SNAP with all possible COPYVOLID combinations.

Table 5. SNAP VOLUME COPYVOLID

	Target Volser	Target VTOC	Target VVDS
COPYVOLID(YES)	A	A	A
COPYVOLID(NO)	B	B	B
COPYVOLID(NO) CONDVOL(LBL)	B	A	A

Note: *A* identifies the names related to the source volume and *B* identifies the names related to the target volume.

For DFSMSdss COPYVOLID specifies that the volser from the input DASD volume is to be copied to the output DASD volume. This applies to COPY FULL operations and to COPY TRACKS operations if track 0 is copied. If a full-volume copy is done of an SMS managed input volume, the COPYVOLID keyword is required.

3.6.4 Physical and Logical Dumps

A physical dump moves data at the track-image level, thus eliminating the need to filter the names of data sets to be dumped.

Doing a DFSMSdss CC DUMP results in a volume snap under the cover, rather than snaps of individual data sets, so that the snap that corresponds to the time to logical completion is much faster than a regular DFSMSdss DUMP.

As with DFSMSdss, even with SnapShot, when an entire volume is selected instead of individual data sets, catalog checking is not needed, so the process is faster to snap a volume than to snap data sets.

Table 6 shows the results to dump a volume, by using the OPT(1) DFSMSdss CC keyword and without a background workload, that contains sequential data sets that are 1.6 MB each.

Table 6. Physical and Logical Dumps

	Number of Data Sets	**Time to Logical Completion (sec)**	**Elapsed Time (sec)**
PHYSICAL	100	1	134
LOGICAL	100	14	187
PHYSICAL	200	1	275
LOGICAL	200	28	303

3.6.5 Coexistence Considerations

DFSMSdss DUMP data sets created using the virtual concurrent copy function are identical to DFSMSdss DUMP data sets created using the existing concurrent copy function. Thus they can be restored by all levels of DFSMSdss and DFDSS V2R5.0 as well, with minor exceptions:

- DFDSS V2R5.0 cannot restore PDSEs dumped with DFSMSdss V1R1.0 or higher releases using logical data set DUMP with the CC keyword specified.
- DFSMSdss V1R2.0 and lower releases cannot restore extended addressability VSAM data sets dumped with DFSMSdss V1R3.0 or higher releases, using logical data set DUMP (it does not matter whether or not the CC keyword is used).

See the *DFSMS/MVS V1R4 DFSMSdss Storage Administration Reference*, Appendix A, “Compatibility Considerations” for complete details about restoring DFDSS and DFSMSdss dumps with other levels of DFDSS or DFSMSdss. Toleration PTFs are not required.

3.6.6 Authorization for DFSMSdss SnapShot Functions

To get API services (see 2.1.3, “How It Works” on page 7), the caller must be APF authorized or RACF authorized to use the command.

These are the optional RACF FACILITY class profiles:

- STGADMIN.ANT.SNAPSHOT.COMMANDS
- STGADMIN.ANT.SNAPSHOT.SQUERY

If the profile does not exist, access is granted only if the user is APF authorized.

3.7 DFSMSdss Messages for DFSMSdss SnapShot and Virtual Concurrent Copy

In this section we cover the most useful DFSMSdss messages issued when you use DFSMSdss SnapShot. Using an automation product, you can monitor these messages and take the appropriate actions.

3.7.1 Messages Issued When DFSMSdss SnapShot Copy Is Invoked

When DFSMSdss SnapShot is used and SnapShot is internally successfully invoked, a new DFSMSdss message,

```
ADR806I Volume/Tracks/Data set COPIED USING SNAPSHOT FUNCTION
```

is issued to indicate that the SnapShot function is being used to “instantly” copy the requested volume, tracks, or data set.

If DFSMSdss invokes SnapShot internally, but SnapShot fails, the message,

```
ADR736E AN ERROR CONDITION WAS DETECTED BY THE SYSTEM DATA MOVER.
DIAGNOSTIC INFORMATION: RC=6033 (X'1791')
```

is issued to indicate that DFSMSdss cannot complete the requested SnapShot function because SDM has detected an error condition. The error can be a DFSMSdss internal error, an SDM internal error, an IXPF internal error, or a configuration error, so a traditional data movement technique is used.

The SDM return code and reason code are supplied as diagnostic aids. If the ANTRQST API is called and it returns an unexpected return code, this message is issued along with the appropriate return code and reason code to explain the reason of the failure:

```
ADR701E UNEXPECTED RETURN CODE FROM ANTRQST
```

3.7.2 Messages Issued When Virtual Concurrent Copy Is Invoked

Figure 21 on page 34 shows the joblog output for a DFSMSdss COPY with CC.

```
//*****************************************
//* DSS COPY WITH CC and DYNAMIC ALLOCATION *
//*****************************************
//COPY    EXEC PGM=ADRDSSU,REGION=5M
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
IEF373I STEP/COPY    /START 1998043.1512
IEF374I STEP/COPY    /STOP  1998043.1512 CPU 0MIN 00.17SEC SRB 0MIN 00.01
IEF375I  JOB/CLAIMCT /START 1998043.1512
IEF376I  JOB/CLAIMCT /STOP  1998043.1512 CPU 0MIN 00.17SEC SRB 0MIN 00.01
PAGE 0001  5695-DF175  DFSMSDSS V1R3.0  DATA SET SERVICES   1998.043
 COPY TRACKS(45,0,45,5)   /* SOURCE TRKS            */ -
      OUTTRACKS(30,0)     /* TARGET TRKS            */ -
      INDYNAM(49GS01)     /* ALLOC DYNAM            */ -
      OUTDYNAM(49GS02)    /* ALLOC DYNAM            */ -
      CC                  /* CONCURRENT COPY        */ -
      CANCELERROR         /* STOP ON ERROR          */ -
      WRITECHECK          /* VERIFY DATA WRITTEN    */
ADR101I TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'COPY '
ADR109I 1998.043 15:12:11 INITIAL SCAN OF USER CONTROL STATEMENT COMPLETE
ADR014I 1998.043 15:12:11 ALL PREVIOUSLY SCHEDULED TASKS COMPLETED.
ADR016I RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR006I 1998.043 15:12:11 EXECUTION BEGINS
ADR806I TRACKS COPIED USING SNAPSHOT FUNCTION.
ADR734I 1998.043 15:12:14 CONCURRENT COPY INITIALIZATION SUCCESSFUL
        SERIALIZATION FOR THIS DATA IS RELEASED IF DFSMSDSS HELD IT.
        THE INTERMEDIATE RETURN CODE IS 0000
ADR013I 1998.043 15:12:14 TASK COMPLETED WITH RETURN CODE 0000
ADR012I 1998.043 15:12:15 DFSMSDSS PROCESSING COMPLETE. HIGHEST RC IS 0.
```

Figure 21. Messages Received When Virtual Concurrent Copy Is Invoked

When virtual concurrent copy is specified, you can receive the following messages:

ADR734I

This message indicates that the initialization of a virtual concurrent copy session has completed.

ADR736E/ADR735W

This message indicates that the initialization of a virtual concurrent copy session has not completed successfully for the specified data set or volume.

When the virtual concurrent copy session has been initialized successfully, you receive this message:

ADR734I

If the virtual concurrent copy session used SnapShot to copy the data, you receive this message:

ADR806I TRACKS COPIED USING SNAPSHOT FUNCTION

When the operation is physically completed, you receive the normal DFSMSdss task completed message, that is:

ADR013I TASK COMPLETED WITH RETURN CODE 0000

When the virtual concurrent copy initialization was unsuccessful, ADR734I is preceded by one or more ADR735W or ADR737W to indicate which data was not

successfully initialized and why. If you are doing a logical data set operation, ADR734I is also preceded by:

ADR801I

indicating the number of data sets that passed filtering and are therefore selected for further processing. If DFSMSdss cannot complete the virtual concurrent copy function, this message is issued with RC=8(X'08'):

ADR736E

If the virtual concurrent copy function does not complete because hardware supporting SnapShot is not available, this message is issued with RC=2:

ADR735W

If the virtual concurrent copy function does not complete because SnapShot software support not available, this message is issued with RC=16:

ADR735W

If the virtual concurrent copy function does not complete because of WSDS problems, this message is issued with RC=614x (X'180x'):

ADR736E

For virtual concurrent copy, if the ANTRQST API is called and it returns an unexpected return code, this message is issued along with the appropriate return code and reason code to explain the reason of the failure:

ADR701E UNEXPECTED RETURN CODE FROM ANTRQST

So, we can say that, if you already use 3990 concurrent copy and an automation program is checking for successful and/or unsuccessful initialization, you can let your automation procedure remain unchanged and at the same time get all of the benefits of DFSMSdss SnapShot. If virtual concurrent copy is your first concurrent copy implementation, you can easily code your automation program to check for the above messages.

3.8 Summary

Here is a brief summary of DFSMSdss DUMP and COPY commands with SnapShot support:

- The format of the DFSMSdss COPY command is unchanged when using DFSMSdss SnapShot.
- DFSMSdss DUMP changes the format of the data so DFSMSdss SnapShot cannot be used. When the CONCURRENT keyword is specified, SnapShot is used to create the point-in-time copy, and DFSMSdss then moves the data to tape.
- In the case of DFSMSdss COPY, even if the CC keyword is specified, DFSMSdss can call SnapShot internally and transparently to the user and, if all the requirements are met, execute a DFSMSdss SnapShot instead of a traditional DFSMSdss COPY, that is, the source is directly "snapped" to the target, greatly reducing job execution time.
  - So, when DFSMSdss SnapShot can be used and the SnapShot is successfully invoked, this DFSMSdss informational message is issued to indicate that the SnapShot function has been used:

```
ADR806I Volume/Tracks/Data set COPIED USING SNAPSHOT FUNCTION
```

- A WSDS is not used.
- When DFSMSdss COPY with CC is specified and a DFSMSdss SnapShot is performed, you receive the logically completed and physically completed DFSMSdss messages at the same time:

  ```
  ADR734I 1998.043 15:12:14 CONCURRENT COPY INITIALIZATION SUCCESSFUL
  ADR013I 1998.043 15:12:14 TASK COMPLETED WITH RETURN CODE 0000
  ```

- If DFSMSdss COPY with CC is required, but not all of the requirements are met, virtual concurrent copy is done, that is, the source is first "snapped" to the WSDS, and then the data set is copied by DFSMSdss COPY from the WSDS to the target. The ADR734I message is received when the data set has been snapped to the WSDS and is considered *logically complete*. The ADR013I message is received once the data set has been copied to the target data set and is *physically complete*.

Chapter 4. Working Space Data Sets

In this chapter we explain how to allocate WSDSs, discuss their interaction with the catalog search and the SDM, and consider some resource requirements and performance issues.

To perform a virtual concurrent copy, space is required on one or more volumes in the same RAMAC Virtual Array subsystem as the source data set. During a virtual concurrent copy, data is “snapped” from the source location to an intermediate location (the WSDS) and the gradually copied to the target location through normal I/O methods.

The operation is logically complete after the source data is “snapped” to the intermediate location. The operation is physically complete after the data is moved to the target media. The WSDS acts as a target data set for a snap operation. Thus a point-in-time copy of the data can be captured and the source data can be available for update, once the logical copy is complete (the snap to the WSDS). The data can be updated during the actual movement of the data from the WSDS to the output tape or disk data set.

When the WSDS is used as an interim target for the snap, the target of a virtual concurrent copy can be on a different RVA from the source.

WSDSs are not usable data sets, other than for DFSMSdss. You must preallocate them.

As the target of a snap, the WSDS does not take up any back-end space on its own. If the source tracks are updated, the source updates take up additional space. After the snapped tracks are written out, the SDM releases the back-end space, in order to reduce the NCL. All space for a data set is released once the data set physical copy is complete.

4.1 Allocation

Allocation of some number of WSDS to be used as working space by virtual concurrent copy is necessary. The naming convention for these data sets is:

```
SYS1.ANTMAIN.sysname.SNAPnnnn
```

where *nnnn* is a four-digit decimal number in the 0001—9999 value range, and *sysname* is the system name used by JES. The data set names must be in sequence. If there is a gap in the sequence numbers, the SDM stops looking for the WSDS (for performance reasons).

The WSDS must meet these requirements:

- It must be a physical sequential (PS) data set.
- LRECL and BLKSIZE may be any valid combination.
- It must not be multivolume or extended format (striped).
- It must be RACF protected to prevent casual use. If a WSDS is protected, ANTMAIN must be authorized to access it.
- It must not be migrated.

WSDSs do not have to be SMS managed.

© Copyright IBM Corp. 1998

Each SDM issuing CONCURRENT copy commands to the RVA must have its own WSDSs defined. If you have multiple SDMs, multiple WSDSs must be allocated in the RVA subsystem. Each WSDS can be shared among multiple DFSMSdss jobs that are running in the same system; however, each DFSMSdss job will prefer a different WSDS if available.

Because the WSDS must be on the same RVA, device type, and partition as the source, multiple WSDSs must be allocated.

You must allocate one WSDS per combination of:

- System processing concurrent copy commands at a given time
- RVA
- Device type defined on the RVA
- RVA partition (if test devices are present)

If each system and each device type for DFSMSdss jobs (with concurrent copy specified) run simultaneously from more than one system and access data on the same RVA, they each require at least one WSDS. For example, DFSMSdss must allocate three WSDSs to process data on an RVA from three MVS systems, on devices of each device type containing data processed with virtual concurrent copy.

WSDS tracks are used for the duration of the physical copy; that is, from this message:

```
ADR734I (002)-ANNA (03), 1998.047 13:51:20 CONCURRENT COPY
        INITIALIZATION SUCCESSFUL FOR VOLUME XXXXXX.
        SERIALIZATION FOR THIS DATA IS RELEASED IF DFSMSDSS HELD IT.
        THE INTERMEDIATE RETURN CODE IS 0000.
```

to the end of the copy, that is, until this message:

```
ADR013I (002)-CLTSK(01), 1998.047 13:51:21 TASK COMPLETED WITH
        RETURN CODE 0000
```

Figure 22 shows an example of a job that can be used to allocate a single WSDS.

```
//*
//* ALLOCATE SYS1.ANTMAIN.SYSNAME.SNAP0001 WSDS
//*
//ALOCSAM EXEC PGM=IEBGENER
//SYSUT1   DD DSN=SYS1.ANTMAIN.SYSNAME.SNAP0001,
//         DISP=(NEW,CATLG),SPACE=(TRK,(30000)),
//         VOL=SER=WP32S1,UNIT=3390,
//         DCB=(DSORG=PS,RECFM=FB,LRECL=100,BLKSIZE=100)
//SYSUT2   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
/*
```

Figure 22. Allocating a Working Space Data Set

Figure 23 on page 39 shows the part of a job log with the output messages received when you allocate a single WSDS, as in Figure 22.

```
IEF236I ALLOC. FOR SNAPNO1 ALOCSAM
IGD103I SMS ALLOCATED TO DDNAME JOBLIB
IGD101I SMS ALLOCATED TO DDNAME (SYSUT1  )
        DSN (SYS1.ANTMAIN.SYSNAME.SNAPOOO1                        )
        STORCLAS (SC90WK31) MGMTCLAS (        ) DATACLAS (        )
        VOL SER NOS= WP32S1
IEF142I SNAPNO1 ALOCSAM - STEP WAS EXECUTED - COND CODE OOOO
IGD106I TEMP.LIB.LOAD                         PASSED,    DDNAME=JOBLIB
IGD104I SYS1.ANTMAIN.SYSNAME.SNAPOOO1            RETAINED,  DDNAME=SYSUT1
IEF373I STEP/ALOCSAM /START 1998033.0900
IEF374I STEP/ALOCSAM /STOP  1998033.0900 CPU OMIN OO.O1SEC SRB    OMIN
```

Figure 23. Working Space Data Set Allocation Output Messages

Figure 24 on page 40 shows an example of a multistep job that can be used to allocate several WSDSs, to accommodate your needs.

```
//ANTMAIN  JOB (SYS10000),,,
//    CLASS=A,MSGCLASS=T,MSGLEVEL=(1,1),REGION=6M,
//ALOCSAM EXEC PGM=IEBGENER
//SYSUT1   DD DSN=SYS1.ANTMAIN.SYSNAME.SNAP0001,
//        DISP=(NEW,CATLG),SPACE=(TRK,(30000)),
//        VOL=SER=WP31S1,UNIT=3390,
//        DCB=(DSORG=PS,RECFM=FB,LRECL=100,BLKSIZE=100)
//SYSUT2   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
/*
//ALOCSAM EXEC PGM=IEBGENER
//SYSUT1   DD DSN=SYS1.ANTMAIN.SYSNAME.SNAP0002,
//        DISP=(NEW,CATLG),SPACE=(TRK,(16700)),
//        VOL=SER=WP3ES1,UNIT=3380,
//        DCB=(DSORG=PS,RECFM=FB,LRECL=100,BLKSIZE=100)
//SYSUT2   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
/*
//ALOCSAM EXEC PGM=IEBGENER
//SYSUT1   DD DSN=SYS1.ANTMAIN.SYSNAME.SNAP0003,
//        DISP=(NEW,CATLG),SPACE=(TRK,(1500)),
//        VOL=SER=WP32S1,UNIT=3390,
//        DCB=(DSORG=PS,RECFM=FB,LRECL=100,BLKSIZE=100)
//SYSUT2   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
/*
//ALOCSAM EXEC PGM=IEBGENER
//SYSUT1   DD DSN=SYS1.ANTMAIN.SYSNAME.SNAP0004,
//        DISP=(NEW,CATLG),SPACE=(TRK,(1500)),
//        VOL=SER=WP3JS1,UNIT=3380,
//        DCB=(DSORG=PS,RECFM=FB,LRECL=100,BLKSIZE=100)
//SYSUT2   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
/*
//ALOCSAM EXEC PGM=IEBGENER
//SYSUT1   DD DSN=SYS1.ANTMAIN.SYSNAME.SNAP0005,
//        DISP=(NEW,CATLG),SPACE=(TRK,(1500)),
//        VOL=SER=WP3KS1,UNIT=3380,
//        DCB=(DSORG=PS,RECFM=FB,LRECL=100,BLKSIZE=100)
//SYSUT2   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
/*
//ALOCSAM EXEC PGM=IEBGENER
//SYSUT1   DD DSN=SYS1.ANTMAIN.SYSNAME.SNAP0006,
//        DISP=(NEW,CATLG),SPACE=(TRK,(150)),
//        VOL=SER=WP31S1,UNIT=3390,
//        DCB=(DSORG=PS,RECFM=FB,LRECL=100,BLKSIZE=100)
//SYSUT2   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
/*
```

Figure 24. Allocating Many and Different Working Space Data Sets

4.2 Catalog Search

WSDSs must be cataloged. The catalog search is done sequentially for each WSDS, starting with SYS1.ANTMAIN.SYSNAME.SNAP0001 until a catalog locate error (superlocate request) is encountered indicating that the WSDS was not found.

WSDSs with a decimal number beyond the data set that was not found is not used as working space. That is why you have to allocate them with names that have contiguous decimal numbers as the last four digits of the last qualifier.

If the catalog search for a WSDS indicates that it is multivolume, it is not used as a WSDS.

You can preallocate up to 9999 WSDSs. The SDM must find a WSDS that matches the source data set requirement as quickly as possible.

Each DFSMSdss job will attempt to select a WSDS that is not in use by another job.

DFSMSdss uses a selection algorithm to locate one or more WSDSs to be used to copy the source data set. So, a subset of the WSDS list, having a size greater than or equal to the source data set, is sorted by WSDS size. When the first WSDS that meets the requirements is found, it is used to copy the source data set. If a WSDS that meets the requirements is not found, you can use as many WSDSs as you need to get the size required to copy the source data set. Figure 25 shows how a source data set (WSDS 3) can be split in to two WSDSs (WSDS 1 and WSDS 2).

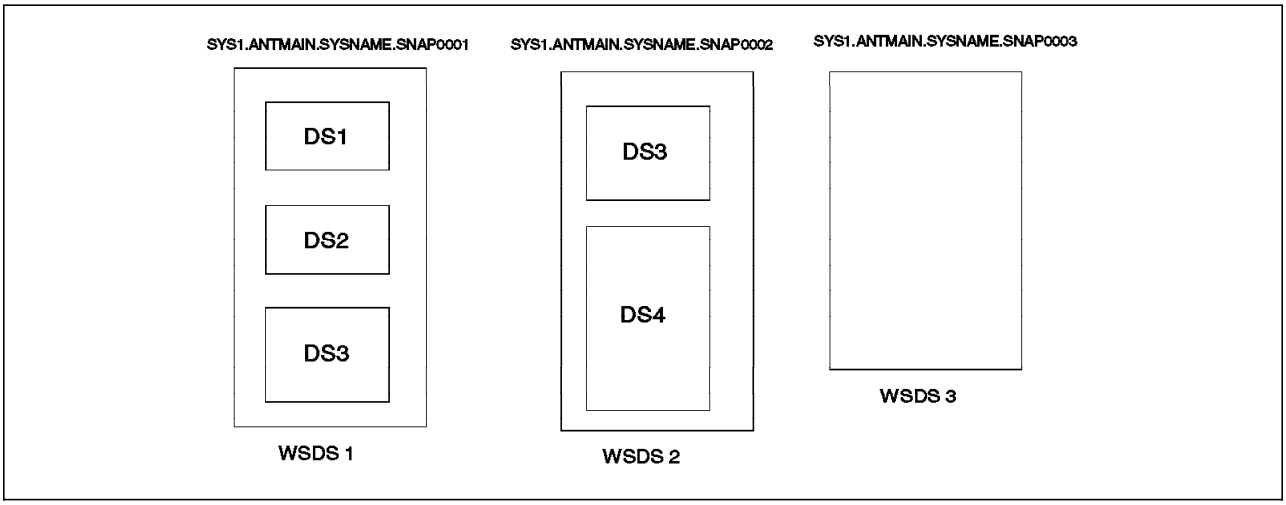


Figure 25. Source Data Set Spanning Two WSDSs

4.3 System Data Mover

The SDM does not extend a WSDS. If secondary space allocation is desired, you must extend the data set by filling it with data that compresses well, before starting the DFSMSdss job.

The SDM has the allocated WSDS, so it holds an enqueue for it, when the working space is being used during a SnapShot operation, and deallocates it, releasing the acquired enqueue, after the operation is completed.

A WSDS can be reallocated or extended when the SDM does not hold an enqueue on it. When the SDM subsequently uses the reallocated or extended WSDS, it uses the new size of the data set. Additional WSDSs can be added after the SDM has completed the initialization process.

The SDM uses these data sets the first time an out-of-working condition is encountered during a SnapShot operation. It refreshes the list of WSDSs by performing a catalog search starting with SYS1.ANTMAIN.SYSNAME.SNAP0001.

