Save Set Manager for OpenVMS

User and Installation Guide

This manual contains installation and user information for the Save Set Manager software.

Operating Systems:	OpenVMS VAX Version 7.3, OpenVMS Alpha Version 7.3-2 & 8.2 and OpenVMS I64 Version 8.2.
Software Version:	Save Set Manager Version 1.8

January 2005

Printed in the U.S.A.

This document was prepared using VAX DOCUMENT, Version 2.1.

Contents

Tables

Preface

The *Save Set Manager User and Installation Guide* describes the procedures for configuring, installing, and operating the Save Set Manager (SSMgr) software.

Intended Audience

This manual is intended primarily for use by system managers, operators, and workstation users.

Structure

This manual is organized as follows:

Chapter 1	Provides an overview of the SSMgr
Chapter 2	Explains how to install the SSMgr
Chapter 3	Describes the command language interface of the SSMgr
Chapter 4	Describes SSMgr reports
Chapter 5	Lists the error messages generated by the SSMgr

Related Documents

The following table lists documents that contain information related to this product:

Document Title	Order Number
The Save Set Manager for OpenVMS User and Installation Guide	AA–QDKQJ–TE
OpenVMS System Management Utilities Reference Manual: A–L	AA–PV5Px–TK
OpenVMS License Management Utility Manual	AA–PVXUx–TK

Documentation Conventions

The following conventions are used in this manual:

boldface type	Boldface type indicates the first instance of terms being defined in text, in the glossary, or both.
italic type	Italic type indicates emphasis and complete manual titles. In the glossary, italic type is also used to indicate cross-references.

1
Introduction

The Save Set Manager for OpenVMS™ (SSMgr) is a layered software product that reduces the time used to create OpenVMS BACKUP save sets, while providing greater flexibility in save set management.

OpenVMS BACKUP users have had a rich set of options and a high level of data integrity available at the cost of greater down time during the backup operation.

SSMgr runs as a post-processor on save sets created by BACKUP, or other storage management software that conforms to the BACKUP save set structure. It allows some functions that are currently done online with BACKUP, such as checking the integrity of save sets, making multiple copies of a save set, and XOR or cyclic redundancy checking (CRC), to be done offline using SSMgr. SSMgr does not replace OpenVMS BACKUP for creating or restoring save sets.

SSMgr is supported for OpenVMS VAX Version 7.3, for OpenVMS Alpha Versions 7.3-2 & 8.2 and for OpenVMS I64 Version 8.2. SSMgr supports a DCL interface and all disk and tape technologies supported by OpenVMS systems.

1.1 Advantages of SSMgr

Using SSMgr to manage your save sets provides the following benefits:

- You can run OpenVMS BACKUP with fewer time-consuming data integrity options, such as /VERIFY, to complete the operation with minimal down time. Once you have the backup save set, you perform the validation and addition of data integrity features offline using SSMgr.
- You can reduce the number of full backups in your backup cycle by substituting incremental backups and using the save set merge feature of the SSMgr.
- Restore time is greatly reduced, because you can use SSMgr to merge incremental backups into image save sets to maintain a current image save set at all times. This process replaces the practice of using OpenVMS BACKUP to restore an image save set plus many incremental save sets.
- SSMgr does not require access to any data outside of its input save sets to run, so it can be run on any OpenVMS system that can directly access those save sets or to which they can be moved.
- You can transfer save sets between tape technologies.
- The data integrity of SSMgr save sets is comparable to save sets created with other OpenVMS utilities using full data checking.
- You gain additional flexibility in managing your save set files by using the SSMgr commands for save set validation, copy, and merge functions.

- You can monitor the integrity of archived save sets to determine how badly the media has degraded over time, and you can use SSMgr to reconstruct and restore the save sets to fresh media.
- You can create OpenVMS BACKUP Journal Files (.BJL) for previously created save sets.
- You can create up to five copies of a save set simultaneously.

SSMgr performs its operations on save sets that strictly conform to the OpenVMS BACKUP save set format for disk or tape devices. You can use save sets and journal files that have been created by SSMgr as input to OpenVMS BACKUP.

1.2 SSMgr Functions

SSMgr performs the following functions:

- **VALIDATE**—Validates that all data blocks in the save set can be read without error. Validation includes the verification that any CRC or XOR protection recorded in the save set is consistent with the data they are protecting. This command may eliminate your need for a /VERIFY pass when you run OpenVMS BACKUP.

  Tapes degrade over time, and the VALIDATE command allows you to monitor the degradation of your archived save sets so that you can detect that degradation early, thus minimizing data loss. If the save sets contain XOR redundancy protection and degradation is detected early, you may avoid any consequent data loss.
- **COPY**—Used to make a copy of a save set, to reconstruct data that has been found to be degenerated by VALIDATE, or to transfer a save set to a different media technology. You also can change the format of save sets, such as block or group size, and you can insert or remove XOR and CRC protection in the save set.

  Using OpenVMS BACKUP, if you wished to create multiple copies of a save set, such as a local copy and a vault copy, you had to perform your online backup operation multiple times. With SSMgr, you need to perform the OpenVMS BACKUP operation only once, after which you can produce up to 5 copies offline with a single COPY operation.
- **MERGE**—Allows for a more flexible backup policy, including merging incremental save sets into image save sets to create a fully restorable save set, or merging incremental save sets into each other. The MERGE command can be used to reduce the frequency of image save set operations (also called full backup) done with OpenVMS BACKUP.
- **SAVESET HELP**—Used to get help on SSMgr operations.

Both COPY and MERGE commands include the option to validate their input save sets while performing their operations.

1.3 How SSMgr Changes Save Set Management Policy

A common BACKUP policy is to perform an image backup once a week and incremental backups all other days. In the worst case scenario, a disk could be lost on the seventh day, after the last incremental backup had been performed and before the next image backup. To restore the disk, the image save set must first be restored and then the six incremental save sets applied in sequence. The

time needed to restore the data is longer if the period of time between image backups is longer.

Backup Strategies

SSMgr allows you to select alternative policies as the best backup strategy for your installation. The following two different policies using the SSMgr MERGE command could dramatically reduce the time needed to restore the disk:

- Perform an image backup on day one. Each following day, perform an incremental backup and merge the incremental save set for the day with the image backup save set, creating a new image save set. Repeat this process for each subsequent day, merging that day's incremental save set with the image save set created by the previous day's merge operation until you wish to perform your next image backup.

  This strategy results in a single large save set to restore in the event of data loss.

- Perform an image backup on day one and an incremental backup on day two. Perform another incremental backup on day three and merge its save set with the incremental save set from day two to produce a cumulative incremental save set. Repeat for each subsequent day, merging that day's incremental save set with the cumulative incremental save set created by the previous day's merge operation until you wish to perform your next image backup.

  This strategy results in one image save set and one incremental save set to restore in the event of data loss.

Merging an incremental save set with an image save set requires two SSMgr passes over the incremental save set and one pass over the image save set. Merging two incremental save sets requires one pass over each save set. Therefore, depending upon the relative size of the image and incremental save sets, you may prefer to use the second strategy instead of the first one.

Controlling Image Backup Save Set Size

Image save sets created by SSMgr merge operations of an image save set and one or more incremental save sets can grow to be larger than the backed-up disk's capacity. Note that this same condition occurs when using OpenVMS BACKUP to restore an image save set and all subsequent incremental save sets.

The increase in size in both cases is due to directories or files on the volume that were deleted or renamed subsequent to the original image backup. Both BACKUP and SSMgr take a conservative approach to the files that used to be contained in those deleted or renamed directories and their subdirectories; BACKUP does not delete these files when restoring an incremental save set that shows the directory deleted, and SSMgr retains these files when merging in such an incremental save set.

Therefore, SSMgr MERGE operations do not eliminate the need for periodic image backups, but by using MERGE you can generally increase the interval between image backups.

Other Ways to Reduce the Time Spent on BACKUP Operations

Time spent on online OpenVMS BACKUP operations could be further reduced by omitting the calculation of CRC protection in BACKUP, and adding in such protection in a COPY or MERGE operation in SSMgr. CRC protection and XOR redundancy were originally put into BACKUP as a protective mechanism against the relatively poor quality tapes (and tape drives) of previous storage

technologies; their main use on modern tape systems is to protect against long-term degradation of tape media. CRC calculations are compute intensive and can reduce BACKUP throughput on less powerful VAX systems. XOR redundancy blocks can limit BACKUP throughput on slower tape technologies such as QIC or RDAT.

Omitting these protections in BACKUP save set operations results in greater throughput during the backup operation. These protections can then be inserted during a subsequent SSMgr COPY or MERGE operation. In the event that SSMgr detects an error on a recently produced save set, you can run OpenVMS BACKUP again.

1.4 OpenVMS Compatibility

The following general comments apply to using SSMgr with OpenVMS systems.

1.4.1 ODS and Extended File Specification Support

SSMgr V1.8 supports all operations on ODS-5 disks on Alpha Systems running OpenVMS V7.3-2 and V8.2. This includes support for EFS.

The following information is useful in understanding how a disk may be configured to enhance data access for improved performance.

1.4.1.1 Disk Concepts

Disk structures may be defined as either logical or physical and the two types interact with each other to some degree. That is, you cannot manipulate a logical structure without considering the effect on a corresponding physical structure.

Note that RMS disk files reside on Files–11 On-Disk Structure (ODS) disks. Files–11 is the name applied to the disk structures supported by the operating system.

Files–11 disk structures are further characterized as being either on-disk structures or CD-ROM volume and file structures. The Files–11 structure is a hierarchical organization of files, their data, and the directories needed to gain access to them.

The OpenVMS file system implements the Files–11 on-disk structure and provides random access to the files located on the disk or CD-ROM. Users can read from and write to disks; they can only read from CD-ROMs.

On-disk structures include Levels 1, 2, and 5. (Levels 3 and 4 are internal names for ISO and High Sierra CD formats.) ODS-1 and ODS-2 structures have been available on OpenVMS systems for some time.

Beginning with OpenVMS Version 7.2-1 on Alpha systems, you can also specify ODS-5 format disks.

Table 1–1 compares the specific characteristics of Files–11 On-Disk Structure (ODS) Levels 1, 2, and 5.

Table 1–1 OpenVMS File Structure Options - A Comparison

Characteristic	ODS-1(VAX only)	ODS-2	ODS-5
File names	9.3	39.39	238 bytes, including the dot. For Unicode, that is 119 characters including the dot.
Character set	Uppercase alphanumeric	Uppercase alphanumeric plus hyphen (-), dollar sign ($), and underscore (_)	ISO Latin-1, Unicode
File versions	32,767 limit; version limits are not supported	32,767 limit; version limits are supported	32,767 limit; version limits are supported.
Directories	No hierarchies of directories and subdirectories; directory entries are not ordered [1]	Alpha: 255[2] VAX: 8 (with rooted logical, 16)	Alpha: 255 VAX: 8 (with rooted logical, 16)
OpenVMS Cluster Access	Local access only; files cannot be shared across a cluster	Files can be shared across a cluster	Files can be shared across a cluster. However, only computers running OpenVMS Version 7.2 and 7.3 or later can mount ODS-5 disks. VAX computers running Version 7.2 and 7.3 or later can see only files with ODS-2 style names.
Disk	Unprotected objects	Protected objects	Protected objects
Disk quotas	Not supported	Supported	Supported
Multivolume files and volume sets	Not supported	Supported	Supported
Placement control	Not supported	Supported	Supported
Caches	No caching of file header blocks, file identification slots, or extent entries	Caching of file header blocks, file identification slots, or extent entries	Caching of file header blocks, file identification slots, or extent entries
Clustered Allocation	Not supported	Supported	Supported
Backup Home Block	Not supported	Supported	Supported

[1]RSX–11M, RSX–11D, RSX–11M–PLUS, and IAS systems do not support subdirectories and alphabetical directory entries.

[2]Prior to OpenVMS Version 7.2 and 7.3, RMS limited directory levels to 8 or 16.

(continued on next page)

Table 1–1 (Cont.) OpenVMS File Structure Options - A Comparison

Characteristic	ODS-1(VAX only)	ODS-2	ODS-5
Protection code E	E means "extend" for the RSX–11M operating system but is ignored by system but is ignored by OpenVMS	E means "execute access"	E means "execute access"
Enhanced protection features (for example, access control lists)	Not supported	Enhanced protection features supported	Enhanced protection features supported
RMS journaling	Not supported	Supported	Supported

For details please read Chapter 1 Introduction of Guide to OpenVMS File Applications.

1.4.2 OpenVMS BACKUP Utility

Save sets created by any version of OpenVMS BACKUP, up to and including V7.3, are accepted by SSMgr. Also, all save sets created by SSMgr are valid inputs to all versions of OpenVMS BACKUP. All save sets created by any version of SSMgr are accepted by SSMgr, but not all SSMgr operations are valid for all types of save sets. For example, while SSMgr can copy physical save sets, it cannot merge two physical save sets.

Unlike OpenVMS BACKUP, output tapes used by SSMgr must be initialized before use and all tapes must be mounted without the /FOREIGN qualifier.

1.4.3 VMScluster Support

SSMgr runs as a single image on a single node and does not include any explicit VMScluster™ support.

1.4.4 OpenVMS License Management Facility

SSMgr uses the OpenVMS License Management Facility (LMF) to determine if a node is authorized to use the software.

1.4.5 Privileges

SSMgr requires no privileges other than TMPMBX and NETMBX in order to perform its functions.

1.4.6 Internationalization of Messages

The OpenVMS MESSAGE utility is used by the SSMgr Command Language Interface (CLI) to allow internationalization of error and status messages.

1.4.7 OpenVMS HELP Utility

An OpenVMS HELP utility file that describes the CLI is provided on the SSMgr distribution media and is installed with the software. Use the HELP SAVESET command to access this feature from the DCL prompt.

2
Installation Procedure

2.1 Preinstallation Requirements and Preparations

This section discusses the system requirements and preparations necessary for installing the SSMgr software.

The SSMgr kit includes online and hardcopy release notes. HP strongly recommends that you read the release notes before proceeding with the installation. The release notes for printing to hardcopy are in a file named SYS$HELP:SSM018_RELEASE_NOTES.PS. The online release notes are in a file named SYS$HELP:SSM018.RELEASE_NOTES.

SSMgr requires that your system to be running OpenVMS VAX Version 7.3 or OpenVMS Alpha 7.3-2 & 8.2 or OpenVMS I64 8.2. Installation may only be completed on a per-node basis.

In case you have SSMGR V1.7 installed on the system please remove it using the following command:

```
$ product remove ssmgr
```

2.2 License Registration

Before you install and run SSMgr on a new node, you must first register a License Product Authorization Key (License PAK) using the License Management Facility (LMF).

For complete information on using LMF, see the *OpenVMS License Management Utility Manual*.

2.2.1 Privileges and Disk Space

To install the SSMgr software, you must be logged into an account that has either of the following privileges:

- SETPRV
- CMKRNL, WORLD, and SYSPRV

On VAX system SSMgr V1.8 requires 3600 free disk blocks on the system disk to install. On Alpha system SSMgr V1.8 requires 4200 free disk blocks on the system disk to install. On I64 system SSMgr V1.8 requires 8000 free disk blocks on the system disk to install. To determine the number of free disk blocks on your system disk, enter the following command at the DCL prompt:

```
$ SHOW DEVICE SYS$SYSDEVICE
```

2.2.2 Process Account Quotas

The account you use to install SSMgr must have sufficient quotas to enable you to perform the installation. Table 2–1 summarizes the process quotas required for the installation account. Unless you are installing SSMgr in your system account, these process quotas are in addition to the normal process quotas and are necessary to ensure the proper level of resources for your system.

