VERITAS Storage Migrator™ 3.4
VERITAS Storage Migrator Remote™ 3.4

System Administrator's Guide

UNIX

June 2000
100-001497

VERITAS

Disclaimer

The information contained in this publication is subject to change without notice. VERITAS Software Corporation makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. VERITAS Software Corporation shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this manual.

Copyright

Printed in the USA, June 2000.

VERITAS Software Corporation.
1600 Plymouth St.
Mountain View, CA 94043
Phone 650.335.8000
Fax 650.335.8050
http://www.veritas.com

Contents

Figures

Tables

Preface

Introduction

This guide describes how to configure and manage VERITAS Storage Migrator and VERITAS Storage Migrator Remote, and is intended for administrators who have a good working knowledge of the UNIX operating system.

Except where otherwise noted, all information in this manual applies to both VERITAS Storage Migrator and VERITAS Storage Migrator Remote, and refers to them collectively as *VSM* (VERITAS Storage Manager). Differences are indicated by specific references to VERITAS Storage Migrator or VERITAS Storage Migrator Remote. See the release notes for specific information on the hardware and operating systems to which this software applies.

Note This release of VSM uses two types of graphical user interfaces: Java-based and Motif-based. The Motif-based GUI (`xhsmadm`) will not be supported in future releases of VSM.

Note In this publication, the term *Media Manager* refers to the media-management software that is part of VSM. The term *volume* refers to removable storage media, either tape or optical disc.

Audience

This guide is intended for system administrators responsible for configuring and maintaining VSM systems using UNIX.

This guide assumes:

- A basic understanding of system administration
- A good working knowledge of the UNIX operating system
- A basic understanding of storage management principles

Scope

The purpose of this guide is to explain how to configure, maintain, and use VERITAS Storage Migrator and VERITAS Storage Migrator Remote.

Organization

This guide is organized as follows:

- Chapter 1, “About VSM” on page 1, is an overview of VSM. Read this chapter first to get an idea of what these products do.
- Chapter 2, “Planning VSM Configuration” on page 51, provides information useful in developing a migration policy for your site. Read this chapter before configuring VSM.
- Chapter 3, “Configuring VSM (Motif-based GUI)” on page 121, describes the steps to perform in configuring VSM using the Motif-based GUI, xhsmadm. Read this chapter before configuring VSM.
- Chapter 4, “Configuring VSM (Java-based GUI)” on page 179, describes the steps to perform in configuring VSM using the Java-based GUI, VSM-Java. Read this chapter before configuring VSM.
- Chapter 5, “The VSM Activity Monitor” on page 235, describes how to use the job monitor, VSM Activity Monitor, after you have VSM configured and running. Read this chapter after configuring VSM.
- Chapter 6, “The VSM File Browser” on page 249, describes how to use the end-user GUI, VSM File Browser, after you have VSM configured and running. Read this chapter after configuring VSM.
- Chapter 7, “Managing VSM” on page 259, contains topics related to managing operations after you have VSM configured and running. Read this chapter after configuring VSM.

In addition to these chapters, there are several additional tools for your reference and convenience:

- Appendix A,“Man Pages” on page 325, contains man pages for commands that relate specifically to VSM. You can also use the `man` command to view these command descriptions online.
- Appendix B, “Command Requirements” on page 521, contains blank configuration worksheets suitable for duplication.
- Appendix C, “Planning Worksheets” on page 525, contains blank configuration worksheets suitable for duplication when planning your VSM configuration. Read this chapter before configuring VSM.

- Appendix D, “Setting Up a Schedule” on page 537, offers complete information on the VSM File Browser Scheduling interface.
- The glossary, on page 575, offers a complete list of VSM terms.
- The index, on page page 585, offers an alphabetic subject index helps you easily find the information you need.

Using This Guide

The following are guidelines for using this document:

System Administrators

Read the chapters in numerical order. Careful planning, as outlined in Chapter 2, will make the configuration procedures described in Chapters 3 and 4 easier. Chapter 5 contains helpful information once the system is running in production.

Users

Information about those migration controls not requiring root privileges are described in the *VERITAS Storage Migrator User’s Guide*.

Related Documents

- The *VERITAS Storage Migrator Release Notes - UNIX* provides important information such as the platforms and operating systems that are supported, new features, and problems fixed since the last release.
- The *VERITAS Storage Migrator Installation Guide - UNIX* describes how to install VERITAS Storage Migrator and VERITAS Storage Migrator Remote, and includes some operational notes that may not be in the other manuals.
- The *VERITAS Storage Migrator User’s Guide - UNIX* describes migration operations not requiring root privileges, and the end user interface.

Note The following manuals are applicable to VERITAS Storage Migrator, but not to VERITAS Storage Migrator Remote:

- The *VERITAS NetBackup Release Notes - UNIX* provides important information about NetBackup and Media Manager such as supported platforms, operating systems, and peripheral storage equipment.
- The *VERITAS NetBackup Installation Guide - UNIX* describes how to install VERITAS NetBackup.

- The *Media Manager Device Configuration Guide* describes how to configure storage devices controlled by Media Manager.
- The *Media Manager System Administrator's Guide - UNIX* describes Media Manager, its components, and how they are used to manage media volumes, drives, and robots.

Conventions

The following explains typographical and other conventions used in this guide.

Type Style

Table 1. Typographic Conventions

Typeface	Usage
`Bold fixed width`	Input. For example, type **`cd`** to change directories.
`Fixed width`	Paths, commands, filenames, or output. For example: The default installation directory is `/opt/VRTSxx`.
Italics	Book titles, new terms, or used for emphasis. For example: *Do not* ignore cautions.
Sans serif (italics)	Placeholder text or variables. For example: Replace *filename* with the name of your file.
Sans serif (no italics)	Graphical user interface (GUI) objects, such as fields, menu choices, etc. For example: Enter your password in the Password field.

Notes and Cautions

Note This is a Note and is used to call attention to information that makes it easier to use the product or helps you to avoid problems.

Caution This is a Caution and is used to warn you about situations that can cause data loss.

Key Combinations

Some keyboard command sequences use two or more keys at the same time. For example, you may have to hold down the Ctrl key before you press another key. When this type of command is referenced, the keys are connected by plus signs. For example:

Press Ctrl+t

Command Usage

The following conventions are frequently used in the synopsis of command usage.

brackets [ ]

The enclosed command line component is optional.

Vertical bar or pipe (|)

Separates optional arguments from which the user can choose. For example, when a command has the following format:

`command` *arg1*|*arg2*

the user can use either the *arg1* or *arg2* variable.

Getting Help

- For updated information about this product, including system requirements, supported platforms, supported peripherals, and a list of current patches available from Technical Support, visit our web site:

  `http://www.veritas.com/products/vhsm/`

- For product assistance, contact VERITAS Customer Support.

  US and Canadian Customers: 1-800-342-0652

  International Customers: +1 (650) 335-8555

- VERITAS Customer Support can also be reached through electronic mail at:

  `support@veritas.com`

About VSM 1

VERITAS Storage Migrator and VERITAS Storage Migrator Remote are hierarchical storage management products that increase the amount of file space available to users by migrating files from a local UNIX file system to secondary storage (such as a tape, optical disc, or another magnetic disk) as space is needed in the local file system. When a user accesses a migrated file, it is automatically retrieved from secondary storage and cached in the online file system. Except for the delay to perform the retrieval, users and programs are unaware that file migration and retrieval are taking place.

Note Except where otherwise noted, *VSM* (VERITAS Storage Manager) refers to both VERITAS Storage Migrator and VERITAS Storage Migrator Remote. Differences are indicated by specific references to VERITAS Storage Migrator or VERITAS Storage Migrator Remote.

VSM implementations fall into two groups: kernel-based implementations or nonkernel-based implementations. Kernel-based implementations use a parallel inode (`.PAIN`) file. Nonkernel-based implementations (the Data Management Application Programming Interface, DMAPI) use an inode-to-handle (`.IHAND`) file. These implementations are platform-dependent, and any differences between them are noted in the documentation. Refer to your Release Notes for a detailed breakdown of which implementation applies to your system.

VERITAS Storage Migrator and VERITAS Storage Migrator Remote are very similar, both in design and function. They both use the same techniques for managing the local file system and for most administrative procedures. The main difference between them is the secondary storage methods each supports:

- VERITAS Storage Migrator uses directly connected tape, optical disc, or magnetic disk devices as well as magnetic disk file systems on remote volume servers for secondary storage. Media Manager provides the interface to the tape and optical storage devices. Support for large-capacity library devices with robotic access mechanisms eliminates the need for operator action to either migrate or cache files. The net result is apparently unlimited online storage but at a lower cost per megabyte because the extra storage is on lower-cost media such as cartridge tape or optical disc.

VERITAS Storage Migrator offers several methods for secondary storage. They are disk file for premigration (`dk`), alternate magnetic disk (`ad`), three tape methods (`ct`, `dt`, and `mt`), optical disc as tape with random seek (`op` and `ow`), remote using ftp (`ft`), and integration with VERITAS NetBackup (`nb`).

VERITAS Storage Migrator is able to share secondary storage devices with VERITAS NetBackup through the use of a common Media Manager.

- VERITAS Storage Migrator Remote is a subset of VERITAS Storage Migrator. It uses only four methods: disk file for premigration (`dk`), alternate magnetic disk (`ad`), remote using ftp (`ft`), and NetBackup (`nb`). The `ad` method can be used to migrate files to a remote file system by using NFS. Using the `ft` method you can combine a local UNIX file system with one or more remote file systems to create the impression of one large *virtual* file system. The `nb` method migrates files using VERITAS NetBackup.

  The remote volume servers for the `ft` method can be from a variety of different vendors and have different capacities. Files are migrated and retrieved by using standard ftp commands to access a remote file system.

  Neither the tape nor optical methods are available with VERITAS Storage Migrator Remote. Media Manager is required only by VERITAS Storage Migrator and VERITAS NetBackup, and not by VERITAS Storage Migrator Remote.

Note Except where otherwise noted, the information in this guide applies to both VERITAS Storage Migrator and VERITAS Storage Migrator Remote and refers to them collectively as *VSM* (Hierarchical Storage Manager). Differences are indicated by specific references to VERITAS Storage Migrator or VERITAS Storage Migrator Remote.

The *managed server* in a VSM configuration can play three different roles:

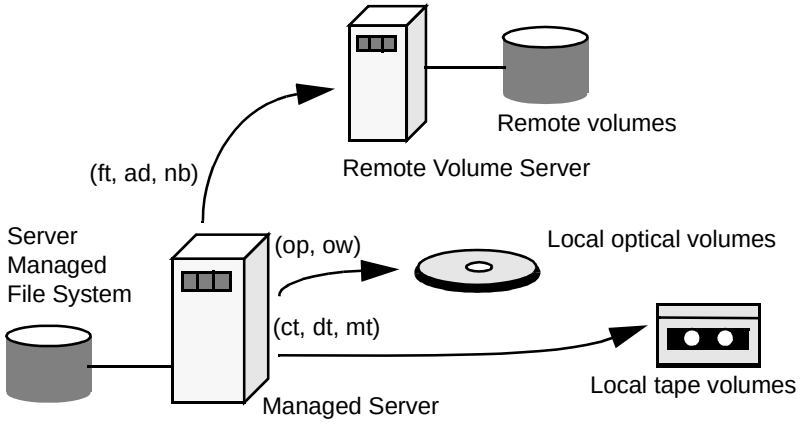
- It is the server on which one or more VSM managed file systems reside. This is the platform where the server component of VSM executes.
- It can also be the server that communicates with a *remote volume server*. Communication between the two servers is either via NFS (`ad` method) or by ftp (`ft` method).

A *remote volume server* refers to the server on which a remote volume used by VSM resides. A remote `ad` volume is NFS mounted on the *managed server*.

A *client workstation* is a system with one or more attached file systems.

In the example of a basic VSM configuration shown in Figure 1, a VSM managed file system resides on the *managed server*. The *managed server* migrates files to local optical and tape volumes. In this example the *managed server* also migrates files to remote volumes on a *remote volume server*. (A second copy of VERITAS Storage Migrator can be running on the *remote volume server*.)

Figure 1. Basic VSM Configuration Terminology



There are two main steps in the migration process. In the first step, VSM selects files according to predefined selection criteria and premigrates them within the file system. In the second step, VSM copies the files to one or more secondary storage devices. However, the premigrated files remain available on the disk.

When the file system reaches a predefined high-water mark or becomes full, VSM automatically purges premigrated files that have been copied to secondary storage, thus providing additional file space. The file name and its attributes still remain in the user's directory and information about each migrated file (size, number of copies, location, and type of media for each copy) resides in a database on the server disk.

If a user accesses a migrated file, VSM makes it available by caching the data back to disk. a VSM managed file system can be accessed by using FTP or NFS. After VSM caches migrated files back in the file system, FTP or NFS transfers occur as if the file had never been migrated.

VSM can also use an NFS mounted file system as a secondary storage device. Although you can NFS mount a VSM partition, VSM cannot manage an NFS mounted file system.

Caution It is recommended practice to mount NFS file systems in a dedicated subdirectory of the root directory, neither in the root directory itself nor in any other location. VSM and other applications perform `stat()` operations on entries in the root directory to determine the path to the current working directory. This operation can block on an NFS mount point in root if the NFS server is unreachable.

The remaining topics in this chapter provide an overview of the tasks that VSM administrators and users can perform. There is also a functional overview that provides a more detailed explanation of how VSM operates.

Administrative Controls

The VSM administrator configures and manages the operation of VSM. The administrator can choose the file systems that VSM manages and tailor VSM to meet the migration requirements of those file systems.

For example, with VERITAS Storage Migrator, small and frequently accessed files can be migrated to an optical disc. If files are very large or infrequently accessed, they can be migrated to tape.

Areas that the administrator can configure for each file system include:

- Space thresholds at which VSM starts and stops migration.
- Criteria that VSM uses when selecting individual files to migrate.
- Criteria that VSM uses when selecting individual files to purge from disk.
- Media on which VSM writes the selected files. Also, how data is written on that media (for example, block sizes). The choice of media also implies the device.
- Number of copies to make of each migrated file, and which of those copies to use for caching.
- Registration of volumes on which to store each copy of the files.
- Whether to use concurrent recording to speed up migrations (when multiple devices are available).
- Quota for the amount of space a user can restrict from migration.
- Number of migration levels to use.

After configuration, the VSM administrator can initiate and control file migrations to manage disk space either by manual commands that the administrator can execute directly or by using `crontab` entries to schedule automatic migrations. Commands to start and stop VSM can also be added to startup scripts. By using these techniques, an administrator can set up VSM to operate automatically, requiring little or no action outside of exception conditions.

Migration controls include:

- Making space available before it is needed by selecting and copying files to secondary storage but leaving a copy the premigrated data online. Premigrated files remain available on disk unless free space falls below configured limits and VSM has to purge (delete) some of them to make space available.
- Checking file system free space and keeping it within configured limits. If space is below configured limits, VSM purges premigrated files or migrates additional files as necessary to provide enough space.
- Activating and deactivating VSM configured file systems.

- Saving and reinitializing VSM log files to conserve disk space.
- Listing which files you always want to migrate in a global migrate file, and listing which files you never want to migrate in a global stop file.
- Enabling constant sweeping of the file system to select migration candidates.