You can refresh the list yourself by using the following SDM user command:

```
F ANTMAIN,REFRESHWS
```

Other SDM debug commands are:

```
F ANTMAIN,L .module
F ANTMAIN,D .module.length
F ANTMAIN,DUMP
F ANTMAIN,DEBUG .status
```

4.3.1 Working Space Data Set Not Found

If SYS1.ANTMAIN.SYSNAME.SNAP0001 is incorrectly allocated on a non-RVA subsystem and SYS1.ANTMAIN.SYSNAME.SNAP0002 is correctly allocated on an RVA, there are two alternatives: One is for when you initially create the WSDS list, and the other is for when you refresh the list.

INITIAL CREATE OF WSDS LIST: Data sets are selected by a catalog search based on the WSDS naming convention. A gap in the series results in the “data set not found” condition. When you encounter the first data set not found condition, no further searching is performed.

REFRESH OF WSDS LIST: The catalog is searched for all WSDSs currently in the WSDS list. When the end of the current list is reached the search continues until you get the “data set not found” condition from the CATALOG. Any data sets in the current list that are not found are marked as non usable, but the search continues to the end of the current list.

4.3.2 System Data Mover Messages

These are a number of SDM-related messages:

```
ANTM6000I SNAPSHOT WORKING SPACE DATASETS BEING REFRESHED

ANTM6001I number SNAPSHOT
WORKING SPACE DATASETS BEING REFRESHED

ANTM6002W ERROR REFRESHING SNAPSHOT
WORKING SPACE DATASETS - RC=rc REAS=reas

ANTM6003E SNAPSHOT READ ERROR ON DEVICE=device
VOLSER=volume RC=rc REAS=reas
CSW=csw SENSE=sense
```

On SMS 1.3.0 these messages are written to the system log or console. On SMS 1.4.0, they are also written to the job log. A software LOGREC entry accompanies any ABEND. A LOGREC entry is written to accompany any ANTM6003E message as well.

4.4 Size Requirements

Virtual concurrent copy uses WSDS space to contain track image copies of the data that DFSMSdss processes.

The total size of all work data sets allocated on each RVA should be equal to or exceed the largest total amount of data to be processed in a single DFSMSdss COPY or DUMP command on that RVA. If there is insufficient space, the virtual concurrent copy initialization for one or more data sets in the job fails.

To determine the WSDS size required for a given DFSMSdss job or set of jobs, you must, for each RVA, system, and device type combination determine the total size of all source data sets that will be processed as a unit, in tracks. If multiple DFSMSdss jobs run simultaneously, add the tracks needed for all of the jobs on that RVA, system, and device type. The total number of WSDS tracks on that RVA for that system must be equal to or exceed the maximum number calculated above. Multiple WSDSs can be allocated on the same or different volumes. Ideally one WSDS should be available for each DFSMSdss job.

The impact on NCL for WSDS space can vary from 0% for a data set that has no updates during the virtual concurrent copy operation to 100% if the entire data set is updated before the operation is physically completed, that is, until you receive the DFSMSdss task completed message:

```
ADR013I TASK COMPLETED WITH RETURN CODE 0000
```

For example, a 100-cylinder data set may require up to 100 cylinders of data space storage if the data set is completely rewritten before the virtual concurrent copy operation is physically completed.

Based on concurrent copy experience, a typical impact on NCL is likely to be about 10% of the data being dumped or copied with virtual concurrent copy. The SDM overwrites any user data written to a WSDS. Once the physical copy is complete, the SDM releases the back-end space.

4.5 Numbers of Working Space Data Sets

Table 7 shows the effect of varying the number of WSDSs that DFSMSdss can use. To use the numbers shown in Table 7, we did a DFSMSdss concurrent copy DUMP of 10 sequential data sets, each 100 MB.

Table 7. Effect of Varying the Number of Working Space Data Sets

Number of WSDSs	Size of Each WSDS	Logical Completion (sec)	Physical Completion (sec)
1	3 GB	7	290
10	300 MB	8	291
100	30 MB	44	327
1,000	3 MB	357	641

The reason for the increase in time is that DFSMSdss must determine the size of each WSDS and then determine which one will be the best fit; the data set is then snapped to that WSDS and dumped. If none of the WSDSs can completely contain one of the source data sets, parts of that source must be snapped to different WSDSs. However, for performance reasons, it is preferable to allocate the required space in as few WSDSs as possible.

For experimental reasons, the size of each WSDS is always the same. As you know, you can preallocate several WSDSs, each with a different size, but for our purposes, we ran every test with fixed-size WSDSs.

4.6 Recommendations

You must preallocate several groups of WSDS. You can choose to allocate each of them with a different size; however, from a performance point of view, it is better to have just a few large WSDSs. If unit control block (UCB) constraint is not a concern, we recommend that you dedicate a full volume for each WSDS: as many volumes as possible. If you keep other data sets on the same volume where your WSDSs reside, the response time for the other data sets could be affected because of an increase in IOSQ time.

From a serialization point of view, if you use only a few WSDSs with a large number of virtual concurrent copy operations that run at the same time, you may get I/O queuing issues because DFSMSdss can share a single WSDS for multiple operations. The time to physical completion can be increased; however, logical completion is not affected by sharing a WSDS among multiple DFSMSdss operations. The physical completion time is usually removed from the critical path, and application availability is affected only by the time to logical completion.

WSDSs are dedicated to each system and identified by the sysname.

4.7 Performance Issues

Many of the performance considerations related to virtual concurrent copy are similar to those for 3990 concurrent copy. In fact a major component of virtual concurrent copy session initialization includes the time DFSMSdss spends filtering data sets. Therefore, the more precisely you specify the data sets to be processed, the sooner the initialization is completed and the sooner you can update your data again.

Here are some suggestions for reducing initialization time:

- Keep the data to be dumped by one DUMP command cataloged in one catalog, if possible.
- Do not specify the DYNALLOC keyword if you do not need dynamic allocation for your data sets.
- Specify fully or almost fully qualified data set names, to reduce the time that DFSMSdss searches catalogs for the data set to be processed.
- Specify small groups of data sets to be processed together on a DFSMSdss operation.
- Minimize the use of wildcards with the INCLUDE keyword.

- Minimize the use of sophisticated BY filtering to determine the data to be processed.
- Ensure that DFSMSdss can obtain serialization on all data sets being processed.
- Specify WAIT(0,0) to prevent DFSMSdss from waiting for serialization when it can not be obtained.
- Do not specify the NOTIFYCONCURRENT keyword if you do not need notification of each data set included in the virtual concurrent copy session.
- Do not specify the SPHERE keyword if you are not processing VSAM spheres.
- Use the ADMINISTRATOR keyword or DASDVOL RACF protection (where applicable) to bypass authorization checks for each data set being processed.
- Ensure that the volumes containing the VTOC and catalog entries for the data set to be processed have caching enabled. Also ensure that the catalogs involved are enabled for in-storage cache (ISC) or catalog data space cache (CDSC).
- Ensure that data sets being processed have not been migrated by DFSMShsm.

4.8 Restrictions

A WSDS cannot be migrated by DFSMShsm. DFSMShsm by default does not migrate any data set that has *SYS1* as the high level qualifier (HLQ), but, because you can change the WSDS name, you need to make sure that WSDS DFSMShsm migration does not occur. The best way to avoid migration is to use the WSDS naming convention: SYS1.ANTMAIN.SYSNAME.SNAPnnnn.

The SDM does not extend WSDSs above the allocation extents but uses additional extents if they are available.

Chapter 5. DFSMSdss SnapShot Performance Considerations

In this chapter we discuss the performance considerations related to DFSMSdss SnapShot.

With DFSMSdss installations using the virtual concurrent copy COPY or DUMP option for data sets or entire volumes on an RVA subsystem can take advantage of greatly improved flexibility in performing their backups.

DFSMSdss SnapShot and virtual concurrent copy significantly reduce the amount of time that data sets are unavailable during the copy process. This enhanced performance may also allow for more frequent backups, so recovery is faster.

5.1 Measurement Environment

The information written in this chapter is based on measurements made at the IBM San Jose Performance Laboratory (SJPL) in San Jose, California.

The DASD environment used for the measurments consisted of an eight-path RVA T82 with ESCON attachment using 18 MB per second LED/multimode channels with 420 GB of DASD attached.

Other environments, configurations, and processors will experience differing levels of performance. Accordingly, the numbers reported in this chapter do not constitute a performance guarantee or warranty, and the considerations that follow should be taken only as a reference.

For a complete description of the environment used, refer to the DFSMSdss SnapShot Performance Flash, which is available from MKTTOOLS in the DSS_PERF PACKAGE or from your IBM representative.

The SJPL has developed a VSAM database workload that is intended to mimic the I/O characteristics of a variety of database management systems. It is implemented as an application program using VSAM RRDSs. The program can generate a variety of random access patterns and read-write characteristics.

The workload uses 4K record sizes for all variations. Each logical volume in the workload has one database data set. Each database data set is approximately 800 MB. For DFSMSdss SnapShot and virtual concurrent copy performance testing, we used only cache standard, that is, a mixture of intermediate and hostile caching volumes.

5.2 Performance Results

A dump is logically complete after the data to be dumped has been “snapped” to the WSDS and physically complete when the job ends. It is important to realize that these two times depend on different factors: time to logical completion depends only on the number of source data sets, and time to physical completion depends on the total size of the source data sets.

A full-volume virtual CONCURRENT COPY (or DUMP) running on an RVA subsystem runs at SnapShot speed, as Table 8 shows.

© Copyright IBM Corp. 1998

Table 8. SnapShot, CC Copy, and CC Dump	
Method	**Time per volume (sec)**
Virtual CONCURRENT COPY	4
Virtual CONCURRENT DUMP (logical completion)	1
SNAP VOLUME	1

The performance of a virtual concurrent copy volume DUMP and SNAP VOLUME are equivalent.

A virtual CONCURRENT COPY to snap a volume typically takes about 4 sec.

The impact of one or even several DFSMSdss CONCURRENT DUMPs on the background workload, both in terms of response time and throughput, is approximately 5%-10%, no matter which DFSMSdss OPTIMIZE level is used.

When comparing the performance of 3990 concurrent copy and virtual concurrent copy, we see further benefits of virtual concurrent copy. The CPU time for a DFSMSdss CONCURRENT DUMP on a device that supports SnapShot is substantially less than on a device that supports concurrent copy. 3990 concurrent copy also uses subsystem cache resources, which may affect the performance of other work running on the same system. Expanded storage dataspace of approximately 10% of the data being copied can also be consumed during a concurrent copy operation. The way in which virtual concurrent copy uses NCL is similar to the way 3990 concurrent copy uses its cache and dataspace “sidefile”; however, this is likely to have a less significant and more predictable effect on performance.

In addition, with the RVA T82, you have the option of eight host data transfers, which significantly increases the number of DFSMSdss DUMP jobs that can be run concurrently without affecting other work on the subsystem.

5.3 Recommendations

In this section we summarize the recommendations based on the measurements and test performed by the SJPL.

5.3.1 Working Space Data Sets

The volumes on which the WSDSs reside may experience very high response times, even several hundred milliseconds. This is a typical response time for volumes with data sets on them that are being dumped.

Consequently, the WSDSs should be the only data sets on the volumes on which they reside, to prevent such volumes from being used by applications that are running at the same time and thus to avoid any performance degradation of the applications.

If multiple dump jobs run at the same time, we recommend placing the WSDSs on as many different volumes as possible. If different jobs run concurrently, requiring different WSDSs that are on the same volume, queuing delays for that volume can occur.

The biggest WSDS should be at least as large as the largest source data set to be dumped (see Table 7 on page 43).

5.3.2 CPU Time

Usually CPU time is not a concern with DFSMSdss SnapShot, but here are a few points to consider:

- Using the DFSMSdss COMPRESS option can lead to an increase in CPU, by a factor of 1.2 to 1.9, for the DFSMSdss job. So avoid using the COMPRESS option, unless space is a much more important factor than CPU time. Using the COMPRESS option leads to a lesser benefit on an RVA because the RVA itself compresses data.
- Higher levels of the OPT parameter result in decreased CPU times.
- A background workload that makes heavy use of the CPU results in queuing the DFSMSdss jobs to use the CPU, leading to an increase in elapsed time.

5.3.3 Workload Throughput and Response Time

The impact on background workload when running concurrent copy dumps depends on the profile of the background work and the number of concurrent dumps.

The performance measurements indicate a 5%-10% impact on throughput and response time of background workload when running parallel dump jobs up to the largest suggested number, which is six for a T82. The above is true when you use the DFSMSdss DUMP command with any value of OPTIMIZE.

5.3.4 Parallel Jobs

When you run multiple dump jobs, there is no performance advantage in running simultaneously more jobs than the number of tape channels. Also there is a significant saving in time to physical completion, although there is essentially no saving in time to logical completion.

We recommend an upper limit of 2 on the number of simultaneous dump jobs per RVA2 model T42 or IBM 3990-6 control unit. When you use an RVA2 model T82, you can increase the number of simultaneous dump jobs to 6. In both cases, the performance results are workload dependent.

For application serialization reasons, you may want to overcommit the tape channels, by issuing more virtual concurrent copy jobs. This will provide logical completion to more tasks, but in this case you should run the DFSMSdss jobs with a lower optimization value.

5.3.5 Volume Snaps

If the task is to snap a volume, do so without the parameter DATASET(INCLUDE(**)), because DFSMSdss will make a list of all data sets on the volume and snap them one by one, rather that snapping the volume itself. With the new DFSMSdss SnapShot support, there is an ease-of-use advantage in using virtual concurrent copy to do this, although there is no great functional or performance advantage to doing this from DFSMSdss; traditional SnapShot jobs work just as well.

Refer to Table 8 on page 48 for measurement results.

5.3.6 Number of Extents

Before the new DFSMSdss SnapShot support, one of the SnapShot recommendations was to reduce the number of extents for all source data sets. Now, with DFSMSdss SnapShot and virtual concurrent copy, there is no need to reduce the number of extents, as indicated in Table 4 on page 31.

5.3.7 Concurrent Copy

Installing the new DFSMSdss SnapShot support does not affect the performance of 3990 concurrent copy or DFSMSdss. You do not have to modify concurrent copy jobs for performance or functionality reasons.

5.3.8 Other Copies and Dumps

DFSMSdss SnapShot support does not interfere with other copies and dumps for devices that do not support either SnapShot or concurrent copy.

There is only a minor increase in time when copying a small data set from an RVA device to a non-RVA device. The results of performance evaluation tests show that there is only a minor impact, about 10%, for small (50 to 500 KB) sequential data sets due to the overhead of DFSMSdss attempting to make a SnapShot copy and then realizing that it cannot. In all other cases, the elapsed time is the same.

5.3.9 DFSMSdss OPTIMIZE Parameter

The recommended values for 3990 concurrent copy are OPT(2) or OPT(3). OPT(2) provides minimal impact to other application I/O, and OPT(3) provides minimal CPU overhead. The recommended value for virtual concurrent copy is OPT(3).

Because you can easily use the same 3990 concurrent copy batch jobs for virtual concurrent copy, changing only the source and the target, you can set OPTIMIZE to 3 and not worry about whether you are using virtual concurrent copy or 3990 concurrent copy. If there is a great deal of existing JCL with OPT(2) coded, it might not be worth the trouble of changing it to use OPT(3) for virtual concurrent copy.

5.3.10 Frequency of Dumps

Jobs using 3990 concurrent copy to back up data can now be used to back up data on an RVA, which is much easier than doing a direct SnapShot and then dumping to tape. Consequently you may want to back up your data to tape more frequently, because this does not affect a background workload.

You may choose to back up data sets to disk, using DFSMSdss SnapShot, and then delete them at the end of the batch processing.

Each installation must make its own decision about increasing, decreasing, or keeping constant the number of backups to tape.

5.4 Summary

The DFSMSdss COPY command on an RVA (with all the requirements met) invokes the DFSMSdss SnapShot function in a way similar to invoking concurrent copy on a 3990-6. The times to filter data set names and perform the catalog function and the CPU times should be comparable (+/- 5%) to the equivalent functions in concurrent copy, to the extent that the functions can be compared.

The introduction of SnapShot does not degrade performance of DFSMSdss COPY in environments that do not support SnapShot (either for hardware or software reasons or because SnapShot is not applicable), except for small sequential data sets when there is a minor impact of about 10% due to DFSMSdss attempting to perform a DFSMSdss SnapShot and finding that it cannot.

For the DFSMSdss COPY and DUMP commands with the CONCURRENT COPY (CC) keyword specified, when the virtual concurrent copy SnapShot function can be used, all performance measurements are comparable to previous levels of DFSMSdss where the concurrent copy function can be used.

Chapter 6. Using DFSMSdss SnapShot in Batch

In this chapter we discuss some practical applications to reduce your batch window by using virtual concurrent copy and DFSMSdss SnapShot.

Before the availability of DFSMSdss SnapShot and virtual concurrent copy, the only way of invoking SnapShot in batch was through the SIBBATCH program.

As you know, the commands and related operands of SnapShot and DFSMSdss are quite different, so it is not easy to replace the DFSMSdss commands without changes to procedures and JCL.

Now DFSMSdss can copy RVA data at SnapShot speed and preserve certain benefits such as filtering and sphere processing. The hardware and software work together to maximize the functionality of your applications.

With the implementation of SnapShot, most customers can dramatically reduce the amount of nonproductive time associated with backup operations. Full merges of critical information are reduced by hours and become easier to schedule around peak or unexpected workloads. During the nightly batch cycle, for certain data volumes, batch runs can be reduced from hours to minutes, which leads to an increase in system availability and improves access for application development and test.

6.1 Identifying SnapShot Candidates

Most batch processing is sequential; that is, multiple jobs are processed one after another. You may have a more complex job structure with many interdependencies among individual jobs. However, the critical path through those jobs (the paths that determine how fast the overall set of jobs can run) is nevertheless essentially sequential.

All of the preparatory activities to follow to identify the best SnapShot candidates in your test environment remain the same as before DFSMSdss SnapShot. You have to identify candidates that would benefit most, in terms of CPU and elapsed time, if traditional data copying is replaced by SnapShot.

Furthermore, if SnapShot is used to back up few large data sets, the backup window is dramatically reduced. In other words, *pick the low-hanging fruit first*, to get the biggest results with the smallest effort.

What has really changed with DFSMSdss SnapShot is that, if all of the requirements listed in 2.1.2, “Requirements” on page 6 are met, SnapShot is internally and automatically invoked by DFSMSdss each time you perform a DFSMSdss COPY operation. The new DFSMSdss SnapShot support is designed to be transparent to the user.

Be aware that if data sets span different RVAs, DFSMSdss moves all of the data sets according to the “all or nothing” rule, and the movement can take a long time to complete.

© Copyright IBM Corp. 1998

6.2 Implementation

Your batch window is probably the easiest place to incorporate SnapShot. Now that DFSMSdss fully benefits from SnapShot characteristics and SnapShot benefits from DFSMSdss characteristics, you must think about exploring new opportunities. Consider not only which applications can use DFSMSdss to take advantage of the CPU and elapsed time reductions that SnapShot affords, but also how your applications could benefit from using the new support.

Consider, for example, how many backup copies are made today during a daily batch process. Most customers make one or a maximum of two backup copies, so if something goes wrong, they lose a lot of time in batch reprocessing, starting from the last valid backup copy, thus delaying the time when online service is available. Using DFSMSdss SnapShot and virtual concurrent copy, you can increase the number of backups that you make and reduce the rerun time in the event of a job failure. Online activity is the core business process for all companies, so you must minimize the possibility of having reduced online service availability.

Before SnapShot, for traditional batch processing (Figure 26) you could only make one backup copy, before running the various phases of your batch processing; otherwise you would risk delaying time *C* when online processing becomes available again.

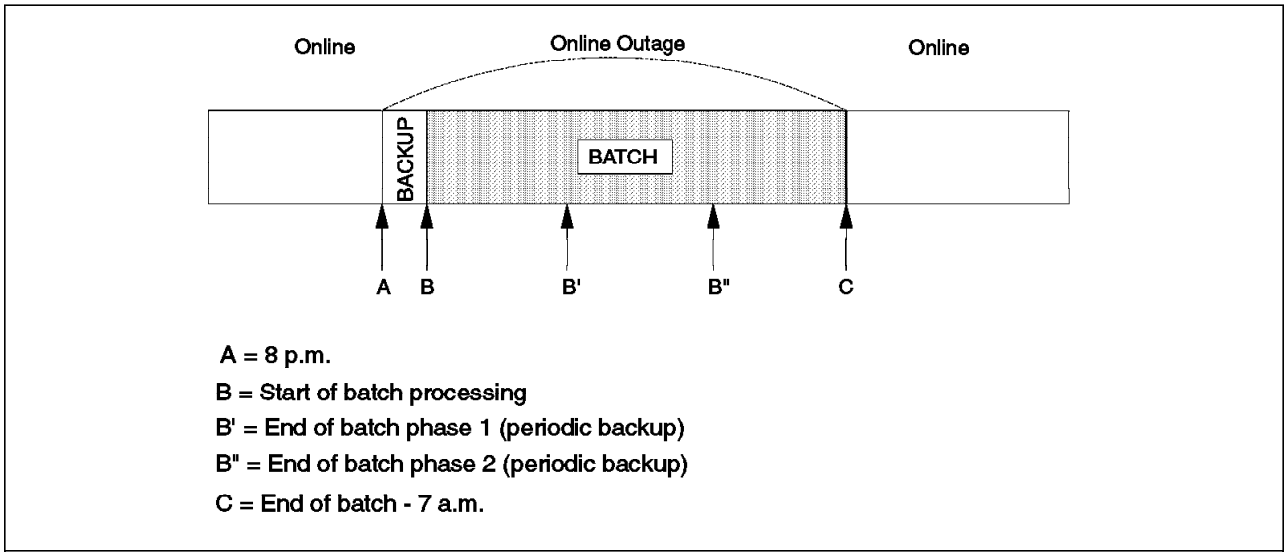


Figure 26. Traditional Batch without SnapShot

With SnapShot, the amount of elapsed time backups take to complete is reduced, thereby reducing the outage of online applications.

SnapShot reduces the outage to online services by hours rather than only minutes; in the example shown in Figure 27 on page 55 online availability is increased by 3 hours.