Table 2–1 Process Account Quotas for the Installing Account

Process Account	Quota
ASTLM	10
BIOLM	10
BYTLM	4000
DIOLM	10
ENQLM	20
FILLM	300
PRCLM	2
TQLM	10
PGFLQUOTA	10000

2.2.3 User Account Quotas

User account quotas are stored in the SYSUAF.DAT file. Use the OpenVMS AUTHORIZE utility to verify and change user account quotas. First set your directory to SYS$SYSTEM and then run AUTHORIZE, as shown in the following example:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF>
```

At the UAF> prompt, enter the SHOW command with an account name to check a particular account. For example:

```
UAF> SHOW SMITH
```

To change a quota, enter the MODIFY command. The following example changes the FILLM quota for the SMITH account and then exits from the utility:

```
UAF> MODIFY SMITH /FILLM=50
UAF> EXIT
```

Any account from which SSMgr is to be run must have at least the TMPMBX and NETMBX privileges. Use the OpenVMS AUTHORIZE utility to determine whether users who will run SSMgr have the privileges they require.

After you exit from the AUTHORIZE utility, the system displays messages indicating whether or not changes were made. Once the changes have been made, you must log out and log in again for the new quotas to take effect.

For more information on modifying account quotas, see the description of the AUTHORIZE utility in the *OpenVMS System Management Utilities Reference Manual: A–L*.

2.2.4 Backing Up Your System Disk

HP recommends that you back up your system disk before installing any software.

Use the backup procedures that are established at your site. For details on performing a system disk backup, see the section on the BACKUP utility in the *OpenVMS System Management Utilities Reference Manual: A–L*

2.3 Sample Installation Procedure for OpenVMS I64

Same procedure will be followed for VAX and ALPHA.

$product install SSMgr

The following product has been selected: HP I64VMS SSMGR V1.8 Layered Product [Installed]

Do you want to continue? [YES]

Configuration phase starting ...

You will be asked to choose options, if any, for each selected product and for any products that may be installed to satisfy software dependency requirements.

HP I64VMS SSMGR V1.8: HP Save Set Manager for OpenVMS I64

```
© 2005 Hewlett-Packard Development Company, L.P.

 The following process are still active ---

     TCPIP$PORTM_1
     TCPIP$FTP_1
     SRINATHV
     _OPA0:
     AKILA
     GAYATHRIS
     NIPUN

  Do you want to continue[NO] ? y

  * Does this product have an authorization key registered and loaded? y
Do you want the defaults for all options? [YES]
Process Account Quotas needed for the Installing Account

          ASTLM     10
          BIOLM     10
          BYTLM     4000
          DIOLM     10
          ENQLM     20
          FILLM     300
          PRCLM     2
          TQLM      10
          PGFLQUOTA 10000

    Installtaion account privileges

          To install the SSMgr software, you must be logged into an account
          that has either of the following privileges:
          SETPRV  or
          CMKRNL, WORLD, and SYSPRV

    It is recommended that you take backup of system disk.

    Do you want to continue? [YES] y
Do you want to review the options? [NO] y
```

```
HP I64VMS SSMGR V1.8: HP  Save Set Manager for OpenVMS I64 [Installed]
    Do you want to run IVP ?: YES

Are you satisfied with these options? [YES]

Execution phase starting ...

The following product will be installed to destination:
    HP I64VMS SSMGR V1.8                DISK$OPENVMS:[VMS$COMMON.]

Portion done: 0%...10%...20%...30%...40%...50%...70%...80%...90%...100%

The following product has been installed:
    HP I64VMS SSMGR V1.8    Layered Product

%PCSI-I-IVPEXECUTE, executing test procedure for HP AXPVMS SSMGR V1.8 ...

This is the SAVESET IVP command procedure.

$ saveset validate blue.bck
ERRORS terminal option enabled
EVENTS terminal option enabled
Primary input save set DKA100:[NIPUN.V18PCSI.NEWNAME]BLUE.BCK;1 opened

  Save Set Manager V1.8  Time: 6-Dec-2004 14:08:06.80

    SAVESET function: VALIDATE
    Primary input save set: DKA100:[NIPUN.V18PCSI.NEWNAME]BLUE.BCK;1
      No journal file

    Final status of each save set:

      Save set name:  DKA100:[NIPUN.V18PCSI.NEWNAME]BLUE.BCK;1
        No errors detected

$ saveset validate yellow.bck
ERRORS terminal option enabled
EVENTS terminal option enabled
Primary input save set DKA100:[NIPUN.V18PCSI.NEWNAME]YELLOW.BCK;1 opened

  Save Set Manager V1.8  Time: 6-Dec-2004 14:08:07.35

    SAVESET function: VALIDATE
    Primary input save set: DKA100:[NIPUN.V18PCSI.NEWNAME]YELLOW.BCK;1
      No journal file

    Final status of each save set:

      Save set name:  DKA100:[NIPUN.V18PCSI.NEWNAME]YELLOW.BCK;1
        No errors detected

$ saveset copy blue.bck red.bck
ERRORS terminal option enabled
EVENTS terminal option enabled
Primary input save set DKA100:[NIPUN.V18PCSI.NEWNAME]BLUE.BCK;1 opened
Output save set DKA100:[NIPUN.V18PCSI.NEWNAME]RED.BCK; opened

  Save Set Manager V1.8  Time: 6-Dec-2004 14:08:08.01

    SAVESET function: COPY
    Primary input save set: DKA100:[NIPUN.V18PCSI.NEWNAME]BLUE.BCK;1
      No journal file
    Output save set: DKA100:[NIPUN.V18PCSI.NEWNAME]RED.BCK;
      No journal file

    Final status of each save set:

      Save set name:  DKA100:[NIPUN.V18PCSI.NEWNAME]BLUE.BCK;1
        No errors detected
```

```
      Save set name:  DKA100:[NIPUN.V18PCSI.NEWNAME]RED.BCK;
        No errors detected

$ saveset merge blue.bck yellow.bck green.bck
ERRORS terminal option enabled
EVENTS terminal option enabled
Primary input save set DKA100:[NIPUN.V18PCSI.NEWNAME]BLUE.BCK;1 opened
Secondary input save set DKA100:[NIPUN.V18PCSI.NEWNAME]YELLOW.BCK;1 opened
Output save set DKA100:[NIPUN.V18PCSI.NEWNAME]GREEN.BCK; opened

  Save Set Manager V1.8  Time: 6-Dec-2004 14:08:09.02

    SAVESET function: MERGE
    Primary input save set: DKA100:[NIPUN.V18PCSI.NEWNAME]BLUE.BCK;1
      No journal file
    Secondary input save set: DKA100:[NIPUN.V18PCSI.NEWNAME]YELLOW.BCK;1
      No journal file
    Output save set: DKA100:[NIPUN.V18PCSI.NEWNAME]GREEN.BCK;
      No journal file

    Final status of each save set:

      Save set name:  DKA100:[NIPUN.V18PCSI.NEWNAME]BLUE.BCK;1
        No errors detected

      Save set name:  DKA100:[NIPUN.V18PCSI.NEWNAME]YELLOW.BCK;1
        No errors detected

      Save set name:  DKA100:[NIPUN.V18PCSI.NEWNAME]GREEN.BCK;
        No errors detected

The SAVESET IVP command procedure has successfully completed.
%PCSI-I-IVPSUCCESS, test procedure completed successfully

HP I64VMS SSMGR V1.8: HP Save Set Manager for OpenVMS I64

    Users of this product require the following privileges:
        (TMPMBX,NETMBX)

    Release notes for Save Set Manager available.

        The release notes for Save Set Manager are available in the file
        SYS$HELP:SSM018.RELEASE_NOTES.

    User  guides for Save Set Manager available.

           SYS$HELP:SSM018_USER_GUIDE.TXT
           SYS$HELP:SSM018_USER_GUIDE.PS
           SYS$HELP:SSM018_USER_GUIDE.PDF
```

Errors can occur during the installation if any of the following conditions exist:

- The operating system version is incorrect.
- Quotas necessary for successful installation are insufficient.
- System parameter values for successful installation are insufficient.
- An image mismatch error is issued while running the IVP.

2.4 Reporting Product Problems

If you encounter a problem while using this software, report it to HP Customer Services in the U.S.A. 1-800-354-9000. In other countries, report it through your usual support channels.

3
Using SSMgr

This chapter describes how to use SSMgr, from preparing save sets to using the command language interface.

3.1 Input and Output Save Set Specifiers

SSMgr operates on save sets. Each SSMgr command takes at least one input save set as a parameter. Some SSMgr commands create output save sets. Input save sets may be on disk or tape, and SSMgr can create output save sets on disk or tape. OpenVMS BACKUP defines several modes under which save sets may be created. Three of those save set types are fully supported by SSMgr. They are:

- Image save set—This also is called a full backup. It is a functionally equivalent copy of an entire volume or volume set.
- Incremental save set—This save set contains files that were created or modified since the last save operation.
- Selective—This also is called a file-oriented save set. It is a collection of files saved for some particular reason.

Physical save sets (copies of entire volumes) are supported by SSMgr for VALIDATE and COPY operations, but not for MERGE operations.

3.1.1 Input Save Sets

Input save sets may have been created by OpenVMS BACKUP, by SSMgr, or by some other entity that creates valid, OpenVMS save sets. For SSMgr to operate on an input save set, the device on which that save set resides must be mounted Files–11 (that is, **not** mounted /FOREIGN), must be accessible by the process performing the SSMgr operation, and must not require network access.
A save set is contained in a file whose contents have a well-defined structure. A save set file may reside on a disk, a bound volume set, a single tape, or a multivolume tape set. SSMgr does not validate, copy, or merge disks, tapes, or files. The units of operation are save sets. In order for SSMgr to operate on these save sets, the files containing them must be accessible in the appropriate manner. The following sections describe how to ensure that SSMgr can appropriately access input and output save sets that are contained in files on various media.

3.1.1.1 Input Save Sets on Disk

SSMgr can read save sets on any disk that is locally connected and mounted Files–11. The process wishing to read those files must be suitably privileged for read access. The disk may be a single disk volume or a bound volume set.

Standard OpenVMS defaults apply in specifying the pathname of the file containing the save set. That is, the default directory specification is assumed unless otherwise specified. If more than one input save set is being specified (such as the primary input save set and the secondary input save set in a

MERGE operation), the save sets may be on the same or different devices, or in the same or different directories.

For the use of wildcards in specifying pathnames, see Section 3.2

3.1.1.2 Input Save Sets on Tape

Input save sets may be located on magnetic tape. SSMgr can read save sets on any tape that is locally connected and mounted Files–11. SSMgr will always open any tape file containing a save set read only. It does not matter if the tape is write-locked or not. In mounting a tape Files–11, the tape label must be specified, or the /OVERRIDE=ID qualifier used on the $MOUNT command.

Tape are sequential devices. SSMgr will begin searching for specified file at the current position on the tape unless the /REWIND qualifier is specified on the $SAVESET command line. If /REWIND is specified, SSMgr will first rewind to BOT and then search for the specified file.

If there is more than one input save set on tape (such as the primary input save set and the secondary input save set in a MERGE operation), each must be on a different volume and mounted in a different tape drive.

3.1.1.3 Input Save Sets on Multivolume Tape Sets

A file containing a save set may span tapes in a multivolume tape set. If that is so, the tapes must have been mounted in a manner that allows continuation to the next tape in the set. There are several factors that need to be considered.

Tape switching of Files–11-mounted tapes is handled by the OpenVMS magnetic tape ancillary control process (MTAACP), which is part of the OpenVMS operating system, not SSMgr. The MTAACP performs a number of checks on the continuation volume that is loaded into the tape drive, be it by a human or by a mechanical loader. Two checks that are significant for input save sets that span tapes are the label (volume identifier) check and the set id (fileset identifier) check.

Multivolume tape sets that are created by OpenVMS BACKUP or by other OpenVMS utilities follow a canonical tape labeling scheme if the creator of the multivolume tape set does not explicitly specify the tape labels to be used for continuation tapes.

Labels generated for continuation tapes fill the six-character volume identification field. The first four characters of the field contain the first four characters of the previous volume in the volume set. (If the label is less than four characters, the volume identifier field is padded with underscores.) The fifth and sixth characters contain the relative volume number for that reel or cartridge in the volume set. For example, if the first tape has a label of ABCDEF, the second and third would be labeled ABCD02 and ABCD03. If the first tape has a label of AB, the second and third would be labeled AB__02 and AB__03.

If SSMgr will be processing an input save set that spans tapes (whether it is an operation on a single large save set or an operation that processes many save sets through the use of wildcards and the /ALL qualifier), the multivolume set must be appropriately mounted for continuation. If the labeling of the continuation volume does not fit the labeling scheme recognized by MTAACP, it will not mount and make available that volume for reading. If the multivolume tape set follows the canonical labeling scheme described above, it will mount and make available that volume for continued processing. If not, the tape set must be mounted specifying the names of each tape volume in the set. For example:

```
$ MOUNT MUA0: BEATLE,JOHN,PAUL,GEORGE,RINGO
```

The use of the /OVERRIDE=ID qualifier on the $MOUNT command will override the label check on the **first volume only**. It does not carry over to continuation volumes. Therefore, if the labels are like the ones in the previous example, the following $MOUNT command:

```
$ MOUNT/OVERRIDE=ID MUA0:
```

would result in the volume BEATLE being mounted and processed, but when the tape switch occurred, the MTAACP would send OPCOM messages asking for relative volume 2, BEAT02, to be mounted. At this point, the operation cannot continue, because the MTAACP will accept nothing other than a tape labeled BEAT02.

A second check made by the MTAACP is the fileset identifier. In some instances, continuation volumes that are initialized by OpenVMS BACKUP may not be recognized by the MTAACP as the correct next volume, even though it is. In that case, an OPCOM message is issued that says that the tape is not the next in the set.

SSMgr does its own checking to ensure that what it finds on the next tape is indeed the continuation of the save set it had been processing. Therefore, this check made by the MTAACP can be safely overridden with the /OVERRIDE=SETID qualifier on the $MOUNT command. For example,

```
$ MOUNT/OVERRIDE=SETID MUA0:  BEATLE,JOHN,PAUL,GEORGE,RINGO
```

3.1.2 Output Save Sets

Output save sets may be created by SSMgr on any nonnetwork-accessed disk, bound volume set, or tape device. As with input save sets, that device must be mounted Files–11.

3.1.2.1 Output Save Sets on Disk

SSMgr can create and write to save sets on any disk that is locally connected and mounted Files–11. The process wishing to create and write those files must be suitably privileged for create and write access. The disk may be a single disk volume or a bound volume set.

Standard OpenVMS defaults apply in specifying the pathnames of the files containing the save sets. That is, the default directory specification is assumed unless otherwise specified. See Section 3.1.4 for special considerations around the creation of multiple output save sets.

For the use of wildcards in specifying pathnames, see Section 3.2.

3.1.2.2 Output Save Sets on Tape

Output save sets may be created on magnetic tape. SSMgr can create save sets on any tape device that is locally connected and mounted Files–11. In mounting a single tape Files–11, the tape label must be specified, or the /OVERRIDE=ID qualifier used on the $MOUNT command. Further, the tape must have been initialized ($INITIALIZE) prior to mounting for use by SSMgr.

Tapes are sequential devices. SSMgr will create the specified save set file after the last file on the tape unless the /REWIND qualifier is specified on the $SAVESET command line. If /REWIND is specified, SSMgr will first rewind to BOT and then create the specified save set file, overwriting whatever may have been on the tape.

See Section 3.1.4 for special considerations around the creation of multiple output save sets.

For the use of wildcards in specifying pathnames, see Section 3.2.

3.1.2.3 Output Save Sets on Multivolume Tape Sets

A save set created by SSMgr may span tapes in a multivolume tape set. If that will be so, the tapes must have been initialized and mounted in a manner that allows continuation to the next tape in the set. There are several factors that need to be considered.

Tape switching of Files–11-mounted tapes is handled by the OpenVMS magnetic tape ancillary control process (MTAACP), which is part of the OpenVMS operating system, not SSMgr. The MTAACP has certain expectations of a continuation tape that is loaded into the drive, be it by a human or a mechanical loader. Those expectations are set by the parameters and qualifiers that had been specified in the $MOUNT command for that device. There are several different ways in which a multivolume tape set can be mounted Files–11 for use by SSMgr to create and write to output save sets:

```
$ MOUNT MUA0: BEATLE,JOHN,PAUL,GEORGE,RINGO
```

In this example, five tapes have been initialized with the tape labels BEATLE, JOHN, PAUL, GEORGE, and RINGO. The tape that had been initialized as BEATLE is loaded, and the $MOUNT command issued. When that tape is full, the MTAACP expects a tape that had been initialized as PAUL to be loaded next. If a mechanical loader is being used, and the next tape in the loader has that label, operations will proceed. If a tape is not automatically loaded, the MTAACP issues an OPCOM message requesting that relative volume 2, JOHN, be mounted. If a tape initialized with label JOHN is loaded into the drive, operations will continue on that tape. If, however, an uninitialized tape or a tape with a label other than JOHN is loaded into the drive, the MTAACP will reject that tape and continue sending OPCOM messages requesting relative volume 2, JOHN.