VSM also offers a comprehensive set of tools for managing the secondary media. These management capabilities include:

- Registering media for use with VSM.
- Consolidating volumes to recover obsolete space.
- Listing database information about all VSM volumes.
- Scanning volumes and displaying information about their contents.
- Displaying the location of a migrated file.
- Validating database consistency.
- Restoring lost files.
- Reconstructing lost VSM databases.
- Moving migrated files to a new volume set.
- Exporting migrated files (and volumes) to another VSM managed file system.
- Specifying how often file marks are written to tape.

See chapter 6 of this manual for additional information on how to manage VSM.

User Controls

The administrator can provide users some control over their file migrations and caches by allowing them to:

- Force the premigration of specific files or directories.
- Force the purging of specific files or directories.
- Specify which files they want to prevent from migrating automatically by listing them in local stop files (`.migstop`).
- Specify which files they want to migrate by listing them in local migrate files (`.migrate`).
- Declare a group of files that need to be cached together.
- Read migrated data without caching the files.

See chapter 5 of this manual for additional information on how to use VSM.

VSM Functional Description

The introduction to this chapter provides an overview of VSM. The following topics provide a more detailed description of how VSM operates:

- VSM Architecture
- Basic Migration Operations and Features
- NetBackup Storage Method
- Disk-Space Management
- Multilevel Migration
- File Export / Import
- VSM System Files and Directories

VSM Architecture

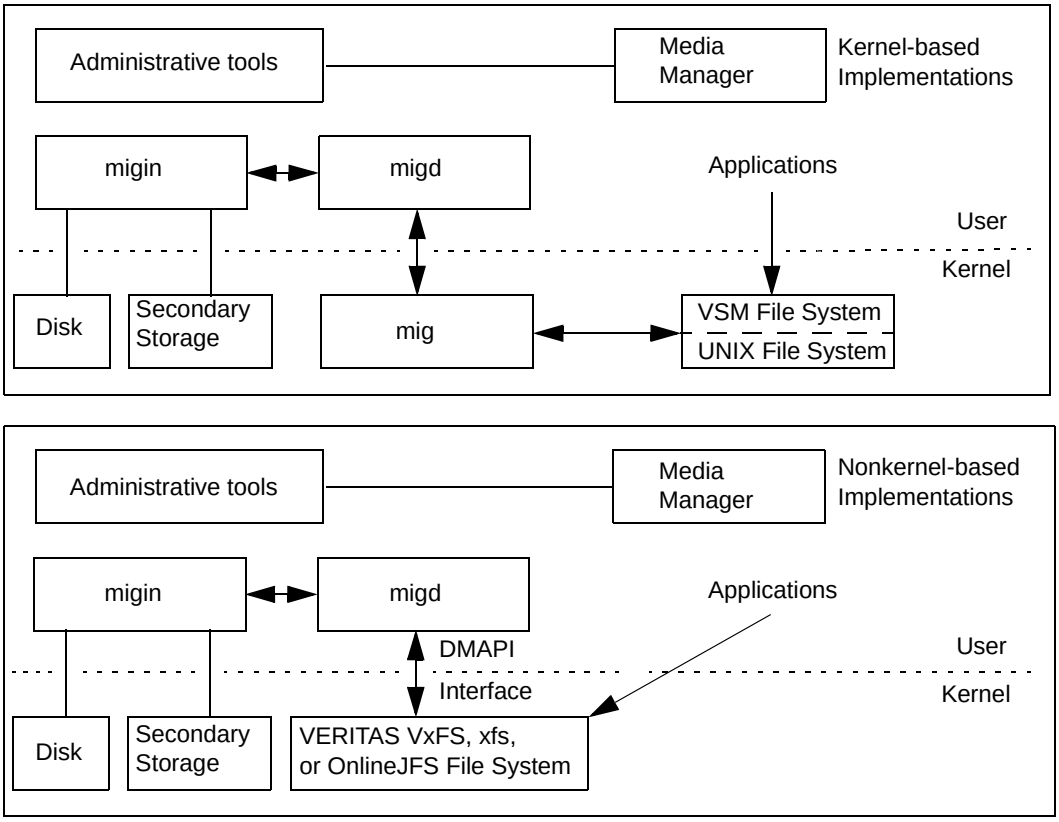
The main elements of VSM architecture are shown in Figure 2.

- Migration daemon (`migd`)
- Media Manager

  (not applicable to VERITAS Storage Migrator Remote)
- Kernel resident VSM file system module (`hsm`), Solaris *ufs* file systems only
- Kernel resident pseudodevice driver (`mig`), Solaris *ufs* file systems only
- Administrative tools to control the migration and cache processes (for example, `migbatch`, `mignospace`)
- Other utilities for volume and database management

Nonkernel-based implementations do not include the kernel resident modules `hsm` and `mig`. Instead, the migration daemon (`migd`) interfaces with the *OnlineJFS* file system on HP-UX, the *xfs* file system on IRIX or the VERITAS *VxFS* file system on Solaris using the DMAPI interface.

Figure 2. Basic VSM Architecture



Kernel-based Implementation

Note Kernel-based implementations in this release run on Solaris *ufs* file systems.

The migration daemon (`migd`) runs in user space and processes events generated by the VSM file system when a user references a migrated file.

The VSM file system is a layer of software that is linked to the system kernel. Its purpose is to intercept file-system calls to migrated files before those calls get to the underlying UNIX file system.

VSM passes requests for unmigrated files directly to the underlying UNIX file system. However, requests for migrated files are blocked while VSM passes its own request to `mig`, which in turn passes it to `migd`.

migd calls the appropriate processor, based on the original request. For instance, on read and write requests, migd calls the reload processor (migin) to cache the requested file to disk. When migin finishes, migd sends a completion message to the VSM file system through mig. The completion message, causes the VSM file system to pass the original read or write request to the underlying UNIX file system for further processing.

The VSM pseudodevice driver (mig) implements special system calls required for:

- Migrating files
- Caching files
- Determining the status of migrated files
- Providing a communication channel between the file-system layer and the migration daemon (migd)

The mig_make_migrated call is an example of a special system call implemented in mig. This call is used to premigrate a file. VSM uses these special system calls internally. They are not available or necessary for application development.

As a communications channel, mig implements a device driver for a pseudodevice called /dev/mig.

The VSM protocol uses the pseudodevice, /dev/mig, as two separate devices:

- /dev/mig0
- /dev/mig1

/dev/mig0 is the input channel to mig and implements a system-call mechanism. /dev/mig1 is the output channel from mig and implements a call-out mechanism. The VSM file system uses the call-out mechanism to initiate cache requests and no- or low-space processing. migd processes these requests in user space.

Nonkernel-based Implementation

The migration daemon (migd) runs in user space and processes events generated by the HP-UX *OnlineJFS*, SGI IRIX *xfs*, or VERITAS *VxFS* file system when a user references a migrated file.

Object events are:

DM_EVENT_READ

DM_EVENT_WRITE

DM_EVENT_TRUNCATE

File system events are:

DM_EVENT_NOSPACE

DM_EVENT_REMOVE

DM_EVENT_DESTROY (Solaris and HP-UX only)

DM_EVENT_UNMOUNT

When `migd` starts, it requests control from the kernel whenever a migrated file is referenced. `migd` then calls the appropriate processor, based on the original request. For instance, on `read`, and `write` requests, `migd` calls the reload processor (`migin`) to cache the requested file to disk. When `migin` finishes, `migd` sends a completion message to the kernel. The completion message, causes the kernel to pass the original `read` or `write` request to the underlying UNIX file system for further processing.

Nonkernel-based implementations use the DMAPI interface to inform `migd` when the managed file system is out of space. Threshold processing is done completely by `migd`.

Common Implementation

In all implementations, the administrator uses the `migbatch` or `miglow` commands to manage free space on the disk. These two commands control migration and removal of files from managed file systems. During the migration process, `migcopy` copies files from the managed file system to secondary storage. See “Migration Commands” on page 26 for more information on `migbatch`, `miglow`, and `migcopy`.

VSM records its activities in log files. The migration daemon (`migd`) records its activities in a VSM global-log file. Other VSM processes use individual log files configured for each managed file system.

The VSM volume daemon (`migvold`) controls the unmounting of Media Manager volumes mounted in read mode. VERITAS Storage Migrator uses Media Manager to access secondary storage media such as tapes or optical discs.

Basic Migration Operations and Features

Basic migration operations and features are described in the following sections:

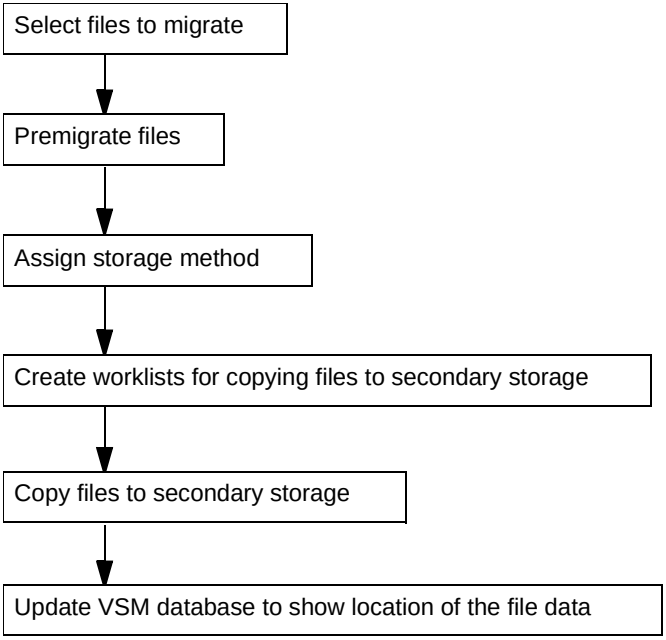
- File Migration
- File Caching
- File Slice
- Total File Caching
- Partial File Caching
- Directory Level Migration
- Tape and Optical Volume Management

File Migration

During migration, VSM selects files, premigrates them, and then copies them to secondary storage. For information on how to migrate a group of files together, see “Directory Level Migration” on page 20.

Figure 3 shows the main steps in the migration process and the following topics explain them. The `migbatch` command can be used to perform these steps.

Figure 3. File Migration Process



Select Files

VSM selects files by scanning the VSM managed file system and evaluating each file according to the file-selection criteria set during configuration. This criteria is based on file size and time elapsed since the last access or since last modification, whichever is most recent. Files that meet the criteria become candidates for migration and are placed on a list.

VSM selects files until there are no more files that meet the selection criteria or until it selects enough to reduce space used to a predefined level called the *low-water mark*. See “Define Criteria for Selecting Files” on page 70 for details on selection criteria.

Note VSM customarily sweeps the file system on demand or periodically as a crontab entry. To implement constant sweeping, see “Constant Sweeps” on page 35.

Premigrate Files

Premigration is implemented in two different ways on VSM.

- On Solaris *ufs* systems, VSM creates a file in the premigration directory, assigns data blocks to it, and removes the data blocks from the original file.
- On all nonkernel-based implementations, VSM sets appropriate file regions (DMAPI) as appropriate. VSM does not use the premigration directory for this purpose.

In each case, a migrated flag marks a file as migrated. A migrated file is also assigned a file handle. The file handle is a sequence number (hexadecimal) that VSM uses to identify migrated files. The file handle is stored in the file-handle database (FHDB) for the file system. Each file system can use a separate file-handle database or several file systems can share a database.

Although the user perceives the file to be migrated, VSM holds a copy of the migrated data on disk until space is needed in the managed file system, at which time the premigrated file is purged or truncated. See “Disk-Space Management” on page 24.

Premigration is a quick and “atomic” operation that neither moves nor copies the data from one place to another. The data remains in the same file system. Premigrated files take no additional disk space on all nonkernel-based implementation servers, while on Solaris *ufs* file system implementations additional space is needed only for the slice (nominally 8 kilobytes) and one additional inode for each premigrated file.

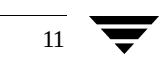
Some other VSM systems on the market are less efficient. They accomplish this operation by marking a file migrated, replacing it with a stub file, and copying the file data to a prestaging disk. This prestaging process requires additional time to copy a file from one disk to another and doubles the disk overhead. VSM’s premigration is like prestaging, but without the burden of copying or doubling the disk overhead.

If a user executes the VSM `fls -l` command on a migrated file, the resulting display shows the migrated flag, the file handle, the original file size and dates. See man page fls(1) or the *Storage Migrator User’s Guide.* After purging, the file that is in the user’s directory contains only a slice of the migrated data. See “File Slice” on page 16.

Certain input/output and name space operations on a migrated file cause special processing. Examples include reading past the slice value, which causes VSM to cache the file, and removing a file, which causes VSM to update VSM databases.

Assign the Storage Method

After premigrating the file, VSM reads the storage methods defined in the configuration file (`migconf`) for the file system and assigns them to the selected files. Storage methods are a combination of the method name, volume set number, volume set availability (called *hint*), and volume pool; they determine the secondary storage media to which files are migrated.



During configuration, the administrator configures the storage method that is most suitable for the file system. The volume set number specifies a set of volumes. The availability provides an indication of how long it takes to access the volume (for example, library indicates an online library and therefore immediate access). The volume pool specifies where the volume set is located.

See “Choose Storage Methods for Migrating Files” on page 82 for more information on storage methods and volume sets.

Create Migration Work Lists

The `migpolicy` script creates a work list, called a `copydb` file, for each secondary storage method and volume set configured for VSM. `migpolicy` also assigns files to storage methods and writes file entries in the proper work list. For information on how to divert specific files to suitable media, see “Customize the VSM Policy and Method for Migrating Files” on page 295.

The work lists provide input for the copy processors that copy premigrated files to secondary storage. VSM stores these work lists in the working directory specified in the global configuration file (`migconfg`).

See “Work Lists (copydb files)” on page 303 for a more detailed description of work lists.

For more information on how VSM can use VERITAS NetBackup to make copies of migrated files, see “NetBackup Storage Method” on page 23.

Copy to Secondary Storage

The `migworker` script initiates a copy processor for each work list (`copydb`) file. The copy processors read entries from their assigned work list and copy the indicated files to secondary storage. If the configuration specifies concurrent recording and sufficient devices are available, different files can be copied to different devices simultaneously. Copy 1 and copy 2 of a file can also be recorded simultaneously.

A copy processor attempts to copy every premigrated file on its work list from disk to secondary storage and upon completion sets the status of each file operation. VSM periodically removes work list entries whose status field indicates proper completion. Copy operations that do not complete successfully are reprocessed the next time VSM starts that copy processor.

After using up previously registered media, VERITAS Storage Migrator automatically registers additional tape and optical disc media as needed from one or more volume pools for writing the premigrated files on the work list to secondary storage.

Update File-Handle Database and Volume Database

The final step in the migration process is to update the file-handle database (FHDB) and VSM volume database (VOLDB) to show the location of the migrated files. The FHDB contains at least one entry for each copy of a migrated file. If VERITAS Storage Migrator divides larger files into parts called granules before writing them to secondary storage on tape or optical disc, the FHDB may contain more entries for each file copy (see "File-Handle Database Space Requirement" on page 106). Granule size is configurable. Files written using remote methods `ft` and `nb` are migrated as single granules.