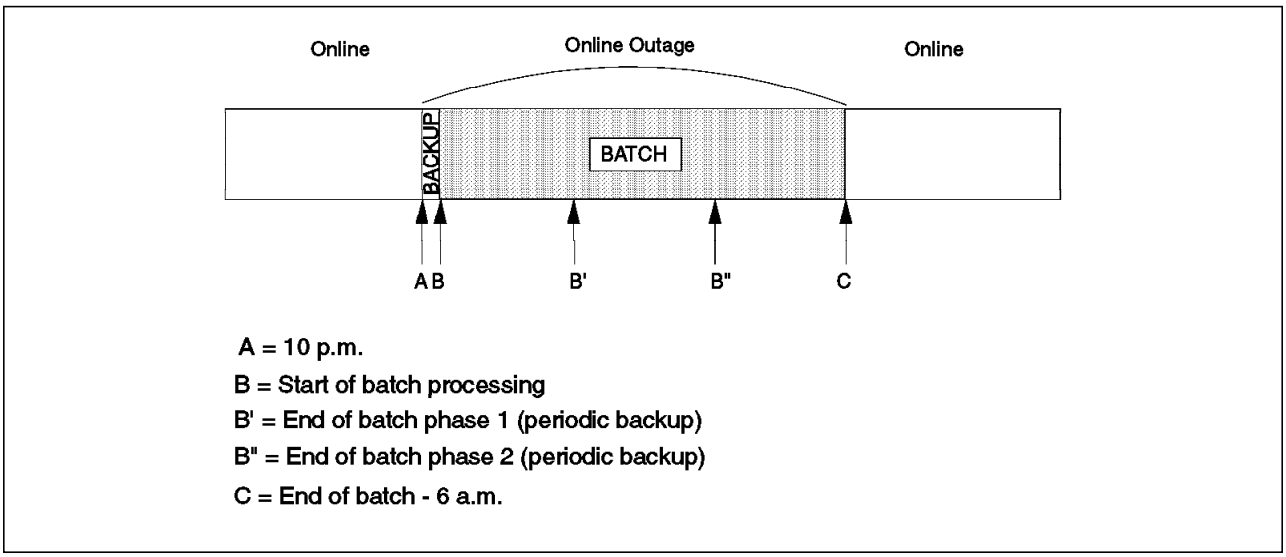


Figure 27. Batch with SnapShot

With DFSMSdss SnapShot, you have a further reduction in batch processing time and can make additional backup copies during your batch process (for example, at the end of each phase of your batch). These periodic backups enable you to get more frequent backup copies, saving a lot of hours for recovery in case of a batch failure (see Figure 28).

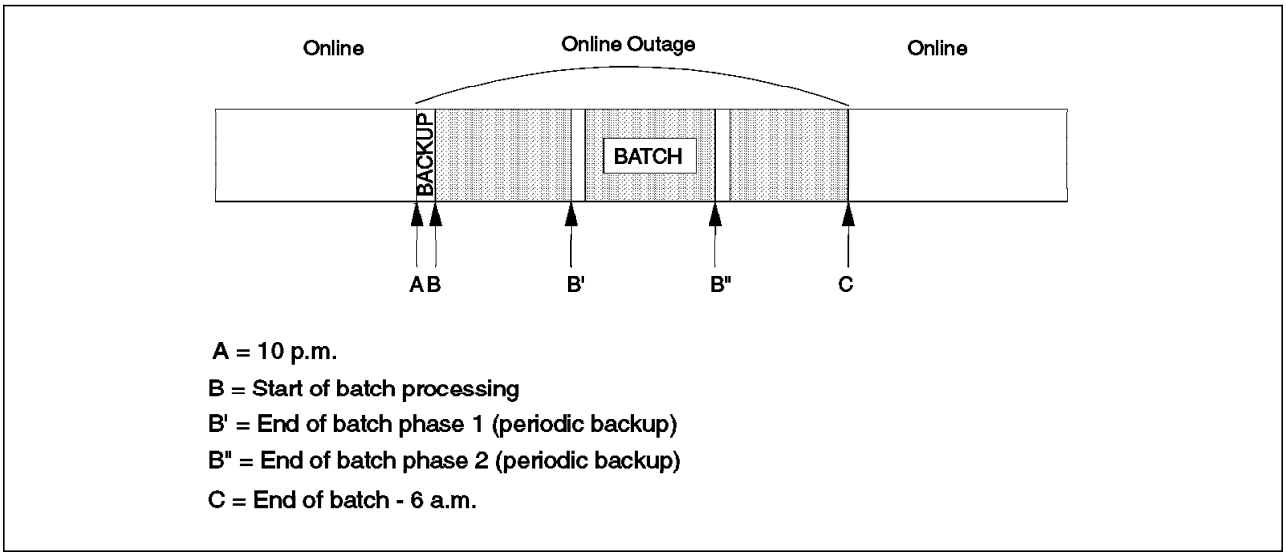


Figure 28. Batch with DFSMSdss SnapShot

Without DFSMSdss SnapShot support you must invoke a SNAP DATASET command for each data set individually from a SIBBATCH batch job.

Now that SnapShot is internally and automatically called by DFSMSdss, even small data sets can benefit from SnapShot speed data movement, and you do not have to change your applications.

When go to implement DFSMSdss SnapShot, consider that most batch processing streams include multiple processing and backup steps, which may be in one or more jobs, so you may find unexpected benefits. Older job streams may contain many backup steps that were added over time as the application

developed. Some of these backup steps are redundant and could be removed, thus reducing your batch window and the amount of resources consumed.

You can gradually add the above advantages to the advantage gained from using SnapShot.

DFSMSdss SnapShot allows you to transparently exploit the time-saving benefits of SnapShot without changing your JCL. If you have multiple subsystems (RVA and/or non-RVA), you can consider modifying your automatic class selection (ACS) routines and/or using VOLPREF to drive the allocation to a specific volume or volumes, to maximize the benefits of DFSMSdss SnapShot.

You can consider using the CONCURRENT keyword for both DFSMSdss COPY and DUMP to benefit from virtual concurrent copy.

6.3 3990 Concurrent Copy and Virtual Concurrent Copy

DFSMSdss concurrent copy works in exactly the same way as virtual concurrent copy. From a conceptual point of view they are very similar; the major difference is performance.

The concurrent copy function of DFSMSdss is a hardware and software solution. The database, or any collection of data, is copied to 3990 cache, and after that the update activity on the database may resume; this is a very small fraction of the time that the complete backup takes. The copy that is made is a point-in-time copy, without any of the updates that were subsequently made to the source data.

With virtual concurrent copy, the data is “snapped” from the source location to an intermediate location (WSDS), and the data is then gradually copied to the target location through normal I/O methods.

6.4 DFSMSdss Concurrent Copy Optimize

The duration of a particular concurrent copy operation is determined by the number of tracks DFSMSdss is to read at a time. The OPTIMIZE parameter of the DFSMSdss DUMP command specifies the number of tracks to be read at a time:

- If OPT(1) is coded, DFSMSdss reads one track at a time.
- If OPT(2) is coded, DFSMSdss reads two tracks at a time.
- If OPT(3) is coded, DFSMSdss reads five tracks at a time.
- If OPT(4) is coded, DFSMSdss reads one cylinder at a time.

By using an optimize value greater than 1, you reduce the elapsed time and do fewer I/O operations on the DASD device whenever the load on the tape channel is low enough and the tape speed is high enough to keep pace with the data being read from the DASD volume.

In addition DFSMSdss uses more real and virtual storage and places greater strain on subsystem resources of cache and host paths.

6.4.1 RVA Concurrent Copy Optimize

The measurements made in a variety of system environments show the OPT(3) parameter gives the best results, independently of the background workload.

The recommended values for 3990 concurrent copy are OPT(2) or OPT(3).

Now that you can easily use the same 3990 concurrent copy batch jobs even for virtual concurrent copy, changing only the source and the target, you can set OPTIMIZE to 3 and not worry about whether you are using virtual concurrent copy or 3990 concurrent copy

If the background workload is not a concern, use OPT(4) to get a fast DUMP operation with fewer I/O operations.

6.5 SnapShot Recovery and DFSMSdss Recovery

In this section we compare recovery procedures for virtual concurrent copy with those for SnapShot.

6.5.1 Virtual Concurrent Copy Recovery Considerations

In many batch streams, recovery from a job failure is relatively straightforward. If a processing step fails, the data being processed is restored from a backup, some recovery action is performed, and the batch processing stream continues.

The output from a virtual concurrent copy DUMP is a DFSMSdss data set, and recovery requires that the DFSMSdss data set be restored. Currently there is no way of accessing the WSDS or recovering a data set from the snapped copy in the WSDS.

If you are using DFSMSdss SnapShot, a DFSMSdss copy back to the source location results in a SnapShot, so recovery may be faster.

Concurrent copy alters the possible failures that can arise and the recovery actions that you follow because the operation progresses in parallel with application processing. Although unlikely, three different failure modes are possible when you use concurrent copy:

- Application processing fails, but the concurrent copy completes normally. This failure is exactly the same as a failure of an application processing job without concurrent copy. Recovery may be slightly more complex because the application could fail before the concurrent copy dump is complete. Once the concurrent copy dump job completes, you can restore the dump and continue with recovery in the same way as without concurrent copy
- The concurrent copy dump fails but the application processing completes normally. In general, this failure is not serious. If the dump was being taken for recovery in case of application processing failure, it is not needed. If, however, the dump was being taken for other reasons, you might need to take another dump.
- Both the concurrent copy dump and the application processing fail. This is the most serious failure because you do not have a complete dump available to use for recovery.

If a dual failure occurs, the most direct recovery technique is to recover from a dump taken earlier in the batch processing and rerun all the work done in the meantime.

To complete recovery successfully in case of a dual failure, you must ensure that a dump of the data has completed successfully before you start batch processing, for example, using SnapShot. If such a dump is not available, recovery from a dual failure could involve restoring a dump from the previous day and forward recovering an entire day's processing, which would be extremely time consuming.

6.5.2 SnapShot Recovery Considerations

If you have updated your source and discovered that the update must be backed out to the last backup copy, with SnapShot you can instantaneously snap the copy back to your data set, thereby backing out the update.

The snap that allows you to restore a previously snapped data set or volume from its snapshot, is commonly called a *snap back*. This benefit also applies to DFSMSdss copies that are taken with DFSMSdss SnapShot, as these, if copied back within the same RVA subsystem, again result in a snap.

6.6 Concurrent Copy Automation

If your batch processing stream is structured with application processing and backup work divided into separate jobs, few or no changes to the jobs that you run are required to implement virtual concurrent copy. If, however, you combine application processing and backup work as multiple steps within the same job, you must separate the two functions into separate jobs, so that they can both run at the same time.

When backup and application processing are separated, start the backup processing first and then start application processing jobs when DFSMSdss indicates with this message that the backup is logically complete:

```
ADR734I (002)-ANNA (03), 1998.047 13:51:20 CONCURRENT COPY
        INITIALIZATION SUCCESSFUL FOR VOLUME XXXXXX.
        SERIALIZATION FOR THIS DATA IS RELEASED IF DFSMSDSS HELD IT.
        THE INTERMEDIATE RETURN CODE IS 0000.
```

To get the real benefit of using concurrent copy, you must release the work that runs in parallel with concurrent copy jobs when the concurrent copy session is logically completed.

Instead of using traditional batch processing, with multiple jobs being processed sequentially one after another, as shown in Figure 29 on page 59, you should take advantage of the parallelism allowed by concurrent copy and restructure your batch stream.

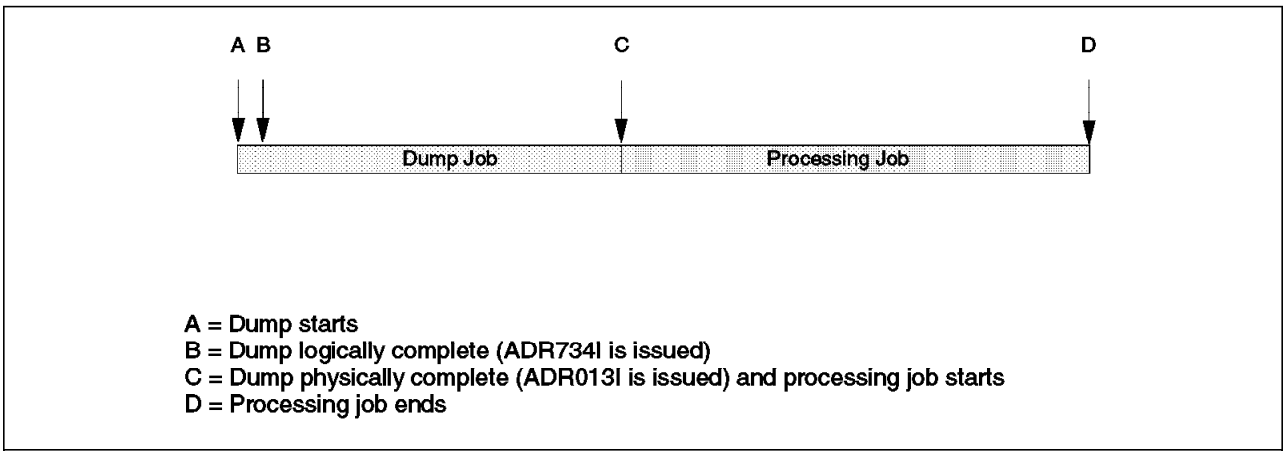


Figure 29. Batch Processing Stream without Concurrent Copy

Figure 30 shows a restructured batch job stream which exploits concurrent copy to reduce batch elapsed time.

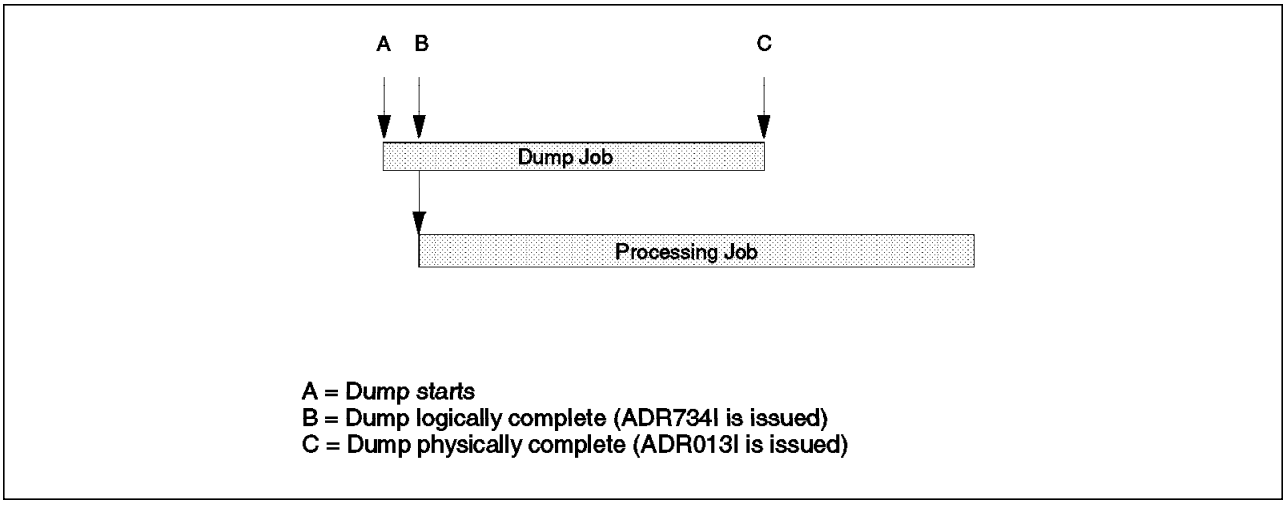


Figure 30. Logically Complete and Physically Complete Dump

In a multistep job, the next step is released only when this message indicating that the step is physically complete has been issued:

```
ADR013I (002)-CLTSK(01), 1998.047 13:51:21 TASK COMPLETED WITH
        RETURN CODE 0000
```

Because today virtual concurrent copy is not restartable, if you absolutely need a point-in-time copy of your data, virtual concurrent copy may not be the best solution. Use SnapShot instead to guarantee a point-in-time copy of your data.

If you do not use a batch scheduling system, you can use one of the following techniques to coordinate batch processing:

- All jobs are submitted to JES2 at the start of batch processing but placed in HOLD status. When one job completes, its last step releases the next job to be processed.
- Only the first job is submitted at the start of batch processing. When one job completes, its last step submits the next job to be submitted.

After that you can use any automation product to intercept the following message and then release or submit the subsequent jobs:

```
ADR734I CONCURRENT COPY INITIALIZATION SUCCESSFUL
```

You can also use the DFSMSdss NOTIFYCONCURRENT keyword. For a logical data set dump or copy operation this keyword specifies that DFSMSdss issue this message for every data set that is successfully included in the concurrent copy operation:

```
ADR767I CONCURRENT COPY INITIALIZATION SUCCESSFUL
FOR DATA SET dsname
```

Intercepting this message enables you to perform processing based on individual data sets. If you decide to perform individual data set processing, you should also intercept and respond to the following message, which indicates that concurrent copy initialization failed for a particular data set:

```
ADR735W USE OF CONCURRENT COPY FAILED FOR DATA SET
dsname
```

For more information about automating concurrent copy using NetView and OPC/ESA, and code samples, refer to *Implementing Concurrent Copy*, GG24-3990.

6.7 Volume Preferencing and Automatic Class Selection Routines

In this section we discuss two solutions that enable you to drive the allocation of your target data set to a required volume or volumes and maximize your benefit from DFSMSdss SnapShot and virtual concurrent copy.

6.7.1 Volume Preferencing

Volume preferencing enables SnapShot to filter the choice of an SMS target volume for newly allocated data sets. It is very useful in an SMS-managed environment when you have defined storage groups that cover more than one RVA subsystem or span RVA and non-RVA subsystems. See 7.2.3, “Volume Preferencing” on page 65 for a complete discussion of volume preferencing implementation.

6.7.2 Automatic Class Selection Routines

In an SMS-managed environment, you generally allow the system to place data sets for you. If you want to control the placement of the data sets (for example, because you want the target data set to reside on the same RVA subsystem as the source, to reap the benefits of DFSMSdss SnapShot), you must take special steps.

If you use OUTDDNAME or OUTDYNAM to specify a volume list, the volume serial numbers are passed as input to the ACS routines. Depending on how your ACS routines are written, this input might or might not be used in determining where to place the data sets.

One way to guarantee that the data sets go to particular volumes is to write your storage group ACS routine such that data sets are moved to the volume you selected.

Alternatively, if the storage class of a data set has the guaranteed-space attribute, the data set is placed on the volumes specified by the user, if such

volumes reside in the same storage group and the ACS routine selects that storage group for the data set. By using the BYPASSACS and STORCLASS DFSMSdss COPY keywords, you can ensure that the storage group selected contains the volumes you specify with OUTDDNAME or OUTDYNAM. However, this procedure works only if your storage group ACS routine uses a storage class to determine the storage group for a data set. If you code the NULLSTORCLAS and BYPASSACS DFSMSdss keywords together, the target data set becomes non-SMS-managed. Alternatively you could define a storage group that consists entirely of volumes in the same RVA subsystem, to maximize your potential use of DFSMSdss SnapShot.

6.8 Using SnapShot for Data Set Backup

Traditionally you keep your data set backups on tape or DASD. Before the DFSMSdss SnapShot support, you had to make some changes and carry out some manual operations to reap the benefits of SnapShot. There are some considerations that are applicable especially for environments where you are currently using SnapShot.

When you keep your saved copy on tape, you must snap the data set to DASD first and then back it up to tape. In this case the following considerations are applicable:

- Snapped data sets have different names.
  - Changes are required to backup and restore jobs.
  - Recataloging may be required
  - There are specific considerations for VSAM data sets (see 2.1.7, “VSAM Considerations” on page 9 for further details).
- Data sets can be recovered from a volume snap.
  - A logical dump does not back up uncataloged VSAM data sets.
  - Original data sets must be uncataloged before snapped data sets can be renamed.
- Back up related data sets together for recovery purposes.

When you keep your saved copy on DASD, you can snap the data set to DASD directly, so in this case, these considerations are applicable:

- Inline backups may be temporary copies.
  - SnapShot is ideal for time and space savings.
  - You have the opportunity to do batch reviews.
- JCL changes are required.
  - SnapShot requires fully qualified data set names.
  - DFSMSdss SnapShot uses DFSMSdss filtering and wildcard techniques.

When choosing which data set backup technique to use, remember that as either copy of the data set on DASD is changed after the snap, NCL usage will increase. So if you choose to keep your saved copy on tape, and your available NCL is limited, delete the snapped copies of the data set as soon as they have been successfully copied to tape.

With DFSMSdss SnapShot support, most of the above limitations have been removed, so you just can leave your original JCL as it was before, and, if the requirements are met, SnapShot is internally and transparently invoked.

You can fully benefit from the flexibility of DFSMSdss, for such functions as filter lists and wildcard characters, and benefit also from the DFSMSdss serialization technique, which ensures consistency across related data sets.

Chapter 7. DFSMS/MVS and Virtual Concurrent Copy

In this chapter we explain how DFSMS/MVS exploits DFSMSdss SnapShot and virtual concurrent copy.

7.1 Software Prerequisites

The following APARs provide the new SMS storage class support:

- OW27857 - volume selection
- OW27860 - accessibility parameter

7.2 Recognizing Virtual Concurrent Copy Capability

DFSMS recognizes the RVA as a high-availability device, like RAMAC 9392 or dual copy devices. However, DFSMS can also distinguish between an IBM 3990-6 control unit and an RVA because the RVA also indicates that it is SnapShot capable. In addition DFSMS can detect whether the SDM has the support for DFSMSdss SnapShot and virtual concurrent copy.

Therefore SMS device selection routines can be used to direct allocations to or away from a virtual concurrent copy capable device. There are some changes to the volume preferencing operation to support these enhanced functions. There are also additional subparameters to the ACCESSIBLITY settings that are supported through Interactive Storage Management Facility (ISMF) panels.

7.2.1 Defining Accessibility

The storage class ACCESSIBILITY attribute defines the function of the hardware supporting concurrent copy. Concurrent copy allows database management systems (DBMSs) to take what appears to be an instantaneous copy of data. The copy can be for backup purposes (generally to tape) or for copying a database from one set of DASD volumes to another. The ACCESSIBILITY attribute enables you to direct allocation of new data sets to DASD connected to an IBM 3990 storage control unit or an RVA that supports virtual concurrent copy. New parameters for the ACCESSIBILITY attribute enable you to distinguish between a 3990 storage control and an RVA depending on whether you need concurrent copy support (including virtual concurrent copy), or SnapShot support (including DFSMSdss SnapShot).

In the ACCESSIBILITY field, you specify whether allocation to a concurrent copy capable volume is required (CONTINUOUS), preferred (CONTINUOUS PREFERRED), or discouraged (STANDARD). NOPREF is the default:

- CONTINUOUS(C) Allocate the data set on volumes that support concurrent copy. When selecting volumes for allocation, SMS considers only volumes for which concurrent copy is available. If no such volumes are available, the allocation fails. Only concurrent copy volumes are selected.
- CONTINUOUS PREFERRED(P) If possible, allocate the data set on volumes that support concurrent copy. If no such volumes are available, or if an insufficient number of such volumes are available, proceed with the allocation, using volumes where concurrent copy is not available.