```
$ MOUNT MUA0: BEATLE
```

In this example, only the label of the first tape is specified on $MOUNT command line. A tape that had been initialized as BEATLE is loaded and the $MOUNT command issued. When that tape is full, the MTAACP expects a tape that been initialized as BEAT02 to be loaded next. See Section 3.1.1.3 for a description of the canonical continuation tape labeling algorithm for a description of what labels are expected when continuation tape labels are not explicitly specified on the command line.

```
$ MOUNT/INITIALIZE=CONTINUATION MUA0: BEATLE
```

In this example, only the label of the first tape is specified on the $MOUNT command line. When continuation tapes are loaded, either by a human or by a mechanical loader, the MTAACP will change the tape label of the next tape that is loaded, using the algorithm described above. That is, the label of the second tape would be reinitialized to BEAT02, the third to BEAT03, etc.

In each of the examples above, the MTAACP expects continuation tapes to have been initialized. It is also possible to use uninitialized (unformatted) tapes as continuation tapes. This requires responding to MTAACP-generated OPCOM messages with the /BLANK_TAPE qualifier. For example:

```
%%%%%%%%%%  OPCOM, 23-JAN-2002 15:23:31.78  %%%%%%%%%%
request 3, from user SNYDER
MOUNT new relative volume 2 (BEAT02) on MUA0:
```

With OPER and VOLPRO privileges, the following reply will result in the uninitialized tape being formatted and labeled, allowing operations to continue:

```
$ REPLY/BLANK_TAPE=3 "BEAT02"
```

The tape will be formatted and labeled as BEAT02 (or whatever you specify as the label you want), and mounted in the drive, allowing continuation.

If a continuation tape is loaded that has been formatted, but the tape has a label that is not the one expected by the MTAACP, the following reply will result in the tape being relabeled, allowing operations to continue:

```
$ REPLY/INITIALIZE_TAPE=3
```

The tape will be relabeled as BEAT02, and operations will continue. An alternative is:

```
$ REPLY/INITIALIZE_TAPE=3 "JOHN"
```

In this case, the tape will be relabeled as JOHN, and operations will continue.

3.1.3 Files–11 Mount Example

Example 3–1 shows a session in which an image save set on tape is merged with an incremental save set on disk. The resulting "virtual" image save set is written to a multivolume tape.

Example 3–1 A Files–11 Mount

```
$ INITIALIZE MKB200: MERGE0 1
$ INITIALIZE MKB200: MERGE1 2
$ MOUNT MKB100: IMAGE 3
%MOUNT-I-MOUNTED, IMAGE mounted on MKB100:
$ MOUNT MKB200: MERGE0,MERGE1 4
%MOUNT-I-MOUNTED, MERGE0 mounted on MKB200:

$ saveset merge mkb100:full.bck incr.bck mkb200:merge.bck 5
ERRORS terminal option enabled
EVENTS terminal option enabled
Opening file MKB100:[]FULL.BCK;1
Primary input save set MKB100:[]FULL.BCK;1 opened
Secondary input save set DISK$USER:[ELVIS]INCR.BCK;1 opened
Opening file MKB200:[]MERGE.BCK;
Output save set MKB200:[]MERGE.BCK; opened

 Save Set Manager V1.8 Time: 24-Dec-2004 13:29:09.58

    SAVESET function: MERGE
    Primary input save set: MKB100:[]FULL.BCK;1
      No journal file
    Secondary input save set: DISK$USER:[ELVIS]INCR.BCK;1
      No journal file
    Output save set: MKB200:[]MERGE.BCK;
      No journal file

    Final status of each save set:

      Save set name:  MKB100:[]FULL.BCK;1
        No errors detected
```

(continued on next page)

Example 3–1 (Cont.) A Files–11 Mount

```
      Save set name:  DISK$USER:[ELVIS]INCR.BCK;1
        No errors detected

      Save set name:  MKB200:[]MERGE.BCK;
        No errors detected

$ DISMOUNT MKB100: 6
$ DISMOUNT MKB200: 7
```

1 The output save set will span two volumes. In this line, the first tape is initialized with the label *merge0*.

2 The first tape is removed from mkb200 and the second tape is loaded. This second tape is then initialized with the label *merge1*. After initializing that tape, it is removed and the first tape is reloaded into the drive.

3 The device containing the primary input save set is mounted Files–11.

4 The device to which the "virtual" image save set will be written also is mounted Files–11 as a multivolume save set.

5 The SAVESET MERGE operation is performed. The second output tape is loaded either manually or with a loader when the first output tape reaches EOT.

6 The input device is dismounted.

7 The output device is dismounted, completing the merge operation.

3.1.4 Multiple Output Save Sets

SSMgr provides the ability to make up to 5 copies of any output save set simultaneously. These copies may be written to disk or tape. Each output save set pathname is specified as a separate parameter on the $SAVESET command line. For example,

```
$ SAVESET COPY MUA0:SS.BCK *.* [-]*.* MKA100:*.* MUA1:SS_ARCHIVE.BCK
```

In this example, the save set named SS.BCK is on a tape mounted on tape drive MUA0:. Four copies are being made. Two of the copies are being written to disk, and two other are being written to tape. The copies being written to disk are being written to the default directory and to the parent directory, and in each case, the name SS.BCK is preserved. One tape copy is being written to a tape mounted on MKA100: and the file name is again the same. The last copy is being written to a tape mounted on MUA1: and is being name SS_ARCHIVE.BCK.

With the /ALL qualifier, SSMgr may be processing multiple input save sets. With one exception, in the event of a failure to open or to write to any output save set, SSMgr will attempt to continue operation to as many output save sets as it can, from as many input save sets as it can.

If there is a failure to open any one of the output save sets for the second or subsequent input save sets, an error will be reported and operation will continue to all output save sets that could be opened. If none could be opened, an error will be reported and operation will continue with the next input save set, if any.

If SSMgr cannot successfully write to an output save set once it has been opened, that output file will be closed, an error reported, and operation will continue to all other output save sets. If SSMgr encounters a condition where it can write to

none of the output save sets, it will move to the next input save set, if any, and continue operation from there.

The one exception to the continue-if-possible philosophy is if there is a problem opening any output save set for the first (or only) input save set for a given SSMgr command. In that case, the command execution is terminated to allow the offending condition to be corrected.

There are several performance and resource implications to producing multiple output save sets:

- Multiple copies can be written to the same disk, but for each tape copy desired, there must be a dedicated tape drive.
- A SSMgr operation to multiple output save sets can proceed only as fast as the slowest output device permits. If one device is involved in error recovery, all others must wait until it's complete.
- A set of I/O buffers is allocated for each output save set, with a maximum of four 64KB buffers allocated for each.
- XOR and CRC calculations are performed for each output save set, thereby significantly increasing the compute load of the operation.
- For $ SAVESET COPY/IDENTICAL operation, there will be significant slowdown with multiple output save sets. The operation is optimized for a single output save set, an optimization that does not transfer to multiple output save set operations.

3.2 Wildcards in Save Set and Journal File Specifiers

SSMgr supports standard OpenVMS wildcarding of input and output save set specifiers, as well as journal file name specifiers. However, because of the semantics of SSMgr commands, there are some constraints.

3.2.1 Wildcard Constraints on Input Save Sets

By default, if a wildcard is used in the name of an input save set, only the first matching file found will be used. This is true for VALIDATE, COPY, and MERGE. For VALIDATE and COPY, you can elect to operate on all files that match the wildcarded save set specifier by using the /ALL qualifier. Because the MERGE command takes two input save set specifiers and one output save set specifier, the possible combinations of names of input and output save sets all using wildcards presents a dizzying semantic matrix of resultant operations. Therefore, the /ALL qualifier is not supported for the MERGE command.

As it is, use of /ALL with the COPY command has semantic implications. For example:

```
$ SAVESET COPY *.BCK FOO.BCK/ALL
```

If there are 10 files that match the wildcarded save set specifier, *.BCK, there will be 10 versions of foo.bck created. That is, the output save set for each of the 10 COPY operations will have the same name, FOO.BCK, but a different version number. If there is a version limit placed on the directory, the first several copies could be deleted on creation of subsequent copies.

3.2.2 Wildcard Constraints on Output Save Sets

Wildcarding of output save set specifiers is permitted. Resolution of the actual name of the output save set is based on the location of the wildcard or wildcards in the output save set specifier.

The components of a save set specifier are the device name, the directory string, the file name, the file type, and the version number. For example, the specifier dua1:[foo]bar.baz;3 breaks down as:

```
device:     dua1:
directory:  [foo]
file name:  bar
file type:  .baz
version:    ;3
```

Only the file name and file type components may contain wildcards. The only legal wildcard character is * and it must be the only character in that component of the file specification. When the component contains the wildcard character, the corresponding component from the primary input file specification is used. If the component is missing altogether, it is treated the same as if * were used.

```
$ SAVESET COPY DUA0:[FOO]BAR.BAZ;3 DUA1:[MUMBLE]*;6
```

In this case, the output save set would be DUA1:[MUMBLE]BAR.BAZ;6 because the file name component contains the wildcard and the file type component is missing. Therefore, both of those components mimic the corresponding component from the primary input save set specification.

3.2.3 Wildcard Constraints on Journal File Names

As with output save set specifiers, journal file name specifiers may contain wildcard characters. If the "file name" component of the journal file specifier contains a wild card, the "file name" component of the associated save set will be used. If the "file type" component of the journal file specifier contains a wild card, then the default journal file type (.BJL) will be used.

```
Example 1:  SAVESET VALIDATE DUA0:[FOO]BAR.BAZ;3/JOURNAL=*.JOU

In this case, the journal file would be written to BAR.JOU in the
default directory.

Example 2: SAVESET VALIDATE DUA0:[FOO]BAR.BAZ;3/JOURNAL=FOOBAR.*

In this case, the journal file would be written to BAR.JOU in the
default directory.