The copy processors perform the database update by creating temporary database entries as they complete their work lists. When a copy processor completes its work list, `migworker` calls `migmerge.sh` to merge in the temporary entries. The FHDB is locked while new entries are being merged into it.

File Caching

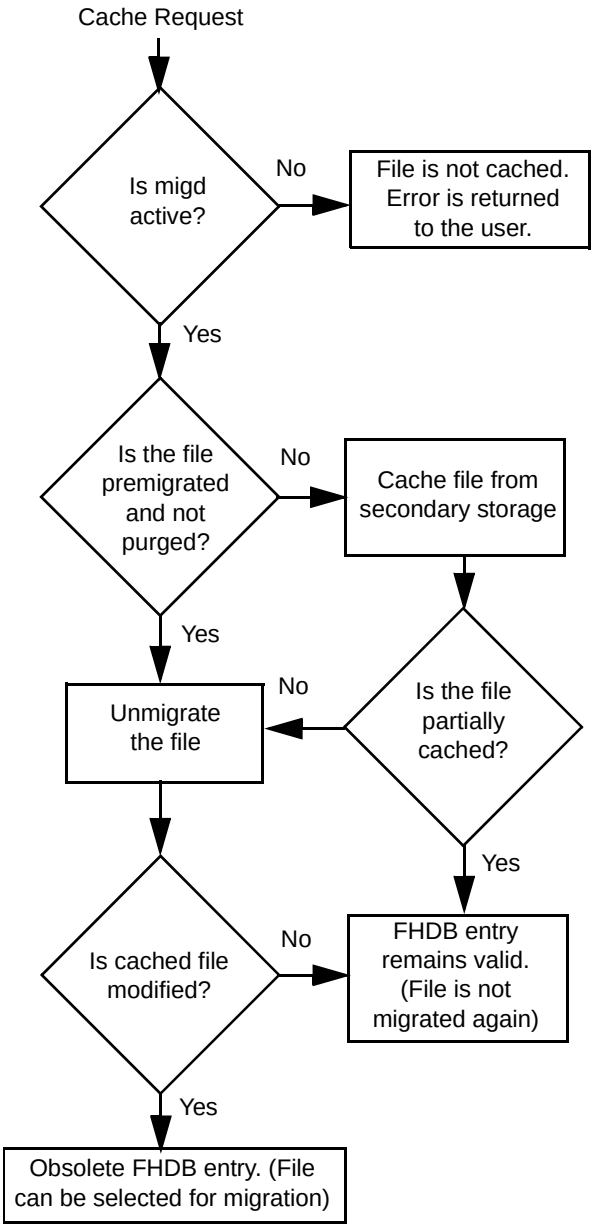
Caching is the process of copying migrated files back to the managed file system for access. The name of a migrated file remains in its original directory and stays visible to the user. Users can determine the status of a file by using the VSM `fls` or `migloc` command. See man pages fls(1) and migloc(1), or the *Storage Migrator User's Guide*. Neither command caches the file if it is migrated.

Before a user can access the data for a migrated file, VSM must cache the data back to its original file system. There are two ways to do this.

- Total file caching (see page 17)
- Partial file caching -- DMAPI implementations only (see page 17)

The following information is common to both total and partial file caching.

Figure 4. File Caches



If a file is premigrated and not purged, VSM can unmigrate the data without having to cache it, and there is no caching delay. Otherwise, the caching process copies the data once from secondary storage and unmigrates it unless it is only partially cached, in which case it remains marked as migrated. VSM unmigrates files in two different ways:

- On Solaris ufs file system implementations, VSM moves the data blocks from the premigration directory back to the original file.
- On nonkernel-based implementations, VSM removes some of the file attributes (DMAPI).

In each case, appropriate flags are set to indicate a cached or partially cached, unmodified file. If a cached or partially cached file is left unmodified, the file-handle database entry remains valid. If unmodified cached files are remigrated, VSM purges the data from disk using the same purging selection criteria applicable to any migrated file. When a user modifies a cached file, VSM sets the FHDB entry to obsolete, making the modified file eligible for migration at a later time. See man page fls(1) or the *Storage Migrator User's Guide*.

If more than one copy of the file is available, VSM attempts to cache remote copies first (if they exist) before attempting to cache copies from local media. See Figure 1 on page 3. If caching from local media, VSM attempts to cache the copy from the volume it can access in the shortest time. If that copy is damaged, VSM automatically switches to another copy and caches that one.

In all instances, caching occurs automatically and without extra effort on the part of the user. Because the application accessing the data is blocked during the cache operation, a noticeable delay can occur when caching migrated files to disk. Implementations supporting partial file caching, however, can reduce or control this delay when caching large files (see "Total and Partial File Caching Trade-offs" on page 20). The length of the delay depends on several factors:

- Availability of drives
- Availability of the volume
- Transfer rates from secondary storage to primary disk
- Size of the file
- Level of activity on the system.

You can minimize the caching delay in several ways:

- Configuring enough tape or optical devices to handle the peak demand
- Matching device characteristics to the size and access frequency for migrated files
- Configuring the unmount delay parameter to enable successive read requests from the same volume to proceed without unmounting and remounting that volume (see "Unmount Delay" on page 128)

- Configuring the demand delay parameter to unmount an unused volume of the same density immediately, and mount the volume containing the file to be cached (see “Demand Delay” on page 144)
- Using partial file caching, if available (see “Partial File Caching” on page 17)

For instance, if one device is available then only one cache or migration request can be processed at a time. All other cache requests must wait. However, if four devices are available to VSM, a maximum of four simultaneous caches are possible, assuming no migrations are active.

File Slice

A slice is the portion of a migrated file which is disk resident at any given time. There are two values for file slice.

- Configured slice

  A portion of the front of a file which VSM retains on disk even when the file is migrated to secondary storage. This is a configurable parameter and can be set to a different value for each file system. See “Slice Size” on page 135.

- Effective slice

  The variable portion of a migrated file which is partially cached to disk. See “Partial File Caching” on page 17.

Note Accesses to files over NFS on some platforms can cause VSM to cache files regardless of the configured slice size. See the *Storage Migrator User’s Guide*.

Note Restoring a migrated file with NetBackup sets the slice value to 0.

Any `write` request or any memory mapped I/O request to a migrated file will cache the file no matter how big the configured slice is.

Depending upon the size of the configured slice, you can prevent some standard utilities like `file` and `head` from accidentally caching a large number of migrated files.

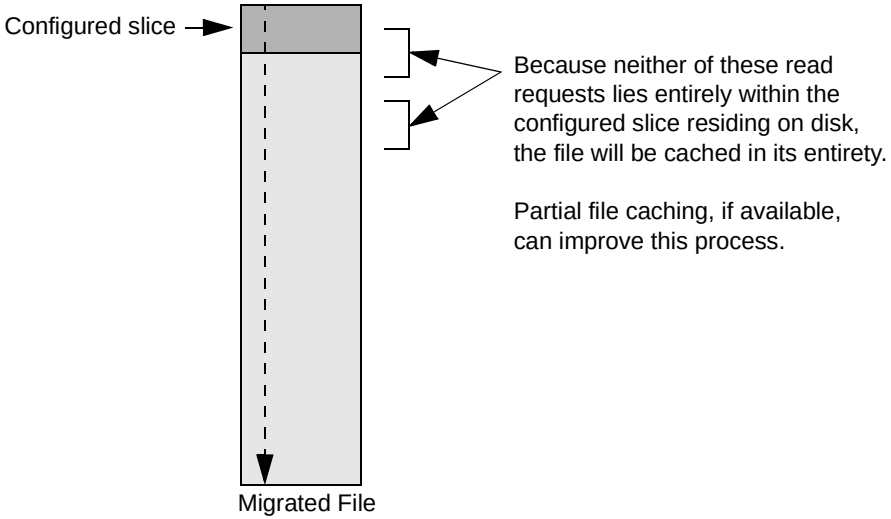
- The `head` utility reads the first 8192 bytes of a file by default. To prevent `head` from caching a file, set the slice value to at least 8192 bytes.
- The `file` utility normally reads the first 512 bytes to determine the type of file. Therefore, setting the slice value to 512 or greater usually prevents the `file` command from caching migrated files; but, as explained below, that is not always true.

After reading the first 512 bytes, the `file` command applies built-in rules to determine the file type. If it fails to determine the file type by using these rules, it reads different parts of the file depending on the platform. If any of these parts are beyond the slice, caches the entire file.

Total File Caching

A file `read` request for a sliced migrated file will be satisfied, if possible, by using the data on disk. If the `read` request exceeds or is completely beyond the configured slice, the entire file is cached before control is returned to the application or user. Partial file caching, if available, can improve this process. See "Total and Partial File Caching Trade-offs" on page 20. A write request always caches the entire file.

Figure 5. Total File Caching



If more than one copy of the file is available, VSM attempts to cache the copy from the volume it can access in the shortest time, regardless of whether it is on remote or local media. See Figure 1 on page 3. This is the volume with the best volume set availability (lowest *hint* value). If hint values are equal, the copy is cached from the volume with the lowest level number. If the first copy is damaged, VSM automatically switches to another copy and caches it completely, regardless of whether it is on remote or local media.

Partial File Caching

Implementations of VSM using the DMAPI interface support partial file caching. This does not apply to files cached from the `ft` or `nb` methods, which are always totally cached. Files cached with either the `migstage` or `migtie` command are also always totally cached.

Partial file caching allows `read` access to a migrated file without caching the entire file. A file `read` request for a sliced migrated file will be satisfied, if possible, by using the data on disk. If a `read` request exceeds or is completely beyond the configured slice, the file is partially cached to satisfy the `read` request. If a `read` request plus a configurable read-ahead exceeds or is completely beyond the partially cached data on disk, additional data is cached to satisfy the `read` request.

A `write` request always caches the entire file. If a read cache is in progress or a file is partially cached when a read request comes in, caching continues until the file is totally cached.

The configurable read-ahead determines the minimum amount of data VSM caches to disk beyond what is needed to satisfy the `read` request. See “Read Ahead” on page 135.

Figure 6. Partial File Caching, Phase 1

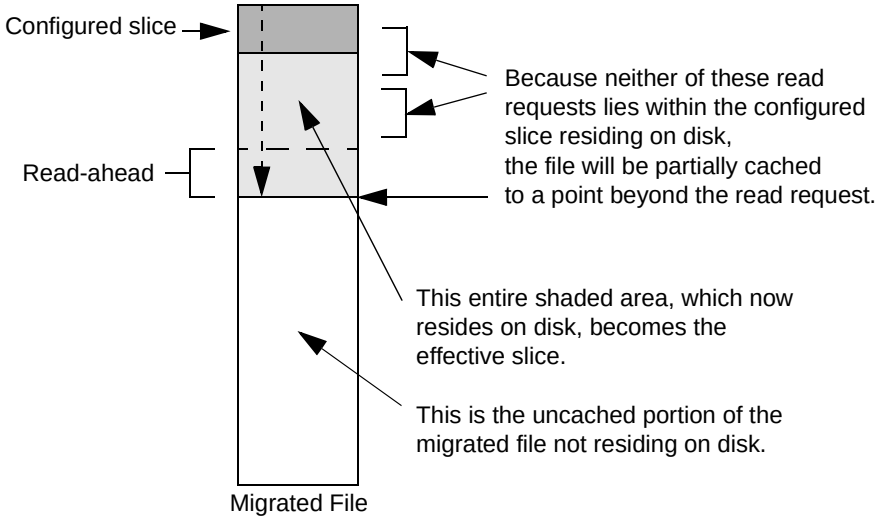
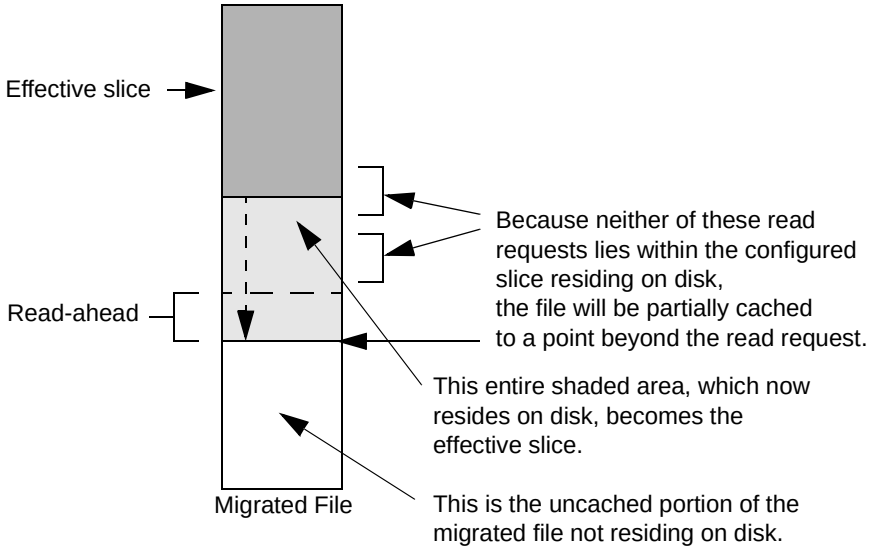


Figure 7. Partial File Caching, Phase 2



The Administrator can disable partial file caching by configuring the read-ahead to be -1, in which case any `read` request that does not lie entirely within the configured slice caches the entire file. If a file is partially cached VSM allows `read` access to cached data as soon as the granules containing the requested data are cached to disk, and then continues with the partial caching operation for a configurable read-ahead amount. All data previously on disk, plus the data required for the read, plus the read-ahead, is rounded up to the next whole granule. This cached portion of the migrated file now becomes the effective slice for the file, superseding the configured slice when processing subsequent `read` requests.

The initial partial caching request reads from the beginning of the file including the configured slice. Subsequent partial caches of the same file do not reread data already cached, but resume caching at the end of the effective slice.

The effective slice and other information pertaining to partial file caching are stored in the file's `.IHAND` entry. See "Administer Inode-to-Handle Files" on page 152.

Note The format of the `.IHAND` file prior to VSM R3.1 has been changed to support partial file caching in R3.1 and higher. These files were converted or rebuilt automatically when VSM R3.1 was installed.

Partially cached files remain marked as migrated. If the effective slice equals the file size, VSM restores the configured slice value and marks the file as cached.

If more than one copy of the file is available, VSM attempts to cache the copy from the volume it can access in the shortest time, regardless of whether it is on remote or local media. See Figure 1 on page 3. This is the volume with the best volume set availability

(lowest *hint* value). If hint values are equal, the copy is cached from the volume with the lowest level number. If the first copy is damaged, VSM automatically switches to another copy and caches it completely, regardless of whether it is on remote or local media. Partial file caching applies only to the first copy VSM attempts to cache from local media. Partial file caching does not apply to files cached from the `ft` or `nb` methods, which are always cached in their entirety.

Partially cached files remain on disk until selected for purging. Selection criteria are the same as those for any unmigrated file. VSM updates the `.IHAND` entry when a partially cached file is purged.

Total and Partial File Caching Trade-offs

The decision to enable or disable partial file caching depends on the way your applications use the migrated data.

Total file caching is preferable if your applications read migrated file data sequentially from start to end. See "Total File Caching" on page 17. Partial file caching, if enabled in this situation, can increase system overhead and make your applications run longer.