© Copyright IBM Corp. 1998

- STANDARD(S) If possible, allocate the data set on volumes where concurrent copy is not available. If no such volumes are available, or if an insufficient number of such volumes are available, proceed with the allocation, using volumes where concurrent copy is available. Non-concurrent-copy volumes are preferred over concurrent copy volumes.
- NOPREF(N) Concurrent copy capability is ignored during volume selection. Both concurrent copy and nonconcurrent copy meet the accessibility request.

NOTE

ACCESSIBILITY=STANDARD *biases* allocations toward non-concurrent-copy volumes; it is the opposite of CONTINUOUS PREFERRED. There is no option to *force* allocations to non-concurrent-copy volumes. However, if a data set is allocated to a concurrent copy volume, you are not forced to use concurrent copy.

The new SMS storage class support introduces new subparameters under ACCESSIBILITY:

```
ACCESSIBILITY .......... CONTINUOUS
                         CONTINUOUS PREFERRED
                         STANDARD
                         NOPREFERENCE
              Level..... Backup=     Y|N|blank
              Level..... Versioning= Y|N|blank
```

This allows DFSMS to distinguish between a concurrent-copy-capable controller and a SnapShot-capable controller.

- Backup is to be used for concurrent copy whether it is provided on a 3990 or RVA subsystem.
- Versioning indicates a SnapShot capable device.

7.2.2 Defining Availability

You can use the storage class AVAILABILITY attribute to assign a data set to a fault-tolerant device. Such devices ensure continuous avilability for a data set in the event of a single device failure. The fault-tolerant devices that are currently available are dual copy devices and RAID architecture devices, such as the RVA.

The options listed below are available for the AVAILABILITY attribute. To ensure that a data set is allocated on a fault-tolerant device, assign the data set a storage class that specifies the AVAILABILITY attribute as CONTINUOUS.

- CONTINUOUS

  Data is placed on a dual copy or RAID device that guarantees that data can be accessed in the event of a single device failure. This establishes a mandatory availability requirement. If neither device is available, allocation fails. Dual copy, RVA, and RAMAC volumes are eligible for this setting.

- PREFERRED

  The system tries to place data on a fault-tolerant RVA device but does not guarantee placement. Dual copy volumes are not eligible for selection.

- STANDARD

This option represents normal storage needs. The system tries to allocate the data set on a device that is not a fault-tolerant device, to avoid wasting resources. In this case, processing of a data set stops in the event of a device failure. All devices, including RVAs, are eligible, except for dual copy devices.

- NOPREF

  The system chooses any devices, except for dual copy devices. NOPREF is the default.

7.2.3 Volume Preferencing

Volume preferencing enables IXPF to filter the choice of an SMS target volume for newly allocated data sets. Volume preferencing builds a list of volumes that are eligible for SnapShot, that is, they are in the same RVA as the source data set. SMS is enhanced to recognize this list of “preferred volumes.”

After determining that there is a list of preferred volumes, SMS ranks all volumes in the storage groups that are eligible for selection. SMS looks into several attributes when ranking volumes. The first two attributes remain threshold (that is below-threshold or above threshold) and volume status (that is enabled, quiesced, or disabled), and they are followed by the new preferred volume attribute in the volume ranking hierachy. The volumes on the preferred volume list are ranked higher than the volumes that are not on the list and thus have a higher chance of being selected. When volume ranking is completed, SMS performs the volume selection.

SMS volume preferencing is not supported for multistriped allocations. For striped data sets, SMS simply ignores the preferred volume interface and allocates the multistriped data set, using the current algorithm. SMS returns a special return code to the caller DFSMSdss if the allocation is not satisfied by the preferred volumes.

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 SnapShot installation. It can optionally be deactivated.

7.3 DFSMShsm and Concurrent Copy

DFSMShsm can automatically invoke concurrent copy when it creates:

- Control data set (CDS) backup copies during automatic backup
- CDS backup copies using the BACKVOL CDS command
- Backup copies of individual data sets during automatic backup
- Backup copies of individual data sets using the HBACKDS command
- Backup copies of individual data sets using inline backup
- Backup copies of an entire volume, using the BACKVOL command

DFSMShsm can also use concurrent copy for both SMS-managed and non-SMS-managed data sets when backing up an aggregate, as described in

7.5.3, “Using Virtual Concurrent Copy with Aggregate Backup and Recovery Support” on page 70.

DFSMShsm uses concurrent copy when performing the functions listed above only if the data sets concerned are system-managed and stored on volumes attached to a 3990 that supports concurrent copy or an RVA with virtual concurrent copy capability. In addition, DFSMShsm must be using DFSMSdss as the data mover for the function, and the data set′s management class must indicate that concurrent copy is required. Provided that these requirements are met, DFSMShsm automatically uses concurrent copy for the operation. There are no special DFSMShsm commands to control whether or not concurrent copy is used.

7.3.1 Concurrent Copy Management Class Support

DFSMShsm manages non-SMS data sets at the volume level, applying the management criteria to all data sets on a given volume. SMS automates the management of storage at the data set level by introducing management classes. When assigned to data sets, management classes replace and expand attributes that otherwise would be specified on JCL DD statements, IDCAMS DEFINE commands, and DFSMShsm commands.

With the BACKUP or ABACKUP COPY TECHNIQUE attribute, you hava the option of specifying whether concurrent copy should be used for data sets to enhance system availability during data set backup processing. IDCAMS data set Alter can be used to alter a management class so that data sets can use concurrent copy during backup processing. You have the option of specifying whether the concurrent copy process is required, preferred, or discouraged. The BACKUP COPY TECHNIQUE attribute is related to the Backup and Aggregate Backup components. These fields appear on the Management Class List, Display, Define/Alter, View, and Sort panels.

As with many other SMS functions, you control allocaton through a data set′s storage class, and, through its management class, you control whether or not a particular function applies to a data set.

When using DFSMSdss as the data mover, the BACKUP COPY TECHNIQUE management class attribute that is assigned to the control data sets tells DFSMShsm whether or not to direct DFSMSdss to use the concurrent copy function when backing up those data sets. (The attribute is used for a similar purpose during automatic or volume backup processing).

The BACKUP COPY TECHNIQUE attribute is used as follows:

- CONCURRENT REQUIRED (R) - Specifies that the concurrent copy facility is required. If a concurrent copy session cannot be established, the data sets will not be backed up.
- CONCURRENT PREFERRED (P) - Specifies that the concurrent copy function is preferred. If a concurrent copy session can be established, the data sets are backed up by using the concurrent copy function. Otherwise, a DFSMSdss logical dump (without concurrent copy) is used.
- STANDARD (S) - Specifies that the concurrent copy function is not to be used. A DFSMSdss logical dump without concurrent copy processing is used.

If the management class associated with a data set specifies that the concurrent copy BACKUP COPY TECHNIQUE (either REQUIRED or PREFERRED) be used, DFSMSdss is invoked specifying the CONCURRENT parameter.

If the management class BACKUP COPY TECHNIQUE is specified as CONCURRENT REQUIRED and a concurrent copy session cannot be established, the backup for that data set fails.

If the management class BACKUP COPY TECHNIQUE is specified as CONCURRENT PREFERRED and a concurrent copy session cannot be established, the data set is backed up without using concurrent copy.

You will probably want to specify BACKUP COPY TECHNIQUE=STANDARD for the majority of your data sets for which continuous accessibility is not crucial. Using concurrent copy where it is not required wastes system resources, provides no benefit, and slows DFSMShsm backup operations, because of the extra processing required to initialize the concurrent copy environment.

7.4 How DFSMShsm Uses Virtual Concurrent Copy

DFSMShsm honors the management class BACKUP COPY TECHNIQUE attribute when it backs up a data set only if it is using DFSMSdss as the data mover for the data set. DFSMShsm uses DFSMSdss as the data mover for the vast majority of data sets. DFSMShsm does not use DFSMSdss as the data mover for PDSEs. For those data sets, DFSMShsm itself is the data mover.

You can find a full list of different data set types and the data mover that DFSMShsm uses for each in the *DFSMShsm Implementation and Customization Guide*, SH21-1078. With DFSMShsm, you cannot use the SETSYS DATAMOVER command to change the data mover that DFSMShsm uses for a particular type of data set.

Regardless of the BACKUP COPY TECHNIQUE specified in the management class, DFSMShsm backs up PDSEs (and any other data set type for which DFSMShsm does not use DFSMSdss as the data mover) without using concurrent copy.

DFSMShsm uses concurrent copy when backing up a data set by creating an individual concurrent copy session for each data set as it processes the data set. There are two implications of this style of processing:

The data sets that DFSMShsm backs up during an automatic backup cycle are not protected by concurrent copy at the same time. If you update some of the data sets during the DFSMShsm backup process, the backup copies that DFSMShsm creates will not reflect a single, consistent point in time. This is the same behavior that you would observe if you updated the data sets during the DFSMShsm backup process without using concurrent copy. To obtain consistent backup copies of multiple data sets processed in this way, consider using the HBACKDS command or inline backup (ARCINBAK) once your processing is complete.

If you open a data set for update before DFSMShsm starts to back up the data set, DFSMSdss will not be able to obtain serialization on the data set and so the data set will not be backed up. DFSMShsm issues messages ARC0734I with

return code 19 and ARC1319I to warn you. Once again, this is the same behavior that you would observe if you did not use concurrent copy with DFSMShsm.

DFSMShsm updates the date last referenced and the change bit in the format 1 data set control block (DSCB) as soon as the backup copy is logically complete. The purpose of updating the format 1 DSCB after the concurrent copy session is initialized is to ensure that the change bit will be correctly set if a user updates the data set while it is being backed up.

DFSMShsm releases the SYSTEM enqueue on data sets to allow applications to update them during backup processing. DFSMShsm does not release the ARCDSN enqueue to prevent users from issuing DFSMShsm commands against data sets in use by the backup process.

During automatic backup processing, DFSMShsm suppresses messages from DFSMSdss about concurrent copy session initialization to prevent excessive console message traffic.

7.5 Backing Up DFSMShsm Control Data Sets

When DFSMShsm uses virtual concurrent copy to back up the control data sets, the period of time during which system functions are suspended is dramatically reduced. DFSMShsm reserves the data sets only for as long as it takes to initialize a concurrent copy session for each of them and to back up and nullify the journal.

To allow CDS backup using virtual concurrent copy, you must:

- Ensure that the data mover for the CDSs is DFSMSdss. If you specify:

  ```
  SETSYS CDSVERSIONBACKUP( DATAMOVER(DSS) )
  ```

  DFSMShsm invokes DFSMSdss to perform a logical dump of the CDSs and uses sequential I/O to back up the journal. DFSMSdss validates the CDSs while backing them up and uses concurrent copy if it was specified in the management class.

- Prevent the CDSs and journal from being backed up as part of data set backup. The CDSs and journal are backed up separately as specified by the SETSYS CDSVERSIONBACKUP command.

  If the CDSs and journal are SMS managed, place them on volumes that are defined in a storage group with AUTO BACKUP ===> NO, or associate them with a management class whose attributes are AUTO BACKUP = = = > N .

  If the CDSs and journal are non-SMS managed, issue the ALTERDS command to prevent them from being backed up outside CDSVERSIONBACKUP.

- Prevent the CDSs and journal from migrating. Allowing the CDSs and journal to migrate is not recommended because you might not be able to recover should any of the CDSs be damaged.

  If the CDSs and journal are SMS managed, place them on volumes that are defined in a storage group with AUTO MIGRATE ===> NO or associate them with a management class whose attributes are COMMAND OR AUTO MIGRATE ===> NONE.

If the CDSs and journal are non-SMS managed, issue the SETMIG command to prevent them from migrating.

- Ensure that the CDSs have a management class BACKUP COPY TECHNIQUE attribute of CONCURRENT REQUIRED or CONCURRENT PREFERRED.
- Ensure that the CDSs are on a DASD volume with a concurrent-copy-capable 3990 or RVA controller.

7.5.1 Control Data Set Backup Using SnapShot

You can also back up DFSMShsm CDSs directly with SnapShot; however this process is outside DFSMShsm control. Information APAR OW30915 contains documentation on how to backup your CDSs with SnapShot. If you want to reduce the time taken for CDS backup, and you do not have the correct software support for virtual concurrent copy, you can follow this procedure:

1. Quiesce DFSMShsm activity
2. Issue SnapShot for the CDSs and journal
3. Do a data set or volume level SnapShot
   - Use HOSTCOPYMODE(SHARED) for a data set SnapShot
   - Use TOLERATEENQFAILURE(YES) for a volume SnapShot
4. Inhibit journaling
5. Issue BACKVOL CDS(NULLJOURNALONLY)
6. Restart Journaling and other Hierarchical Storage Manager (HSM) activity

7.5.2 DFSMShsm Volume Dump with Virtual Concurrent Copy

DFSMSdss DUMP supports virtual concurrent copy as follows:

- It takes a SnapShot of a WSDS and dumps the snapped data in the WSDS to tape.
- DFSMShsm automatically takes advantage of virtual concurrent copy for:
  - CDS backups
  - Full volume dumps
  - Incremental backups
  - Aggregate Backup and Recovery Support (ABARS)

DFSMShsm allows use of virtual concurrent copy for DFSMShsm-managed volume dumps. Because the applicaton must manage serialization, there is no serialization across volumes. If you need to minimize data unavailability during backup, it is preferable to use ABARS. However, if you currently use DFSMShsm-managed volume dumps, use of virtual concurrent copy will reduce the length of time the VTOC is serialized during the backup.

To specify that you want DFSMShsm to use concurrent copy for volume dumps, use the VOLUMEDUMP parameter:

```
VOLUMEDUMP(CC|NOCC)
```

If you specify CC with this parameter, DFSMShsm determines whether the hardware and microcode are capable of this process, and if so, uses concurrent copy when DFSMSdss dumps volumes. If you specify NOCC with this parameter, concurrent copy will not be used when DFSMSdss dumps volumes. NOCC is the default.

The correct level of hardware and microcode must be present in your system environment for the concurrent copy function to work.

In a mixed hardware environment, where some volumes are capable of concurrent copy and others are not, CC will be accepted and applied only to those volumes capable of this process. The non-concurrent-capable volumes will be dumped by using traditional methods.

7.5.3 Using Virtual Concurrent Copy with Aggregate Backup and Recovery Support

ABARS is DFSMShsm′s flagship of disaster recovery functions.

ABARS can use concurrent copy when backing up system-managed and non-system-managed data sets from level zero DASD. ABARS does not use concurrent copy when backing up data sets from ML1 or ML2. There would be no advantage to using concurrent copy because applications cannot update ML1 or ML2 data sets.

ABARS uses a single invocation of DFSMSdss with optimization level 3 to back up data sets from level zero DASD. As a consequence, with concurrent copy, DFSMSdss initializes a single session for all data sets concerned. If you quiesce activity to those data sets before starting the ABACKUP procedure, the backup copy will represent a single, consistent point in time for all of the data sets. ABARS indicates that the ABACKUP is logically complete by issuing the following message:

```
ARC6402I CONCURRENT COPY INITIALIZATION IS COMPLETE FOR AGGREGATE
         agname. ANY SERIALIZATION HAS BEEN RELEASED. JOB= jobname.
```

You can use this message in the same way as you use the DFSMSdss ADR743I message in automation procedures to restart applications that access the data sets contained in the aggregate.

You control whether or not ABARS uses concurrent copy through parameters in the management class and by associating a management class with a particular aggregate group.

7.5.3.1 Controlling When ABARS Uses Virtual Concurrent Copy

The management class ABACKUP COPY TECHNIQUE attribute controls whether or not ABARS uses concurrent copy when backing up an aggregate group. The ABACKUP COPY TECHNIQUE attribute has three possible values:

- CONCURRENT REQUIRED

  ABARS should use concurrent copy when backing up the level zero data sets in the aggregate group. If concurrent copy is not available for all or part of any data set, ABARS fails the entire aggregate backup.

- CONCURRENT PREFERRED

  ABARS should use concurrent copy if possible when backing up the level zero data sets in the aggregate group. If concurrent copy is not available for all or part of any data sets, ABARS uses standard copy (and retains serialization) for that data set and continues with the remainder of the aggregate group. Serialization is held for the entire aggregate until every data set copy is either logically complete (using concurrent copy or virtual concurrent copy) or physically complete.

- STANDARD

  ABARS should not use concurrent copy when backing up the aggregate group, even if it is available. Consequently, ABARS retains serialization on the data sets until the ABACKUP is complete.

The ABACKUP COPY SERIALIZATION parameter controls whether or not ABACKUP uses the DFSMSdss TOLERATE(ENQFAILURE) parameter when backing up the level zero data sets in the aggregate group. The ABACKUP COPY SERIALIZATION attribute has three possible values:

- CONTINUE

  Back up the aggregate group even if DFSMSdss cannot obtain a shared ENQ against one or more of the level zero data sets.

- FAIL

  Back up the aggregate group only if DFSMSdss can obtain a shared ENQ against all of the level zero data sets. If DFSMSdss cannot obtain the ENQ against any of the level zero data sets, ABARS fails the backup.

- blank

  Equivalent to FAIL.

When you specify that ABARS should use concurrent copy in the BACKUP COPY TECHNIQUE attribute, DFSMSdss holds the shared ENQ only long enough to initialize the concurrent copy session. For data sets stored on volumes that do not support concurrent copy, DFSMSdss holds the ENQ until it has dumped the data sets.

If you do not specify a management class for an aggregate, ABARS uses a default management class that uses STANDARD copy and FAIL for copy serialization.

7.5.3.2 Defining Aggregate Backup Attributes

The fifth page of the Management Class Define panel is used to define the aggregate backup attributes (Figure 31 on page 72).

```
********************************************************************************
                 MANAGEMENT CLASS DEFINE                    Page 5 of 5

Command ===>

SCDS Name . . . . . : SMS.SCDS1.SCDS

Management class Name : SMS.SCDS1.SCDS


To DEFINE Management Class, Specify:

  AGGREGATE backup Attributes:

    # Versions . . . . . .        (1 o 9999, NOLIMIT or blank)

    Retain Only Version  . . .   (1 o 9999, NOLIMIT or blank)

      Unit . . . . . .           (D=days, W=weeks, M=months,

                                 Y=years or blank)

    Retain Extra Version   . .   (1 o 9999, NOLIMIT or blank)

      Unit . . . . . .           (D=days, W=weeks, M=months,

                                 Y=years or blank)

    Copy Serialization . . . .   (C=continue, F=fail or blank)

    Abackup Copy Technique . . S (P=Conc Preferred, R=Conc Required

                                 or S=Standard)


Use ENTER to Perform Verification; Use UP Command to View previous
Panel;
Use HELP Command for Help; Use END Command to Save and EXIT;
CANCEL to Exit.

********************************************************************************
```

Figure 31. Management Class Define Panel: Aggregate Backup Attributes

When you define the aggregate backup attributes, the following fields are relevant:

- The Source Control Data Set (SCDS) Name field specifies the name of the SCDS into which the management class is defined.
- The # Versions field specifies the number of versions to be maintained.
- The Retain Only Version field indicates how long the most recent backup version of an aggregate group is kept.
  - The Unit field specifies the unit of measure for the time period specified for the Retain Only Version field.
  - This field cannot be blank if the Retain Only Version field is specified, except when Retain Only Version is NOLIMIT.
- The Retain Extra Versions field indicates how long to keep backup versions of an aggregate group that precede the most recent version.

  - The Unit field specifies the unit of measure for the time period specified for the Retain Extra Versions field. This field cannot be blank if the Retain Extra Versions field is specified, except when Retain Extra Versions is NOLIMIT.
- The Copy Serialization field specifies whether aggregate backup should continue if an enqueue failure is encountered.
- The Abackup Copy Technique field specifies whether concurrent copy should be used in conjunction.

7.6 Using Concurrent Copy with Removable Media Manager

When you make a backup of the Removable Media Manager(RMM) CDS, serialization prevents updates to the CDS and journal from occurring until the backups are complete, thus ensuring the integrity of the backups so that recovery is possible. While the backups are taking place, however, any tape processing involving updates to the DFSMSrmm CDS or journal has to wait until the backups are complete. If the CDS is large, this outage could last for an unacceptably long time.

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

- The CDS and journal can be backed up without affecting tape use.
- Updates resulting from tape use or commands are permitted when backup is in progress and do not affect the resulting backup copy.
- Backup can now be directly to tape.

To use DFSMSrmm nonintrusive backup, an installation must have a DASD subsystem controller that supports concurrent copy, either an IBM 3990-6 control unit or an RVA with virtual concurrent copy support.

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

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

To use concurrent copy to back up the DFSMSrmm CDS, the source and target data sets must be on the same RVA.

Following these steps to use concurrent copy for DFSMSrmm CDS backup:

1. Update the EDGHSKP program.
2. Specify BACKUP(DSS) for BACKUP(AMS|DSS) when you want DFSMSdss to perform the backup.

The DFSMSdss operands are specified in the DSSOPT DD statement, in the EDGHSKP and EDGBKUP utilities. To use DFSMSdss concurrent copy, you must have a concurrent copy environment set up.

When using BACKUP(DSS) without concurrent copy, you can still direct the output to tape, but all updates to the CDS will wait until the backup is complete. BACKUP(AMS) is the default. DFSMSrmm uses the Access Method Services (AMS) REPRO command to perform backup processing. Specifying BACKUP(AMS) prevents updating the CDS during backup processing.

When using BACKUP(DSS) without concurrent copy, you can still direct the output to tape, but all updates to the CDS will wait until the backup is complete.

Protect the DFSMSdss commands in the RACF FACILITY class.

You can use EDGHSKP or EDGBKUP with DFSMSdss DUMP or the AMS REPRO command to create a backup copy of your DFSMSrmm CDS and journal. When you use DFSMSdss concurrent copy you may have to define or update the resource STGADMIN.ADR.DUMP.CNCURRNT.

7.7 Direct Access Device Space Management Enhancements

Deleting a data set using conventional methods calls direct access device space management (DADSM) processing. DADSM deletes VTOC pointers when it deletes a data set. In an RVA it is recommended that freespace in the log structured array be explicitly released. This function is performed by the deleted data space release (DDSR) function of IXFP.

DDSR obtains control in the MVS DADSM scratch and release functions. After a data set is deleted, dynamic DDSR issues a release for all extents used by the deleted data set. These freed extents are then available for collection by the free space collection process.

The IXFP subsystem installs a preprocessing exit that affects IBM user exit IGGPRE00. The hook must be the last installed, which sometimes causes problems with non-IBM products that also have the same requirement.

Two enhancements have been made to DADSM code to integrate DDSR with DFSMS. The new code eliminates the need for the IGGPRE00 preprocessing installation exit to invoke DDSR and improves ″erase-on-scratch″ enhancements. The first enhancement prevents IGGPRE00 collisions between customer and vendor code and IXFP′s DDSR launch from IGGPRE00.

The second enhancement improves DADSM erase-on-scratch handling, bypassing DADSM′s own erasure code when DDSR successfully erases tracks and retrying using DADSM erasure code in the event that DDSR fails to erase requested tracks.