Example 3: SAVESET VALIDATE DUA0:[FOO]BAR.BAZ;3/JOURNAL=*.*

In this case, the journal file would be written to BAR.JOU in the
default directory.
```

3.3 SSMgr Command Language Interface

This section describes the valid commands for SSMgr. A matrix of commands and qualifiers is shown in Table 3–1.

Table 3–1 SSMgr Commands and Qualifiers

Command	SAVESET Command		
Qualifier	VALIDATE	COPY	MERGE
/ALL	Yes	Yes	No
/BLOCK_SIZE	No	Yes	Yes
/BRIEF	Yes	Yes	Yes
/CHECKS	Yes	Yes	Yes
/COMMENT	No	Yes	Yes
/CRC	No	Yes	Yes
/FULL	Yes	Yes	Yes
/GROUP_SIZE	No	Yes	Yes
/IDENTICAL	No	Yes	No
/JOURNAL	Yes	Yes	Yes
/LOG_FILE	Yes	Yes	Yes
/OVERRIDE	No	Yes	No
/REWIND	Yes	Yes	Yes
/TERMINAL	Yes	Yes	Yes

COPY

Copies a save set.

Format

COPY *input-ss output-ss [output-ss] [output-ss] [output-ss] [output-ss]*

Parameters

input-ss
OpenVMS file name of an input save set.

output-ss
OpenVMS file name(s) of the output save set(s). Up to 5 output save sets may be specified. Unless otherwise specified by command qualifiers, the output save set will have the same attributes for XOR group size, block size, CRC, and other attributes as the input save set. Different qualifiers may be applied to each output save set.

Description

The COPY command reads an OpenVMS BACKUP save set, verifies that it is readable and consistent, and creates 1 to 5 copies of that save set. Any rewritten (redundant) blocks in the input save set are eliminated in the COPY operation. For an explanation of rewritten blocks, see Section 4.2.2 for Successful Completion with Save Set Condition Report.

You can optionally specify software CRC and XOR protection to be present, absent, or the same as the source save set. The default is to make the output save set the same as the input save set. The desired block size of the output save set can also be specified as different from the input save set. The COPY command regenerates any bad blocks found during the operation.

Note

The size of the save set after a COPY operation may be slightly smaller than the original save set due to the way the OpenVMS BACKUP utility formats the data. SSMgr repacks its data records during processing to reduce the space required; there is no loss of data from the OpenVMS input save set, and the output save set is compatible with OpenVMS BACKUP.

Qualifiers

/ALL
/[NO]ALL (default)

Command qualifier

This qualifier specifies the behavior of Save Set Manager when save set names contain wildcards.

If ALL is specified, then all save sets matching the wildcarded input save set file specification are processed.

If NOALL is specified, then only the first file name found that matches the wildcarded file specification is processed.

The qualifier is valid only for the VALIDATE and COPY functions.

The default is /NOALL.

/BLOCK_SIZE=*n*

Output save set qualifier

This qualifier specifies the desired block size of the output save set. Valid values for *n* are between 2048 and 65024, and are rounded up to the nearest multiple of 512. The block size of the input save set is used if the BLOCK_SIZE qualifier is not specified.

As required by OpenVMS BACKUP, the upper limit for save sets on disk is 32256. If a larger block size is specified, SSMgr will round it down to 32256.

This qualifier may not be used with the /IDENTICAL qualifier.

The default is the block size of the input save set.

/BRIEF (default)

Command qualifier

This qualifier specifies that a minimum level of detail is to be echoed. The SAVESET function and save set name(s) are echoed; any error information is displayed; and the result of the SAVESET operation is reported. See /FULL for the complete list of output information.

The default is /BRIEF.

/CHECKS=([NO]CRC, [NO]XOR)

Input save set qualifier

This qualifier specifies the optional consistency checks that will be performed on the input save set; the default is CRC, XOR.

You must specify the /CHECKS qualifier immediately after the file name of each input save set when specifying the input save set. Example 3–12 also shows that when you specify both CRC and XOR checks, you must list them in parentheses with a comma separating them.

If CRC checking is specified and the input save set is written with CRC, then the CRC is computed for each block in the input save set. The computed CRC is compared against the CRC stored in the block header.

If XOR checking is specified and the input save set was written with XOR redundancy, then a running XOR is computed across each XOR group of blocks. The running XOR is then compared to the XOR block.

Each of these options requires additional processing time and may impact the performance of SSMgr. Any CRC or XOR inconsistencies are written to the log file and an informational message is returned to the calling program or user when processing completes.

/COMMENT=*string*

Output save set qualifier

This qualifier inserts a comment in the output save set. The comment *string* can be up to 252 characters. If the comment string is longer than one word or if it contains non-alphanumeric characters, it must be enclosed in quotation marks ("). The comment from the primary input save set is not copied to the output save set.

The default is no comment string written to the output save set.

/CRC
/NOCRC

Output save set qualifier

Specifying /CRC causes the CRC to be computed across each block of the output save set and stored in the block header of each block of the output save set. Specifying /NOCRC causes CRC computation to be inhibited on the output save set. If this qualifier is not specified, the CRC is computed and stored in the output save set only if it was present in the input save set.

This qualifier may not be used with the /IDENTICAL qualifier.

The default is /CRC if the primary input save set was written with CRC and /NOCRC if the primary input save set was written without CRC.

/FULL

Command qualifier

This qualifier specifies that more detail is to be echoed. In addition to information displayed with the /BRIEF qualifier, the following information for each save set is displayed:

Save set name
Save set group size
Save set block size
Number of blocks in save set
Nondirectory user files in save set
Directory user files in save set
Alias nondirectory files in save set
Alias directory files in save set
Unrecoverable CRC errors
Recoverable CRC errors
Recoverable checksum errors
Unrecoverable checksum errors
XOR errors
Read errors
Write errors
Record errors
Unrecoverable missing blocks
Recoverable missing blocks
Rewritten blocks

The default is /BRIEF.

/GROUP_SIZE=*n*

Output save set qualifier

This qualifier specifies the XOR group size in the output save set. Valid values for *n* are between 0 and 100. A value of 0 specifies that no XOR should be computed.

If this qualifier is not specified, the group size of the input save set is used for the output save set group size. This is the default action.

This qualifier may not be used with the /IDENTICAL qualifier.

/IDENTICAL

Command qualifier

This qualifier does a fast copy of the input save set. A fast copy operation copies the contents of each input save set block to the output save set without validation of the block's internal record structure. If errors are encountered during this operation, restart the operation and include the /OVERRIDE qualifier to continue operation even in spite of the errors. See the /OVERRIDE qualifier description below.

Each save set is stored with information that reflects the environment in which it was created. In a normal save set copy, the information in the output save set reflects the environment in which the *output* save set was created. With the /IDENTICAL qualifier, this information reflects the environment in which the *input* save set was created.

The /IDENTICAL qualifier is not supported for multivolume output save sets, because you may not be able to restore such a save set.

You may not use /GROUP_SIZE, /BLOCK_SIZE, /CRC, /COMMENT, or /JOURNAL qualifiers with the /IDENTICAL qualifier. You may not use the /TERMINAL=LOG qualifier with the /IDENTICAL qualifier.

/INIT_MOUNT= (TAPE_LABEL:VOLUME_LABEL, MEDIA_FORMAT:[NO]COMPACTION)

Save set qualifier

For magnetic tape volumes only, INIT_MOUNT directs SSMgr to initialize a volume and mount the drive. For input save set the drives are only mounted. INIT_MOUNT is effective only with IDENTICAL qualifier.

You must specify this qualifier immediately after the applicable file name on the command line. Again the qualifier will have no effect if the drive is already mounted.

The following consequences result when using the INIT_MOUNT qualifier:

- If specified with an input save set, SSMgr will mount it if the drive is not mounted already.
- If specified with an output save set, SSMgr initializes the output volume with the value supplied for the TAPE_LABEL field. By default the output volume will be initialized with the same label as that of the input volume. SSMgr will initialize the output volume with the same density as that of the input volume if the density is supported on the output tape drive. Otherwise the volume will initialized with the default density. By default output drives will mounted with "compaction enabled" if the input drive supports compaction and with "compaction disabled" if the input drive doesn't support compaction. As it is obvoius, if the output drive doesn't support compaction, it will always be mounted in "compaction disabled" mode.

/JOURNAL[=*journal-file-name*]

Save set qualifier

This qualifier specifies whether SSMgr will create a journal file for the save set. The journal file is an OpenVMS BACKUP journal file with an OpenVMS file name. If no journal file name is specified, the journal file will be written to <saveset_name>.BJL in the current default directory.

If a journal file with the same name already exists, the new journal file is appended. If it does not already exist, a new journal file is created.

Journal files created by SSMgr will not list the labels of follow-on volumes in multivolume tape sets. Also, if COPY/IDENTICAL is used to copy a multivolume tape to a single volume tape, the journal file will still list it as a multivolume tape set.

These differences from Backup created journal files are due to a limitation in VMS that does not allow a non-privileged process to obtain the volume labels of a multivolume tape set.

You must specify this qualifier immediately after the save set file name specifier on the command line.

This qualifier is valid for all save set file name specifiers, input or output, on all SSMgr commands.

This qualifier may not be combined with the /IDENTICAL qualifier.

Journal files will be created for each save set specified on the command line that is followed by the /JOURNAL qualifier. If more than one journal file is being created, each journal file name must be unique.

For example, SAVESET COPY FILES.BCK/JOURNAL FILES.SAV/JOURNAL is illegal, because the resulting journal file names would each be, by default, FILES.BJL.

The default is /NOJOURNAL.

/LOG_FILE[=*logfile-name*] (default)
/NOLOG_FILE

Command qualifier

This qualifier causes SSMgr to write events to a log file specified by *logfile-name*. The *logfile-name* is an ASCII file with an OpenVMS file name. If not specified, the default log file name is SAVESET.LOG in the current default directory. You can suppress creating a default log file by using /NOLOG_FILE or by including /LOG_FILE=NL: on the command line. The following events are written to the SAVESET.LOG file:

- Each invocation of SSMgr
- All output returned to the user or calling program
- Any errors or warnings encountered while processing input or output save sets
- Operator requests

Wildcards may not be used in the specification of logfile names.

/OVERRIDE
/NOOVERRIDE (default)

Command qualifier

This qualifier allows you to override certain internal consistency checks to allow execution to continue with unreadable or unrecoverable blocks in the input save set. This qualifier allows you to make a complete and identical copy of a damaged save set.

You must use the /IDENTICAL qualifier when specifying the /OVERRIDE qualifier to allow continued processing of the save set after encountering unrecoverable input save set errors. The /IDENTICAL and /OVERRIDE qualifiers must be specified in that order on the command line as shown in Example 3–4.

/REWIND
/NOREWIND (default)

Input or output save set qualifier

For magnetic tape volumes only, REWIND directs SSMgr to rewind the magnetic tape to the beginning-of-tape (BOT) marker before reading or writing the volume. If /ALL was specified, the rewind only happens once before the first SAVESET operation.

You must specify this qualifier immediately after the applicable file name on the command line, as shown in Example 3–12.

The following consequences result when using the REWIND qualifier:

- If specified with an input save set, SSMgr searches for the specified file starting at the BOT position. This allows SSMgr to find files located before the current position of the tape.

- If specified with an output save set, SSMgr overwrites the tape starting at the BOT position. All files on the tape are therefore destroyed.

The default for this qualifier is /NOREWIND. On input, the tape will be searched starting at the current tape position. On output, the new file will be opened at end-of-data.

/TERMINAL=([NO]ERRORS, [NO]EVENTS, [NO]LOG)

Command qualifier

This qualifier specifies what class(es) of information should be displayed on the user's terminal during execution of the SAVESET command.

If ERRORS is specified, then all error conditions occurring during execution are displayed as they occur, in addition to being included in the final report at the end of command execution and in the log file.

If EVENTS is specified, then all nonerror event conditions occurring during execution (e.g., tape switches) are displayed as they occur.

If LOG is specified, then the names of all output user files (for COPY and MERGE) are displayed as they are written to the output save set. For VALIDATE, the names of all input user files are displayed as they are processed. The /TERMINAL=LOG qualifier may not be used with the /IDENTICAL qualifier.

Note

When a multivolume save set is copied using the COPY/IDENTICAL command, volume switch information from the original input save set will be preserved in the single volume copy, and it will be reported as an EVENT.

The default is /TERMINAL=(ERRORS,EVENTS,NOLOG).

/WRITE_CHECK

Output save set qualifier

For magnetic tape volumes only,WRITE_CHECK directs SSMgr to check every block of data written to the output save set.

You must specify this qualifier immediately after the applicable file name on the command line.

The following consequences result when using the WRITE_CHECK qualifier:

- Each block of data written on tape will be compared with the data in memory. Which may make the operation slower. If the block-size of the output saveset is small, the operation may be slower.

Examples

Example 3–2 COPY Command

```
$ saveset copy mkb200:savesetua013.B/rewind disk$user:[kits]copy.sav
ERRORS terminal option enabled
EVENTS terminal option enabled
Rewinding MKB200: to beginning of volume
Rewind of MKB200: complete
Opening file MKB200:[]SAVESET018;1
Primary input save set MKB200:[]SAVESET018;1 opened
Output save set DISK$USER:[KITS]COPY.SAV; opened

Save Set Manager V1.8 Time: 24-Dec-2004 13:45:05.49

    SAVESET function: COPY
    Primary input save set: MKB200:[]SAVESET018;1
      No journal file
    Output save set: DISK$USER:[KITS]COPY.SAV;
      No journal file

    Final status of each save set:

      Save set name:  MKB200:[]SAVESET018;1
        No errors detected

      Save set name:  DISK$USER:[KITS]COPY.SAV;
        No errors detected
```

In Example 3–2, the tape mounted on mkb200 is rewound and then the save set, SAVESET018, is copied from that tape to a file called copy.sav on disk$user:[kits], with all save set characteristics preserved.

Example 3–3 COPY Command with Multiple Output Save Sets

```
$ saveset copy full.bck mkb200:image.bck/block=32768 -
mkb500:full20.bck/group=20
ERRORS terminal option enabled
EVENTS terminal option enabled
Primary input save set DISK$USER:[NDIAMOND]FULL.BCK;1 opened
Opening file MKB200:[]IMAGE.BCK;
Output save set MKB200:[]IMAGE.BCK; opened
Opening file MKB500:[]FULL20.BCK;
Output save set MKB500:[]FULL20.BCK; opened

Save Set Manager V1.8 Time: 24-Dec-2004 13:56:01.74

    SAVESET function: COPY
    Primary input save set: DISK$USER:[NDIAMOND]FULL.BCK;1
      No journal file
    Output save set: MKB200:[]IMAGE.BCK;
      No journal file
    Output save set: MKB500:[]FULL20.BCK;
      No journal file

    Final status of each save set:

      Save set name:  DISK$USER:[NDIAMOND]FULL.BCK;1
        No errors detected

      Save set name:  MKB200:[]IMAGE.BCK;
        No errors detected

      Save set name:  MKB500:[]FULL20.BCK;
        No errors detected
```

In Example 3–3, two tape drives and a disk drive are being used. The input save set is on DISK$USER:[NDIAMOND] and is named FULL.BCK. Two copies of this save set are being made. One copy is being written to the tape mounted on MKB200, with the same name and other characteristics as the input save set, except that the block size on the output tape is being explicitly set to 32768, regardless of the block size on the input save set. A second copy is being written to the tape mounted on MKB500. For this copy, the name of the save set is changed to FULL20.BCK, the group size is being explicitly set to 20, and all other characterists of the input save set are preserved.

Example 3–4 COPY Command with IDENTICAL and OVERRIDE Qualifiers on Save Set with Errors

```
$ saveset copy/identical/override star.sav mkb200:rock.bck/rewind
ERRORS terminal option enabled
EVENTS terminal option enabled
Rewinding MKB200: to beginning of volume
Rewind of MKB200: complete
Primary input save set DISK$USER:[GRD]STAR.SAV;1 opened
Opening file MKB200:[]ROCK.BCK;
Output save set MKB200:[]ROCK.BCK; opened
CRC error: block number 33 in save set DISK$USER:[GRD]STAR.SAV;1
Error recovery successful in save set DISK$USER:[GRD]STAR.SAV;1
%SAVESET-I-MISSINGBLK, Missing block could not be regenerated in file
DISK$USER:[GRDEAD]STAR.SAV;1
CRC error: block number 35 in save set DISK$USER:[GRD]STAR.SAV;1
Recovery unsuccessful in save set DISK$USER:[GRD]STAR.SAV;1
%SAVESET-I-BADCRC, Data CRC error in file DISK$USER:[GRD]STAR.SAV;1
CRC error: block number 42 in save set DISK$USER:[GRD]STAR.SAV;1
Block recovered from XOR in save set DISK$USER:[GRD]STAR.SAV;1
CRC error: block number 48 in save set DISK$USER:[GRD]STAR.SAV;1
Recovery unsuccessful in save set DISK$USER:[GRD]STAR.SAV;1
%SAVESET-I-BADCRC, Data CRC error in file DISK$USER:[GRD]STAR.SAV;1
XOR error at block 55 in save set DISK$USER:[GRD]STAR.SAV;1
CRC error: block number 60 in save set DISK$USER:[GRD]STAR.SAV;1
Recovery unsuccessful in save set DISK$USER:[GRD]STAR.SAV;1
%SAVESET-I-BADCRC, Data CRC error in file DISK$USER:[GRD]STAR.SAV;1
XOR error at block 66 in save set DISK$USER:[GRD]STAR.SAV;1

Save Set Manager V1.8 Time: 24-Dec-2004 14:38:49.31

    SAVESET function: COPY
    Primary input save set: DISK$USER:[GRDEAD]STAR.SAV;1
      No journal file
    Output save set: MKB200:[]ROCK.BCK;
      No journal file

    Final status of each save set:

      Save set name:  DISK$USER:[GRDEAD]STAR.SAV;1
      Unrecoverable CRC errors:             3
      Recoverable CRC errors:               2
      XOR errors:                           2

      Save set name:  MKB200:[]ROCK.BCK;
        No errors detected
```

In Example 3–4, a copy of the save set, STAR.SAV, located in the default disk directory, is being copied to a tape mounted on tape device MKB200. The tape is rewound before the copy is initiated. An "identical" copy is being made; i.e., no internal record structure validation is being done and the save set information stored within the output save set will reflect the environment in which the input save set was created, rather than the environment in which the output save set is being created. Further, the /OVERRIDE qualifier results in continued operation even in the face of errors, accepting the fact that the output save set may not be fully restorable.

Example 3–5 COPY with Wildcarding, Journaling, and No Logfile

```
$ saveset copy/all/nolog
_Input Save Set: dka400:[dcsc030.kit]*.*/checks=(crc,xor)-
_/journal=*.bjl
_Output Save Set: mkb200:*.*/crc/comment="DCSC Copy"
_Output Save Set 2:
ERRORS terminal option enabled
EVENTS terminal option enabled
Primary input save set DKA400:[DCSC030.KIT]DCSC030.A;1 opened
Opening file MKB200:[]DCSC030.A;
Output save set MKB200:[]DCSC030.A; opened

Save Set Manager V1.8 Time: 24-Dec-2004 14:53:20.21

    SAVESET function: COPY
    Primary input save set: DKA400:[DCSC030.KIT]DCSC030.A;1
      Journal file: DISK$GREEN:[DRAGON]DCSC030.BJL
    Output save set: MKB200:[]DCSC030.A;
      No journal file

    Final status of each save set:

      Save set name:  DKA400:[DCSC030.KIT]DCSC030.A;1
        No errors detected

      Save set name:  MKB200:[]DCSC030.A;
        No errors detected


Primary input save set DKA400:[DCSC030.KIT]DCSC030.B;1 opened
Opening file MKB200:[]DCSC030.B;
Output save set MKB200:[]DCSC030.B; opened

Save Set Manager V1.8 Time: 24-Dec-2004 14:56:45.83

    SAVESET function: COPY
    Primary input save set: DKA400:[DCSC030.KIT]DCSC030.B;1
      Journal file: DISK$GREEN:[DRAGON]DCSC030.BJL
    Output save set: MKB200:[]DCSC030.B;
      No journal file

    Final status of each save set:

      Save set name:  DKA400:[DCSC030.KIT]DCSC030.B;1
        No errors detected

      Save set name:  MKB200:[]DCSC030.B;
        No errors detected


Primary input save set DKA400:[DCSC030.KIT]DCSC030.C;1 opened
Opening file MKB200:[]DCSC030.C;
Output save set MKB200:[]DCSC030.C; opened