Partial file caching is preferable if your applications read a small portion of the file data without reading the entire file. See "Partial File Caching" on page 17. Performance improvement is greatest if the `read` access is near the beginning of the file. VSM allows `read` access to partially cached data as soon as the granules containing the requested data are cached to disk. Total file caching, if enabled in this situation, delays `read` access until the entire file is cached.

If possible, separate applications that read entire files from those that read partial files. Place the former in a VSM managed file system with partial file caching disabled, and the latter in another VSM managed file system with partial file caching enabled.

If this is not possible, enable partial file caching and configure the read-ahead to optimize overall performance. If the majority of your applications read migrated file data sequentially from start to end, configure a larger read-ahead. If the majority of your applications only read a small portion of the file data, configure a smaller read-ahead. Reconfigure read-ahead as often as necessary to maintain optimum performance. See "Read Ahead" on page 135.

Directory Level Migration

While migration customarily occurs at the file level, it is also possible in nonkernel-based implementations of VSM to perform migration and caching at the directory level. This feature is only available on DMAPI implementations, not on Solaris *ufs* file systems.

With directory level migration, users can group all of their owned files in an owned directory and all of its owned subdirectories, and migrate them as a group. Users with root privilege can group directories owned by any user. When any file in the group is cached, all of the files in the group are cached.

Note By its very nature, this feature can increase migration and caching activity unnecessarily if used in situations where file level migration would suffice. Use directory level migration only where applicable.

Grouped files are premigrated together and placed in a work list (`copydb` file) as a group. When VSM copies the group to secondary storage, the files are located in close proximity to one another on sequential media. This minimizes the time needed to cache the grouped files.

See “Work Lists (copydb files)” on page 303 for a more detailed description of work lists.

Normal file selection criteria as described in “Select Files” on page 10 are not applicable in directory level migration. However, normal VSM rules apply to any ungrouped files added to grouped directories after the directories were grouped, and these files are selected, migrated, and cached individually.

See man page miggroup(1) for more information on how to implement directory level migration.

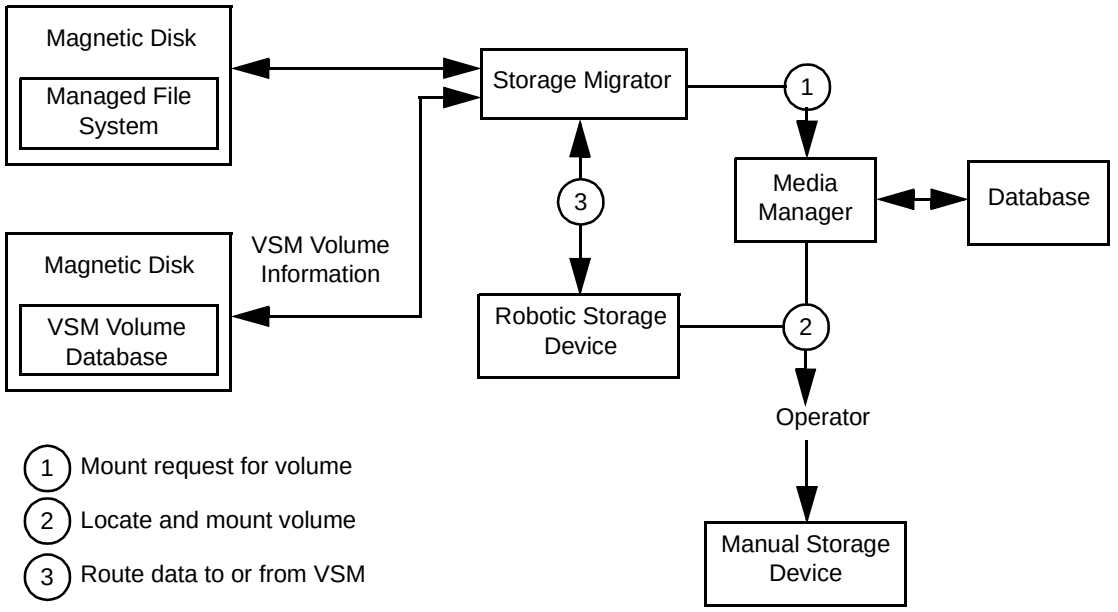
Tape and Optical Volume Management

During configuration, the VSM administrator defines the storage methods that are available to VSM. For VERITAS Storage Migrator, the type of media that these storage methods define can be magnetic disk, tape, or optical disc (used as raw device).

The administrator also defines which method name (and therefore which media) to use for specific file systems. The configuration for one file system can specify that migrated files go on a cartridge tape. Files from another file system can go on the same or another media type.

The tape or optical storage devices that VERITAS Storage Migrator uses are managed by Media Manager. See Figure 8. When transferring data to or from a storage device, VERITAS Storage Migrator queries the volume database (VOLDB) and identifies the volume on which to read or write the migration data. It then includes the volume information in a tape request to the Media Manager device daemon, which allocates or deallocates drives based on availability.

Figure 8. Media Management



The administrator places volumes for VSM in pools from which VSM selects the volumes it will use. Once a volume is selected by VSM, that volume is assigned to VSM and registered in the VOLDB. For information on scratch pools, see “Keeping a Supply of Unused Volumes” on page 273.

Media Manager tracks the location of both online and offline volumes and keeps this information in its own volume database.

If a request from VERITAS Storage Migrator involves a robotic device, the Media Manager device daemon queries the Media Manager volume daemon to determine which robotic device has the volume. The device daemon then issues a mount command to the robotic daemon controlling that device, which automatically mounts the specified volume and returns control to the application or user. No operator intervention is required, provided the required volume is physically in the robotic device.

With a manually operated device, the device daemon issues a mount request to the operator. To satisfy the request, the operator must find the volume, mount it manually, and assign it.

Media Manager is managed separately from VERITAS Storage Migrator and can be used by other applications. See the *Media Manager System Administrator’s Guide* for more information.

NetBackup Storage Method

You can make copies of migrated files using VERITAS NetBackup by specifying the `nb` storage method name. See "NetBackup Method" on page 86.

Migration to VERITAS NetBackup

This process is similar to the basic migration process described in "File Migration" on page 10. The steps of file selection, premigration, storage method determination, and work list creation are identical. VSM creates a granule header file (`GLABEL`) for each entry in the work list (`copydb`). This is done by `migcopy` and is deleted when the copy completes. Then VSM lists both the premigrated file and its `GLABEL` in an input file for NetBackup.

When `copydb` processing is finished, VSM issues a `bpbackup` command to NetBackup, and the listed files are backed up.

On Solaris *ufs* file systems, NetBackup backs up the premigration image of the files. On nonkernel-based implementations, NetBackup backs up files using their full paths. The `bpbkar` command uses invisible input/output to avoid causing unnecessary file cache events.

When the backup operation is complete (or the Deadman Timeout configured for the `nb` method has expired) the `GLABEL` files are deleted from premigration. VSM sets the `copy_date` field in the FHDB for each file successfully backed up to be the UNIX date of the NetBackup image. This value is used to cache the files, and later by `migmdclean` to clean the databases and NetBackup volumes by setting obsolete entries to dead and removing them.

An error is logged for each failed copy attempt unless the entire backup operation failed, in which case the reason for the backup failure is logged. `GLABEL` files are recreated the next time a failed backup is migrated again.

Caching from VERITAS NetBackup

Note Partial file caching, as described on page 17, does not apply to files cached from NetBackup.

This process is similar to the basic caching process described in "File Caching" on page 13. VSM detects cache requests, adds them to a work list (`copydb`), and builds an input file for NetBackup. This input file includes the start date when each file was originally migrated; this minimizes the time NetBackup takes to locate the files by searching its databases.

When `copydb` processing is finished, VSM issues a `bprestore` command to NetBackup, and the listed files are cached.

On Solaris *ufs* file systems, NetBackup caches files to the premigration directory. On nonkernel-based implementations, NetBackup caches files using their full paths.

An error is logged for each failed restore attempt unless the entire restore operation failed, in which case the reason for the restore failure is logged.

Use the `migmdclean` command to clean the databases and NetBackup volumes by setting obsolete entries to dead and removing them. When all copies of files on a given `nb` volume are expired, `migmdclean` also deletes the image from the NetBackup database.

If a file access causes a cache request, VSM attempts to initiate a restore operation from NetBackup before attempting to cache copies from local media.

Disk-Space Management

With VSM, you can manage disk space to best suit your needs in a variety of ways:

- You can anticipate disk space consumption and make space available before it is needed.
- You can set the disk parameters such that the system automatically makes space available when the need arises.
- You can use a combination of these two strategies.

In this last approach you premigrate and copy files to secondary storage based on anticipated needs, and then let the system make space available when needed by purging premigrated files.

VSM provides migration parameters and commands that allow the administrator to maintain free space between configured limits and otherwise tailor VSM to satisfy site requirements.

Migration Parameters

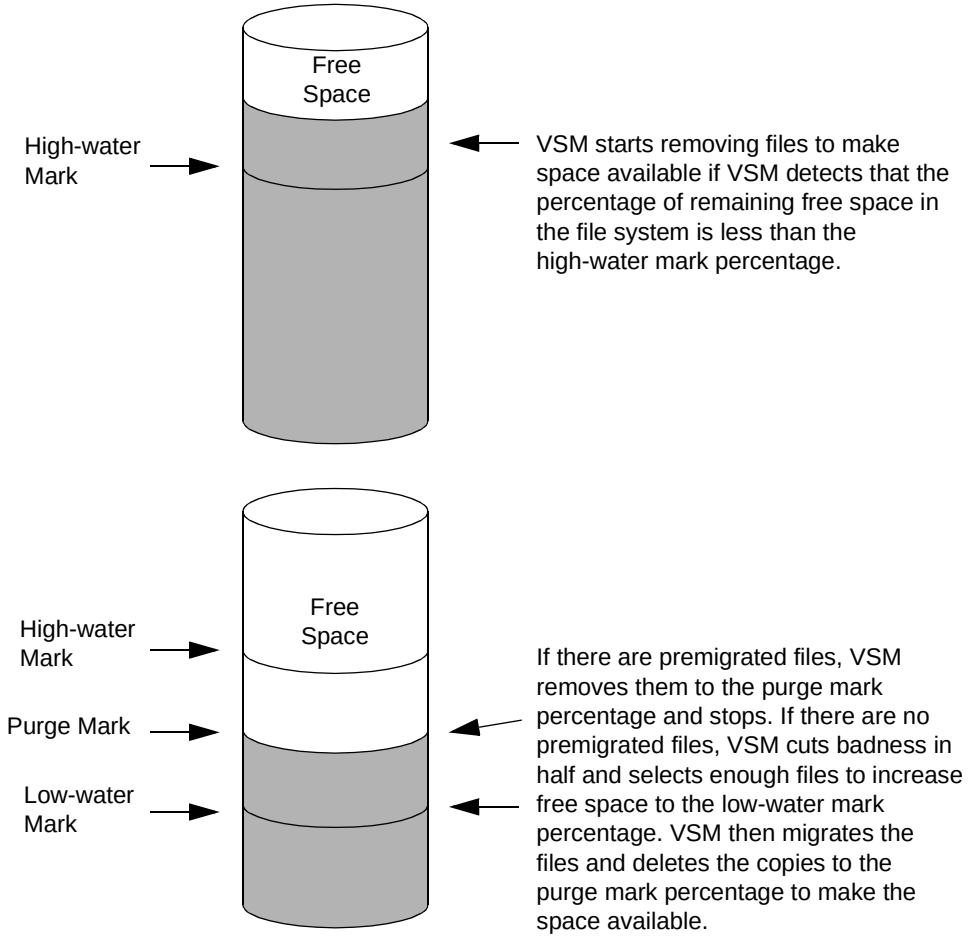
Before reading about the disk-management commands, you need to understand some of the parameters for configuring the file system space limits.

As shown in Figure 9, the main configuration parameters controlling space limits are as follows:

- High-water mark. You can configure VSM to automatically remove premigrated files or migrate files when free space is less than the high-water mark percentage or you can execute commands that start this process (see “Migration Commands” on page 26).

- Low-water mark. You can configure VSM to stop selecting files for migration when free space increases to this percentage.
- Purge mark. You can configure VSM to keep some migrated files on disk for rapid access.

Figure 9. Migration Parameters



In addition to parameters controlling space limits, there are three configuration parameters that control the files that VSM selects for migration:

Minimum age

VSM cannot migrate files accessed within this time period. For example, if minimum age is 2 days, VSM selects only files that were accessed two days or more ago.

Minimum size

Specifies the minimum size file that VSM can migrate. For example, if minimum size is 100 kilobytes, VSM selects only files 100 kilobytes or larger.

Badness

After ruling out files that are less than the minimum age and size, VSM calculates the badness of each remaining file and selects file candidates only if their badness exceeds the value configured for the file system. The VSM formula for badness is configurable and is based on file size and age. Administrators can define a site-selected badness formula in lieu of the VSM formula.

Similar parameters control the files that VSM selects for purging.

Migration Commands

There are three main commands that the administrator can use to manage disk space. They are:

- `migbatch`
- `mignospace`
- `miglow`

The following paragraphs provide an overview of what these commands do and how the administrator uses them. This is followed by separate descriptions of each command.

The `migbatch` command controls the migration of files to secondary storage. This command performs migration in two stages:

- Premigration
- Copy premigrated files to secondary storage

During premigration, VSM selects files according to criteria that you specify and marks the files with a migrated flag and by a VSM file handle. The premigration process continues until:

- VSM premigrates all files meeting the file-selection criteria.

  or

- The space that VSM can potentially free by purging premigrated files is enough to increase the percentage of file-system free space to a level called the low-water mark.

VSM then copies the premigrated files to secondary storage. The premigrated copies remain on the disk until the file system needs more space.

Purging of premigrated files, starts when:

- A user or process writes to the file system when file-system free space is at or less than the high-water mark percentage. When VSM detects this condition, `migd` starts `mignospace`.

  or

- You execute the `mignospace` command.

In either case, `mignospace` purges premigrated files in an attempt to increase free space to the purge mark percentage. See "Define Criteria for Selecting Files" on page 70 for more information on how to configure which files `mignospace` removes first.

If there are no premigrated files, `mignospace` starts a migration by selecting more files and copying them to secondary storage. This second selection and removal process continues until all files meeting the selection criteria have been purged or the amount of free space increases to the purge mark percentage (whichever is first). See Figure 9 on page 25.

If you do not want to use `migbatch` and `mignospace` individually as described above, you can also use a command called `miglow`. This command checks the high-water mark as specified in the *dwpath*`/database/migconf` file and then runs both `migbatch` and `mignospace` to provide space if necessary.

`migbatch`

`migbatch` selects, premigrates, and copies files to secondary storage without purging the premigrated files from the disk. The administrator can start `migbatch` either from the GUI or with a `crontab` entry.