This support is provided as follows:

1. OW29642
    - UW43360 DFSMS/MVS 1.1
    - UW43357 DFSMS/MVS 1.2
    - UW43358 DFSMS/MVS 1.3
    - UW43359 DFSMS/MVS 1.4

2. OW29816
    - L170571 ixfp 2.1

Chapter 8. Backup-While-Open for CICS and IMS

For high-availability environments, it may not be possible or desirable to quiesce an application to produce consistent copies of application data sets. For such environments, including CICS and IMS, DFSMSdfp provides support to back up SMS-managed VSAM data sets that are open for output. The support is useful only when the application can recover a restored database to a point of consistency, typically from a forward recovery log maintained by the application or DBMS.

DFSMSdss supports backup-while-open (BWO) serialization, which can back up data sets that are open for update for long periods of time. BWO is a better method of backup than using SHARE options or TOLERATE(ENQFAILURE) for data sets that are open for update while DFSMSdss is making a copy. Using BWO allows CICS or IMS to log forward recovery images of changes to the data sets. At a later time the backup of the data set can be restored and brought to a point of consistency by using a forward recovery utility.

While the data set is being dumped, a database application can update the data set in a manner that invalidates the data set backup. For example, a control-interval (CI) or control-area (CA) split may occur. If this happens, the backup is unreliable and should be discarded.

8.1 Backup-While-Open and Concurrent Copy

Concurrent copy improves BWO processing by significantly reducing the possibility of invalidating a BWO dump because of updates to the data set.

If DFSMSdss can obtain serialization on the data set, it will be retained for logical completion. Once the concurrent copy initialization is completed, the dump takes place and the data set is available for update.

When BWO is specified, and DFSMSdss cannot obtain serialization on the data set, it is dumped and can remain in use for the entire time. Any updates that occur after logical completion do not cause the dump to be invalidated.

To use concurrent copy, specify the CONCURRENT keyword when you dump BWO data sets.

8.2 Backup-While-Open and CICS Data Sets

BWO is a better method of backup than using SHARE or TOLERATE(ENQFAILURE) for dumping CICS VSAM file-control data sets that are in use and open for update.

When you dump data sets that are designated by CICS as eligible for BWO processing, data integrity is maintained through serialization interactions between CICS; (the database control program); CICS VSAM Recovery (CICSVR), (the forward recovery program); VSAM record management; DFSMSdfp; and DFSMSdss.

The support for BWO in CICS enables some types of VSAM data sets to be backed up by DFSMSdss while CICS is currently updating these data sets. At

© Copyright IBM Corp. 1998

the same time, CICS logs forward recovery images of any changes to these data sets on a forward recovery journal. At a later date the backup of the data set can be restored and brought to a point of consistency by applying the forward recovery logs and using a forward recovery utility such as CICSVR.

BWO is available only for:

- Data sets that are accessed by CICS file control
- Data sets that are SMS managed
- VSAM extended sequential data sets (ESDSs), RRDSs (both fixed and variable), and KSDSs

There are two ways of defining BWO:

1. By defining the cluster with parameter BWO with the value of TYPECICS. TYPECICS indicates that the data set is eligible for BWO in a CICS region. The file resource definition is ignored if the BWO parameter is specified.
2. If the BWO parameter is not defined, it defaults to UNDEFINED. In this case CICS looks at the file resource definition. Clusters that are to be opened in record level sharing (RLS) mode must have BWO specified in the cluster definition.

   CICS defines a data set as eligible for BWO when the file is defined by using resource definition online (RDO), and BACKUPTYPE=DYNAMIC is specified for a VSAM file.

If data sets are updated in RLS mode, BWO is managed entirely by DFSMS. When a BWO copy is made, DFSMSdss sends a message to all CICS systems in the sysplex that have open access control blocks (ACBs) for the VSAM sphere. The CICS systems keep track of all current units of work that have updated files for the sphere. When all of the units of work have completed, CICS writes tie-up records and notifies DFSMSdss. The copy is complete when all CICS systems have responded.

If data sets are updated in non-RLS mode, and the BACKUPTYPE is DYNAMIC, CICS calls DFSMSdfp to update the Integrated Catalog Facility (ICF) catalog to indicate that the base cluster data set is eligible for BWO while it is under the control of CICS. CICS records the fact that the data set is eligible for BWO. When the last file associated with the base cluster is closed, DFSMSdfp updates the ICF catalog to indicate that the data set is no longer eligible for BWO. This prevents BWO during the batch window between CICS sessions.

Before DFSMSdss or DFSMShsm takes a backup of the VSAM sphere, the ICF catalog is examined to check whether BWO is required. If BWO is required, the backup is made without trying to get exclusive control and serialize updates to the data set.

When a backup copy of a data set is restored by DFSMSdss or DFSMShsm, and the backup type is BWO, the catalog is updated to indicate that the data set needs to be forward recovered before it can be used. CICS checks this at data set open time. Once the data set is restored, the forward recovery utility is used to bring the data set to a point of consistency.

8.3 Backup-While-Open and IMS

DFSMSdss supports BWO processing of IMS data sets by allowing concurrent copy dumps. IMS requests the use of concurrent copy through the DFSMSdss API.

BWO is applicable to HISAM, SHISAM, and index (primary and secondary) databases. In addition non-VSAM IMS data sets can also be dumped with BWO.

You can define the base cluster with the BWO(TYPEIMS) paramater. It is not required for the data set to be dumped while open, but it is required for DFSMSdss to do split detection.

A VSAM KSDS cluster must be defined as BWO(TYPEIMS) if concurrent updaters are allowed.

DFSMSdss COPY and DUMP do not provide any special handling for data sets that are defined as BWO(TYPEIMS) unless the BWO request is triggered by IMS through a user interaction module (UIM) request.

For further details on IMS use of virtual concurrent copy, see Chapter 10, “Using Virtual Concurrent Copy for IMS” on page 105.

Chapter 9. Using DFSMSdss SnapShot and Virtual Concurrent Copy with DB2

With the availability of DFSMSdss SnapShot and virtual concurrent copy, customers can now manage and use their DB2 data effectively. Using either DFSMSdss SnapShot or 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
- Quickly replicate large amounts of data for application testing purposes

9.1 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 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 recovers use 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.

© Copyright IBM Corp. 1998

9.2 Benefits of Virtual Concurrent Copy and DFSMSdss SnapShot with DB2

Using virtual concurrent copy and DFSMSdss SnapShot for your DB2 data has many benefits in the areas of DB2 image copies, disaster recovery, creating test data, and making multiple copies of your DB2 data.

9.2.1 Image Copy

Now that that the concurrent copy functionality has been expanded to include virtual concurrent copy, 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.

9.2.1.1 Types of DB2 Image Copies

The DB2 COPY utility enables you to make two levels of copies: SHRLEVEL CHANGE and SHRLEVEL REFERENCE.

SHRLEVEL CHANGE A SHRLEVEL CHANGE copy is sometimes referred to as a *fuzzy* copy, because some of the data on the copy might be uncommitted. Never use SHRLEVEL CHANGE in a recover TOCOPY situation. When SHRLEVEL CHANGE is used in a recover situation, always recover to a *consistent* point after the copy had been taken. A *consistent* point to which to recover a DB2 object could be any of the following:

- The value for when a QUIESCE utility was run against the object. This value can be obtained from SYSIBM.SYSCOPY for records that have an ICTYPE value of *Q* for the object you are going to recover. You need to use the value from the START_RBA column, which is either a relative byte address (RBA) value (in s non-data-sharing environment) or a log record sequence number (LRSN) value (in a data-sharing environment).
- The value for when an ARCHIVE LOG MODE(QUIESCE) was performed, but only in a non-data sharing environment. Use the end RBA value of the archive log that was created as a result of the command as the TORBA value in your recover utility.
- The RBA value when DB2 was stopped, but only in a non-data sharing environment (unless all DB2 members are stopped, at which point you need the highest LRSN value of the member that stopped last). You then use the this RBA or LRSN value as either the TORBA or TOLOGPOINT value in your recover utility. You must use this RBA or LRSN value as your consistent recovery point.

A recovery to a consistent point in time after the copy was made involves the following processes:

1. The image copy for the object being recovered is restored.
2. All changes to the object that occurred between the time the copy was made and up to the consistent recovery point are applied to the object by using DB2 log data.

SHRLEVEL REFERENCE A SHRLEVEL REFERENCE copy is sometimes referred to as a *consistent* copy because all of the data on the copy has been committed. You can use a SHRLEVEL REFERENCE copy in a recover TOCOPY situation. You

can also use it to recover your data to a consistent point in time after the copy was made by applying log data in the recover process, just as you would when using a SHRLEVEL CHANGE copy.

9.2.2 Disaster Recovery

To ensure the consistency of your data for disaster recovery purposes, you can take advantage of virtual concurrent copy to make SHRLEVEL REFERENCE DB2 image copies with minimal outage to your applications. This technique can be used to coordinate your DB2 recovery with another DBMS and to copy all of your data at the same consistent point in time. Also, if your disaster recovery plan involves recovering your DB2 data to the most recent image copy to minimize the amount of DB2 log data to be processed, use virtual concurrent copy to create consistent copies with minimal outage to your applications.

If your DB2 disaster recovery plan uses data set backups or volume dumps to restore your DB2 environment, you can also take advantage of DFSMSdss SnapShot to copy large amounts of data to the same consistent point with minimal outage to your applications.

9.2.2.1 Using Image Copies for Disaster Recovery

Because with the DB2 COPY utility you can create up to 4 output copies of your data, you can also use virtual concurrent copy to make the copies and minimize application outages while the copies are being made. The copies can then be sent offsite, and you can use standard DB2 recovery procedures to recover your data.

Note

Even if you use virtual concurrent copy to copy all of your DB2 data, we recommend having all DB2 log data and the most recent copy of your DB2 bootstrap data set (BSDS) available at the disaster recovery site. If any of your DB2 data cannot be recovered with the most recent copy, the previous copy and log data can be used to recover an object to the point in time to which you are recovering the rest of your DB2 subsystem.

9.2.2.2 Using DFSMSdss SnapShot at Data Set Level for Disaster Recovery

You can now take advantage of wild-carding and filtering within DFSMSdss to back up all DB2 data needed for a disaster recovery with minimal outage to your applications and using very few DFSMSdss SnapShot control statements.

9.2.2.3 Using DFSMSdss SnapShot at Volume Level for Disaster Recovery

If you are currently using volume dumps and restores for your disaster recovery planning, you can use virtual concurrent copy to significantly reduce the time required to back up your data, thus minimizing the outage to your applications. You can realize the benefits of virtual concurrent copy and not have to change your current disaster recovery processes.

9.2.3 Creating Test Data and Systems

DFSMSdss 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 DFSMSdss 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, DFSMSdss SnapShot 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.

9.2.4 Making Multiple Copies of Your DB2 Data

If you are currently using the DB2 COPY utility to make more than one backup of your DB2 data with the DB2 COPY utility you can make up to four output copies at a time, virtual concurrent copy can further help minimize your application outage during the copy process. As you may well be aware when you create multiple backups, the amount of I/O performed during the copying increases. Because the I/O for each copy can contend with one another, the COPY utility takes longer to complete. Because with virtual concurrent copy the copy process is logically complete in a very short amount of time and very little I/O is performed during the logical copy phase, making additional copies has no impact on the time required to complete the logical copy. The I/O for reading the source and writing to multiple targets is performed during the physical copy phase, and during this time applications are allowed to update the source data of the copy operation.

The output from a virtual concurrent copy is a DFSMSdss DUMP data set. This output cannot be processed directly by the DB2 utilities. If you use your image copy output for processing with another DB2 utility such as DSN1COPY, make such copies separately from your virtual concurrent copies.

9.3 SnapShot Positioning

SnapShot provides benefits for customers who require a point-in-time copy of data, where data consistency is of prime importance. Recovering to a consistent point-in-time copy enables you to recover quickly and coordinate recovery of multiple separate applications that may be related. This is especially relevant where your application suite spans both DB2 and non-DB2 applications. If your prime objective is service availability, DB2 provides a comprehensive series of utilities that enable you to run DB2 with a minimum of disruption.

Table 9 on page 85 makes a quick comparison between a number of copy options to help you decide which software utility may best suit your business objective.

Table 9. Copy options

Business Objective	SHRLEVEL CHANGE	SHRLEVEL REFERENCE	SHRLEVEL REFERENCE CONCURRENT	SHRLEVEL REFERENCE CONCURRENT List of Objects	SHRLEVEL REFERENCE CONCURRENT FILTERDDN(xxx) List of Objects	DFSMSdss Data Set COPY or DUMP CONCURRENT	DFSMSdss VOLUME COPY or DUMP CONCURRENT
Availability	*Continuous*	*High*	*Very high with minimal outage*	*High (1)*	*Very high with minimal outage*	*Very high with minimal outage*	*High with minimal outage*
Effort to implement	*Easy; use DB2 utilities*	*Easy; use DB2 utilities*	*Easy; use DB2 utilities*	*Easy; use DB2 utilities*	*Easy; use DB2 utilities*	*More work for user (2) (3)*	*More work for user (2) (3)*
Recoverablity	*Use DB2 utilities; will require log apply to recover to a consistent point*	*Use DB2 utilities (4)*	*Use DB2 utilities (4)*	*Use DB2 utilities (4)*	*Use DB2 utilities (4)*	*DFSMSdss RESTORE (3) (5)*	*DFSMSdss VOLUME RESTORE (3) (5)*
Application testing (including Year 2000)	*Not recommended (data not guaranteed to be consistent)*	*Good for a single object*	*Excellent for a single object*	*Excellent for list of objects*	*Excellent for list of objects*	*Might be easy to implement using wildcards*	*Might be easy to implement using wildcards*
Cloning DB2	*Not recommended (data not guaranteed to be consistent)*	*Not likely*	*Not likely*	*Excellent for list of objects*	*Excellent for list of objects*	*Might be easy to implement using wildcards*	*Might be easy to implement using wildcards*
Disaster recovery	*Requires log apply to restore data to a consistent point*	*Good for recovery of single object to a consistent point (4)*	*Good for recovery of single object to a consistent point (4)*	*Good for recovery of a list of objects to a consistent point (4)*	*Good for recovery of a list of objects to a consistent point (4)*	*User performs data set restore procedures*	*User performs volume recovery procedures*

Note:
1: Though more objects and large object size can can make the outage longer. **2:** The copy process will involve several steps, and, for any future changes, each step has to be modified. **3:** If recovering the objects to a point in time other than TOCOPY, log apply will be necessary. **4:** Because DFSMSdss is invoked for each data set that is recovered, each step has to be modified. **5:** Might be easily integrated into existing disaster recovery plans.

9.4 DB2 Backup and Recovery Procedures Using Virtual Concurrent Copy

In this section we describe DB2 backup and recovery procedures using virtual concurrent copy. If you are not already using the CONCURRENT keyword with your DB2 COPY utility, all you have to do to begin using virtual concurrent copy is add the CONCURRENT keyword to the control statements of your existing DB2 COPY jobs. If you are going to copy a list of objects, using virtual concurrent copy, you should also make one additional syntax change and a job control language (JCL) change. To take full advantage of virtual concurrent copy and minimize as much as possible the outage of your data during the copy operation, you have to add the FILTERDDN(xxxxxxxx) keyword to your DB2 COPY utility control statements. Then change the JCL of your DB2 COPY utility to have only a single output data set for the copy operation with a ddname that matches the value you associated with the FILTERDDN keyword.

Figure 32 on page 86 shows a DB2 COPY of a list of data sets without virtual concurrent copy capability.

```
//CCCOPY   EXEC PGM=DSNUTILB,REGION=1024K,
//             PARM='DB2B,DSNTEX'
//STEPLIB  DD DSN=DB2.LINKLIB,DISP=SHR
//*
//*****************************************************
//*  DATA SETS USED BY THE UTILITY
//*****************************************************
//SYSCOPY1  DD DSN=CONCOPY1.GROUP00.OSMOTS32.SYSCOPY1,
//             DISP=(MOD,CATLG,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(42,5),RLSE)
//SYSCOPY2  DD DSN=CONCOPY1.GROUP01.OSMOTS32.SYSCOPY2,
//             DISP=(MOD,CATLG,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(42,5),RLSE)
//SYSCOPY3  DD DSN=CONCOPY1.GROUP02.OSMOTS32.SYSCOPY3,
//             DISP=(MOD,CATLG,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(42,5),RLSE)
//SYSCOPY4  DD DSN=CONCOPY1.GROUP03.OSMOTS32.SYSCOPY4,
//             DISP=(MOD,CATLG,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(42,5),RLSE)
//SYSCOPY5  DD DSN=CONCOPY1.GROUP04.OSMOTS32.SYSCOPY5,
//             DISP=(MOD,CATLG,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(42,5),RLSE)
//SYSCOPY6  DD DSN=CONCOPY1.GROUP05.OSMOTS32.SYSCOPY6,
//             DISP=(MOD,CATLG,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(42,5),RLSE)
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD *
  COPY TABLESPACE GROUP00.OSMOTS32
     COPYDDN(SYSCOPY1)
       TABLESPACE GROUP01.OSMOTS32
     COPYDDN(SYSCOPY2)
       TABLESPACE GROUP02.OSMOTS32
     COPYDDN(SYSCOPY3)
       TABLESPACE GROUP03.OSMOTS32
     COPYDDN(SYSCOPY4)
       TABLESPACE GROUP04.OSMOTS32
     COPYDDN(SYSCOPY5)
       TABLESPACE GROUP05.OSMOTS32
     COPYDDN(SYSCOPY6)
    SHRLEVEL REFERENCE
```

Figure 32. DB2 COPY without Virtual Concurrent Copy Capability

Figure 33 on page 87 shows the same DB2 COPY utility job virtual concurrent copy capability.

```
//CCCOPY   EXEC PGM=DSNUTILB,REGION=1024K,
//             PARM='DB2B,DSNTEX'
//STEPLIB  DD DSN=DB2.LINKLIB,DISP=SHR
//*
//*****************************************************
//*  DATA SETS USED BY THE UTILITY
//*****************************************************
//FILTERDD DD DSN=&&TEMPFIL,DISP=(,DELETE,DELETE),
//            UNIT=SYSALLDA,SPACE=(CYL,(1,1),RLSE)
//SYSCOPY   DD DSN=CONCOPY1.GROUP00.OSMOTS32.SYSCOPY1,
//             DISP=(MOD,CATLG,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(200,25),RLSE)
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD *
  COPY TABLESPACE GROUP00.OSMOTS32
       TABLESPACE GROUP01.OSMOTS32
       TABLESPACE GROUP02.OSMOTS32
       TABLESPACE GROUP03.OSMOTS32
       TABLESPACE GROUP04.OSMOTS32
       TABLESPACE GROUP05.OSMOTS32
    FILTERDDN(FILTERDD)  COPYDDN(SYSCOPY)
    CONCURRENT SHRLEVEL REFERENCE
```

Figure 33. DB2 COPY with Virtual Concurrent Copy Capability

Once the logical copy for every object has completed, all of the objects are available for updating. You must modify existing jobs that have steps that first back up your data followed by steps that update your data. You need to place your update steps in a new separate job, so that the updates can begin as soon as the logical copy portion of the virtual concurrent copy has completed. The new job can then be scheduled to execute after the DFSMSdss message indicating logical copy completion for all objects. You can automate the execution of jobs that are to follow the logical copy completion, by having them execute after the message shown in Figure 34 is issued to the console from the DB2 COPY job step.

```
15.39.46 JOB00310  ADR734I (002)-DTDSC(01), 1998.063 15:39:46 CONCURRENT
                   INITIALIZATION SUCCESSFUL FOR 6 OF 6 SELECTED DATA SERIALIZATION
                   FOR THIS DATA IS RELEASED IF DFSMSDSS HELD IT. THE INIATE RETURN
                   CODE IS 0004.
```

Figure 34. Console Message Issued after Logical Copy Completion

If you implement virtual concurrent copy and continue to allow your update processes to reside in the same job as the backups, you will not realize any benefit from virtual concurrent copy. The backup step will not end until the data has been physically copied, even though the data is available for update as soon as the logical copy completes.

9.4.1 DB2 Copy Using SHRLEVEL CHANGE CONCURRENT

To copy a single DB2 object and allow it to be updated by your applications during the entire copy process, use the DB2 COPY utility with the SHRLEVEL CHANGE keyword. If you execute the DB2 COPY utility specifying SHRLEVEL CHANGE CONCURRENT keyword values (Figure 35), virtual concurrent copy will be used to actually perform the copy operation. When SHRLEVEL CHANGE is specified, uncommitted data might be copied, thus creating a *fuzzy* copy.

Note

Image copies taken using SHRLEVEL CHANGE are not recommended for use with RECOVER TOCOPY. Always recover to a consistent point in time after the copy has been taken.

```
//CCCOPY   EXEC PGM=DSNUTILB,REGION=1024K,
//             PARM='DB2B,DSNTEX'
//STEPLIB  DD DSN=DB2.LINKLIB,DISP=SHR
//*
//*****************************************************
//*  DATA SETS USED BY THE UTILITY
//*****************************************************
//SYSCOPY1  DD DSN=CONCOPY1.GROUP00.OSMOTS32.SYSCOPY1,
//             DISP=(MOD,CATLG,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(42,5),RLSE)
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD *
    COPY TABLESPACE GROUP00.OSMOTS32
         COPYDDN(SYSCOPY1)
         CONCURRENT SHRLEVEL CHANGE
```

Figure 35. DB2 COPY Using the SHRLEVEL CHANGE CONCURRENT Option

When a single object is copied with the DB2 COPY utility using the SHRLEVEL CHANGE CONCURRENT option, the following actions occur:

1. The DB2 object is placed in a status of utility read write (UTRW).
2. DFSMSdss is invoked to make the copy (which is done as a DFSMSdss DUMP).
3. After the data sets have been logically dumped, DFSMSdss issues message ADR734I, indicating that the logical dump completed.
4. DFSMSdss begins physically dumping the data sets to the output data set.
5. Once the physical copy of all data sets has completed, control is returned to DB2.
6. A *C* value is placed in the SHRLEVEL column and an *F* value is placed in the ICTYPE column for the row that is inserted into SYSIBM.SYSCOPY for the object that was copied.
7. DB2 places the object back into a read write (RW) status, and the utility ends.

This copy technique creates a *fuzzy* copy and should not be used in a RECOVER TOCOPY recovery. A copy made specifying SHRLEVEL CHANGE should be used only when recovering to a consistent point in time.

Note

If you use the DB2 COPY utility with the SHRLEVEL CHANGE CONCURRENT keywords, we recommend that, prior to the COPY you run the DB2 QUIESCE utility, specifying the WRITE(YES) option, to force DB2 to externalize any updated pages that are in its buffers for the object. You need to do this to minimize the amount of log required to recover to a point in time after the copy has been taken.