Save Set Manager V1.8 Time: 24-Dec-2004 15:02:50.30

    SAVESET function: COPY
    Primary input save set: DKA400:[DCSC030.KIT]DCSC030.C;1
      Journal file: DISK$GREEN:[DRAGON]DCSC030.BJL
    Output save set: MKB200:[]DCSC030.C;
      No journal file

    Final status of each save set:

      Save set name:  DKA400:[DCSC030.KIT]DCSC030.C;1
        No errors detected

      Save set name:  MKB200:[]DCSC030.C;
        No errors detected
```

(continued on next page)

Example 3–5 (Cont.) COPY with Wildcarding, Journaling, and No Logfile

```
  Summary of COPY operations

    Total operations attempted:             3
    Operations completing successfully:     3
$
```

In Example 3–5, all the save sets in dka400:[dcsc030.kit] are copied to the tape mounted on mkb200. No log file will be generated. CRC checking and XOR checking will be performed on all the save sets as they are being copied. A journal of the files is created for each save set and given the name of each save set with the .bjl extension. CRC is computed for each save set on the tape, and the comment, "DCSC copy" is entered into the save set comment field because the journal file name is the same for all three save sets copied. The second and third save sets' journal file entries are appended to the journal file created for the first save set.

Example 3–6 COPY with /JOURNAL and /TERMINAL Qualifiers

```
$ saveset copy/terminal=(errors,events,log)
_Input Save Set: mkb200:SAVESET018/journal
_Output Save Set: saveset018
_Output Save Set 2:
LOG terminal option enabled
ERRORS terminal option enabled
EVENTS terminal option enabled
Opening file MKB200:[]SAVESET018;1
Primary input save set MKB200:[]SAVESET018;1 opened
Output save set DISK$USER:[SOFTWARE]SAVESET018; opened
Processing user file: [SOFTWARE.SAVESET018]KITINSTAL.COM;29
Processing user file: [SOFTWARE.SAVESET018]SAVESET$IVP.COM;14
Processing user file: [SOFTWARE.SAVESET018]SAVESET.CLD;4
Processing user file: [SOFTWARE.SAVESET018]SAVESET.HLP;4
Processing user file: [SOFTWARE.SAVESET018]
 SAVESET018.RELEASE_NOTES;1
Processing user file: [SOFTWARE.SAVESET018]
 SAVESET018_RELEASE_NOTES.PS;1
Processing user file: [SOFTWARE.SAVESET018]
 SAVESET_COVER_LETTER.PS;1
Processing user file: [SOFTWARE.SAVESET018]
 SAVESET_USER_GUIDE.PS;1
Processing user file: [SOFTWARE.SAVESET018]
 SAVESET_USER_GUIDE.TXT;1

Save Set Manager V1.8 Time: 24-Dec-2004 15:31:56.90

    SAVESET function: COPY
    Primary input save set: MKB200:[]SAVESET018;1
      Journal file: SAVESET018JL;
    Output save set: DISK$USER:[SOFTWARE]SAVESET018;
      No journal file

    Final status of each save set:

      Save set name:  MKB200:[]SAVESET018;1
        No errors detected

      Save set name:  DISK$USER:[SOFTWARE]SAVESET018;
        No errors detected
```

Example 3–6 shows how to enable all classes of information that can be displayed to a terminal during a copy operation that copies the save set SAVESET018 from the tape mounted on MKB200 to the directory from which the command is issued. Note the prompt for another output save set, should you decide to create another copy during the same operation. If you don't want another copy, hit <return>. A journal file of the name, SAVESET018JL, is created in that same directory.

Example 3–7 COPY with /BLOCK_SIZE, /GROUP_SIZE, and /FULL Qualifiers

```
$ saveset copy/full mkb200:SAVESET018
_Output Save Set: copy.sav/block_size=3072/group_size=20
_Output Save Set 2:
ERRORS terminal option enabled
EVENTS terminal option enabled
Opening file MKB200:[]SAVESET018;1
Primary input save set MKB200:[]SAVESET018;1 opened
Output save set DISK$USER:[SOFTWARE]COPY.SAV; opened

Save Set Manager V1.8  Time: 24-Dec-2004 15:42:28.56

    SAVESET function: COPY
    Primary input save set: MKB200:[]SAVESET018;1
      No journal file
    Output save set: DISK$USER:[SOFTWARE]COPY.SAV;
      No journal file

    Final status of each save set:

      Save set name:  MKB200:[]SAVESET018;1
      Save set group size:                  25
      Save set block size:                  9216
      Blocks in save set:                   106
      Nondirectory files in save set:       9
      Directory files in save set:          0
      Alias nondirectory files in save set: 0
      Alias directories in save set:        0
      Unrecoverable CRC errors:             0
      Recoverable CRC errors:               0
      Unrecoverable checksum errors:        0
      Recoverable checksum errors:          0
      XOR errors:                           0
      Read errors:                          0
      Write errors:                         0
      Record errors:                        0
      Unrecoverable missing blocks:         0
      Recoverable missing blocks:           0
      Rewritten blocks:                     0
        No errors detected

      Save set name:  DISK$USER:[SOFTWARE]COPY.SAV;
      Save set group size:                  20
      Save set block size:                  3072
      Blocks in save set:                   361
      Nondirectory files in save set:       9
      Directory files in save set:          0
      Alias nondirectory files in save set: 0
      Alias directories in save set:        0
      Unrecoverable CRC errors:             0
      Recoverable CRC errors:               0
      Unrecoverable checksum errors:        0
      Recoverable checksum errors:          0
      XOR errors:                           0
      Read errors:                          0
      Write errors:                         0
      Record errors:                        0
      Unrecoverable missing blocks:         0
      Recoverable missing blocks:           0
      Rewritten blocks:                     0
        No errors detected
```

(continued on next page)

Example 3–7 (Cont.) COPY with /BLOCK_SIZE, /GROUP_SIZE, and /FULL Qualifiers

In Example 3–7, the save set, SAVESET018, is copied from the tape mounted on MKB200 to DISK$USER:[SOFTWARE]COPY.SAV. The block size will be 3072 and the group size = 20 in the save set, COPY.SAV. Note the prompt for an additional output save set if all the information is not entered on the initial command line. A full report will be generated upon completion of the copy operation.

HELP

Use this command to access online Help for SSMgr commands.

Format

SAVESET HELP [keyword]

MERGE

Merges two input save sets.

Format

MERGE *primary-input-ss secondary-input-ss output-ss [output-ss] [output-ss] [output-ss] [output-ss]*

Parameters

primary-input-ss
OpenVMS file name of an input save set. The primary save set can be an image, incremental, or file-oriented save set.

secondary-input-ss
OpenVMS file name of a secondary input save set. The secondary input save set is an incremental or file-oriented save set that covers a period of time starting at the time the primary save set was created.

output-ss
OpenVMS file name(s) of the output save set(s). Up to five output save sets may be specified. Unless otherwise specified by command qualifiers, the output save set will have the same attributes for XOR group size, block size, CRC, and other attributes as the primary input save set. Different qualifiers may be applied to each output save set.

Description

The MERGE operation takes two OpenVMS BACKUP save sets and produces 1 to 5 output save sets. Acceptable input save sets include image backups, incremental backups, and file-oriented save sets. The following combinations of input primary and secondary save sets result in the listed output save sets:

Primary Input Save Set	Secondary Input Save Set	Output Save Set
Image	Incremental	Image
Incremental	Incremental	Incremental
Incremental	File-oriented	File-oriented
File-oriented	Incremental	File-oriented
File-oriented	File-oriented	File-oriented

No other combinations of save sets are legal.

The output save set contains the merge of the two input save sets. If the primary save set was an image backup, the output saves set contain an image backup up to the time that the secondary input save set was created. If the primary and secondary save sets were incrementals, the output save set will contain an incremental save set that spans the combined time frame of the two input incrementals.

The output save sets' CRC and XOR protections can be independently specified to be present, absent, or the same as the primary save set. If detected by SSMgr, an attempt to merge a primary save set with an inappropriate incremental save set, such as from another disk or noncontiguous time period, will result in a diagnostic message and an aborted operation. However, SSMgr cannot detect all

such inappropriate combinations; the result of such an inappropriate merge will be a legal save set, but its contents may not be meaningful.

The MERGE operation does not accept physical save sets.

Qualifiers

/BLOCK_SIZE=*n*

Output save set qualifier

This qualifier specifies the desired block size of the output save set. Valid values for *n* are between 2048 and 65024, and are rounded up to the nearest multiple of 512. The block size of the primary input save set is used if the BLOCK_SIZE qualifier is not specified.

As required by OpenVMS BACKUP, the upper limit for save sets on disks is 32256. If a larger block size is specified, SSMgr will round it down to 32256.

/BRIEF (default)

Command qualifier

This qualifier specifies that a minimum level of detail is to be echoed. The SAVESET function and save set name(s) are echoed; any error information is displayed; and the result of the SAVESET operation is reported. See /FULL for a complete list of output information.

/CHECKS=([NO]CRC,[NO]XOR)

Input save set qualifier

This qualifier specifies the optional consistency checks that will be performed on the input save set; the default is CRC, XOR.

You must specify this qualifier immediately after the file name of the save set on the command line, as shown in Example 3–10. Example 3–12 shows that when you specify both CRC and XOR checks, you must list them in parentheses with a comma separating them.

If CRC checking is specified and the input save set is written with CRC, then the CRC is computed for each block in the input save set. The CRC is compared against the CRC stored in the block header.

If XOR checking is specified and the input save set was written with XOR redundancy, then a running XOR is computed across each XOR group of blocks. The running XOR is compared to the XOR block.

Each of these options requires additional processing time and may impact the performance of SSMgr. Any CRC or XOR inconsistencies are written to the log file and an informational message is returned to the calling program or user when the command completes.

/COMMENT=*string*

Command qualifier

This qualifier inserts a comment in the output save set. The *string* can be up to 252 characters. If the comment string is longer than one word or if it contains non-alphanumeric characters, it must be enclosed in quotation marks ("). If not specified, no comment is written to the output save set.

/CRC
/NOCRC

Output save set qualifier

Specifying /CRC causes the CRC to be computed across each block of the output save set and stored in the block header of each block of the output save set. Specifying /NOCRC causes CRC computation to be inhibited on the output save set. If this qualifier is not specified, the CRC is computed and stored in the output save set only if it was present in the primary input save set.

/FULL

Command qualifier

This qualifier specifies that more detail is to be echoed. In addition to information displayed with the /BRIEF qualifier, the following information for each save set is displayed:

Save set name
Save set group size
Save set block size
Number of blocks in save set
Nondirectory user files in save set
Directory user files in save set
Alias nondirectory files in save set
Alias directory files in save set
Unrecoverable CRC errors
Recoverable CRC errors
Recoverable checksum errors
Unrecoverable checksum errors
XOR errors
Read errors
Write errors
Record errors
Unrecoverable missing blocks
Recoverable missing blocks
Rewritten blocks

/GROUP_SIZE=*n*

Output save set qualifier

This qualifier specifies the XOR group size in the output save set. Valid values for *n* are between 0 and 100. A value of 0 specifies that no XOR should be computed. If this qualifier is not specified, the group size of the primary input save set is used.

/INIT_MOUNT= (TAPE_LABEL:VOLUME_LABEL, MEDIA_FORMAT:[NO]COMPACTION)

Save set qualifier

For magnetic tape volumes only,INIT_MOUNT directs SSMgr to initialize a volume and mount the drive. For input save set the drives are only mounted. INIT_MOUNT is effective only with IDENTICAL qualifier.

You must specify this qualifier immediately after the applicable file name on the command line. Again the qualifier will have no effect if the drive is already mounted.

The following consequences result when using the INIT_MOUNT qualifier:

- If specified with an input save set, SSMgr will mount it if the drive is not mounted already.
- If specified with an output save set, SSMgr initializes the output volume with the value supplied for the TAPE_LABEL field. By default the output volume will be initialized with the same label as that of the input volume. SSMgr will initialize the output volume with the same density as that of the input volume if the density is supported on the output tape drive. Otherwise the volume will initialized with the default density. By default output drives will mounted with "compaction enabled" if the input drive supports compaction and with "compaction disabled" if the input drive doesn't support compaction. As it is obvoius, if the output drive doesn't support compaction, it will always be mounted in "compaction disabled" mode.

/JOURNAL[=*journal-file-name*]

Save set qualifier

This qualifier specifies whether SSMgr will create a journal file for the save set. The journal file is an OpenVMS BACKUP journal file with an OpenVMS file name. If no journal file name is specified, the journal file will be written to <saveset_name>.BJL in the current default directory.

If a journal file with the same name already exists, the new journal file is appended. If it does not already exist, a new journal file is created.

Journal files created by SSMgr will not list the labels of follow-on volumes in multivolume tape sets. Also, if COPY/IDENTICAL is used to copy a multivolume tape to a single volume tape, the journal file will still list it as a multivolume tape set.

These differences from Backup created journal files are due to a limitation in VMS that does not allow a non-privileged process to obtain the volume labels of a multivolume tape set.

You must specify this qualifier immediately after the save set file name specifier on the command line.

This qualifier is valid for all save set file name specifiers, input or output, on all SSMgr commands.

Journal files will be created for each save set specified on the command line that is followed by the /JOURNAL qualifier. If more than one journal file is being created, each journal file name must be unique. For example, SAVESET MERGE FULL.BCK/JOURNAL INCR.BCK/JOURNAL FULL.BCK/JOURNAL is illegal, because the resulting journal file names would be FULL.BJL, INCR.BJL, and FULL.BJL.

The default is /NOJOURNAL.

/LOG_FILE[=*logfile-name*] (default)
/NOLOG_FILE

Command qualifier

This qualifier causes SSMgr to write events to a log file specified by *logfile-name*. The *logfile-name* is an ASCII file with an OpenVMS file name. By default, a logfile is created for every execution of an SSMgr command. If not specified, the default log file name is SAVESET.LOG in the current default directory. You can suppress the creation of a default log file by using /NOLOG_FILE or bu including /LOG_FILE=NL: on the command line. The following events are written to the SAVESET.LOG file:

- Each invocation of SSMgr
- All output returned to the user or calling program
- Any errors or warnings encountered while processing input save sets
- Operator requests

Wildcards may not be used in the specification of logfile names.

/REWIND
/NOREWIND (default)

Input or output save set qualifier

For magnetic tape volumes only, REWIND directs SSMgr to rewind the magnetic tape to the beginning-of-tape (BOT) marker before reading the volume.

You must specify this qualifier immediately after the applicable file name on the command line, as shown in Example 3–12 for the /CHECKS qualifier.

The following consequences result when using the REWIND qualifier:

- If specified with an input save set, SSMgr searches for the specified file starting at the BOT position. This allows SSMgr to find files located before the current position of the tape.
- If specified with an output save set, SSMgr overwrites the tape starting at BOT. All files on the tape are therefore destroyed.

The default for this qualifier is /NOREWIND, which causes SSMgr to start processing the tape from the current position.

/TERMINAL=([NO]ERRORS, [NO]EVENTS, [NO]LOG)

Command qualifier

This qualifier specifies what class(es) of information should be displayed on the user's terminal during execution of the SAVESET command.

If ERRORS is specified, then all error conditions occurring during execution are displayed as they occur, in addition to being included in the final report at the end of command execution and in the log file.

If EVENTS is specified, then all nonerror event conditions occurring during execution (e.g., tape switches) are displayed as they occur.

If LOG is specified, then the names of all output user files (for COPY and MERGE) are displayed as they are written to the output save set. For VALIDATE, the names of all input user files are displayed as they are processed.

The default is /TERMINAL=(ERRORS,EVENTS,NOLOG).

/WRITE_CHECK

Output save set qualifier

For magnetic tape volumes only,WRITE_CHECK directs SSMgr to check every block of data written to the output save set.

You must specify this qualifier immediately after the applicable file name on the command line.

The following consequences result when using the WRITE_CHECK qualifier:

- Each block of data written on tape will be compared with the data in memory. Which may make the operation slower. If the block-size of the output saveset is small, the operation may be slower.

Examples

Example 3–8 MERGE Command