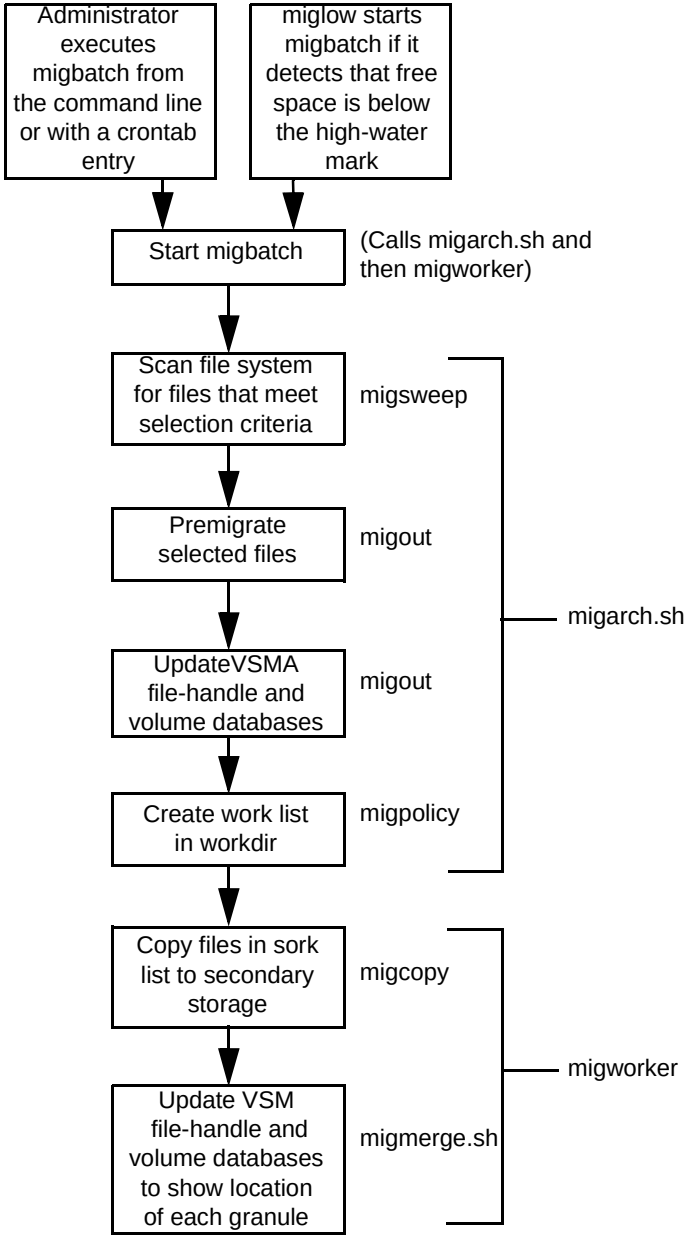
When selecting files for migration, VSM performs round-robin sweeps that start at the root of the managed file system or directory and traverse the entire directory tree. Sweeping stops when enough files have been selected to satisfy the low-water mark. VSM saves the path of the last selected file in *dwpath*`/workdir/`*hsmname*`.sweep.restart`, and starts the next sweep from the last component of the saved path that still exists in the file system. The `sweep.restart` file is removed once the starting point for the sweep is determined. If the `sweep.restart` file does not exist, VSM starts sweeping from the root of the tree. Round-robin sweeping eventually scans all of the files in the managed file system or directory.

Note Files listed in local and global migrate control files are not necessarily selected in any particular sweep of the file system, but eventually they will be selected as migration candidates. See the *Storage Migrator User's Guide* and "Global Migration Control" on page 290 for more information about these control files.

Because the migration process can take a long time, most administrators run `migbatch` on nights or weekends. This creates premigrated files without interfering with user processes. VSM can then quickly provide space during normal working hours by purging the premigrated files. The `miglow` command can also start `migbatch` (see “miglow” on page 32).

Figure 10 shows what occurs when you run `migbatch`. This figure also indicates the other programs that `migbatch` calls to perform each task.

Figure 10. Migrating Files with migbatch



mignospace

mignospace creates space by purging premigrated files (Figure 11). It begins by checking for premigrated files.

- If some premigrated files exist that exceed purge badness, mignospace either purges this premigrated file space to the purge mark percentage and stops or purges all this premigrated file space and stops, whichever comes first.
- If premigrated files exist but none exceed purge badness, mignospace cuts the current purge badness in half and exits.
- If there are no premigrated files, mignospace cuts the current badness in half and selects enough files to increase free space to the low-water mark percentage. mignospace then premigrates the selected files, copies them to secondary storage, and determines if any exceed purge badness. If some premigrated files exceed purge badness, mignospace either purges this premigrated file space to the purge mark percentage and stops or purges all this premigrated file space and stops, whichever comes first. If no premigrated files exceed purge badness, mignospace cuts the current purge badness in half and exits.

For Solaris *ufs* file systems, the kernel starts mignospace processing automatically if the migration daemon (migd) is active, the VSM state is Active, and file-system free space is less than a point called the high-water mark threshold.

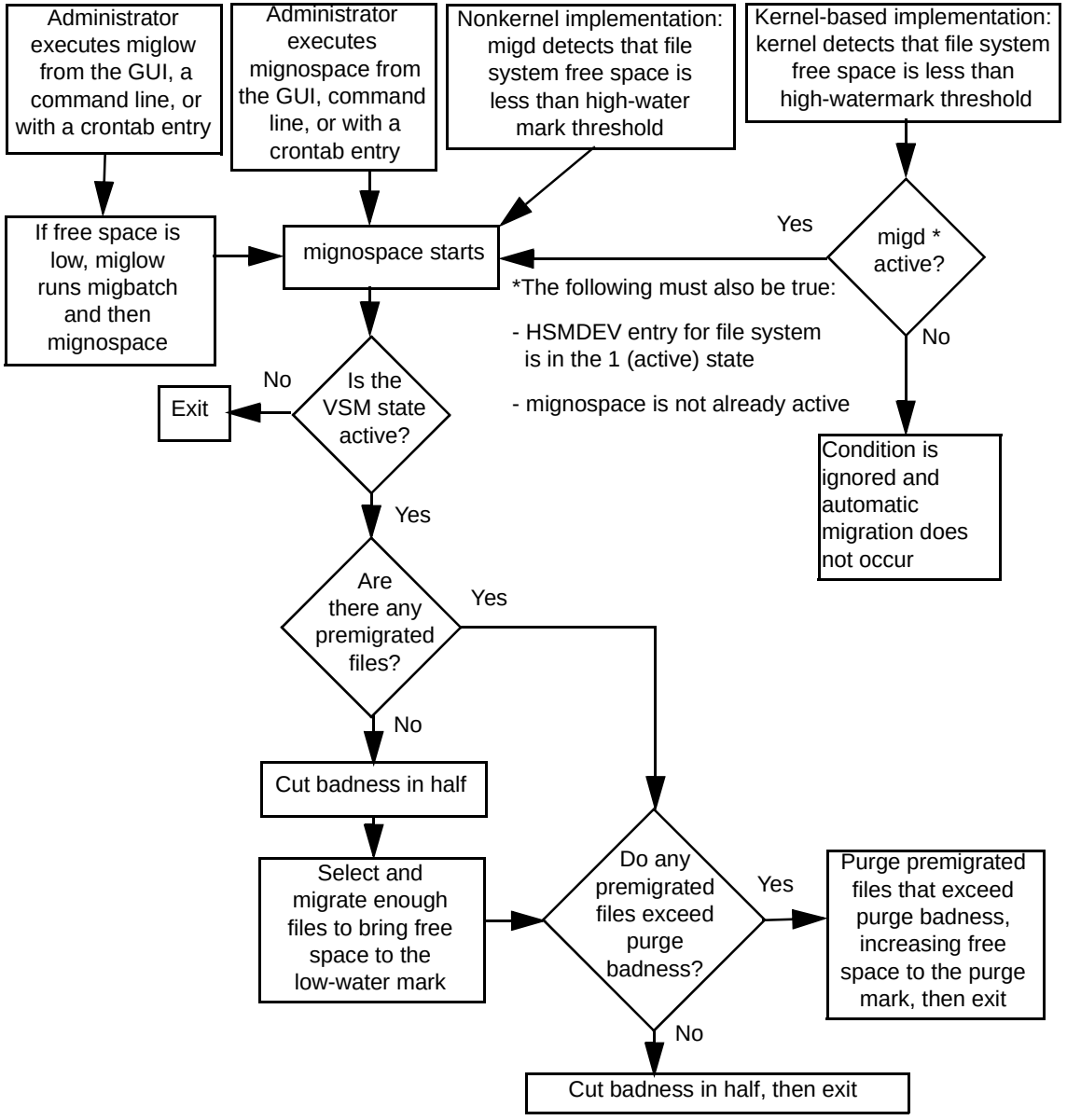
For nonkernel-based implementations, migd periodically checks the high-water mark threshold and starts mignospace if the threshold is exceeded. The frequency with which migd checks the high-water mark threshold is determined by startmigd. See man page startmigd(1M) for more information.

By default, this threshold is the same as the freespace value that you set in the migconf file but you can use the migthreshold command to set it to another value. See “Disabling Automatic Space Management and Caching” on page 62 for more information on resetting the kernel threshold value.

You can also invoke mignospace from the GUI, at the system prompt or by running the miglow command (see “miglow” on page 32).

See “Define Criteria for Purging Files” on page 78 for more information on how to configure which files mignospace removes first.

Figure 11. Making Space Available with mignospace



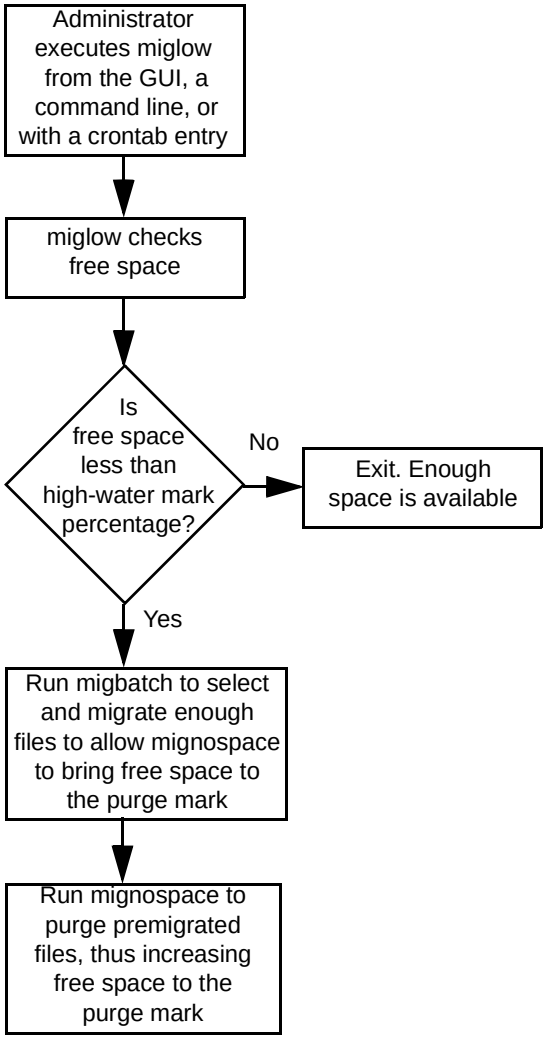
In those cases when VSM automatically detects file system free space less than the high-water mark threshold, it calls `mignospace` with the `-N` option. If there are no premigrated files to purge, this accelerates file space availability by selecting, migrating, and purging files incrementally. These increments are determined by three configurable factors: run time, number of files processed, and size of file space to free. See “Time

Increment," "File Increment," and "Space Increment" on page 35. When any of the three factors are satisfied, users can resume accessing available free space without waiting for the entire `mignospace` operation to finish. See "Accelerated File Space Availability" on page 34.

`miglow`

`miglow` checks the free space available in a managed file system (see Figure 12). If free space is less than specified by the high-water mark, `miglow` calls `migbatch` to premigrate files and then calls `mignospace` to purge them. You can start `miglow` from the GUI, from the command line, or with a `crontab` entry.

Figure 12. Checking High-Water Mark with miglow



`miglow` uses the high-water mark value specified by the `freespace` parameter in the `migconf` configuration file. See “Disabling Automatic Space Management and Caching” on page 62 for more information on using `miglow`.

User Migration Commands

Users can migrate and purge files with the `migrate` and `migpurge` commands if user permissions are enabled by the administrator.

migrate

migrate allows administrators and users to premigrate individual files from their directories. This command does not actually copy the file to secondary storage or free disk space. However, it does make a copydb file (work list) entry so that the next time migbatch executes, VSM copies the file to secondary storage.

migpurge

migpurge allows administrators and users to purge individual files from disk.

Accelerated File Space Availability

When the need arises, VSM automatically makes disk file space available according to the high-water mark and low-water mark parameters you set in migconf (see "Migration Parameters" on page 24). Whenever VSM detects ENOSPC (no space left on device) or file system free space to be less than the high-water mark threshold, it purges premigrated files that have been copied to secondary storage. Disk space only becomes available when VSM starts purging files.

- If premigrated and copied files exist, VSM purges them immediately and makes disk file space available without delay.
- If no such files exist, VSM sweeps the file system, premigrates selected files, copies them to secondary storage, and then purges the files. The delay caused by this process depends on the size of the file system being swept, the number of files selected, and the time it takes to copy data to secondary storage. This delay can be significant, but you can avoid it by keeping some premigrated and copied files available for VSM to purge immediately when the need arises.

The accelerated file space availability feature of VSM reduces the delay in freeing disk space when no premigrated and copied files exist. Rather than waiting for the entire process to run before making disk space available, VSM can optionally interrupt this process and purge files incrementally.

Note Accelerated file space availability gives users faster access to freed file space by interrupting the migration process. This makes the migration process less efficient because the destination storage media must be remounted after each interruption if files remain to be migrated.

The mignospace -N command triggers accelerated file space availability. VSM creates the */dwpath*/workdir/*hsmname*.nospace file to indicate that mignospace -N is running. The existence of this file causes VSM to interrupt migsweep and migcopy processing as soon as any one of three configurable file system attributes is satisfied. When mignospace -N is finished, this file is removed.

Time Increment

Maximum time in minutes `migcopy` runs before being interrupted to make file space available. The default is 30. A value of 0 signifies no limit. See page 135.

File Increment

Maximum number of files processed by `migsweep` and `migcopy` before being interrupted to make file space available. The default is 50. A value of 0 signifies no limit. See page 135.

Space Increment

Minimum amount of disk space (in kilobytes) freed by `migsweep` and `migcopy` before being interrupted to make file space available. The default is 1,048,576. A value of 0 signifies no limit. See page 135.

If `migcopy` terminates before the work list is complete, Media Manager will remount the destination tape when `migcopy` starts up again.

`mignospace` may run more than once to reach the high-water mark threshold.

Constant Sweeps

You can tune VSM to perform constant sweeping of the managed file system instead of the normal sweeping process. To enable constant sweeping, execute this shell script:

`/usr/openv/hsm/bin/migconsweep [-s` *`sleep_time`*`] hsmname &`

where `-s` *`sleep_time`* is the time in seconds that this command sleeps before resuming a sweep of the file system. The default is 60.

Constant sweeping attempts to keep the file list of migration candidates from becoming empty by periodically checking the list and resuming sweeping if necessary.

If `mignospace` is running when VSM sweeps the file system, the accelerated file space availability feature of `mignospace` is implemented. See “Accelerated File Space Availability” on page 34.

Constant sweeping uses system resources. This may adversely affect overall system performance, particularly during periods of heavy system usage. Once initiated, constant sweeping continues to run until the process is terminated with the `kill` command.

Multilevel Migration

Note VERITAS Storage Migrator Remote does not support multilevel migration. Neither the `ft` or `nb` methods are eligible as a source or destination level.