9.4.2 DB2 Copy Using SHRLEVEL REFERENCE CONCURRENT

To copy a single DB2 object and ensure that the copy you create is a *consistent* copy, use the DB2 COPY utility with the SHRLEVEL REFERENCE keyword. If you execute the DB2 COPY utility specifying SHRLEVEL REFERENCE CONCURRENT keyword values (Figure 36), virtual concurrent copy will be used to actually perform the copy operation. After virtual concurrent copy *logically* completes the copy, the data is made available for updating. Without the CONCURRENT keyword, the data would be copied by the DB2 utility and would not be available for updating until all the data has been *physically* copied.

```
//CCCOPY   EXEC PGM=DSNUTILB,REGION=1024K,
//             PARM='DB2B,DSNTEX'
//STEPLIB  DD DSN=DB2.LINKLIB,DISP=SHR
//*
//*******************************************************
//*  DATA SETS USED BY THE UTILITY
//*******************************************************
//SYSCOPY1  DD DSN=CONCOPY1.GROUP00.OSMOTS32.SYSCOPY1,
//             DISP=(MOD,CATLG,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(42,5),RLSE)
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD *
    COPY TABLESPACE GROUP00.OSMOTS32
         COPYDDN(SYSCOPY1)
         CONCURRENT SHRLEVEL REFERENCE
```

Figure 36. DB2 COPY Using the SHRLEVEL REFERENCE CONCURRENT Option.

When a single object is copied with the DB2 COPY utility using the SHRLEVEL REFERENCE CONCURRENT option, the following actions occur:

1. The DB2 object is placed in a status of utility read only (UTRO), all writers are drained for the object, the object is quiesced, and a row is inserted into SYSIBM.SYSCOPY with an ICTYPE value of *Q*.
2. DFSMSdss is invoked to make the copy (which is done as a DFSMSdss DUMP).
3. After the data sets have been logically dumped, DFSMSdss issues message ADR734I, indicating that the logical dump completed.
4. The DB2 object now has its status changed to utility read write (UTRW) and is available for updating.
5. DFSMSdss begins physically dumping the data sets to the output data set.
6. Once the physical copy of all data sets has completed, control is returned to DB2.

7. The *Q* row that had been inserted into SYSIBM.SYSCOPY for the object that has been just been copied now has it's ICTYPE value updated to be an *F* and its SHRLEVEL value is changed to a *R*.
8. DB2 places the object back in a RW status and the utility ends.

This copy technique ensures that the copy being made is a consistent copy and that inability to update the object during the copy process has been minimized.

9.4.3 DB2 Copy Using CONCURRENT for List of Objects

To copy a group of DB2 objects to the same consistency point, you can use the DB2 COPY utility with the SHRLEVEL REFERENCE CONCURRENT option and list of DB2 objects specified see Figure 37.

```
//CCCOPY   EXEC PGM=DSNUTILB,REGION=1024K,
//             PARM='DB2B,DSNTEX'
//STEPLIB  DD DSN=DB2.LINKLIB,DISP=SHR
//*
//*******************************************************
//*  DATA SETS USED BY THE UTILITY
//*******************************************************
//SYSCOPY1  DD DSN=CONCOPY1.GROUP00.OSMOTS32.SYSCOPY1,
//             DISP=(MOD,CATLG,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(42,5),RLSE)
//SYSCOPY2  DD DSN=CONCOPY1.GROUP01.OSMOTS32.SYSCOPY2,
//             DISP=(MOD,CATLG,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(42,5),RLSE)
//SYSCOPY3  DD DSN=CONCOPY1.GROUP02.OSMOTS32.SYSCOPY3,
//             DISP=(MOD,CATLG,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(42,5),RLSE)
//SYSCOPY4  DD DSN=CONCOPY1.GROUP03.OSMOTS32.SYSCOPY4,
//             DISP=(MOD,CATLG,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(42,5),RLSE)
//SYSCOPY5  DD DSN=CONCOPY1.GROUP04.OSMOTS32.SYSCOPY5,
//             DISP=(MOD,CATLG,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(42,5),RLSE)
//SYSCOPY6  DD DSN=CONCOPY1.GROUP05.OSMOTS32.SYSCOPY6,
//             DISP=(MOD,CATLG,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(42,5),RLSE)
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD *
  COPY TABLESPACE GROUP00.OSMOTS32
     COPYDDN(SYSCOPY1)
       TABLESPACE GROUP01.OSMOTS32
     COPYDDN(SYSCOPY2)
       TABLESPACE GROUP02.OSMOTS32
     COPYDDN(SYSCOPY3)
       TABLESPACE GROUP03.OSMOTS32
     COPYDDN(SYSCOPY4)
       TABLESPACE GROUP04.OSMOTS32
     COPYDDN(SYSCOPY5)
       TABLESPACE GROUP05.OSMOTS32
     COPYDDN(SYSCOPY6)
    CONCURRENT SHRLEVEL REFERENCE
```

and List of DB2 Objects.

Figure 37. DB2 COPY Using the SHRLEVEL REFERENCE CONCURRENT Option

When a list of objects is copied with the DB2 COPY utility specifying the SHRLEVEL REFERENCE CONCURRENT option, the following actions occur:

1. All DB2 objects are placed in a status of UTRO, all writers are drained for all the objects, each object is quiesced and a row with an ICTYPE value of *Q* is inserted into SYSIBM.SYSCOPY for each object in the list.
2. DFSMSdss is invoked to logically copy (which is done as a DFSMSdss DUMP) the first object specified in the list.
3. After the data set has been logically dumped, DFSMSdss issues message ADR734I, indicating that the logical dump has completed.

   Note

   Message ARD734I is issued once for each data set that is *logically* copied.

4. The DB2 object that has been *logically* copied has its status changed to UTRW and is available for updating.
5. DFSMSdss begins physically dumping the data sets to the output data set.
6. Once the physical copy of a DB2 object has completed, Steps 2 through 5 are repeated for every object in the list.
7. Once the physical copy of all data sets has completed, control is returned to DB2.
8. The *Q* rows that had been inserted into SYSIBM.SYSCOPY for the objects that have been copied now have their ICTYPE values updated to an *F* and their SHRLEVEL values changed to an *R*.
9. DB2 places all objects back in RW status, and the utility ends.

Note

If all of the output data sets reside on different volumes, either tape or DASD, the logical copies are completed for all objects in the list, all objects are then placed in a status of UTRW, and the physical copy process for all of the objects occurs.

This copy technique ensures that the copy being made is a consistent copy.

9.4.4 DB2 Copy Using FILTERDDN for a List of Objects

If you want to copy a group of DB2 objects to the same consistency point minimize the outage to your applications during the copy process, use the DB2 COPY utility with the SHRLEVEL REFERENCE CONCURRENT FILTERDDN(xxxxxxxx) option and list of DB2 objects specified (Figure 38 on page 92).

```
//CCCOPY   EXEC PGM=DSNUTILB,REGION=1024K,
//             PARM='DB2B,DSNTEX'
//STEPLIB  DD DSN=DB2.LINKLIB,DISP=SHR
//*
//*****************************************************
//*  DATA SETS USED BY THE UTILITY
//*****************************************************
//FILTERDD DD DSN=&&TEMPFIL,DISP=(,DELETE,DELETE),
//            UNIT=SYSALLDA,SPACE=(CYL,(1,1),RLSE)
//SYSCOPY   DD DSN=CONCOPY1.GROUP00.OSMOTS32.SYSCOPY1,
//             DISP=(MOD,CATLG,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(200,25),RLSE)
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD *
  COPY TABLESPACE GROUP00.OSMOTS32
       TABLESPACE GROUP01.OSMOTS32
       TABLESPACE GROUP02.OSMOTS32
       TABLESPACE GROUP03.OSMOTS32
       TABLESPACE GROUP04.OSMOTS32
       TABLESPACE GROUP05.OSMOTS32
    FILTERDDN(FILTERDD)  COPYDDN(SYSCOPY)
    CONCURRENT SHRLEVEL REFERENCE
```

CONCURRENT Option with FILTERDDN

Figure 38. DB2 COPY Using the SHRLEVEL REFERENCE

Note

The order in which you specify the objects within your DB2 COPY command determines the order in which each object is logically copied. If any of the objects within the list of table spaces being copied needs to be made available for updating sooner than other objects in the list, specify it ahead of the other objects. If an object is first in the list, its logical copy will complete first, and it will be available for updating ahead of other objects in the list.

When a list of objects is copied with the DB2 COPY utility specifying the SHRLEVEL REFERENCE CONCURRENT and FILTERDDN keywords, the following actions occur:

1. All DB2 objects are placed in a status of UTRO, all writers are drained for all the objects, each object is quiesced and a row with an ICTYPE value of *Q* is inserted into SYSIBM.SYSCOPY for each object in the list.
2. DFSMSdss is invoked to logically copy each data set associated with the DB2 objects specified in the DB2 COPY control cards.
3. After each object has been logically copied, control is returned to DB2, so that the object can have its status changed to UTRW and be available for updating
4. After all the data sets have been logically dumped, DFSMSdss issues message ADR734I, indicating that the logical dump has completed

Note

Message ADR734I is issued only once after all data sets have been *logically* copied.

5. DFSMSdss begins physically dumping the data sets to the output data set.
6. Once the physical copy of all data sets has completed, control is returned to DB2.
7. The *Q* rows that had been inserted into SYSIBM.SYSCOPY for the objects that have been copied now have their ICTYPE values updated to an *F* and their SHRLEVEL values are changed to an *R*.
8. DB2 places all the objects back in RW status, and the utility ends.

This copy technique ensures that the copies being made are consistent copies and that inability to update the objects during the copy process has been minimized.

Note

If you choose to copy several objects within the same DB2 COPY CONCURRENT command with FILTERDD specified, all of the data is placed in a single output data set.

9.4.5 Backing up your DB2 Data Using DFSMSdss SnapShot COPY

You can also back up your DB2 data utilizing DFSMSdss SnapShot COPY, but the process involves more work on your part.

To obtain a consistent image copy by using DFSMSdss SnapShot COPY:

1. Start the DB2 objects being backed up for read-only access by issuing the following command:

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

   This command ensures that no updates to data occur during this procedure.
2. Run QUIESCE with the WRITE(YES) option to quiesce all DB2 objects being backed up. You must ensure that this step completes with RC=0. If the QUIESCE utility does not end with RC=0, there is something preventing the QUIESCE from completing successfully. A long-running update transaction that is not performing commits could cause the QUIESCE not to complete with RC=0, and a copy taken after this step may not be *consistent*.
   Figure 39 on page 94 shows how to use the QUIESCE utility for a list of DB2 objects.

```
//CCQUIESC EXEC PGM=DSNUTILB,REGION=1024K,
//             PARM='DB2B,DSNTEQ'
//STEPLIB  DD DSN=DB2.LINKLIB,DISP=SHR
//*
//*****************************************************
//*  DATA SETS USED BY THE UTILITY
//*****************************************************
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD *
  QUIESCE TABLESPACE GROUP00.OSMOTS32
          TABLESPACE GROUP01.OSMOTS32
          TABLESPACE GROUP02.OSMOTS32
          TABLESPACE GROUP03.OSMOTS32
          TABLESPACE GROUP04.OSMOTS32
          TABLESPACE GROUP05.OSMOTS32 WRITE(YES)
```

Figure 39. DB2 QUIESCE DB2 Objects with WRITE(YES) Option

3. Run your DFSMSdss SnapShot job to copy your data. When you copy to an output data set on the same RVA, you automatically get the time and space savings of DFSMSdss SnapShot. Figure 39 shows an example of a DFSMSdss SnapShot step to copy a list of DB2 data sets.

   Note

   You can copy table spaces and indexes together and you will obtain a consistent copy for all objects.

```
//CPYDB2   EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
COPY DATASET(INCLUDE(DB2B.DSNDBC.GROUP00.**)) -
     RENAMEU(DB2BCOPY) CATALOG -
     OUTDYNAM(WORK01) CONCURRENT
```

Figure 40. DB2 COPY Set of DB2 Objects with DFSMSdss

4. Once the backup has completed successfully, issue the following command to allow transactions to access the data:

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

   Note

   You can issue the start command after the ADR734I message indicating the data has been *logically* copied.

9.4.6 DB2 Recover Using Virtual Concurrent Copy Copies

When using the DB2 RECOVER utility to recover data backed up with the DB2 COPY utility with the CONCURRENT keyword specified, you can recover your data as if a DB2 image copy was being used. You can recover to current, to a specific copy, or to a specific RBA or logpoint. You can also recover your data by specifying the LOGONLY keyword if the object was restored outside DB2s control.

When you recover a list of objects that had been backed up at the same time by specifying the CONCURRENT and FILTERRDN keywords, the order in which the objects are listed for the RECOVER utility does not have any impact on the performance of the recover utility. However, if the list of objects that had been backed up at the same time by specifying the CONCURRENT keyword but not the FILTERRDN keyword and the output of the copies was to tape, then the order in which the objects are specified for the RECOVER utility affects the performance of the recover utility. To improve the performance of the recover utility when using copies that had been made to tape, refer to “Retaining Tape Mounts” in the *DB2 Utility Guide and Reference*.

The DB2 RECOVER utility example in Figure 41 shows a list of objects being recovered to current. The example assumes that the copies had been made specifying the FILTERDDN keyword, because it is in this way that the copy outage is minimized the most.

```
//CCRECVR  EXEC PGM=DSNUTILB,REGION=1024K,
//             PARM='DB2B,DSNTEX'
//STEPLIB  DD DSN=DB2.LINKLIB,DISP=SHR
//*
//*******************************************************
//*  DATA SETS USED BY THE UTILITY
//*******************************************************
//SYSCOPY   DD DSN=CONCOPY1.GROUP00.OSMOTS32.SYSCOPY1,
//             DISP=OLD
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD *
RECOVER TABLESPACE GROUP00.OSMOTS32
        TABLESPACE GROUP01.OSMOTS32
        TABLESPACE GROUP02.OSMOTS32
        TABLESPACE GROUP03.OSMOTS32
        TABLESPACE GROUP04.OSMOTS32
        TABLESPACE GROUP05.OSMOTS32
```

Figure 41. DB2 RECOVER to Current from FILTERDDN Copy

This same RECOVER job can also be used to recover the data to a specific point in time—whether it be TOCOPY, TORBA, or TOLOGPOINT. All you have to do to recover to a specific point in time is add the appropriate RECOVER keyword and value associated with it to the example in Figure 41. Figure 42 on page 96 shows an example of recovering a list of objects by specifying the TOCOPY keyword with the input for the RECOVER being a copy made using virtual concurrent copy.

```
//CCRECVR  EXEC PGM=DSNUTILB,REGION=1024K,
//             PARM='DB2B,DSNTEX'
//STEPLIB  DD DSN=DB2.LINKLIB,DISP=SHR
//*
//*****************************************************
//*  DATA SETS USED BY THE UTILITY
//*****************************************************
//SYSCOPY   DD DSN=CONCOPY1.GROUP00.OSMOTS32.SYSCOPY1,
//             DISP=OLD
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD *
RECOVER TABLESPACE GROUP00.OSMOTS32
        TOCOPY CONCOPY1.GROUP00.OSMOTS32.SYSCOPY1
        TABLESPACE GROUP01.OSMOTS32
        TOCOPY CONCOPY1.GROUP00.OSMOTS32.SYSCOPY1
        TABLESPACE GROUP02.OSMOTS32
        TOCOPY CONCOPY1.GROUP00.OSMOTS32.SYSCOPY1
        TABLESPACE GROUP03.OSMOTS32
        TOCOPY CONCOPY1.GROUP00.OSMOTS32.SYSCOPY1
        TABLESPACE GROUP04.OSMOTS32
        TOCOPY CONCOPY1.GROUP00.OSMOTS32.SYSCOPY1
        TABLESPACE GROUP05.OSMOTS32
        TOCOPY CONCOPY1.GROUP00.OSMOTS32.SYSCOPY1
```

Figure 42. DB2 RECOVER Specifying TOCOPY Keyword with FILTERDDN Copy

Note

The data set name specified with the TOCOPY keyword is the same value for each DB2 object being recovered because all objects were copied at the same time with the DB2 COPY utility by specfying the CONCURRENT and FILTERDDN keywords. When the DB2 COPY utility is used in this manner, only a single output data set is created and it contains the copy of every object specified in the list.

9.4.7 Recover DB2 Data Using DFSMSdss COPY Backups

If you backed up your DB2 data using DFSMSdss COPY commands, you cannot use the DB2 RECOVER utility to recover all of your data. You must first use DFSMSdss to restore your DB2 data to the point at which the backup was made. Then if it is necessary to recover your data to a more current point in time after the backup had been made, you can use the DB2 RECOVER utility, specifying the LOGONLY keyword to accomplish that. The recover process described below can be used for recovering your DB2 data even if the DB2 COPY utility was used to back up your data.

To recover your DB2 data from a DFSMSdss backup, follow these steps:

1. Ensure that no transactions can access the DB2 objects you are recovering until you have completed the recovery process.
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)
   ```

Note

Before you restore the DB2 data sets, if you believe it is necessary, you can at this time take a backup of the objects in order to restore your DB2 data to the state it was in before you began the recover process.

4. Restore all of the DB2 data sets being recovered. If you used DFSMSdss COPY to back up your DB2 objects to data sets in the same RVA, you can restore them, using another DFSMSdss COPY from your backup data sets, back to your original source data sets. This procedure will give you the time and space savings of SnapShot.

   Note

   If you used DFSMSdss to back up your indexes at the same time you backed up your table spaces, restore your indexes at this time also. If your plan is to recover your data to the point in time at which the backups had been made and no log data is to be applied to the DB2 objects, you will be able to skip the step in which the indexes are recovered.

   Figure 43 shows an example of recovering a list of objects to current by specifying the LOGONLY and TORBA keywords.

```
//CCRECVR  EXEC PGM=DSNUTILB,REGION=1024K,
//             PARM='DB2B,DSNTEX'
//STEPLIB  DD DSN=DB2.LINKLIB,DISP=SHR
//*
//*******************************************************
//*  DATA SETS USED BY THE UTILITY
//*******************************************************
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD *
RECOVER TABLESPACE GROUP00.OSMOTS32
        TABLESPACE GROUP01.OSMOTS32
        TABLESPACE GROUP02.OSMOTS32
        TABLESPACE GROUP03.OSMOTS32
        TABLESPACE GROUP04.OSMOTS32
        TABLESPACE GROUP05.OSMOTS32
        LOGONLY TORBA (X'000007425468')
```

Figure 43. Using LOGONLY and TORBA to Recover a List of DB2 Objects

5. If you are recovering your DB2 objects only to the point in time at which the backups had been taken, you can skip this step. However, if you need to recover the DB2 objects to a point in time after the backups had been made, you must run the DB2 RECOVER utility specifying the LOGONLY keyword. When the LOGONLY keyword is specified, the target objects will be recovered from their existing data sets by applying only log records to the target objects. The DB2 RECOVER utility will apply all log records that were written after a point that is recorded in the data set itself.

   Figure 44 on page 98 shows an example of recovering a list of objects to current specifying the LOGONLY keyword.

```
//CCRECVR  EXEC PGM=DSNUTILB,REGION=1024K,
//             PARM='DB2B,DSNTEX'
//STEPLIB  DD DSN=DB2.LINKLIB,DISP=SHR
//*
//*******************************************************
//*  DATA SETS USED BY THE UTILITY
//*******************************************************
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD *
RECOVER TABLESPACE GROUP00.OSMOTS32
        TABLESPACE GROUP01.OSMOTS32
        TABLESPACE GROUP02.OSMOTS32
        TABLESPACE GROUP03.OSMOTS32
        TABLESPACE GROUP04.OSMOTS32
        TABLESPACE GROUP05.OSMOTS32
        LOGONLY
```

Figure 44. Using LOGONLY to Recover a List of DB2 Objects

Figure 45 shows an example of recovering a list of objects to current specifying the LOGONLY and TORBA keywords.

```
//CCRECVR  EXEC PGM=DSNUTILB,REGION=1024K,
//             PARM='DB2B,DSNTEX'
//STEPLIB  DD DSN=DB2.LINKLIB,DISP=SHR
//*
//*******************************************************
//*  DATA SETS USED BY THE UTILITY
//*******************************************************
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD *
RECOVER TABLESPACE GROUP00.OSMOTS32
        TABLESPACE GROUP01.OSMOTS32
        TABLESPACE GROUP02.OSMOTS32
        TABLESPACE GROUP03.OSMOTS32
        TABLESPACE GROUP04.OSMOTS32
        TABLESPACE GROUP05.OSMOTS32
        LOGONLY TORBA (X'000007425468')
```

Figure 45. Using LOGONLY and TORBA to Recover a List of DB2 Objects

6. If you recovered your DB2 objects to current, there is no need to recover the indexes for the objects, so you can skip this step. But if you recovered your DB2 objects to a previous point in time, recover all indexes on the recovered objects. Figure 46 on page 99 shows an example of recovering all indexes for a single DB2 table space.