```
$ saveset merge full.bck incr.bck merge.bck mkb500:merge.bck
ERRORS terminal option enabled
EVENTS terminal option enabled
Primary input save set DISK$MUSIC:[MOZART]FULL.BCK;1 opened
Secondary input save set DISK$MUSIC:[MOZART]INCR.BCK;1 opened
Output save set DISK$MUSIC:[MOZART]MERGE.BCK; opened
Opening file MKB500:[]MERGE.BCK;
Output save set MKB500:[]MERGE.BCK; opened

Save Set Manager V1.8 Time: 24-Dec-2004 16:03:43.18


    SAVESET function: MERGE
    Primary input save set: DISK$MUSIC:[MOZART]FULL.BCK;1
      No journal file
    Secondary input save set: DISK$MUSIC:[MOZART]INCR.BCK;1
      No journal file
    Output save set: DISK$MUSIC:[MOZART]MERGE.BCK;
      No journal file
    Output save set: MKB500:[]MERGE.BCK;
      No journal file

    Final status of each save set:

      Save set name:  DISK$MUSIC:[MOZART]FULL.BCK;1
        No errors detected

      Save set name:  DISK$MUSIC:[MOZART]INCR.BCK;1
        No errors detected

      Save set name:  DISK$MUSIC:[MOZART]MERGE.BCK;
        No errors detected
      Save set name:  MKB500:[]MERGE.BCK;
        No errors detected
```

In Example 3–8, an image save set, full.bck, is being merged with an incremental save set, incr.bck, in the default disk directory. Two copies of the resultant virtual image save set are being created, one in the current default directory and another on the tape mounted on device mbk500:. Each output save set is named merge.bck.

Example 3–9 MERGE Command with /FULL, /LOG, and /TERMINAL Qualifiers

```
$ saveset merge/full/log=merge.log/terminal=error incr2.bck
_Secondary Save Set: incr3.bck
_Output Save Set: incr4.bck/group_size=0/block_size=32256
_Output Save Set 2:
ERRORS terminal option enabled
EVENTS terminal option enabled
Primary input save set DISK$MUSIC:[EAGLES]INCR2.BCK;1 opened
Secondary input save set DISK$MUSIC:[EAGLES]INCR3.BCK;1 opened
Output save set DISK$MUSIC:[EAGLES]INCR4.BCK; opened

Save Set Manager V1.8 Time: 24-Dec-2004 16:11:55.01

    SAVESET function: MERGE
    Primary input save set: DISK$MUSIC:[EAGLES]INCR2.BCK;1
      No journal file
    Secondary input save set: DISK$MUSIC:[EAGLES]INCR3.BCK;1
      No journal file
    Output save set: DISK$MUSIC:[EAGLES]INCR4.BCK;
      No journal file

    Final status of each save set:

      Save set name:  DISK$MUSIC:[EAGLES]INCR2.BCK;1
      Save set group size:                  10
      Save set block size:                  32256
      Blocks in save set:                   146
      Nondirectory files in save set:       140
      Directory files in save set:          17
      Alias nondirectory files in save set: 0
      Alias directories in save set:        0
      Files removed by merge:               0
      Unrecoverable CRC errors:             0
      Recoverable CRC errors:               0
      Unrecoverable checksum errors:        0
      Recoverable checksum errors:          0
      XOR errors:                           0
      Read errors:                          0
      Write errors:                         0
      Record errors:                        0
      Unrecoverable missing blocks:         0
      Recoverable missing blocks:           0
      Rewritten blocks:                     0
        No errors detected
```

(continued on next page)

Example 3–9 (Cont.) MERGE Command with /FULL, /LOG, and /TERMINAL Qualifiers

```
Save set name:  DISK$MUSIC:[EAGLES]INCR3.BCK;1
Save set group size:                  10
Save set block size:                  32256
Blocks in save set:                   234
Nondirectory files in save set:       4
Directory files in save set:          17
Alias nondirectory files in save set: 0
Alias directories in save set:        0
Files removed by merge:               0
Unrecoverable CRC errors:             0
Recoverable CRC errors:               0
Unrecoverable checksum errors:        0
Recoverable checksum errors:          0
XOR errors:                           0
Read errors:                          0
Write errors:                         0
Record errors:                        0
Unrecoverable missing blocks:         0
Recoverable missing blocks:           0
Rewritten blocks:                     0
  No errors detected

Save set name:  DISK$MUSIC:[EAGLES]INCR4.BCK;
Save set group size:                  0
Save set block size:                  32256
Blocks in save set:                   326
Nondirectory files in save set:       144
Directory files in save set:          17
Alias nondirectory files in save set: 0
Alias directories in save set:        0
Files removed by merge:               0
Unrecoverable CRC errors:             0
Recoverable CRC errors:               0
Unrecoverable checksum errors:        0
Recoverable checksum errors:          0
XOR errors:                           0
Read errors:                          0
Write errors:                         0
Record errors:                        0
Unrecoverable missing blocks:         0
Recoverable missing blocks:           0
Rewritten blocks:                     0
  No errors detected
```

Example 3–9 shows an incremental save set in the file, incr2.bck being merged with a later incremental save set in the file, incr3.bck, and being written to a save set in the file, incr4.bck. A full report is generated upon completion of the merge operation. All errors, if there are any, are displayed to the terminal. The group size is set to 0, and the block size is set to 32256 in the merged save set, incr4.bck.

Example 3–10 MERGE Command with /FULL, /TERMINAL, /CHECKS, and CRC Qualifiers

```
$ saveset merge/full/terminal=event
_Primary Save Set: incr2.bck/checks=(crc,xor)
_Secondary Save Set: incr3.bck/checks=(crc,xor)
_Output Save Set: incr4.bck/crc
_Output Save Set 2:
ERRORS terminal option enabled
EVENTS terminal option enabled
Primary input save set DISK$BIRD:[HUMMING]INCR2.BCK;1 opened
Secondary input save set DISK$BIRD:[HUMMING]INCR3.BCK;1 opened
Output save set DISK$BIRD:[HUMMING]INCR4.BCK; opened

Save Set Manager V1.8 Time: 24-Dec-2004 16:30:30.88

    SAVESET function: MERGE
    Primary input save set: DISK$BIRD:[HUMMING]INCR2.BCK;1
      No journal file
    Secondary input save set: DISK$BIRD:[HUMMING]INCR3.BCK;1
      No journal file
    Output save set: DISK$BIRD:[HUMMING]INCR4.BCK;
      No journal file

    Final status of each save set:

      Save set name:  DISK$BIRD:[HUMMING]INCR2.BCK;1
      Save set group size:                  10
      Save set block size:                  32256
      Blocks in save set:                   146
      Nondirectory files in save set:       140
      Directory files in save set:          17
      Alias nondirectory files in save set: 0
      Alias directories in save set:        0
      Files removed by merge:               0
      Unrecoverable CRC errors:             0
      Recoverable CRC errors:               0
      Unrecoverable checksum errors:        0
      Recoverable checksum errors:          0
      XOR errors:                           0
      Read errors:                          0
      Write errors:                         0
      Record errors:                        0
      Unrecoverable missing blocks:         0
      Recoverable missing blocks:           0
      Rewritten blocks:                     0
        No errors detected
```

(continued on next page)

Example 3–10 (Cont.) MERGE Command with /FULL, /TERMINAL, /CHECKS, and CRC Qualifiers

```
Save set name:  DISK$BIRD:[HUMMING]INCR3.BCK;1
Save set group size:                  10
Save set block size:                  32256
Blocks in save set:                   234
Nondirectory files in save set:       4
Directory files in save set:          17
Alias nondirectory files in save set: 0
Alias directories in save set:        0
Files removed by merge:               0
Unrecoverable CRC errors:             0
Recoverable CRC errors:               0
Unrecoverable checksum errors:        0
Recoverable checksum errors:          0
XOR errors:                           0
Read errors:                          0
Write errors:                         0
Record errors:                        0
Unrecoverable missing blocks:         0
Recoverable missing blocks:           0
Rewritten blocks:                     0
  No errors detected

Save set name:  DISK$BIRD:[HUMMING]INCR4.BCK;
Save set group size:                  10
Save set block size:                  32256
Blocks in save set:                   359
Nondirectory files in save set:       144
Directory files in save set:          17
Alias nondirectory files in save set: 0
Alias directories in save set:        0
Files removed by merge:               0
Unrecoverable CRC errors:             0
Recoverable CRC errors:               0
Unrecoverable checksum errors:        0
Recoverable checksum errors:          0
XOR errors:                           0
Read errors:                          0
Write errors:                         0
Record errors:                        0
Unrecoverable missing blocks:         0
Recoverable missing blocks:           0
Rewritten blocks:                     0
  No errors detected
```

Example 3–10 shows the merge of two incremental save sets, incr2.bck and incr3.bck, into the save set, incr4.bck. A full report is generated upon completion of the merge operation. All events are displayed to the terminal. CRC and XOR checking is performed on both of the input save sets, incr2.bck, and incr3.bck. CRC is added to the merged save set, incr4.bck.

Example 3–11 MERGE Command with Journaling on All Save Sets

```
$ saveset merge incr2.bck/journal=incr2.bjl
_Secondary Save Set: incr3.bck/journal=incr3.bjl
_Output Save Set: incr4.bck/journal=incr4.bjl
_Output Save Set 2:
ERRORS terminal option enabled
EVENTS terminal option enabled
Primary input save set DISK$BEE:[BONNET]INCR2.BCK;1 opened
Secondary input save set DISK$BEE:[BONNET]INCR3.BCK;1 opened
Output save set DISK$BEE:[BONNET]INCR4.BCK; opened

Save Set Manager V1.8 Time: 24-Dec-2004 16:40:04.67

    SAVESET function: MERGE
    Primary input save set: DISK$BEE:[BONNET]INCR2.BCK;1
      Journal file: INCR2.BJL
    Secondary input save set: DISK$BEE:[BONNET]INCR3.BCK;1
      Journal file: INCR3.BJL
    Output save set: DISK$BEE:[BONNET]INCR4.BCK;
      Journal file: INCR4.BJL

    Final status of each save set:

      Save set name:  DISK$BEE:[BONNET]INCR2.BCK;1
        No errors detected

      Save set name:  DISK$BEE:[BONNET]INCR3.BCK;1
        No errors detected

      Save set name:  DISK$BEE:[BONNET]INCR4.BCK;
        No errors detected
```

Example 3–11 shows the merge of two incremental save set, incr2.bck and incr3.bck, into the save set, incr4.bck. A journal file is created for each save set.

VALIDATE

Validates the internal consistency and readability of a save set.

Format

VALIDATE *input-ss*

Parameters

input-ss
OpenVMS file name of the input save set.

Description

The SSMgr VALIDATE command reads an OpenVMS BACKUP save set and verifies that at least one copy of each block of the specified save set is readable or can be regenerated with OpenVMS BACKUP or by using the SSMgr COPY command.

The VALIDATE command differs from the OpenVMS BACKUP/VERIFY operation in that it does not compare the save set contents with the data on disk. It only verifies that the data in the save set is readable and internally consistent.

Any data integrity problems found during the VALIDATE operation, including unreadable data, CRC and XOR consistency errors, and regenerated blocks, are reported to the user.

Qualifiers

/ALL
/[NO]ALL (default)

Command qualifier
This qualifier specifies the behavior of Save Set Manager when save set names contain wildcards.

If ALL is specified, then all save sets matching the wildcarded file specification are processed.

If NOALL is specified, then only the first file name found that matches the wildcarded file specification is processed.

The qualifier is valid only for the VALIDATE and COPY functions.

The default is /NOALL.

/BRIEF (default)

Command qualifier
This qualifier specifies that a minimum level of detail is to be echoed. The SAVESET function and save set name(s) are echoed; any error information is displayed; and the result of the SAVESET operation is reported. See /FULL for a complete list of output information.

The default is /BRIEF.

/CHECKS=([NO]CRC,[NO]XOR)

Input save set qualifier

This qualifier specifies the optional consistency checks that will be performed on the input save set.

You must specify this qualifier immediately after the file name of the save set on the command line, as shown in Example 3–12. Example 3–12 also shows that when you specify both CRC and XOR checks, you must list them in parentheses with a comma separating them.

If CRC checking is specified and the input save set is written with CRC, then the CRC is computed for each block in the input save set. The CRC is compared against the CRC stored in the block header. If the input save set was not written with CRC, then this qualifier is ignored.

If XOR checking is specified and the input save set was written with XOR redundancy, then a running XOR is computed across each XOR group of blocks. The running XOR is compared to the XOR block. If the input save set was not written with XOR redundancy, then this qualifier is ignored.

Each of these options requires additional processing time and may impact the performance of SSMgr. Any CRC or XOR inconsistencies are written to the log file and a diagnostic message is returned to the calling program or user when processing completes.

The default is /CHECKS=(CRC,XOR).

/FULL

Command qualifier

This qualifier specifies that more detail is to be echoed. In addition to information displayed with the /BRIEF qualifier, the following information for each save set is displayed:

Save set name
Save set group size
Save set block size
Number of blocks in save set
Nondirectory user files in save set
Directory user files in save set
Alias nondirectory files in save set
Alias directory files in save set
Unrecoverable CRC errors
Recoverable CRC errors
Recoverable checksum errors
Unrecoverable checksum errors
XOR errors
Read errors
Write errors
Record errors
Unrecoverable missing blocks
Recoverable missing blocks

Rewritten blocks

The default is /BRIEF.

/JOURNAL[=*journal-file-name*]

Save set qualifier

This qualifier specifies whether SSMgr will create a journal file for the save set. The journal file is an OpenVMS BACKUP journal file with an OpenVMS file name. If no journal file name is specified, the journal file will be written to <saveset_name>.BJL in the current default directory.

If a journal file with the same name already exists, the new journal file is appended. If it does not already exist, a new journal file is created.

Journal files created by SSMgr will not list the labels of follow-on volumes in multivolume tape sets. Also, if COPY/IDENTICAL is used to copy a multivolume tape to a single volume tape, the journal file will still list it as a multivolume tape set.

These differences from BACKUP created journal files are due to a limitation in VMS that does not allow a non-privileged process to obtain the volume labels of a multivolume tape set.

You must specify this qualifier immediately after the save set file name specifier on the command line.

This qualifier is valid for all save set file name specifiers, input or output, on all SSMgr commands.

The default is /NOJOURNAL.

/LOG_FILE[=*logfile-name*] (default)
/NOLOG_FILE

Command qualifier

This qualifier causes SSMgr to write events to a log file specified by *logfile-name*. The *logfile-name* is an ASCII file with an OpenVMS file name. By default, a logfile is created for every execution of an SSMgr command. If not specified, the default log file name is SAVESET.LOG in the current default directory. You can suppress the creation of a default log file by using /NOLOG_FILE or by including /LOG_FILE=NL: on the command line. The following events are written to the SAVESET.LOG file:

- Each invocation of SSMgr
- All output returned to the user or calling program
- Any errors or warnings encountered while processing input save sets
- Operator requests

Wildcards may not be used in the specification of logfile names.

The default is /LOG_FILE=SAVESET.LOG.

/REWIND
/NOREWIND (default)

Input save set qualifier

For magnetic tape volumes only, REWIND directs SSMgr to rewind the magnetic tape to the beginning-of-tape (BOT) marker before reading the volume. This qualifier allows SSMgr to find save sets that are located before the current tape position.

You must specify this qualifier immediately after the applicable file name of the save set on the command line, as shown in Example 3–12.

The default for this qualifier is /NOREWIND, which causes SSMgr to start processing the tape from the current position. Any input save set that is located before the current position of the tape will not be found.

/TERMINAL=([NO]ERRORS, [NO]EVENTS, [NO]LOG)

Command qualifier

This qualifier specifies what class(es) of information should be displayed on the user's terminal during execution of the SAVESET command.

If ERRORS is specified, then all error conditions occurring during execution are displayed as they occur, in addition to being included in the final report at the end of command execution and in the log file.

If EVENTS is specified, then all nonerror event conditions occurring during execution (e.g., tape switches) are displayed as they occur.

If LOG is specified, then the names of all output user files (for COPY and MERGE) are displayed as they are written to the output save set. For VALIDATE, the names of all input user files are displayed as they are processed.

The default is /TERMINAL=(ERRORS,EVENTS,NOLOG).

Examples

Example 3–12 VALIDATE Command