With multilevel migration, you can configure and manage cost-effective storage hierarchies that make the best use of your storage equipment investment.

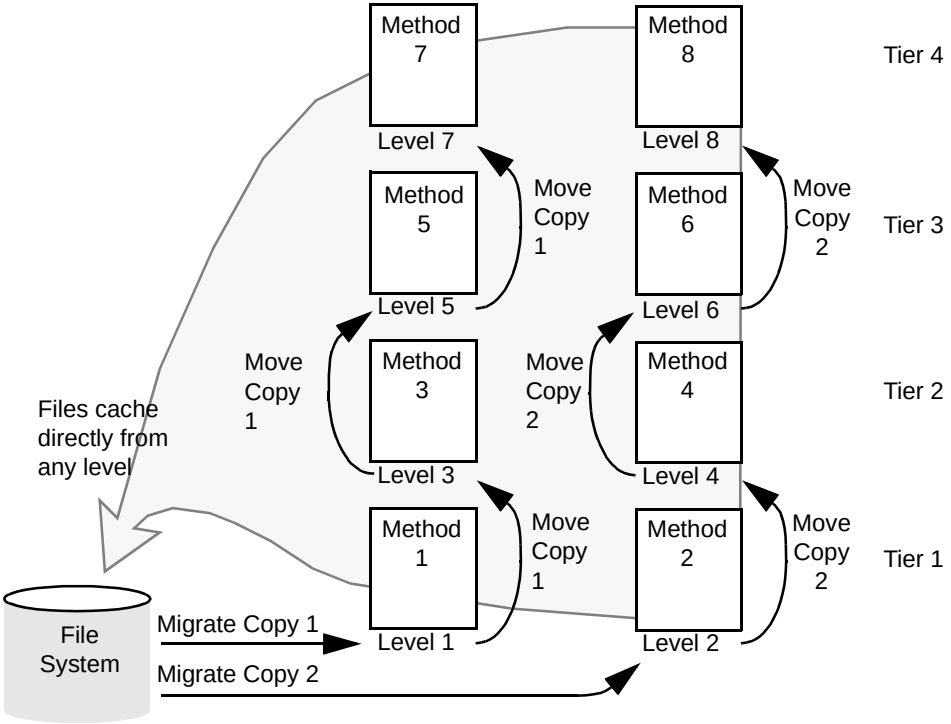
Multilevel migration is compatible with the single migration level and dual recording features of earlier releases, but adds optional migration levels up to a maximum of eight. In the default configuration the eight levels comprise four migration tiers, where four levels are associated with copy 1 and the other four are associated with copy 2. See Figure 13 on page 37.

Once configured, multilevel migration operations occur automatically on a schedule, moving files from one migration level to the next based on site-specified criteria. These operations remain invisible to users, but can reduce operating costs by retaining frequently used data close at hand while moving aging data to lower cost storage media at other levels.

You can configure multilevel migration criteria for each file system managed by VERITAS Storage Migrator. Each migration level consists of a set of storage methods defined in the configuration file for the file system. Typically the secondary storage devices at higher levels have progressively larger capacities, longer access times, and lower unit storage costs, but you are free to configure the storage methods at each level without such arbitrary restrictions.

VERITAS Storage Migrator caches migrated data back to the server disk directly without going through intervening levels. By tracking multiple copies of migrated files, VERITAS Storage Migrator caches this data from the the volume with the best volume set availability (lowest *hint* value), regardless of its migration level. If that volume is not available, VSM requests another volume.

Figure 13. Default Configuration for Multilevel Migration



The destination of initial file migration is to the lowest level. Thereafter, as migrated files age and data access becomes less urgent, VERITAS Storage Migrator automatically selects files and moves them to higher levels whenever the `migmove` command is executed. You can execute `migmove` from the GUI, from the command line, or with a `crontab` entry.

By default, VERITAS Storage Migrator marks files at the source level dead after moving them to the destination level. (As an option, you can mark source level files obsolete or have them remain active instead.) At some later time, it is possible to reclaim wasted filespace in source level volumes by consolidating active files to other volumes and removing all remaining obsolete or dead files.

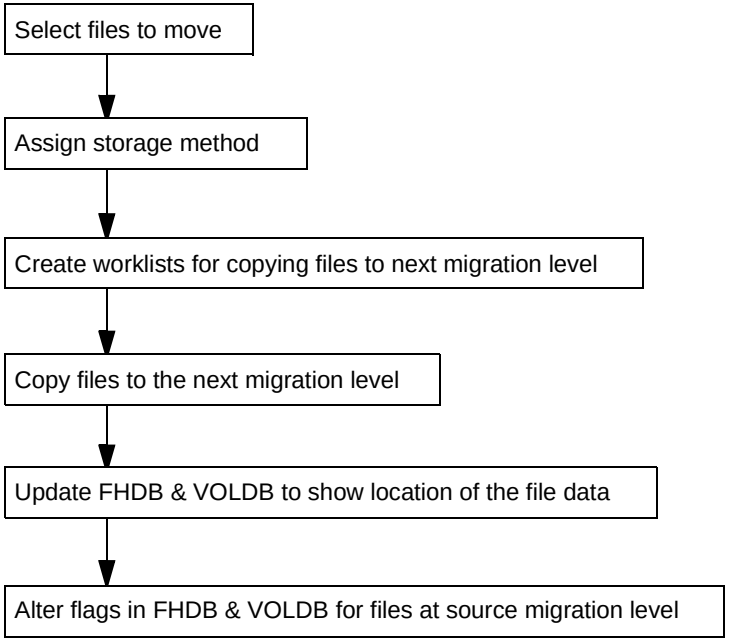
Default Configuration

The default configuration for moving files with VERITAS Storage Migrator has four migration tiers, with the odd-numbered migration levels (1, 3, 5, 7) associated with the first copy of migrated files and even-numbered migration levels (2, 4, 6, 8) associated with the second copy of migrated files.

File Movement

The `migmove` command controls file movement. VERITAS Storage Migrator selects files and then copies them to the next migration level. Figure 14 shows the main steps in this process and the following topics explain them.

Figure 14. File Movement Process



Select Files

VERITAS Storage Migrator selects files by scanning the source migration level and evaluating each file according to the file-selection criteria set during configuration. This criteria is based on file size and time elapsed since the file was migrated or moved to the source migration level if it has not been modified since. Files that meet the criteria become candidates for moving to the next migration level and are placed on a list.

VERITAS Storage Migrator selects files until there are no more files that meet the selection criteria. See “Define Criteria for Migrating Files” on page 64 for details on selection criteria.

Determine the Storage Method

VERITAS Storage Migrator reads the storage methods defined in the configuration file (`migconf`) for the destination migration level and uses those methods to copy files to that migration level. Storage methods are a combination of the method name, volume set number, volume set availability (called *hint*), and volume pool.

During configuration, the administrator configures the storage method that is most suitable for each migration level, up to a maximum of eight levels. For example, lower levels can have optical storage methods and higher levels can have tape storage methods.

See “Choose Storage Methods for Migrating Files” on page 82 for more information on storage methods and volume sets.

Create Movement Work Lists

The `migmove` command creates a work list, called a `copydb` file, for each secondary storage method and volume set configured for VERITAS Storage Migrator. `migmove` also assigns files to storage methods and writes file entries in the proper work list.

The work lists provide input for the copy processors that move files to the next migration level. VERITAS Storage Migrator stores these work lists in the working directory specified in the global configuration file (`migconfg`).

See “Work Lists (copydb files)” on page 303 for a more detailed description of work lists.

Copy to Next Migration Level

The `migworker` script initiates a copy processor for each work list (`copydb`) file. The copy processors read entries from their assigned work list and copy the indicated files to the destination migration level. If the configuration specifies concurrent recording and devices are available, different files can go to different devices simultaneously. Copy 1 and copy 2 of a file can also be moved simultaneously.

A copy processor attempts to copy every file on its work list from the source to the destination migration level and upon completion sets the status of each file operation. If the file is in premigration (method dk), the copy comes from there instead of from the source migration level.

VERITAS Storage Migrator periodically removes work list entries whose status field indicates proper completion. Copy operations that do not complete successfully are reprocessed the next time VERITAS Storage Migrator starts that copy processor.

Files are copied only once to a destination migration level.

Update File-Handle Database and Volume Database

VERITAS Storage Migrator updates the file-handle database (FHDB) and VSM volume database (VOLDB) to show the location of the moved files. The FHDB contains at least one entry for each copy of a migrated file. If VERITAS Storage Migrator divides larger files into parts called granules before writing them to secondary storage on tape or optical disc, the FHDB may contain more entries for each file copy at both source and destination levels (see “File-Handle Database Space Requirement” on page 106). Granule size is configurable.

The copy processors perform the database update by creating temporary database entries as they complete their work lists. When a copy processor completes its work list, `migworker` calls `migsetdb` to update the FHDB and then calls `migmerge.sh` to merge in the temporary entries.

Clean Up FHDB and VOLDB Flags

By default, VERITAS Storage Migrator marks files at the source level dead after moving them to the destination level. Other options are available with `migmove`. Volumes containing large numbers of obsolete and dead files can be consolidated and recycled as new volumes. See man pages migcons(1M) and migsetdb(1M) for more information.

File Export/Import

Note VERITAS Storage Migrator Remote does not support export/import.

The export/import feature of VERITAS Storage Migrator makes it possible to transfer migrated files from one VSM managed file system to another VSM managed file system with the same storage methods. A like server (same platform/OS) is needed for optical volumes. Only tape and optical volumes managed by Media Manager can be exported. The export/import operation does not require file caching, and can process files of any size (including those greater than 2GB).

For example, if an organization begins working on a project at one office but later transfers that project to a second office in another city, the administrator at the first office can export to tape all the migrated files for that project. The administrator at the second office can import these migrated files, adding them to a VSM managed file system already running at the new location.

VERITAS NetBackup is required for file export/import. The `mignbexport` command identifies all VSM volumes containing migrated files to be exported, and backs up all unmigrated files to be exported to NetBackup volumes. `mignbexport` also backs up FHDB and VOLDB data for all files being exported and then deletes these FHDB and VOLDB entries from the exporting VSM managed file system. The exporting administrator removes the VSM volumes containing the migrated files and the NetBackup volumes containing the unmigrated files and database entries from their storage devices and sends them to the new location.

The import process is the reverse of export. The importing administrator places the VSM volumes containing file data to be imported and the NetBackup volumes used in the `mignbexport` operation in the appropriate storage devices, and registers them with Media Manager for the importing VSM managed file system. The importing administrator uses NetBackup's import feature to make NetBackup at the importing site aware of the new data. The `mignbimport` command uses the NetBackup `restore` command to load this data and updates the FHDB and VOLDB for the importing VSM

managed file system to include this information about all files being imported. `mignbimport` now uses the NetBackup `restore` command to restore the imported files into the the VSM managed file system at the new location.

Note Imported files lose any designations they have as tied files, and must be redesignated. See the *Storage Migrator User's Guide*.

For more information on how to plan export/import operations, see "VSM Export/Import Management" on page 298. Also see man pages mignbexport(1M) and mignbimport(1M).

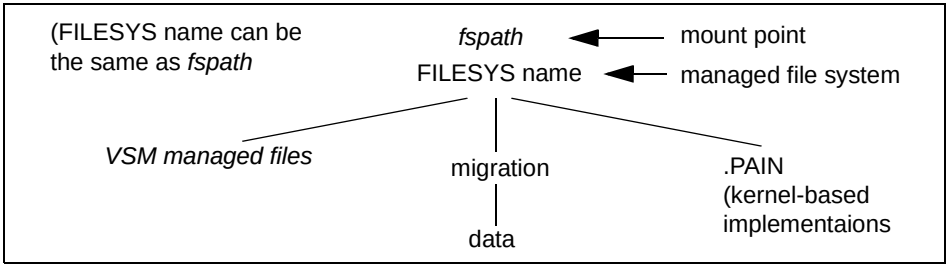
VSM System Files and Directories

Managed Server Files and Directories

Files in the VSM managed file system or managed directory fall under the *fspath* pathname. For examples, see "File Systems to Manage" on page 58.

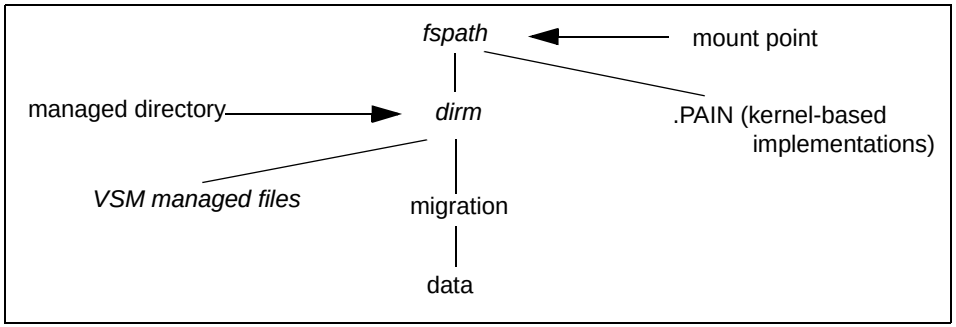
The managed file system structure is shown in Figure 15.

Figure 15. Managed File System Structure



VSM can also manage a directory, *dirm*, in a mounted file system at *fspath*. A managed directory must be below the file system mount point. The managed directory structure is shown in Figure 16.

Figure 16. Managed Directory Structure



fspath

Path that leads to the file system or directory that VSM is managing. It is the mount point for the managed file system. VSM can manage one or more file systems or directories. You specify each *fspath* during configuration. See “Mount Point” on page 125.

fspath`/migration/data` or *fspath*`/dirm/migration/data`

On Solaris *ufs* file systems, this directory contains premigrated data. During migration, VSM premigrates files to this directory and then copies them to secondary storage. The premigrated copies remain in this directory until VSM has to purge them in order to free file system space.