```
//CCRECVR  EXEC PGM=DSNUTILB,REGION=1024K,
//             PARM='DB2B,DSNTEX'
//STEPLIB  DD DSN=DB2.LINKLIB,DISP=SHR
//*
//*****************************************************
//*  DATA SETS USED BY THE UTILITY
//*****************************************************
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(CYL,(50,30))
//SYSIN    DD *
RECOVER INDEX (ALL) TABLESPACE GROUP00.OSMOTS32
```

Figure 46. Recovering Table Space Indexes

Note

If you are recovering indexes for multiple DB2 table spaces, we recommend recovering the indexes for each table space in a separate job. With this approach the recovers run in parallel, and the length of time required to recover all the indexes is minimized.

7. Once all objects have been recovered successfully, issue the following command to allow access to the objects:

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

9.5 Application Testing

DFSMSdss Snapshot or virtual concurrent copy can be used to help make multiple copies of your data, giving you more flexibility when testing new applications or program modifications. If your testing process involves restoring your DB2 to a previous point in time with either the DB2 LOAD utility or restoring a previous copy of your data, DFSMSdss Snapshot or virtual concurrent copy can help reduce the time required to restore your data.

Note

The application testing processes referred to here can be used to perform repeated Year 2000 testing.

You can use the DB2 COPY utility with virtual concurrent copy to back up your data before beginning your testing and then use the DB2 RECOVER utility to restore your data.

However, a much faster way to perform application testing is to use DFSMSdss COPY to copy your data before beginning testing and then use DFSMSdss RESTORE to restore your data. If your source and target data sets are in the same RVA, using DFSMSdss COPY will result in a DFSMSdss SnapShot, with the benefits of SnapShot's speed and space saving.

9.5.1 Using DFSMSdss with Application Test Data

To use DFSMSdss COPY and RESTORE to rebuild your test data very quickly follow this procedure:

1. Choose the source data sets to be backed up. If you will be restoring your data to the point in time at which the copy is made, then also choose which indexes you want to back up. This will enable you to be able to skip the step to RECOVER your indexes, thus saving even more time in the rebuilding of your test data. Before backing up your data using DFSMSdss for the very first time, you must load the data into the DB2 objects you will be backing up.
2. Establish the consistency point to which you intend to restore your test environment. Ensure that all of your DB2 test data sets, including all indexes, are at this point, before you back up your data.
3. Back up the DB2 data sets, using the procedure given in 9.4.5, “Backing up your DB2 Data Using DFSMSdss SnapShot COPY” on page 93.
4. Perform your testing.
5. When you need to reset your test environment with the backups taken in step 3, use the procedure given in 9.4.7, “Recover DB2 Data Using DFSMSdss COPY Backups” on page 96.
6. Repeat steps 4 and 5 as many times as necessary to complete your testing.

9.5.2 Application Testing Considerations

When using DFSMSdss to rebuild your DB2 application test data, you must consider the following points:

- If you change the structure of any of your DB2 tables, you must take a new backup of your DB2 data sets. Also, you must perform this backup after the structure change has been made and data has been loaded into the DB2 data sets.
- If you decide to change the point in time to which you intend to restore your test DB2 environment to, you must also take new backups of your data. When taking these backups, always back up all indexes so that the minimal amount of time is required to reset your test DB2 environment.

9.6 Cloning DB2 Data Using DFSMSdss SnapShot

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 recommended that you copy all of the data you are cloning to the same consistent point, however in a test environment, complete data integrity may not present a problem.

9.6.1 Using DFSMSdss SnapShot with Application Test Data

Using DFSMSdss SnapShot to extract DB2 test data from one DB2 subsystem for use in another can be a challenging task, though not impossible using the proper procedures. Because DB2 tracks every object by the internal identifiers assigned to every object within a DB2 subsystem, it is highly likely that you cannot just use DB2 data from one subsystem in another without some additional processes.

The most common method use to be able to access DB2 data created in one subsystem in another is the DB2 utility DSN1COPY. For detailed instructions on how to use DSN1COPY to determine how to translate the internal identifiers from one subsystem to another and then perform the translation, refer to *DB2 for OS/390 Version 5 Utility Guide and Reference*, SC26-8967.

Note

If the copy of your data you plan to copy into another DB2 subsystem was made with the DB2 COPY utility specifying the CONCURRENT keyword you will have one additional step before you are able to run the DSN1COPY utility to translate the data. Because this output is in DFSMSdss DUMP format, you will first need to manually RESTORE the DB2 objects using DFSMSdss into a VSAM data sets before running the DSN1COPY utility to translate the internal identifiers from one DB2 subsystem to another.

9.6.2 Copying a DB2 Subsystem Using DFSMSdss SnapShot

An alternative to translating the internal identifiers is to create a completely cloned DB2 system which will then require no translation. You can do this by using SnapShot to copy the entire DB2 system including:

- The catalog and directory data sets.
- Both boot strap data sets (BSDS).
- All active log data sets.
- All application data sets.
- Your DB2 load libraries and application load libraries.

If you copied your DB2 environment with SnapShot after the original DB2 subsytem had been shutdown normally, you do not have to copy any of the DB2 archive log data sets.

Figure 47 shows an example of a DFSMSdss job to copy all datasets with a specified high-level qualifier to another high-level qualifier.

```
//CPYDB2   EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 COPY DATASET(INCLUDE(DB2P.**)) -
   RENAMEU(DB2X) CATALOG -
      OUTDYNAM(WORK01) CONCURRENT
```

Figure 47. COPY all DB2 Data Dets Using DFSMSdss

Using this copy process, you can copy all your DB2 data to a different high-level qualifier. Both copies of the data are on the same MVS subsystem on which your DB2 data currently resides. Move the new copies of the data to the MVS subsystem on which you intend to run the cloned DB2 and run the DFSMSdss COPY step shown in Figure 48 on page 102 step to copy all datasets back to their original high-level qualifier.

```
//CPYDB2   EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 COPY DATASET(INCLUDE(DB2X.**)) -
   RENAMEU(DB2P) CATALOG -
      OUTDYNAM(WORKO2) CONCURRENT
```

Figure 48. COPY all DB2 Data Sets Using DFSMSdss

In the example above, you may now delete the datasets created with the high-level qualifier of DB2X. You now have an exact duplicate of the original DB2P subsystem that you can start on a different MVS subsystem. This process allows you to run two entirely separate DB2 subsystems on two different MVS subsystems using the same data. The only increase you will see in NCL usage is from the amount of changes that occur on the data.

This cloned DB2 has the same name as the original so it must run in a different MVS environment. You have to ensure that there is no confusion between the data belonging to the original and cloned systems. Maintaining this separation is operationally complex.

Note

If you want to use this process to test your applications on a different MVS subsystem, it is not necessary to copy all of your DB2 data. You do not have to copy any of the application data you do not intend to use in your new DB2 subsystem.

If you need to run the cloned DB2 in the same MVS environment as the original, you will need to rename the cloned system. Internally it will still have the same identifiers, but externally the data sets all have different names. This means the you also have to rename the VCAT, and make some changes so that all the new data set names are known to DB2.

A procedure to rename a DB2 system is documented in the redbook *DB2 for MVS/ESA Version 4 Data Sharing Implementation*, SG24-4791.

9.7 Virtual Concurrent Copy Considerations

If you use the DB2 COPY utility with the CONCURRENT keyword:

- You must supply either a COPYDDN ddname, a RECOVERYDDN ddname, or both.

  Do not specify DASD and tape output data sets in the same COPY utility step. If you mix output types the COPY utility will terminate with the following DFSMSdss error:

  ```
  OADR374E (OO2)-OPNCL(O8), UNABLE TO OPEN DDNAME COPY1, 10
  ```

- If the SYSPRINT DD card points to a data set, you must use a DSSPRINT DD card.
- After a LOAD or REORG LOG(NO), the status of a table space is reset to a copy pending state. With virtual concurrent copy, no write access to the table space is allowed until the copy *physically* completes.

- Check that you have sufficient DASD to support the online virtual concurrent copy images.
- You can specify a list of table spaces to copy.
- You must use the SHRLEVEL REFERENCE option for table spaces with a 32 KB page size.
- You cannot use a copy made with virtual concurrent copy with the PAGE or ERRORRANGE options. If you specify PAGE or ERRORRANGE, RECOVER bypasses any concurrent copy records when searching the SYSCOPY table for a recoverable point.
- The batch ID that invokes COPY with the CONCURRENT option must provide the necessary authority to invoke the DFSMSdss DUMP command.
- You cannot run the following DB2 utilities or stand-alone utilities on copies made by virtual concurrent copy:
  - DSN1COMP
  - DSN1COPY
  - DSN1PRNT
  - MERGECOPY
- When using virtual concurrent copy to make copies of the DB2 catalog and directory for disaster recovery, you must copy each of the following table spaces as a single object:
  - DSNDB01.SYSLGRNX
  - DSNDB06.SYSCOPY
  - DSNDB01.SYSUTILX
- You cannot invoke the CONCURRENT option from the DB2I utilities panel or from the DSNU TSO CLIST.
- The COPY utility will fail if one or more of the table spaces do not reside on a concurrent copy capable device.
- Incremental image copies are disallowed if the previous copy is a virtual concurrent copy.

Chapter 10. Using Virtual Concurrent Copy for IMS

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. Details of using SnapShot with IMS/ESA before Version 6 are documented in *Implementing SnapShot,* SG24-2241.

10.1 IMS/ESA Image Copy 2

In this section we discuss how to use concurrent copy with the new Image Copy 2 utility, DFSUDMT0, introduced in IMS/ESA Version 6.

Before 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.

10.2 Using Concurrent Copy with IMS/ESA Version 6

When concurrent copy is used with IMS/ESA Version 6, the data is unavailable only during the logical copy, or session initialization. Here is the process for initiating a concurrent copy:

1. Determine a suitable time to start a backup of the databases and, if a static copy is required, stop activity to those databases.
2. Submit an image copy job invoking Image Copy 2. GENJCL has been extended to support concurrent copy for backup and recovery. The %SMS

© Copyright IBM Corp. 1998

keyword in skeletal JCL for GENJCL.IC and RECOV is used to control DCB=BUFNO-nn generation for input and output data sets because it is not supported by virtual concurrent copy.

3. Image Copy 2 requests authorization from DBRC and, if granted, invokes DFSMSdss to perform a dump of the data by using concurrent copy.
4. DFSMSdss locates the data set and interfaces with the SDM to initialize a concurrent copy session.
5. The SDM determines the storage control to which the volume is attached, and whether it is a concurrent-copy-capable or virtual-concurrent-copy-capable device.
6. DFSMSdss performs the concurrent copy and, once the logical copy is complete, releases the serialization.
7. Once the copy is logically complete, DFSMSdss informs Image Copy 2 of the status, using a UIM. Image Copy 2 then releases the database authorization.
8. As DFSMSdss is performing the physical copy, the applications can resume processing of the data being copied.

10.2.1 Supported Data Sets

Image Copy 2 can invoke DFSMSdss to dump KSDSs, ESDSs, and OSAM data sets. Multivolume VSAM data sets are supported, but multivolume OSAM data sets are not.

VSAM data sets to be copied and restored by DFSMSdss must satisfy one of the following conditions:

- Be SMS managed
- Be cataloged in the master catalog
- Be cataloged using an alias (The HLQ of the VSAM data set is an alias for the catalog.)

10.2.2 Image Copy 2 Options

Image Copy 2 can be used to create image copies of a single database data set or a group of databases that are logically related. Each database may comprise multiple data sets. IMS GENJCL generates a single DFSMSdss job for the entire group of data sets. Once generated, this JCL can be directed to a data set for editing.

When you back up a group of databases, all of the related data sets are copied by DFSMSdss to a single DFSMSdss dump data set. To recover databases in parallel, you must restore the archive logs from tape to DASD. Once they are on DASD, and IMS has restored the DFSMSdss dump data set, the databases can be recovered in parallel (see Figure 49 on page 107).

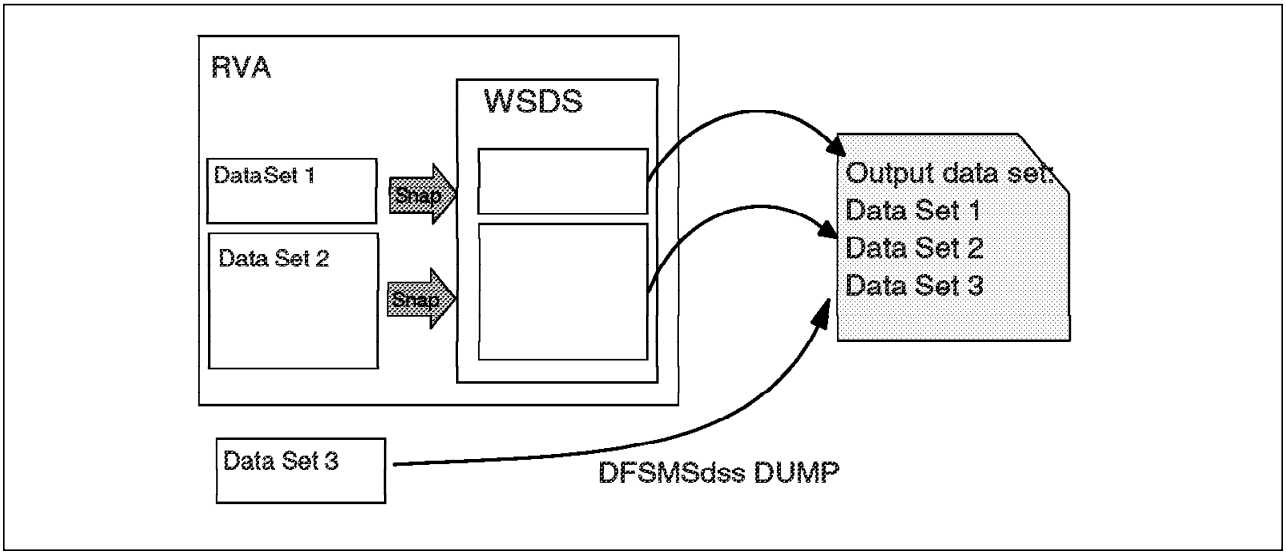


Figure 49. IMS Image Copy 2 of Data Sets 1, 2, and 3

The archive logs are restored to disk through DFSMSdss. Once the log data sets are on disk, you need to NOTIFY or CHANGE the log data set information to RECON to ensure that recovery accesses the correct logs. If you use DFSMShsm to migrate your archive logs the archive log data sets are all cataloged and this is recorded in the RECON data set, so no changes to RECON are required.

When backing up a group of databases, IMS can recognize logical copy completion and release serialization on each database once it is snapped. When IMS backs up a group of databases that reside on both concurrent-copy and non-concurrent-copy-capable devices, serialization will be released as illustrated in Figure 50.

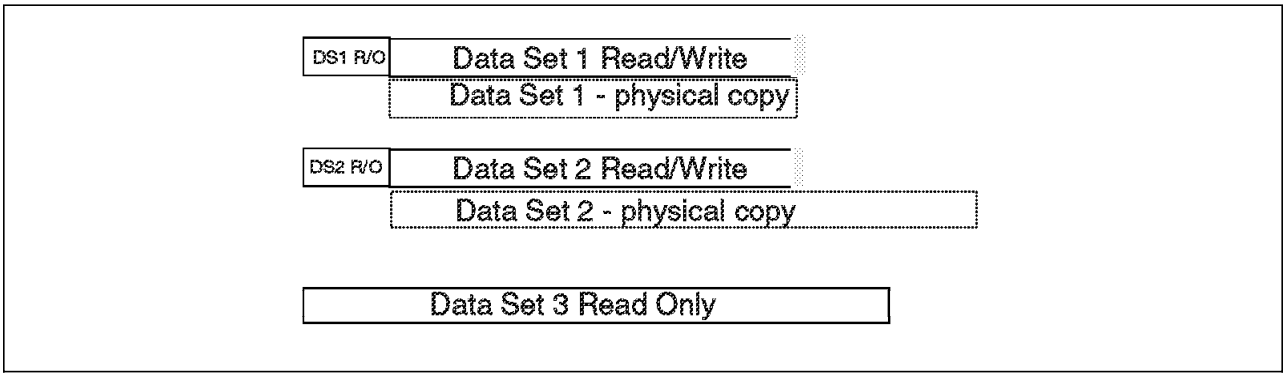


Figure 50. IMS Serialization

Image Copy 2 invokes DFSMSdss in cross-memory mode, using the DFSMSdss API. The Image Copy 2 control statement specifies how many copies are required, whether static or fuzzy copies are required, and, if static, when the database can be released for application use.

10.2.2.1 Copy Count

Image Copy 2 can create up to four copies, two of which can be registered with DBRC. Other copies may be required for shipment offsite where they may be used for disaster recovery purposes.

10.2.2.2 Access Value

The Access value parameter of the Image Copy 2 control statement specifies whether a static or fuzzy copy is required:

- **S** indicates to DBRC that concurrent updates can share access to data with the utility and that a fuzzy copy will be created. The copy is registered with DBRC as SMSCIC. If a fuzzy copy is to be made, the VSAM cluster for a KSDS must be defined as eligible for BWO by IMS. This is specified with the BWO(TYPEIMS) parameter.

  Concurrent copy will be successful only when CI or CA splits are not in process for a KSDS. Image Copy 2 will make up to 10 attempts to initialize; if database update activity prevents the session initialization, it will terminate with a DFS3145A message.

  When you take a fuzzy image copy, you need to have the appropriate logs available for recovery to restore the database to a state of consistency.

- An **X** value indicates that a static image copy is required. This will be registered with DBRC as SMSNOCIC. You must deny all users update access to the database before starting Image Copy 2, by using the/DBRECOVERY database or area command.

10.2.2.3 Authorization Release

When a static image copy is requested with an access value of X, Image Copy 2 uses the database authorization release point to determine when the read-only authorization can be released and the database be authorized for updates:

- **Option L** releases authorization when the logical copy is complete.
- **Option P** releases authorization when the physical copy is complete.

The authorization release parameter enables an installation to be sure that a static copy has been successfully taken. When update access is prevented until the physical copy has completed, a successful static image copy is ensured because you can overcome any failure in the physical copy process by rerunning the copy, knowing that the original database data set has not been updated.

If updates have been permitted after the logical copy is complete, but before the physical copy is complete, a rerun copy will be of a data set that has been updated and may therefore be inconsistent with other data sets in the group.

If you use Option P you ensure that a static copy is taken at a specific point in time. However, database unavailability is extended to the length of the physical copy, which is approximately the same as if you had used the IMS batch image copy utility, so Option P eliminates the principal benefit of concurrent copy.

10.2.2.4 DFSMSdss Options

Image Copy 2 invokes DFSMSdss with several options:

- DUMP - Makes efficient use of the output volume space
- CONCURRENT - Invokes concurrent copy to allow copies to be made of data sets that are open for update

- VALIDATE for KSDS - The index component is validated against the data component during the process. The index component is not dumped but is created as part of the DFSMSdss RESTORE.
- WAIT(0,0) - Prevents waiting if the volume or data set is enqueued by another process.
- OPTIMIZE - Specifies the pacing parameters to DFSMSdss. The OPTIMIZE value defaults to OPT(4) when the database is backed up while in read-only mode, and OPT(1) when concurrent updaters are permitted.

DFSMSdss OPTIMIZE

OPT (4) and OPT (1) are not optimal for RVA concurrent copy because there is no contention on the source data set. Although the job will run fine with these defaults, the DFSMSdss dump job is more efficient at OPT(3) for RVA concurrent copy in any workload environment. It is possible to override these defaults by editing the JCL generated by the GENJCL utility.

10.2.3 Recovery Process

Database recovery is best performed through the IMS utilities, which attempt to preserve data integrity and interface with DBRC.

The recovery utility (DFSURDB0) uses the image copy type that is recorded in DBRC to determine how recovery processing will proceed.

DBRC must be active for the utility to use a copy made with Image Copy 2, whether concurrent copy is used or not. The database is registered by DBRC in the RECON data set. You can use the GENJCL.RECOV command to generate the necessary JCL for the database recovery utility.

The recovery utility invokes DFSMSdss to restore the copy to DASD and continues the recovery process with the subsequent logs or change accumulation data sets.

Chapter 11. Using SnapShot with SAP R/3 on DB2 for S/390

SAP R/3 is a generalized transaction processing system based on a client/server model of transaction handling. The implementation can be in a two or three-tier configuration where the SAP R/3 database server provides database functions to one or more SAP R/3 application servers.

The SAP R/3 database server provides accessibility to the SAP R/3 database. The database contains all of the user data that applications require. In addition, the database contains some application programs. It also contains SAP R/3 configuration, management, and operational tables.

It is possible for the SAP R/3 database server to use one of several DBMS products. In this chapter we discuss only DB2 on the S/390 platform.

11.1 DB2 Features

DB2 for OS/390 is the cornerstone providing SAP R/3 database server function. Several features of DB2 are particularly beneficial in a SAP R/3 installation:

- Continuous operation and high availability

  DB2 can operate for long periods without interruption. With data sharing, work can be transferred between DB2 subsystems within a Parallel Sysplex as a result of planned or unplanned outage. Online reorganization provides greater availability during database unload and reload processing.

- Data sharing

  DB2 exploits the capability of the S/390 Parallel Sysplex through data sharing. With data sharing, different applications from different systems can read from and write to the same set of data concurrently. If a large application can be divided into several components, it is also possible for each component to run on a separate DB2 subsystem and share the single copy of DB2 data with the other components.

  Data sharing makes it possible to use multiple instances of DB2 running on multiple processors, so it is particularly important in installations that have a high availability requirement. Either DB2 instance can provide SAP R/3 service if the other instance fails or is unavailable.

  Additionally, data sharing allows more flexibility in scaling the available processing power for a particular application: additional members can be added without disruption to the existing DB2 members.

- Data integrity

  DB2 provides high data integrity through capabilities such as a sophisticated lock manager and integration with IBM system security products. Support for referential integrity is provided, although not yet implemented in SAP R/3. DB2 also protects data from subsystem, media, and application failures with integrated recovery procedures.

- Very large database support

  Very large databases require rapid processes running in parallel for backup, reorganization, and recovery of data. The large number of independent I/O processors and channels in an S/390 system for DASD and tape operations, as well as the characteristics of DB2 for OS/390, allow parallel processes,

© Copyright IBM Corp. 1998

and are key to meeting the requirements for maintaining the performance of very large databases.

- Database and system administration aids

  To help database administrators manage their database environment, DB2 offers an integrated set of tools and functions. These include flexible security mechanisms, an extensive set of logging and recovery utilities, trace facilities, and functions and tools for monitoring and tuning the subsystems.

11.2 Transaction Concept in SAP R/3

The SAP R/3 architecture separates the various functional elements and allows for multiple independent application or database servers. The application programs requested by the users are executed on an application server host. The interaction between the user and the host is a series of dialog steps, each of which can be executed by the application server *logical dialog service*. All updates to the database are processed by the application server *logical update service*; database updates are asynchronous.

Each *dialog step* of a business transaction in SAP R/3 can be processed on a different work process in the application server host and therefore use a different thread to communicate with DB2. All database changes made in a dialog step are committed at the end of the dialog step. This is acceptable to some applications, but most business transactions consist of multiple dialog steps, which make up a SAP R/3 *logical unit of work* (LUW). In other words the SAP R/3 LUW is expected to consist of the complete business transaction, whereas DB2 treats each dialog step as a discrete unit of work.

After a failure, DB2 recovers the database to a consistent state, rolling back those units of work that were in flight at the time of the failure, and committing the units of work that were completed. This database state, even though consistent from the DB2 point of view, may have incomplete SAP LUWs. The application server must be started to back out changes for SAP R/3 business transactions that were not completed.

11.3 Data Recovery in a SAP R/3 Environment

Correctly recovering your SAP R/3 database is critical after any sort of interruption, either planned or unplanned. The recovery can be to the current state or to a point in time.

Often the recovery is automatic: In the event of a DB2 or OS/390 failure, DB2 recovers the database to a consistent state when it restarts. *Consistent state* means that all database changes made by incomplete units of work are backed out (sometimes called *rolled back*), and the changes made to complete units of work are committed. In the event of a media failure, such as a DASD volume, the database administrator must recover the database tablespaces on that volume, using the RECOVER utility.

Recovery to a point in time always involves the database administrator.

11.4 Recovery to Current

On a DB2 normal restart, DB2 processes the current logs and ensures that all tablespaces are in a consistent state.

In the case where you have had a media failure, and some tablespaces are not available, the RECOVER utility is required. The way in which the DB2 utility recovers the missing data depends on the backup policy that the Database Administrator(DBA) has established in the organization:

- If a full image copy is available for the tablespace, the recovery utility restores the image copy and reapplies the changes recorded in the log data. If the image copy was recorded with SHRLEVEL(REFERENCE) specified, no one was allowed to update the tablespace while the utility was running, so log data from the end of the execution of the image copy will be applied. If the image copy was run with SHRLEVEL(CHANGE) specified, the image copy is a ″fuzzy″ image copy because updates could be applied to the database while the copy was running. In this case, the log data from the time of the beginning of the execution of the image copy will be applied.
- If incremental image copies are available, they can be used to reduce the amount of log data that is required to be processed after the full image copy. Incremental image copies have FULL(NO) specified and only record pages in the tablespace that have been changed since the last image copy.
- Another option is the use of full volume restores, which are outside the control of DB2. If you have a full volume backup taken when the database is known to be consistent, you can restore that volume and recover the tablespaces without using an image copy.

  This full volume backup can be taken with SnapShot. After the restore, the RECOVER utility with LOGONLY specified examines the tablespace and applies any logged changes made since the volume was dumped.

After tuning the tablespace recovery, you must recover all of the indexes on the tablespaces.

Usually a recovery to current is performed after some hardware failure, and the recover utility is required for a few tablespaces, not the whole DB2 environment. If the recovery is for every tablespace and every index, the recovery to current requires a significant amount of time, to the point where some other technique, such as recovering to a previous full system backup, may be much faster, even though you will have lost work performed by your users since the backup was taken.

11.4.1 Tablespace or Index Recovery

In contrast to point-in-time recovery, as discussed in 11.5, “Point-in-Time Recovery” on page 114, *recovery to current* means recovery of a tablespace or index through the last completed and committed unit of recovery. It is assumed that when an object requires recovery, only that object is recovered.

The concept behind the recovery is simple. Recovery is usually at a data set level (tablespaces, partitions, and indexes are stored in VSAM data sets). The data is backed up at the data set level with the DB2 COPY utility. Recovery of a tablespace or partition requires the DB2 RECOVER utility to restore the image copy and then forward apply the DB2 log. The recovery of an index is effected through a rebuild where the table data is read and the index reconstructed.

11.4.2 Using Concurrent Copy with DB2

Concurrent copy is supported by DB2 Version 4 or later as an option of the COPY utility. It is invoked by using the CONCURRENT keyword. Copies made using COPY with the CONCURRENT keyword are recorded in the DB2 catalog table, SYSIBM.SYSCOPY. Concurrent copy can be used for *full* image copies whether these are taken using SHRLEVEL(REFERENCE) or SHRLEVEL(CHANGE):

- SHRLEVEL(REFERENCE) enables you to take a copy while access to the tablespace is restricted to read-only. Using concurrent copy reduces the time during which the database must be kept read-only and improves the availability for users.
- SHRLEVEL(CHANGE) enables you to take a fuzzy copy while users are updating the tablespace. To recover it is necessary to apply the updates that were made during the period of the copy. Use of the concurrent copy option reduces the number of updates that must be applied.

For full details on how to use concurrent copy with DB2, refer to Chapter 9, “Using DFSMSdss SnapShot and Virtual Concurrent Copy with DB2” on page 81.

11.5 Point-in-Time Recovery

A point-in-time recovery restores the state of the database to a previous time. This recovery is often done because the database has some logical damage—perhaps a logic error in an application has caused updates to the database that are difficult to reverse. The SAP R/3 system is integrated with data in the database from all areas of the business. A logic error in a program can cause related errors to the database to spread rapidly through the database. Often restoring everything to a time before the batch job was run is the only way to get back to a known correct state.

It is important to remember that the SAP R/3 database must be considered as a single unit of recovery. All tablespaces, indexes, and DB2 catalog and directory entries must be recovered to the same point in time if the SAP R/3 system is to function correctly. When performing a point-in-time recovery, you must always recover the indexes after the tablespaces have been recovered. These recoveries can take a considerable length of time, so we recommend a point-in-time backup technique using SnapShot volume snaps.

11.6 SAP R/3 Point-in-Time Recovery Using SnapShot

Point-in-time recovery involves setting a table or group of tables to a previous point in time when the data was determined to be consistent. The challenge in a SAP R/3 environment is to determine the set of tables that are logically related. It is not likely that you will be able to determine a subset of the SAP R/3 tables to be reset. It is likely that you will be required to reset *all* SAP R/3 tables to a previous point of consistency.

The issue regarding point-in-time backup is the number of SAP R/3 tables and indexes that must be recovered to a preestablished point of consistency. In a SAP R/3 system there could be more than 10,825 tables and indexes that have to be recovered. Using SnapShot for volume-based copies is clearly faster than a recovery-based scenario.

The major disadvantage of using SnapShot for point-in-time backup is that the dumping of the data is disruptive to the user. The SAP R/3 must be stopped to ensure that the data is consistent; however, this is necessary for any point-in-time backup and recovery technique. An additional disadvantage inherent in any point-in-time recovery scenario is that work is lost when data is reset to a previous point in time.

To recover DB2 to a known point in time you can use a previously established quiesce point (established through the quiesce utility, the ARCHIVE LOG command, or the STOP DB2 command), or you can recover to a specific log relative byte address (RBA) or log record sequence number (LRSN). If you restrict the points in time to which you will recover to those preplanned points where you can perform full volume backups, you can recover much faster. The fastest way to perform a recovery to a preplanned point in time is to use SnapShot to perform the volume backups and recovery.

11.6.1 Backup with SnapShot

If the SAP R/3 database and the DB2 system volumes are stored on an RVA, you can use SnapShot. For each volume used by the database, you must have a second volume defined on the RVA to be used as the backup for that volume. Follow these steps:

1. At the time chosen to take the backups, force the database to read-only mode, using the commands shown in Figure 51.