```
$ saveset validate mkb200:rock.bck/checks=(crc,xor)/rewind
ERRORS terminal option enabled
EVENTS terminal option enabled
Rewinding MKB200: to beginning of volume
Rewind of MKB200: complete
Opening file MKB200:[]ROCK.BCK;1
Primary input save set MKB200:[]ROCK.BCK;1 opened

Save Set Manager V1.8 Time: 24-Dec-2004 19:17:42.17

    SAVESET function: VALIDATE
    Primary input save set: MKB200:[]ROCK.BCK;1
      No journal file
    Final status of each save set:

      Save set name:  MKB200:[]ROCK.BCK;1
        No errors detected
```

In Example 3–12, a save set named ROCK.BCK on MKB200 is validated with CRC checking and XORing. The tape is rewound before the validation begins. A brief report is generated upon completion of the command.

Example 3–13 VALIDATE with /Full Qualifier

```
$ saveset validate mkb200:rock.bck/full/rewind/checks=(nocrc,noxor)
ERRORS terminal option enabled
EVENTS terminal option enabled
Rewinding MKB200: to beginning of volume
Rewind of MKB200: complete
Opening file MKB200:[]ROCK.BCK;1
Primary input save set MKB200:[]ROCK.BCK;1 opened

Save Set Manager V1.8 Time: 24-Dec-2004 19:21:33.48

    SAVESET function: VALIDATE
    Primary input save set: MKB200:[]ROCK.BCK;1
      No journal file
    Final status of each save set:

      Save set name:  MKB200:[]ROCK.BCK;1
      Save set group size:                  10
      Save set block size:                  32256
      Blocks in save set:                   146
      Nondirectory files in save set:       140
      Directory files in save set:          17
      Alias nondirectory files in save set: 0
      Alias directories in save set:        0
      Unrecoverable CRC errors:             0
      Recoverable CRC errors:               0
      Unrecoverable checksum errors:        0
      Recoverable checksum errors:          0
      XOR errors:                           0
      Read errors:                          0
      Write errors:                         0
      Record errors:                        0
      Unrecoverable missing blocks:         0
      Recoverable missing blocks:           0
      Rewritten blocks:                     0
        No errors detected
```

Example 3–13 shows usage of the VALIDATE command to validate ROCK.BCK with full details provided in the report, without the optional CRC and XOR consistency checking, without a journal file, without rewinding the tape, and with ERRORS and EVENTS terminal classes enabled.

Example 3–14 VALIDATE All Save Sets in Directory

```
$ saveset validate/all playoffs.*/checks=crc
ERRORS terminal option enabled
EVENTS terminal option enabled
Primary input save set DISK$AVALANCHE:[STANLEY_CUP]PLAYOFFS.96;1
 opened

Save Set Manager V1.8 Time: 24-Dec-2004 19:31:19.04

    SAVESET function: VALIDATE
    Primary input save set: DISK$AVALANCHE:[STANLEY_CUP]PLAYOFFS.96;1
      No journal file

    Final status of each save set:
      Save set name:  DISK$AVALANCHE:[STANLEY_CUP]PLAYOFFS.96;1
        No errors detected

Primary input save set DISK$AVALANCHE:[STANLEY_CUP]PLAYOFFS.97;1
 opened

Save Set Manager V1.8 Time: 24-Dec-2004 19:31:23.24

    SAVESET function: VALIDATE
    Primary input save set: DISK$AVALANCHE:[STANLEY_CUP]PLAYOFFS.97;1
      No journal file

    Final status of each save set:

      Save set name:  DISK$AVALANCHE:[STANLEY_CUP]PLAYOFFS.97;1
        No errors detected

Primary input save set DISK$AVALANCHE:[STANLEY_CUP]PLAYOFFS.98;1
 opened

Save Set Manager V1.8 Time: 24-Dec-2004 19:31:23.24

    SAVESET function: VALIDATE
    Primary input save set: DISK$AVALANCHE:[STANLEY_CUP]PLAYOFFS.98;1
      No journal file

    Final status of each save set:

      Save set name:  DISK$AVALANCHE:[STANLEY_CUP]PLAYOFFS.98;1
        No errors detected

Primary input save set DISK$AVALANCHE:[STANLEY_CUP]PLAYOFFS.99;1
 opened

Save Set Manager V1.8 Time: 24-Dec-2004 19:31:25.02

    SAVESET function: VALIDATE
    Primary input save set: DISK$AVALANCHE:[STANLEY_CUP]PLAYOFFS.99;1
      No journal file

    Final status of each save set:

      Save set name:  DISK$AVALANCHE:[STANLEY_CUP]PLAYOFFS.99;1
        No errors detected


  Summary of VALIDATE operations

    Total operations attempted:             4
    Operations completing successfully:     4
```

(continued on next page)

Example 3–14 (Cont.) VALIDATE All Save Sets in Directory

Example 3–14 shows the validation of all of the save sets in the DISK$AVALANCE:[STANLEY_CUP] directory. CRC checking is performed on each save set, and a brief report is generated upon the completion of each save set, followed by a summary report of all the operations.

Example 3–15 VALIDATE with Journal Qualifier

```
$ saveset validate mkb500:SAVESET018/journal=kit.contents
ERRORS terminal option enabled
EVENTS terminal option enabled
Opening file MKB500:[]SAVESET018;1
Primary input save set MKB500:[]SAVESET018;1 opened

Save Set Manager V1.8 Time: 24-Dec-2004 19:31:23.24

    SAVESET function: VALIDATE
    Primary input save set: MKB500:[]SAVESET018;1
      Journal file: KIT.CONTENTS

    Final status of each save set:
      Save set name:  MKB500:[]SAVESET018;1
        No errors detected

$ backup/journal=kit.contents/list
Listing of BACKUP journal
Journal file DISK$PEAK:[PIKES]KIT.CONTENTS;1 on 24-May-2004 19:44:23.45
```

4
Interpreting SSMgr Reports

This chapter describes the completion and interim reports generated by SSMgr.

4.1 Monitoring the Progress of SSMgr Operations

While SSMgr is running, you can use Ctrl/T to monitor the progress of the SSMgr operation. Each time you enter Ctrl/T, SSMgr displays a progress report as shown in Example 4–1.

Example 4–1 Ctrl/T Report Example

```
1  JOE::SMITH 15:44:09 SAVESET$S CPU=00:19:00.12 PF=392532 IO=292279 MEM=1131
2   Current input file:    [SMITH]WS_LOGIN.COM;1
3   Input save set blocks:         5      Input files:     28
4   Output save set blocks:        4      Output files:    24
```

1 The first line is the standard OpenVMS Ctrl/T output.

2 Name of last user file read from the input save set.

3 Total number of save set blocks and user files that have been read from the primary input save set.

4 Total number of save set blocks and user files written to the output save set. This is not displayed during a VALIDATE operation.

Entering Ctrl/T before SSMgr has read the first file of the input save set will cause "None" to be displayed as the current input file. On MERGE operations involving an image save set and an incremental save set, SSMgr displays "None" in this field throughout the initial pass over the incremental save set.

4.2 Completion Reporting

When an SSMgr command terminates, the final status and/or a summary report is written to standard output. In most cases, this information also is written to a log file.

4.2.1 Normal Successful Completion

Every SSMgr operation that completes successfully issues a completion report that details any errors or anomalies found in the save sets. Refer to Example 4–2. If no errors were detected, the report has the following format:

Example 4–2 Normal Successful Completion Report

```
1   $ saveset copy mkb200:rock.bck stones.sav/identical/override
2   ERRORS terminal option enabled
    EVENTS terminal option enabled
3   Opening file MKB200:[]ROCK.BCK;1
    Primary input save set MKB200:[]ROCK.BCK;1 opened
    Output save set DISK$MUSIC:[WAVES]STONES.SAV; opened

4   Save Set Manager V1.8 Time: 24-Dec-2004 20:04:29.93


5     SAVESET function: COPY
      Primary input save set: MKB200:[]ROCK.BCK;1
        No journal file
      Output save set: DISK$MUSIC:[WAVES]STONES.SAV;
        No journal file

6     Final status of each save set:

      Save set name:  MKB200:[]ROCK.BCK;1
7     No errors detected

      Save set name:  DISK$MUSIC:[WAVES]STONES.SAV;
        No errors detected
```

1 Command as entered by the user.

2 Selected terminal options displayed.

3 Information about progress of command and save set names.

4 Report header.

5 Echo of the SSMgr command issued, separated into function, input save set, output save set and status of journal file.

6 Status of each save set processed, including the status of multiple output save sets, if any were requested.

7 No errors or anomalies, either recoverable or unrecoverable, were detected.

4.2.2 Successful Completion with Save Set Condition Report

If SSMgr found errors or anomalies in the input save set, but was still able to successfully complete the requested operation, then a save set condition report is produced. This report breaks out the number and type of errors detected, as well as a list of user files affected. An error can be either recoverable or unrecoverable. An error is recoverable if the save set was created with an XOR group size greater than zero, and there was no more than one error in an XOR group. The following types of errors and anomalies are broken out for each input save set:

- recoverable/unrecoverable CRC errors.
- recoverable/unrecoverable block header checksum errors.
- recoverable/unrecoverable missing blocks.
- XOR mismatch errors. The XOR block at the end of a group does not contain the XOR of all the blocks in the group. Note that XOR mismatch errors are only considered recoverable if CRC protection is present in the save set and all data blocks in the XOR group had correct CRC.
- Read errors.

- Record header errors. A record header within a save set block contains an invalid record type or invalid record length.
- Rewritten blocks. This is not an error, but a count of the number of blocks which were rewritten due to conditions such as bad spots on the media. If this number is large, you may be able to significantly reduce the size of your save set by using SSMgr to copy it to better media.

If recoverable errors were detected in the input save set, SSMgr also lists all files in the groups containing the errors. These files are still readable, but are no longer protected by XOR redundancy. If a second error were to develop within the same XOR group, user file data could be lost.

If your save set contains recoverable errors, HP recommends that you use the SSMgr COPY function to regenerate the XOR redundancy protection. If you are dealing with save sets on media that is very old or has been improperly stored, HP recommends that you copy the save set with the SAVESET COPY command with the /IDENTICAL and /OVERRIDE qualifers to another medium, then SAVESET COPY the save set to a final medium with the /CRC and /GROUP qualifiers. This procedure accomplishes the copy operation without the risk of running a second pass over the suspect media.

Example 4–3 Save Set Condition Report

```
$ saveset merge zap4.sav zapcrc.sav foo.sav
  ERRORS terminal option enabled
  EVENTS terminal option enabled
  Opening file DKB200:[ENGINEER]ZAP4.SAV;2
  Primary input save set DKB200:[ENGINEER]ZAP4.SAV;2 opened
  Opening file DISK$100:[ENGINEER]ZAPCRC.SAV;1
  Secondary input save set DISK$100:[ENGINEER]ZAPCRC.SAV;1 opened
  Opening file DISK$100:[ENGINEER]FOO.SAV;
  Output save set DISK$100:[ENGINEER]FOO.SAV; opened
  Resetting to beginning of file DISK$100:[ENGINEER]ZAPCRC.SAV;1
  File DISK$100:[ENGINEER]ZAPCRC.SAV;1 has been reset to beginning of file
  Save Set Manager V1.8 Time: 24-Dec-2004 20:38:36.57

    SAVESET function: MERGE
    Primary input save set: ZAP4.SAV
    Secondary input save set: ZAPCRC.SAV
    Output save set: FOO.SAV

1 Save set name:  DISK$100:[ENGINEER.SSM.SRC.TMP]ZAP4.SAV;

2   Recoverable checksum errors:          1

3 User files affected by error(s):
        [ENGINEER.SSM.VAXV60.OBJ]CONDITION_HANDLER.OBJ;3
        [ENGINEER.SSM.VAXV60.OBJ]CONDITION_HANDLER.OBJ;2
        [ENGINEER.SSM.VAXV60.OBJ]CONDITION_HANDLER.OBJ;1
        [ENGINEER.SSM.VAXV60.OBJ]COPY.OBJ;25

4 Save set name:  DISK$100:[ENGINEER.SSM.SRC.TMP]ZAPCRC.SAV;

5   Recoverable CRC errors:               1

6 User files affected by error(s):
        [ENGINEER]BATCH.COM;26
        [ENGINEER]BL_SETUP.COM;15
        [ENGINEER]BL_SETUP.COM;14
        [ENGINEER]BL_SETUP.COM;13
        [ENGINEER]BYE.COM;2
        [ENGINEER]EVEPLUS.COM;6
        [ENGINEER]FRED.COM;1
        [ENGINEER]INIT$LSE.COM;3
        [ENGINEER]INIT$LSE.COM;2
        [ENGINEER]INIT$LSE.COM;1
        [ENGINEER]LN03.COM;4
        [ENGINEER]LOGIN.COM;6
        [ENGINEER]LOGIN.COM;5
        [ENGINEER]LOGIN.COM;4
        [ENGINEER]MEDIA.COM;1
        [ENGINEER]NEWEVE.COM;5
        [ENGINEER]NOTES$EDIT.COM;2
        [ENGINEER]NOTUP.COM;2
        [ENGINEER]PAGE_COUNT.COM;9
        [ENGINEER]PDC_ORDER.COM;3
        [ENGINEER]RAIDEV_GKSTARTUP.COM;1
        [ENGINEER]READ_DEMO.COM;13
        [ENGINEER]SETDEF.COM;3
        [ENGINEER]STOCK.COM;2
        [ENGINEER]TEMP.COM;2
        [ENGINEER]UUCP.COM;1
        [ENGINEER]WAS_LOGIN.COM;2
        [ENGINEER]WS_LOGIN.COM;1
```

(continued on next page)

Example 4–3 (Cont.) Save Set Condition Report

1 Name of first input save set containing errors/anomalies.

2 Summary of errors found in first save set.

3 List of user files in first input save set that are no longer protected by XOR redundancy.

4 Name of second input save set containing errors/anomalies.

5 Summary of errors found in second input save set.

6 List of user files in second input save set that are no longer protected by XOR redundancy.

4.2.3 Error Reporting

In the case of an SSMgr MERGE or COPY operation (except COPY/IDENTICAL /OVERRIDE), SSMgr cannot continue after detecting an unrecoverable error. If this occurs, SSMgr terminates with an error message as shown in Example 4–4.

Example 4–4 Error Report

```
$ saveset copy zap2.bck zcopy.bck
ERRORS terminal option enabled
EVENTS terminal option enabled
Primary input save set DISK$USER:[SOFTWARE]ZAP2.BCK;1 opened
Output save set DISK$USER:[SOFTWARE]ZCOPY.BCK; opened
CRC error: block number 50 in save set DISK$USER:[SOFTWARE]ZAP2.BCK;1
Block recovered from XOR in save set DISK$USER:[SOFTWARE]ZAP2.BCK;1

  Save Set Manager V1.8 Time: 24-Dec-2004 20:16:28.18

    SAVESET function: COPY
    Primary input save set: DISK$USER:[SOFTWARE]ZAP2.BCK;1
      No journal file
    Output save set: DISK$USER:[SOFTWARE]ZCOPY.BCK;
      No journal file