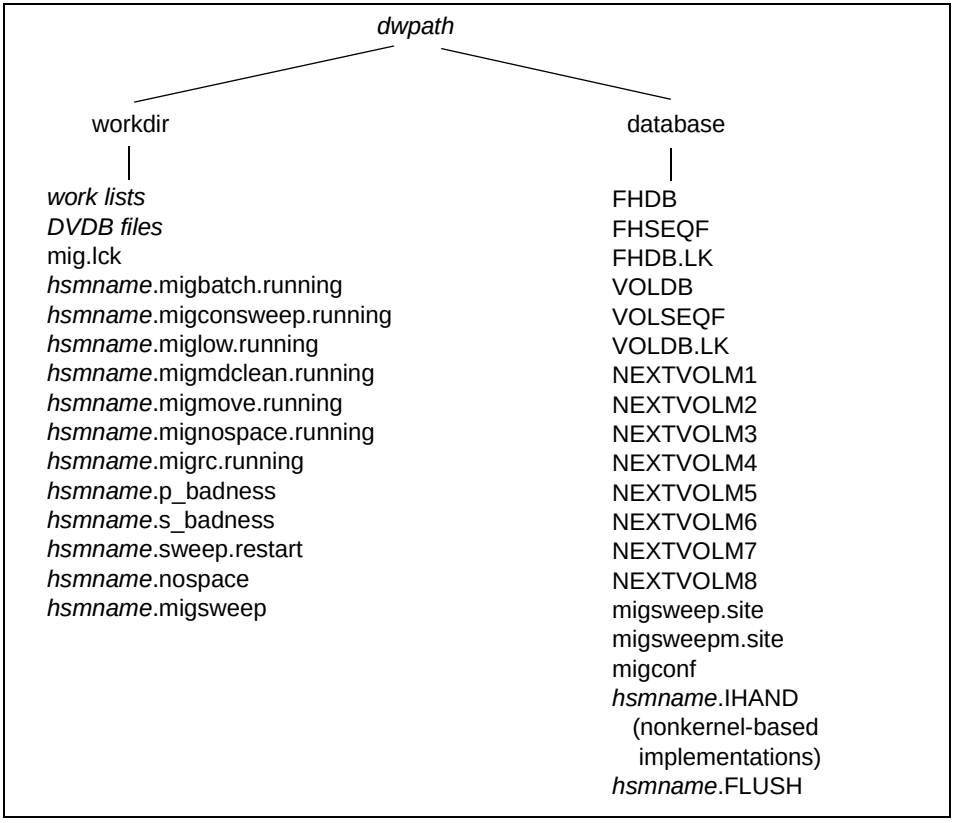
On all nonkernel-based implementations, this directory is used by the `nb` method for migrating files to NetBackup.

fspath`/.PAIN`

.`PAIN` file: Contains parallel inode information about migrated files. (kernel-based implementations, Solaris *ufs* file systems)

Databases and workfiles for the VSM managed file system or directory fall under the *dwpath* pathname. If VSM manages more than one file system or directory, specify each *dwpath* during global configuration.

Figure 17. Database and Workfile Structure



dwpath

Path to the database files for a managed file system. You specify this path for each entry in the `migconfg` file. The default is `/usr/var/openv/hsm.`*hsmname*.

dwpath`/workdir`

Contains the work lists that VSM uses to copy premigrated files to secondary storage and to move migrated files from source to destination migration levels. The names of these files have the form:

hsmname`.copydb.`*method_name*`.`*vol_set_number*`.`*hint*

For example:

```
alpha.copydb.ad.1.library
```


The work directory also contains DVDB (Destination-volume database) files. VSM creates these files to store the temporary file-handle database entries that it creates during copy operations. After merging the temporary entries into the file- handle database, VSM deletes the DVDB file.

Note See "Databases on a Managed Server" on page 300 for a more detailed description of work lists and the DVDB.

The work directory can also contain the following files:

`mig.lck`: Used by `migbatch` and `mignospace` to lock the managed filesystem while sweeping and premigrating files.

hsmname`.migbatch.running`: Contains process ID (pid) of `migbatch` if that process is running.

hsmname`.migconsweep.running`: Contains process ID (pid) of `migconsweep` if that process is running.

hsmname`.miglow.running`: Contains process ID (pid) of `miglow` if that process is running.

hsmname`.migmdclean.running`: Contains process ID (pid) of `migmdclean` if that process is running.

hsmname`.migmove.running`: Contains process ID (pid) of `migmove` if that process is running.

hsmname`.mignospace.running`: Contains process ID (pid) of `mignospace` if that process is running.

hsmname`.migrc.running`: Contains process ID (pid) of `migrc` if that process is running.

hsmname`.p_badness`: Contains the current purge badness value.

hsmname`.s_badness`: Contains the current badness value.

hsmname`.sweep.restart`: Contains the path of the last file selected by `migsweep` before reaching the low-water mark.

hsmname`.nospace`: Indicates that `mignospace` is running with the `-N` option. (File is removed when `mignospace -N` is not running.)

hsmname`.migsweep`: A list of files selected to be premigrated.

dwpath/database

Contains the following database information:

FHDB: File-handle database. This database contains entries for each copy of a migrated file. If VSM divides a file into granules (smaller parts) during migration then this database has an entry for each granule. Granularity depends on the media used to store the data and allows VSM to divide the files into manageable parts.

FHSEQF: File-handle-sequence file. Contains the six-digit hexadecimal value that VSM assigns to the next file handle.

FHDB.LK: File-handle-database lock file. The VSM database merge process uses this file to provide a master lock on the file-handle database.

VOLDB: VSM Volume database. This database contains an entry for each volume that you register with VSM.

VOLSEQF: Volume-sequence file. Contains the six-digit hexadecimal value that VSM assigns to the next volume ID (handle).

VOLDB.LK: Volume-database lock file. The VSM database merge process uses this file to provide a master lock on the volume database.

NEXTVOL1: Next volume set to use for the first copy of a migrated file.

NEXTVOL2: Next volume set to use for the second copy of a migrated file (if applicable).

NEXTVOL3 ... NEXTVOL8: Next volume set to use for moving a migrated file to migration level 3-8 (if applicable).

migsweep.site: A program which allows the site to provide their own migration and purge criteria control for files.

migsweepm.site: A program which allows the site to provide their own move criteria control for files (if applicable).

migconf: Configuration file containing migration criteria for the file systems using this database. You set up this file during the configuration process.

hsmname.IHAND: Contains inode and handle information about migrated files (nonkernel-based implementations).

hsmname.FLUSH: Controls how often VERITAS Storage Migrator writes tape marks when making copies during file migration.

Note See "Databases on a Managed Server" on page 300 for a more detailed description of the files in *dwpath*/database.

Logfiles for the VSM managed file system or directory fall under the *lgpath* pathname. If VSM manages more than one file system or directory, specify each *lgpath* during global configuration. The global log file is in `/tmp/hsm.log`, and it is not configurable.

lgpath

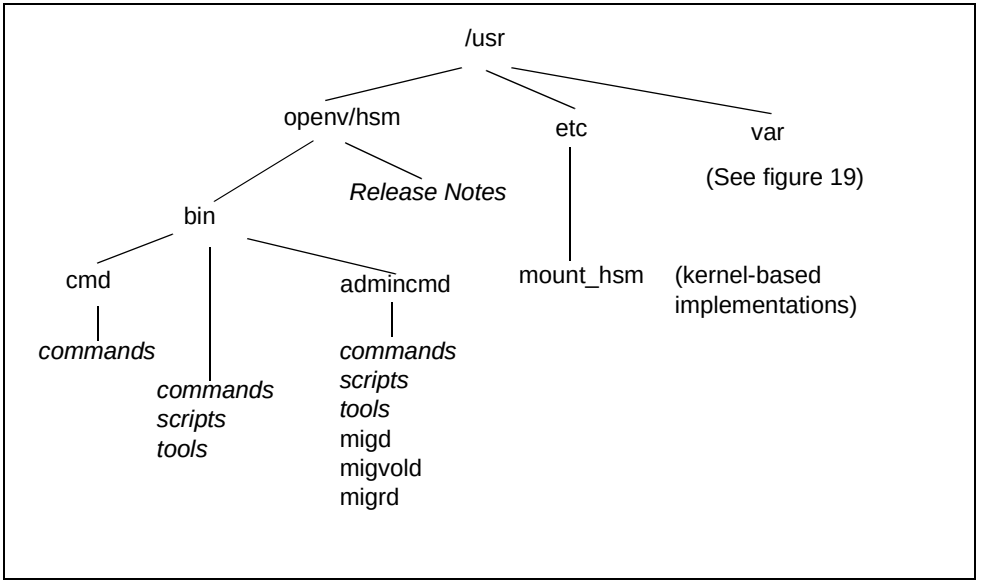
Log for a specific HSMDEV configuration. The name and path is configurable. You specify the path and file name during configuration. The default is `/tmp/hsm.hsmname.log`.

`/tmp/hsm.log`

Global log file to which VSM logs all messages. It is not configurable.

Binaries for VSM reside in the `/usr` directory. See Figure 18. Global configuration information resides in the `/usr/var/openv/hsm` subdirectory. See Figure 19 on page 47.

Figure 18. Binary File Structure



`/usr/openv/hsm/bin`

This directory contains the commands that you execute from the command line. The actual executables used by VSM reside in the subdirectories `/admincmd` and `/cmd`.

`/usr/openv/hsm/bin/admincmd`

This contains the actual executables for some of the commands, scripts, and tools found in `/usr/openv/hsm/bin`. Also included are the VSM daemon (`migd`), the volume daemon (`migvold`), and the Java daemon (`migrd`).

`/usr/openv/hsm/bin/cmd`

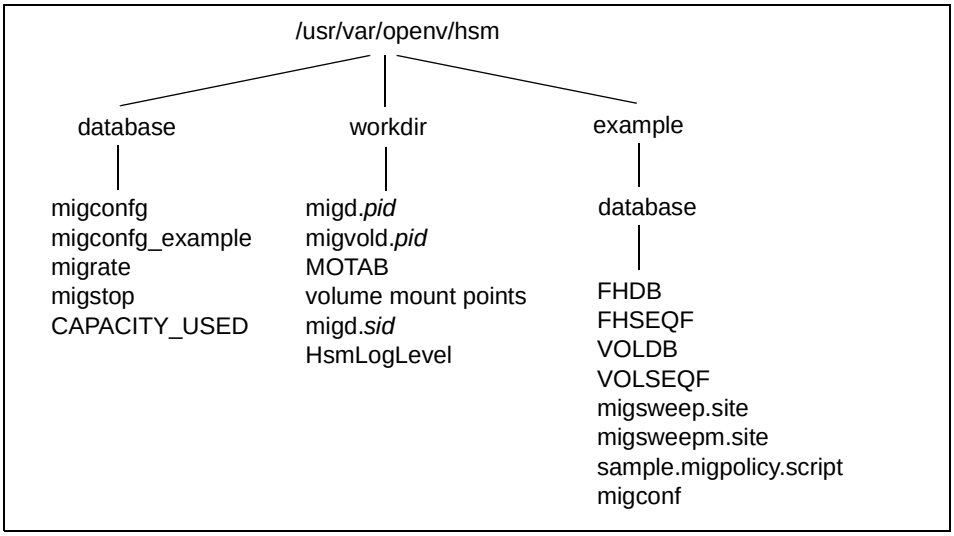
This contains the actual executables for some of the commands found in `/usr/openv/hsm/bin`.

`/usr/etc`

For kernel-based implementations, this directory contains the following file:

`mount_hsm`: Contains the mount processor for the hsm filesystem (Solaris *ufs* file systems only).

Figure 19. Global Configuration Binary File Structure



`/usr/var/openv/hsm/database`

This directory contains the following files:

`migconfg`: VSM global configuration file. During configuration, you create an entry in this file for each file system that you are managing on your system.

`migconfg_example`: A template VSM global configuration file.

`migrate`: Global migrate file, containing a list of the files or directories of files that VSM will migrate during automatic migration.

`migstop`: Global stop file, containing a list of the files that VSM will not migrate.

`CAPACITY_USED`: The calculated current capacity in bytes of VSM storage media used, with reference to licensed capacity.

`/usr/var/openv/hsm/workdir`

This directory contains the following files:

`migd.`*pid*: The process ID (pid) of the VSM migration daemon.

`migvoldb.`*pid*: The process ID (pid) of the VSM volume daemon.

`MOTAB`: The global VSM mount table.

volume mount points in the form: *hsmname*`.`*volume id*`.[W|R]`

`migd.`*sid*: The session ID (sid) of the VSM migration daemon.

`HsmLogLevel`: The level of logged VSM messages.

`/usr/var/openv/hsm/example/database`

Contains template VSM database files.

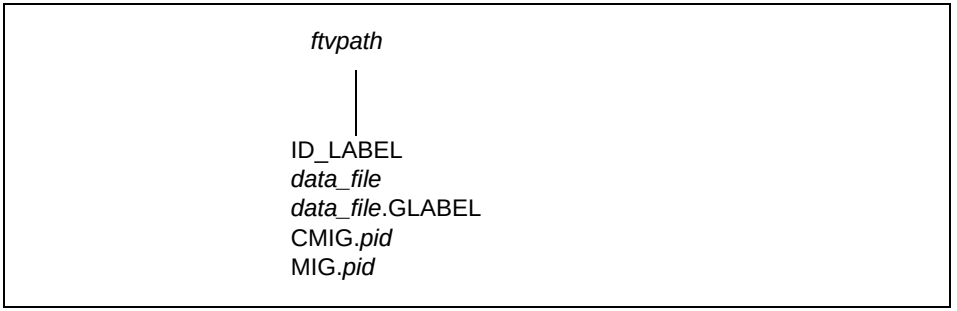
```
FHDB
FHSEQF
VOLDB
VOLSEQF
migsweep.site
migsweepm.site
migconf
sample.migpolicy.script
```

Note After creating your own `migsweep.site` or `migsweepm.site` file, move it to its proper location in the database directory for the VSM managed file system. After creating your own `migpolicy.script` file, move it to `/usr/openv/hsm/bin/admincmd/migpolicy.script`, replacing the file that is there with the new script keeping the same name.

Remote Volume Server Files and Directories

Figure 20 shows the key VSM system files and directories that reside on a remote volume server in a VSM configuration.

Figure 20. Remote Volume Server Files and Directories



ftvpath

Path to the file system containing the ft volume. This file system contains the following types of files:

`ID_LABEL`: Each remote file system containing an `ft` volume has an `ID_LABEL` file. This file contains a single line of text that identifies the label and the client name for the managed server using the volume.

Note See "Databases on a Remote Volume Server" on page 306 for a more detailed description of the ID_LABEL file.

data_file: Each migrated file has a unique id.

data_file . `GLABEL`: There is a `GLABEL` file for each migrated file. The `GLABEL` files contain information pertaining to the migrated file.

CMIG.*pid*: Stage list files being created.

MIG.*pid*: Stage list files available for use.

Note If VSM is running on a remote volume server, use the global stop file to prevent the migration of any *data_file* . `GLABEL` files from that remote server. See "Global Migration Control" on page 290. This will expedite the cleaning of remote file systems by removing obsolete files from `ft` volumes to reclaim filespace. See man page migmdclean(1M). See "Remote Method" on page 85 for more information.

Planning VSM Configuration 2

You should assess the migration requirements for your site and develop a migration plan before configuring VSM. This ensures the most effective use of VSM and the most effective migration procedures. It also makes the configuration process much easier.

This chapter explains factors to consider and steps to perform during each part of the planning process. It also develops a plan based on an example site and contains worksheets you can use for developing your own plan.

Note The worksheets shown in this chapter can be found in “Planning Worksheets” on page 525.

The major topics covered in this chapter are as follows:

- Overview of VSM Configuration
- Example Site
- Plan the VSM Global Configuration
- Define Criteria for Migrating Files
- Define Criteria for Purging Files
- Define Criteria for Moving Migrated Files
- Choose Storage Methods for Migrating Files
- Choose Storage Methods for Moving Files
- Choose Directories for Databases
- Scheduling Migrations
- Completing the Example Configuration Plan

The best approach to configuring VSM is to read this chapter before starting to configure the product. This allows you to work out problems before committing to values in the actual configuration. When you have read what you need from this chapter and are ready to configure your system, proceed to chapter 3 , for the Motif-based interface, or to chapter 4, for the Java-based interface, for configuration instructions and examples.

Overview of VSM Configuration

Before starting your plan, study and understand the overall structure of a VSM configuration. You do not have to know the actual configuration procedures, which are covered in chapter 3.