```
START DATABASE(*) ACCESS(RO)
START DATABASE(DSNDB06) ACCESS(RO)
START DATABASE(DSNDB01) ACCESS(RO)
```

Figure 51. Command to Force the SAP R/3 Databases to Read-Only

2. Quiesce the databases, using control statements similar to those in Figure 52. Quiescing forces any database changes still in the buffer pool to be written to DASD.

```
QUIESCE TABLESPACE DBASE1.TSPACE1
        TABLESPACE DBASE2.TSPACE1
  ⋮
        TABLESPACE DBASEn.TSPACEn
```

Figure 52. Command to Quiesce the SAP R/3 Tablespaces

3. Run the SnapShot utility, using control statements similar to those in Figure 53 on page 116.

```
SNAP VOLUME( -
      SOURCE(VOL(DB2001)) -
      TARGET(VOL(BAK001)) -
      REPLACE(YES) -
      COPYVOLID(NO) -
      CONDVOL(LBL) -
          )
  ⋮
SNAP VOLUME( -
      SOURCE(VOL(DB2nnn)) -
      TARGET(VOL(BAKnnn)) -
      REPLACE(YES) -
      COPYVOLID(NO) -
      CONDVOL(LBL) -
          )
```

Figure 53. Control Statements to Take SnapShot Backup of SAP R/3 Volumes

SnapShot makes a copy of each volume onto the corresponding backup volume. The snap of the volume takes only a few seconds.

4. Return the databases to read-write mode, using the commands in Figure 54.

```
START DATABASE(DSNDB01) ACCESS(RW)
START DATABASE(DSNDB06) ACCESS(RW)
START DATABASE(*) ACCESS(RW)
```

Figure 54. Command to Force the SAP R/3 Databases Back to Read-Write

5. A logical copy of the whole system has now been made, and you can recover to this point in time. You can start a DFSMSdss volume dump to copy these DASD backup volumes to tape. This may take some time, but the creation of the tapes can continue while the SAP R/3 system is running.

If you can afford a slightly longer outage, you can stop DB2 with the STOP DB2 MODE(QUIESCE) command instead of performing steps 1 and 2.

11.6.2 Recovery with SnapShot

To recover to a point in time, you use a similar technique. If the recover is to the time of the latest backup, the data on the backup volumes is still valid, and there is no requirement to restore from tape. If you want to recover to a previous backup, or you deleted the backup DASD volumes to save space, you first have to restore the backup DASD volumes from tape.

With DB2 down, snap the backup volumes back onto the production volumes, using the commands shown in Figure 55 on page 117.

```
SNAP VOLUME( -
      SOURCE(VOL(BAK001)) -
      TARGET(VOL(DB2001)) -
      REPLACE(YES) -
      COPYVOLID(NO) -
      CONDVOL(LBL) -
          )
  ⋮
SNAP VOLUME( -
      SOURCE(VOL(BAKnnn)) -
      TARGET(VOL(DB2nnn)) -
      REPLACE(YES) -
      COPYVOLID(NO) -
      CONDVOL(LBL) -
          )
```

Figure 55. Control Statements to Restore Backup of SAP R/3 Volumes

This snap takes only seconds per volume, and your whole DB2 environment is returned to the time of the backup.

This technique assumes that all of the DASD for the SAP R/3 database server is on RVA. It has the advantage of recovering the complete DB2 environment, including the DB2 catalog, to a single point of consistency. Because it does not require any recovery of indexes, this technique is significantly faster than the alternatives.

Chapter 12. Using DFSMSdss SnapShot in a VM/ESA Environment

From a VM/ESA point of view, DFSMSdss SnapShot introduces only a few, but important, modifications. DFSMSdss now supports VM minivolumes for concurrent copy DUMP and COPY, if you are using RVA devices to the extent that they are supported by IXFP device reporting.

The minivolume in an MVS system running on VM/370 is an OS/VS-formatted VM/370 minidisk whose size is equal to or less than that of the real volume. DFSMSdss uses the device size specified in the VTOC. Minivolumes are supported only by the system version of DFSMSdss.

You cannot use concurrent copy on minivolumes of any format unless they were on an RVA.

12.1 Dumping VM-Format Volumes

You can use DFSMSdss to dump VM-format volumes that are accessible to your MVS system. The volumes must have OS-compatible VTOCs starting on track 0, record 5. DFSMSdss can only retrieve device information from the OS-compatible VTOC; it cannot interpret any VM-specific information on the volume.

Exercise caution when using DFSMSdss to copy VM-format volumes because DFSMSdss does not serialize any VM data in any way, so it is your responsibility ensure data integrity in a VM environment.

You cannot copy VM-format volumes to OS-format volumes or OS-format volumes to OS-format volumes.

12.2 SnapShot for VM/ESA

All the other SnapShot characteristics in a VM/ESA environment are still the same, as they were before the introduction of the DFSMSdss SnapShot support. In this section we briefly summarize the most important characteristics.

SnapShot for VM/ESA can be used for:

- Quick copying of volumes and minidisks
- Some traditional DASD dump restore functions
- Quick formatting of minidisk space
- Quick formatting of temporary disk space
- Quick setup of new users
- VM backups
- Database backups

SnapShot for VM/ESA works on the volume and at minidisk level. So, authorized users can copy single volumes and general users can copy 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.

© Copyright IBM Corp. 1998

SnapShot for VM/ESA can also be used for VM backups, database backups, and replicating test data for database testing and Year 2000 testing.

For complete details about using SnapShot in a VM environment, see Chapter 9, “Using SnapShot in a VM/ESA Environment” in *Implementing SnapShot*.

Appendix A. Overview of Ramac Virtual Array and SnapShot

In this appendix we provide an overview of RVA subsystem and SnapShot functions. For a more detailed description, see *IBM RAMAC Virtual Array,* SG24-4951.

A.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 to define 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 256 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.

A.2 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 free space for writing. You can ensure sufficient free space by observing the NCL of the RVA. The RVA′s physical disk space typically should not be filled above 75% NCL.

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.

© Copyright IBM Corp. 1998

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. The data of a functional track may fit on one or more FBA blocks.

If the data of a fuctional 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 56 shows the pointer table structure of the RVA.

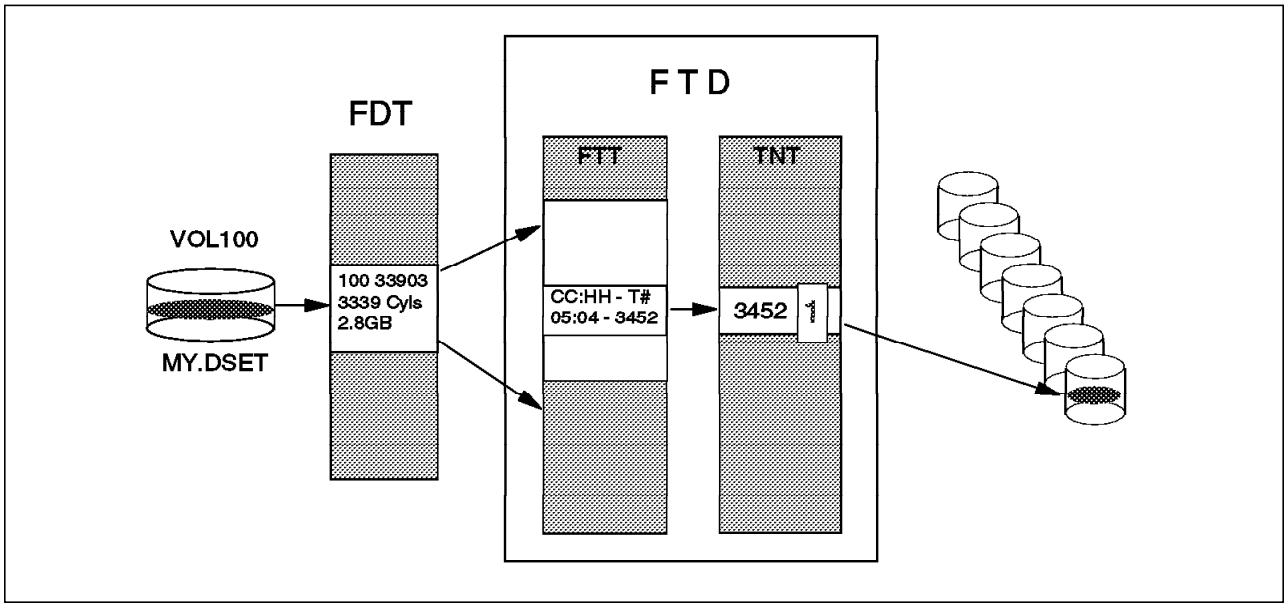


Figure 56. The RVA Tables

A.3 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, are eliminated by compaction. Experience shows that RVA customers can achieve an overall compression and compaction ratio of 3.6:1 to 4.4:1 or higher.

A.4 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 data; as far as the RVA hardware is concerned, it is a virtual copy of the data.

A.5 SnapShot Copy

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

The RVA′s virual 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 57).

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 storagee, 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.

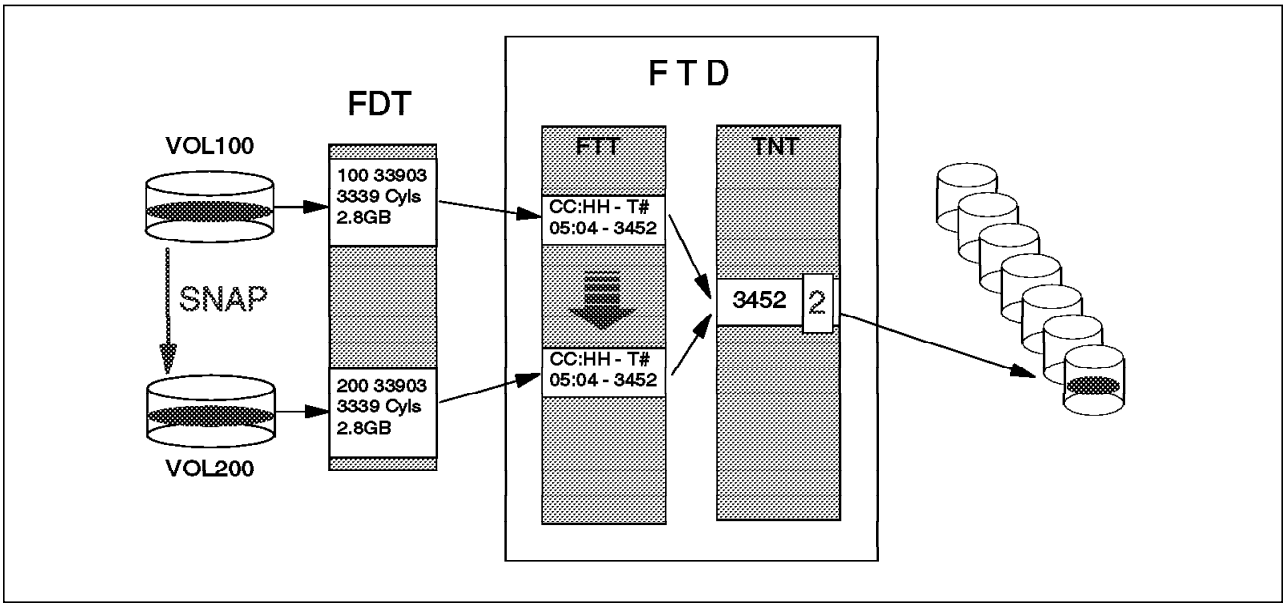


Figure 57. Data Set Snap. SnapShot creates a logical copy by copying the FTD pointers.

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.)

A.6 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.

You invoke SnapShot commands with IXFP from:

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

SnapShot commands can be set up to be invoked by end users as well as by database administrators and system programmers.

A.7 Hardware and Software Requirements

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.

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

A.7.1 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 4.3.26 or later.

A.7.2 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) and MVS/DFP 3.1.1 (5665-XA3); or DFSMS/MVS 1.1 (5695-DF1)
- In a 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

Appendix B. System Data Mover Application Program Interface Functions

When DFSMSdss SnapShot is issued and all requirements for a SnapShot are met, a snap is issued by automatically calling the SDM API, ANTRQST. This is done completely transparently to your applications.

When you let DFSMSdss SnapShot dynamically allocate the target, it prefers the same RVA where the source resides, just as SnapShot prefers.

If the target is SMS managed, the DFSMSdss volume preferencing technique is used.

These are the SnapShot services internally provided through ANTRQST:

SDVCINFO Obtains basic and/or extended information about a functional device

SQRYSSYS Obtains a filtered list of all RVA subsystems attached to host

SQRYDVCS Obtains a filtered list of all online functional devices or those online devices matching the type, partition, and subsystem of a specified input device

SSNAP Copies functional track pointers from a source device to the target device. The target must be the same device type in the same RVA subsystem and partition.

SRELEASE Releases physical back-end space associated with functional tracks

ANTRQST has been externalized so that other subsystems can use it.

Refer to Appendix B in *DFSMS/MVS V1R4 DFSMSdfp Advanced Services* for a complete description of the ANTRQST macro functions.

© Copyright IBM Corp. 1998

Appendix C. Special Notices

This publication is intended to help technical professionals and managers implement IBM RAMAC SnapShot for MVS/ESA. The information in this publication is not intended as the specification of any programming interfaces that are provided by SnapShot, IXFP, or DFSMS/MVS. See the PUBLICATIONS section of the IBM Programming Announcement for IBM RAMAC SnapShot for MVS/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, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The 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.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other

© Copyright IBM Corp. 1998

operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

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

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

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

DATABASE 2	DB2/2
DFSMS	DFSMS/MVS
DFSMS/VM	DFSMSdfp
DFSMSdss	DFSMShsm
DFSMSrmm	DFSORT
ECKD	Enterprise Systems Connection Architecture
ES/3090	ES/9000
ESCON	FFST
Hardware Configuration Definition	IBM
IMS	IMS/ESA
MVS/DFP	MVS/ESA
NetView	OPC
OS/390	Parallel Sysplex
PR/SM	Predictive Failure Analysis
RACF	RAMAC
Resource Measurement Facility	RETAIN
RMF	S/390
Service Director	Virtual Machine/Enterprise Systems Architecture
VM/ESA	VSE/ESA

The following terms are trademarks of other companies:

SnapShot is a trademark and property of Storage Technology Corporation for a duplication product.

IXFP	Storage Technology Corporation
Extended Storage Architecture	Storage Technology Corporation
Iceberg	Storage Technology Corporation
XSA	Storage Technology Corporation
XSA/Reporter	Storage Technology Corporation
StorageTek	Storage Technology Corporation
SAS	SAS Institute, Incorporated
SAS/C	SAS Institute, Incorporated
SAS/GRAPH	SAS Institute, Incorporated
SAP and R/3	SAP AG

Other trademarks are trademarks of their respective companies.

Appendix D. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

D.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 133.

- *Implementing SnapShot*, SG24-2241
- *Implementing Concurrent Copy*, GG24-3990
- *RAMAC Virtual Array*, SG24-4951
- *IMS/ESA Version 6 Guide*, SG24-2228
- *High Availability Considerations: SAP R/3 on DB2 for OS/390*, SG24-2003
- *DB2 for MVS/ESA Version 4 Data Sharing Implementation*, SG24-4791

D.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
Lotus Redbooks Collection	SBOF-6899	SK2T-8039
Tivoli Redbooks Collection	SBOF-6898	SK2T-8044
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
RS/6000 Redbooks Collection (PDF Format)	SBOF-8700	SK2T-8043
Application Development Redbooks Collection	SBOF-7290	SK2T-8037

D.3 Other Publications

These publications are also relevant as further information sources:

- *DFSMSdss/SnapShot Performance White Paper*
- *DFSMS/MVS V1R4 DFSMSdss Storage Administration Reference*, SC26-4929
- *DFSMS/MVS V1R4 DFSMSdss Storage Administration Guide*, SC26-4930
- *DFSMS/MVS V1R4 DFSMSdfp Advanced Services*, SC26-4921
- *DFSMShsm Implementation and Customization Guide*,SH21-1078
- *DB2 Utility Guide and Reference*,SC26-3395

© Copyright IBM Corp. 1998

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at http://www.redbooks.ibm.com/.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Redbooks Web Site on the World Wide Web**

  http://w3.itso.ibm.com/

- **PUBORDER** — to order hardcopies in the United States
- **Tools Disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLCAT REDPRINT
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type the following command:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  ```

  To register for information on workshops, residencies, and redbooks, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1998
  ```

- **REDBOOKS Category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (http://www.redbooks.ibm.com/redpieces.html). 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.

© Copyright IBM Corp. 1998

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1)001-408-256-5422 (Outside USA)** — ask for:

Index # 4421 Abstracts of new redbooks
Index # 4422 IBM redbooks
Index # 4420 Redbooks for last six months

- **On the World Wide Web**

Redbooks Web Site	http://www.redbooks.ibm.com/
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (`http://www.redbooks.ibm.com/redpieces.html`). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number
- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Index

© Copyright IBM Corp. 1998

E

F

G

H

I

K

L

M

N

O

P

R

S

V

W

ITSO Redbook Evaluation

Implementing DFSMSdss SnapShot and Virtual Concurrent Copy
SG24-5268-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.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**
- **Independent Software Vendor**
- **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!)**

_____

_____

_____

_____

_____

© Copyright IBM Corp. 1998

SG24-5268-00
Printed in the U.S.A.