    Final status of each save set:

      Save set name:  DISK$USER:[SOFTWARE]ZAP2.BCK;1
      Recoverable CRC errors:              1

      User files affected by error(s):
        [SOFTWARE.SAVESET018]SAVESET_USER_GUIDE.PS;17

      Save set name:  DISK$USER:[SOFTWARE]ZCOPY.BCK;
        No errors detected
```

4.2.4 Log File

In addition to standard output, SSMgr messages and reports also are sent to a log file as shown in Example 4–5. Each log file begins with a diagnostic message containing the time the SSMgr command was issued, followed by a report or error message identical to that sent to standard output. The default log file name is SAVESET.LOG, written to the user's default directory. See the /LOGFILE= qualifier to change the log file name or supress creating the log file.

Example 4–5 Log File

```
$ saveset copy ssm.bck copy.sav/identical/override

Log file:

ERRORS terminal option enabled
EVENTS terminal option enabled
Primary input save set DISK$ENGINEER:[SOFTWARE]SSM.BCK;1 opened
Output save set DISK$ENGINEER:[SOFTWARE]COPY.SAV; opened
CRC error: block number 15 in save set DISK$ENGINEER:[SOFTWARE]SSM.BCK;1
Recovery unsuccessful in save set DISK$ENGINEER:[SOFTWARE]SSM.BCK;1
%SAVESET-I-BADCRC, Data CRC error in file DISK$ENGINEER:[SOFTWARE]SSM.BCK;1
XOR error at block 26 in save set DISK$ENGINEER:[SOFTWARE]SSM.BCK;1
CRC error: block number 45 in save set DISK$ENGINEER:[SOFTWARE]SSM.BCK;1
Recovery unsuccessful in save set DISK$ENGINEER:[SOFTWARE]SSM.BCK;1
%SAVESET-I-BADCRC, Data CRC error in file DISK$ENGINEER:[SOFTWARE]SSM.BCK;1
XOR error at block 52 in save set DISK$ENGINEER:[SOFTWARE]SSM.BCK;1
CRC error: block number 68 in save set DISK$ENGINEER:[SOFTWARE]SSM.BCK;1
Recovery unsuccessful in save set DISK$ENGINEER:[SOFTWARE]SSM.BCK;1
%SAVESET-I-BADCRC, Data CRC error in file DISK$ENGINEER:[SOFTWARE]SSM.BCK;1
XOR error at block 78 in save set DISK$ENGINEER:[SOFTWARE]SSM.BCK;1

  Save Set Manager V1.8 Time: 24-Dec-2004 20:22:12.74

    SAVESET function: COPY
    Primary input save set: DISK$ENGINEER:[SOFTWARE]SSM.BCK;1
      No journal file
    Output save set: DISK$ENGINEER:[SOFTWARE]COPY.SAV;
      No journal file

    Final status of each save set:

      Save set name:  DISK$ENGINEER:[SOFTWARE]SSM.BCK;1
      Unrecoverable CRC errors:             3
      XOR errors:                           3

      Save set name:  DISK$ENGINEER:[SOFTWARE]COPY.SAV;
        No errors detected
```

A new log file is created for each invocation of SSMgr.

5
SSMgr Messages

This section contains descriptions and user actions for messages issued by SSMgr.

5.1 SSMgr INFORMATIONAL Level Messages

NOLOGFILE, Unable to create log file. Continuing operation.

Explanation: Attempt to create log file failed. The reason for the failure follows this message. Operation continues without logfile.

Action: Correct problem indicated by secondary message.

NOLOGFILEWRITE, Unable to write log file. Continuing operation.

Explanation: Attempt to write to the log file failed. The reason for the failure follows this message. Operation continues without logfile.

Action: Correct problem indicated by secondary message.

5.2 SSMgr ERROR Level Messages

BADCHECKSUM, Block header checksum error in file *filename*

Explanation: An unrecoverable error was found in a save set block header. The data contained in the block is not reliable.

Action: Use OpenVMS BACKUP to restore the recoverable portion of the save set.

BADCRC, Data CRC error in file *filename*

Explanation: An unrecoverable error was found in a save set block CRC. The data contained in the block is not reliable.

Action: Use OpenVMS BACKUP to restore the recoverable portion of the save set.

BADXOR, XOR error in file *filename*

Explanation: The XOR block of a save set group in save set *filename* does not contain the valid XOR of the data blocks in that group and CRC protection is not present. The data in the entire group is unreliable.

Action: Use OpenVMS BACKUP to restore the recoverable portion of the save set.

INVSAVESET, *filename* is not a valid saveset

Explanation: Save set *filename* is not a valid OpenVMS save set.

Action: None.

MISSINGBLK, Missing block could not be regenerated in file *filename*

Explanation: There was an unrecoverable missing block in save set *filename* and there was insufficient metadata to reproduce the data in the block.

Action: None. The data in the missing block is lost.

5.3 SSMgr FATAL Level Error Messages

BUG, A software bug was detected at line *line*, file *file*

Explanation: SSMgr detected an internal bug.

Action: Please submit an SPR and copies of the log file and the input save sets. If this bug prevents you from copying the input save sets, submit the output of BACKUP/LIST on those save sets instead.

DEVACCESS, Cannot access device *devicename*

Explanation: Attempts to access a save set on device *devicename* have failed.

Action: Verify that the path name given by the command line is correct, that the device is mounted, and that you have privileges to access the device.

DUPJNLFIL, Duplicate journal file name: *filename*

Explanation: For any single SAVESET command with more than one journal file being created, each journal file name must be unique.

Action: Verify that the save set file names used are unique.

FAILURE, SAVESET operation was unsuccessful

Explanation: The SAVESET operation could not complete. Additional messages follow indicating the nature of the failure.

Action: Base user action on the immediately following messages.

FILECLOSE, could not close file *filename*

Explanation: SSMgr could not close the save set with *filename*. The reason for the failure appears immediately after this message.

Action: Dependent on the reason for failure.

FILEOPEN, could not open file *filename*

Explanation: SSMgr could not open the file with *filename*. The reason for the failure appears immediately after this message.

Action: Dependent on the reason for failure.

FILEREAD, error reading file *filename*

Explanation: SSMgr encountered and error reading file *filename*. The reason for the failure appears immediately after this message.

Action: Dependent on the reason for failure.

FILERESET, could not reset to beginning of file *filename*

Explanation: SSMgr could not reset to the beginning of the file *filename*. The reason for the failure appears immediately after this message.

Action: Dependent on the reason for failure.

FILEWRITE, error writing file *filename*

Explanation: SSMgr encountered an error writing file *filename*. The reason for the failure appears immediately after this message.

Action: Dependent on the reason for failure.

IDENTCOPYFAIL, Identical copy of *filename* failed

Explanation: The user specified /IDENTICAL on the SAVESET COPY command line, and the operation failed. The reason for the failure appears immediately after this message.

Action: Dependent on the reason for failure.

INSUFSPACE, Insufficient space for *filename* to be allocated

Explanation: There is insufficient space on the target output disk device to allow allocation of a file the size required by the operation.

Action: Specify a device that has sufficient space to hold the output file.

INVADDRESS, Invalid address in save set record in save set *filename*

Explanation: There is a nonzero value in address field of record header in non-LBN/VBN record. The associated data is therefore suspect.

Action: Use OpenVMS BACKUP to restore the save set and check for data corruption.

INVBLKSIZE, Invalid block size. Valid range: 2048—65535

Explanation: Save set blocks may be no smaller than 2048 bytes and no larger than 65535 bytes.

Action: Reenter the SAVESET command and specify a /BLOCK_SIZE value in the range of 2048 to 65535 bytes.

INVDEVCLASS, Could not open *devicename*. Device must be disk or tape

Explanation: SSMgr operates on disk and tape devices only. The device specified by *devicename* was not a disk or tape device.

Action: None.

INVFILATTR, Invalid file attribute in save set file record

Explanation: An invalid file attribute is stored in a save set file record, indicating a corrupt or otherwise invalid save set record.

Action: Re-enter the SAVESET command with a /GROUP_SIZE value in the range 0 - 100.

INVGRPSIZE, Invalid group size. Valid range: 0—100

Explanation: Save set XOR groups may be no larger than 100. A value of zero indicates no XOR should be written to the save set.

Action: Reenter the SAVESET command with a GROUP_SIZE value in the range 0 to 100.

INVRECSIZE, Invalid save set record size in save set *filename*

Explanation: A save set record contains invalid metadata.

Action: Use OpenVMS BACKUP to restore the save set and check for data corruption.

INVRECTYPE, Invalid save set record type in save set *filename*

Explanation: A save set record contains invalid metadata.

Action: Use OpenVMS BACKUP to restore the save set and check for data corruption.

INVOUTWC, Invalid output save set wild card specification

Explanation: The only legal wildcarding of the 'name' or 'type' portions of an output save set file name specification is a single '*'. Use of the '%' wildcard or combining the '*' with any other characters in the specification is illegal.

Action: Re-enter the SAVESET command with a legal wildcard specification or with a nonwildcard specification.

JNLOPNERR, Error opening journal file: *filename*

Explanation: The journal file specified could not be opened successfully.

Action: Ensure that the file name is correct, that you have write access to the device, and that the device is not full.

JNLWRTERR, Error writing to Journal file: *filename*

Explanation: The journal file specified could not be written to successfully.

Action: Ensure that you have write access to the device, and that the device is not full.

MEMALLOC, Memory allocation failure

Explanation: SSMgr was unable to allocate necessary memory.

Action: Ensure sufficient resources for SSMgr operaton.

NODEVNAM, No device name in save set specifier *specifier*

Explanation: No device name was specified in a context in which an explicit device name is required.

Action: Specify the device name with the save set specifier.

NOMERGEPHYS, *AZ*, is a physical save set, invalid as input to MERGE

Explanation: MERGE operations cannot be performed on physical backup save sets.

Action: None.

NOMULTITAPE, COPY/IDENTICAL output save set may not span multiple tapes

Explanation: End of tape (EOT) on the output tape was encountered during a SAVESET COPY/IDENTICAL operation. This operation requires that the copied save set does not span multiple tapes.

Action: Use an output tape that has sufficient space for the save set or use the SAVESET COPY command without the /IDENTICAL qualifier.

NONLOCALACC, *device* must be accessed locally

Explanation: The save set must be accessed on a local device.

Action: Use only save sets that do not require network access.

REWINDFAIL, Could not rewind *device*

Explanation: Attempts to rewind the specified device failed.

Action: This error message is always accompanied by a secondary system message. Correct the problem identified by that secondary message.

SECNOTINCR, Secondary save set not an incremental

Explanation: If the primary input save set in a SAVESET MERGE operation is an image save set, then the secondary input save set must be an incremental save set. See Table 3–1 for a list of valid save set combinations for MERGE.

Action: Retry the SAVESET MERGE operation with the appropriate secondary input save set.

TOOMANYERRS, Too many errors reading save set *filename*. Giving up.

Explanation: Too many errors were encountered while reading the save set. Probably a problem with the drive or media.

Action: If the problem is with the drive, try mounting the media on a different drive.

UNSRECTYPE, Unsupported save set record type in save set *filename*

Explanation: The save set contains a record type inappropriate for the SAVESET operation being performed.

Action: Check that the save set being used is appropriate.

5.4 SSMgr WARNING level Error Mesages

NOUSERFILES, No user files written to output save set

Explanation: The output save set is empty: i.e. no user files were written to the output save set.

Action: Ensure that this is what was expected.

5.5 Terminal Messages

The following informational messages are written to SYS$OUTPUT, SYS$ERROR, and the logfile based on the setting of /TERMINAL options. See the /TERMINAL qualifier descriptions for each of the commands in Chapter 3.

5.5.1 ERRORS Option Terminal Messages

Block recovered from XOR in save set *filename*

Explanation: The ERRORS terminal option is enabled. An unreadable block in save set *filename* was recovered via XOR.

Block recovered via read-ahead in save set *filename*

Explanation: The ERRORS terminal option is enabled. The error specified immediately above was recovered via read-ahead.

Checksum error: block number *number* in save set *filename*

Explanation: The ERRORS terminal option is enabled. A checksum error was detected in block number *number* in save set *filename*.

Current volume is not the next volume in this set

Explanation: The ERRORS terminal option is enabled. While reading a multivolume tape set, the SSMgr has determined that the currently loaded volume is not the next member of the volume set.

CRC error: block number *number* in save set *filename*

Explanation: The ERRORS terminal option is enabled. A CRC error was detected in block number *number* in save set *filename*.

Error recovery successful in save set *filename*

Explanation: The ERRORS terminal option is enabled. The error specified immediately above was successfully recovered by SSMgr.

ERRORS terminal option enabled

Explanation: The ERRORS terminal option is enabled for this SAVESET command.

Missing Block: number *number* in save set *filename*

Explanation: The ERRORS terminal option is enabled. A missing block was detected in block number *number* in save set *filename*.

Read error: block number *number* in save set *filename*

Explanation: The ERRORS terminal option is enabled. A read error was detected in block number *number* in save set *filename*.

Recovery unsuccessful in save set *filename*

Explanation: The ERRORS terminal option is enabled. The error specified immediately above was not recoverable.

Rewritten Block: number *number* in save set *filename*

Explanation: The ERRORS terminal option is enabled. A rewritten block error was detected in block number *number* in save set *filename*.

XOR error at block *block-number* in save set *filename*

Explanation: The ERRORS terminal option is enabled. An XOR error was detected in block number *number* in save set *filename*.

5.5.2 EVENTS Option Terminal Messages

EVENTS terminal option enabled

Explanation: The EVENTS terminal option is enabled for this SAVESET command.

Opening file *filename*

Explanation: The EVENTS terminal option is enabled. SSMgr is attempting to open the file *filename*.

Output save set: *filename*

Explanation: The EVENTS terminal option is enabled. The file *filename* has been successfully opened for use as the output save set.

Primary input save set: *filename*

Explanation: The EVENTS terminal option is enabled. The file *filename* has been successfully opened for use as the primary input save set.

Resetting to beginning of file *filename*

Explanation: The EVENTS terminal option is enabled. The merge of an image save set with an incremental save set requires two passes over the incremental save set. This message indicates that SSMgr is resetting the secondary input save set for the second pass.

Resuming operation on volume *volume-number* of save set *filename*

Explanation: The EVENTS terminal option is enabled. The save set *filename* is on a multivolume tape and operation is resuming on volume number *volume-number*.

Rewind of *filename* complete

Explanation: The EVENTS terminal option is enabled. /REWIND was specified and the tape has been successfully rewound to BOT.

Rewinding *filename* to beginning of volume

Explanation: The EVENTS terminal option is enabled. /REWIND was specified and the tape is being rewound.

Scanning secondary input save set

Explanation: ... EVENTS ...enabled. An incremental save set is being merged with an image save set. SSMgr is scanning the incremental save set for file system information.

Secondary input save set: *filename*

Explanation: The EVENTS terminal option is enabled. The file *filename* has been successfully opened for use as the secondary input save set.

Unable to continue writing to save set *filename*. Continuing operation.

Explanation: The EVENTS terminal option is enabled. SSMgr could not write to one of the multiple output save sets. SSMgr will continue the COPY or MERGE operation, writing to the remaining output save sets.

5.5.3 LOG Option Terminal Messages

LOG terminal option enabled

Explanation: The LOG terminal option is enabled for this SAVSESET command.

Processing user file: *filename*

Explanation: The LOG terminal option is enabled. The user file *filename* is currently being processed.

5.6 Process Quota Exceeded (System Message)

If you see the "Process Quota Exceeded" system message, the process quota exceeded is most likely PGFLQUO or DIOLM. You must rerun the SSMgr operation in a process with sufficient quota to complete the SSMgr operation.

Index