Figure 21 shows the two levels of VSM configuration:

- Global (`migconfg`)
- VSM-specific (`migconf`)

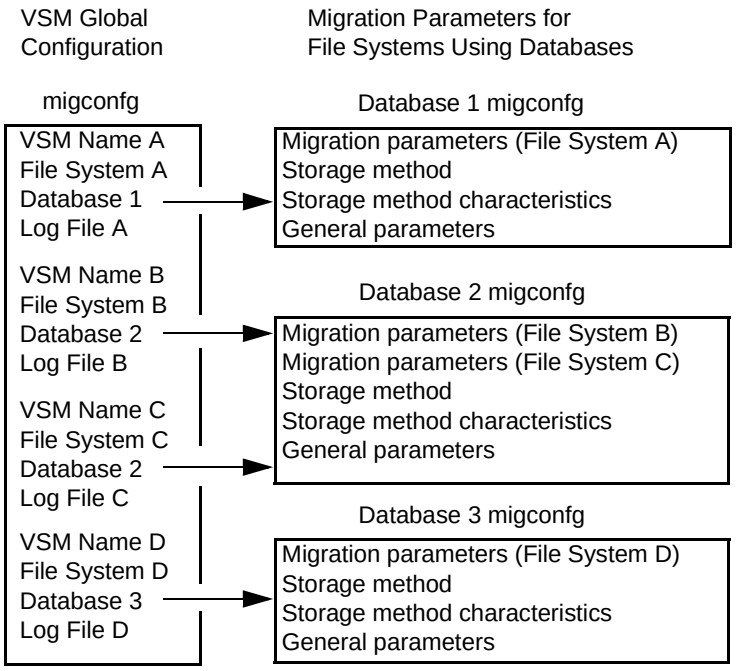
Global Configuration Overview

The global configuration defines a named collection of managed file systems along with their migration attributes. You set up the global configuration in the `/usr/var/openv/hsm/database/migconfg` file. Within the global-configuration file, you create HSMDEV entries that specify the managed file systems. You can choose to configure only one HSMDEV entry and manage one file system or you can configure several HSMDEV entries with each specifying a different file system.

The main parameters for an HSMDEV entry are as follows:

- Name of the individual VSM configuration (unique for each entry).
- File system that VSM manages with this entry (unique for each entry).
- Path to the database directory that VSM uses to control and store migration information about this entry's file system.

Figure 21. VSM Configuration Structure



- Path to the log file in which VSM stores its status and error information about operations performed on this entry's file system and databases (more than one HSMDEV entry can specify the same log file).
- Management state of the file system. If Active (1), VSM provides automatic disk-space management.

VSM-Specific Configuration Overview

In addition to global configuration, you also must set up a configuration file, named `migconf`, within each database directory that you define during global configuration. A `migconf` file includes the following types of parameters:

- Names of file systems using the database along with disk-space management parameters for those file systems. When more than one HSMDEV entry uses the same database for its file system, the `migconf` file must contain a separate set of management parameters for each file system. Also, the common parameters such as storage methods must be appropriate for all file systems. On Figure 21, HSMDEV entries B and C both point to database 2.

Use a separate database for each file system to prevent the database's FHDB (file-handle database) from getting large enough to adversely affect VSM performance. The FHDB contains at least one entry for each copy of a migrated file. See "File-Handle Database Space Requirement" on page 106.

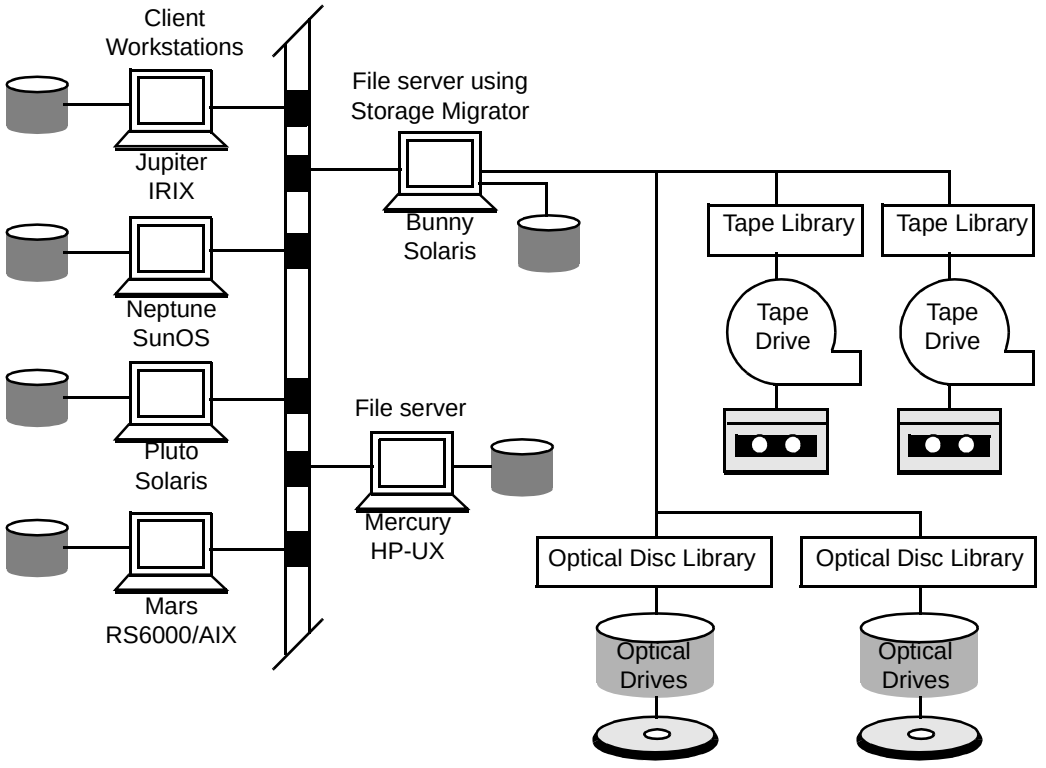
- Methods for storing each copy of files migrated from any file system using this database, and for moving migrated files to another migration level.
- Characteristics of each type of storage method (for example, capacity and access time of the media).
- General parameters for controlling migrations of files from file systems using the database. For example, you can list files that VSM will exclude from migration.

Example Site

Figure 22 on page 55 shows an example of a department level network at a site that uses VSM. This department has four workstations and two file servers. The server named `mercury` provides file service for the workstations. The server named `bunny` provides file service and backup for both this and other departments at the site.

In the past, this site has frequently added more disk storage to `mercury` and `bunny` in order to keep up with a growing number of users and workstations. Now, by using VSM to manage `bunny`, this site can migrate files from both `mercury` and the workstations to `bunny` and increase `bunny`'s storage as necessary with lower-cost, high-capacity storage devices such as tape libraries and optical robots.

Figure 22. Example VSM Site



Plan the VSM Global Configuration

It is important to carefully plan your global configuration because it affects all the other steps that you perform. It can also have an important impact on VSM and server performance.

The topics in this section explain the following aspects of planning the global configuration:

- Choose file systems to manage.
- Choose directory in which to locate the log file for each managed file system.
- Decide whether to enable automatic VSM management of each file system.

Selecting the database directory to use for each managed file system is also part of global configuration. However, you cannot actually make that choice until after you know the migration requirements for your file systems. Therefore, these are explained later in “Choose Directories for Databases” on page 99.

The worksheet on Table 2 (below) summarizes the types of data you need to complete the VSM global configuration.

Choose the File Systems to Manage

VSM allows you to manage standard UNIX file systems. The actual file system types that VSM supports depends on the type of server.

As used in this manual, the term *managed file system* refers to the file system containing the set of directories that VSM searches for files to migrate (these file systems are also mounted as hsm type file systems). For example, if you use VSM to migrate files from the `/home1` file system and its underlying directories then `/home1` is the managed file system.

Before you can choose the file systems to manage, you must gather some information about all the file systems that currently exist on the server. Table 2 shows a list of the file systems on `bunny` at the example site, and includes the types of information that will be useful in evaluating your migration needs. The `du` and `df` commands give you most of the information in this table.

Table 2. Example File system Planning Information

Host	File System	Capacity (Mbytes)	% Used
Bunny	/	47	72%
Solaris	/var	150	30%
	/home1	1160	80%
	/home2	450	60%
	/dbrecs	2000	90%
	/sd3	752	90%

Figure 23. Global Configuration Worksheet

Global Configuration Worksheet

VSM Name: _____

Mount Point: _____

Database Pathname _____

Logfile Pathname: _____

State: _____

VSM Name: _____

Mount Point: _____

Database Pathname _____

Logfile Pathname: _____

State: _____

VSM Name: _____

Mount Point: _____

Database Pathname _____

Logfile Pathname: _____

State: _____

VSM Name: _____

Mount Point: _____

Database Pathname _____

Logfile Pathname: _____

State: _____

The following topics discuss areas that you need to consider when selecting file systems to manage with VSM. The last topic is a list of the managed file systems for `bunny` at the example site.

- File Systems to Manage
- File Systems Not to Manage
- Considering the Number of Files
- Managed File Systems at Example Site

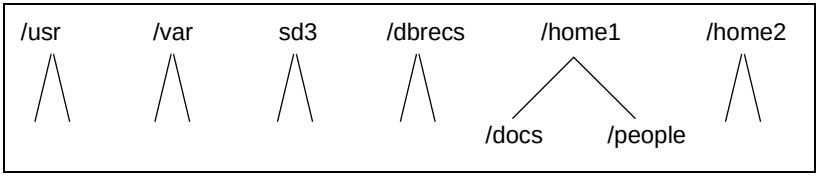
File Systems to Manage

After gathering information on the file systems, the next step is to identify those that you want to manage with VSM. The best way to accomplish this is to first examine the file systems that are most likely to have space problems. For example, this typically includes `/home` because it grows so rapidly.

When you designate a file system, you can include either all or part of it. The path name to the set of directories and files that you are going to manage can start either at the top (mount point) or at any lower directory level. However, you can have only one managed directory below each mount point.

Figure 24 shows a part of the directory tree for `bunny` at the example site. In this example you can manage the entire `/home1` file system or only a certain managed directory, such as `/home1/docs`. You can also manage other file systems such as `/dbrecs` and `/sd3`. However, if you manage the `/home1/docs` directory you cannot also manage the `/home1/people` directory because they are both under `/home1`.

Figure 24. Example Directory Tree



It is best to manage the entire file system. If you want to manage only part of a file system, use quotas to restrict the amount of space the unmanaged part can use. Otherwise, the unmanaged part can use enough space to interfere with VSM's ability to keep free space available.

VSM manages disk space in the managed file system or managed directory associated with the mount point configured in `migconf`. See "Perform Global Configuration" on page 123. If another file system is mounted in the VSM managed file system or managed directory, it will not be managed by VSM.

File Systems Not to Manage

VSM cannot manage the following:

- Root (/) or any subdirectory in the root file system
- NFS mounted file systems
- Raw disk partitions, including swap partitions
- `/tmp` file system

You also want to avoid migrating files that are critical to system operation or those in file systems that do not grow in size enough to cause problems. For example, be careful if you migrate from `/usr`. This file system can contain information essential to system startup and operation.

Note VSM executables reside under `/usr/openv/hsm/bin`.

All your managed file systems will contain some files that you do not want to migrate. Some specific types of files include:

- Temporary files
- Core files
- History files
- Files labelled junk or testing
- `.login`, `.cshrc`, and other `.` (dot) files

Administrators and users can list specific files, those VSM will migrate or those VSM will prevent from migrating, in special VSM control files. See the *Storage Migrator User's Guide* and "Global Migration Control" on page 290 for more information on how to create and use these VSM control files.

Considering the Number of Files and Inodes

The number of files in the managed file system affects the time required to select files for migration. As the number of files increases, so does the selection time.

How many files you expect to migrate from the file system is also an important consideration because it affects the cache time. As the number of migrated files grows, so does the file-handle database and therefore the time to process cache requests. In addition, the number of migrated files affects the number of inodes and space required for slices:

Inodes

In kernel-based implementations (Solaris *ufs* file systems), VSM uses an inode for the premigrated file in addition to the one occupied by the original file. Similarly, caching requires an additional inode. Therefore, a VSM-managed file system must have an

extra inode available for *each file* that you expect to migrate, although the extra inode is needed only while the file is in premigration. You set the number of inodes when you create the file system.

Slice space

VSM optionally retains a configured slice of data on disk from the head of each file it migrates, thus allowing this number of bytes to be read without caching the file. During configuration, you use the `slice` parameter to set the number of bytes that VSM stores in a slice. This number can range from 0 (default) to 65536 (64 kilobytes). On DMAPI implementations, the upper limit to the slice is 2147483648 (2 gigabytes).

The file system must have enough extra space to allow for the configured slices that will reside on disk. For example, if you expect to maintain 1000 files in a migrated state and set slice to the default 8 kilobytes, you need 8 megabytes just to store the slice data. VSM continues to use this space for the slice even after the file (including the slice data) is completely migrated to secondary storage.

For kernel-based implementations (Solaris *ufs* file systems), you must also allow space for the `.PAIN` (Parallel Inode) file. This file contains a 40 byte entry for every inode in the file system. VSM stores the status of migrated files in these entries. The size of a `.PAIN` file for a file system with *N* inodes is 40 x *N* bytes. See "Configure Servers" on page 126.

Note For nonkernel-based implementations, you must allow space for the *hsmname*`.IHAND` (Inode to Handle) file outside of the managed file system. This file is a sparse file with 48 bytes per inode maximum size.

Managed File Systems at Example Site

Table 3 on page 61 shows the list of managed file systems for `bunny` at the example site and the name of the HSMDEV entry that specifies each file system. The list also includes the following:

- Device on which each file system resides (in parentheses below file system)
- Total space available in each file system
- Amount and percentage of space currently used
- Total number of inodes
- Number and percentage of inodes currently used in each file system
- Access frequency (average time between file accesses).

This additional information is useful in determining your database and media requirements, as explained later in this chapter. A subject not mentioned in Table 3 is the fill rate for each file system. This is the average size and number of new files that users create and is useful in determining the frequency with which you must migrate files.

It is important to remember that this is a preliminary list. As you progress through the planning process, you can alter the list to optimize VSM performance. For example, you can use a larger number of smaller file systems in order to reduce the size of the file-handle databases. The reason for creating the list now is to provide a basis for those other considerations.

Table 3. Example Managed File Systems

		File System Capacity						
		File Space			Inodes			
HSMDEV Name	**Managed File System**	**Total**	**Used**	**%Full**	**Total**	**Used**	**%Used**	**Access Frequency**
`hsm1`	`/home1` (partition1)	1160	930	80%	200	50	25%	6 days
`hsm2`	`/home2` (partition2)	450	270	60%	200	16	8%	5 days
`hsm3`	`dbrecs` (partition3)	2000	1800	90%	200	1	.5%	30 days
`hsm4`	`/sd3` (partition4)	752	276	90%	578	450	78%	10 days

Note File system space is in megabytes and inodes are in thousands. Also, the device on which the file system resides is shown in parentheses under the Managed File System.

Choose Path for Log File

As VSM performs operations, it writes informative messages to a log file that you define for the file system. You specify the path to this log file in the HSMDEV entry. By default, the path name to each log file is as follows:

`/tmp/hsm.`*hsmname*`.log`

where *hsmname* is the name you assign to the HSMDEV entry (for example, `hsm1`). Always change it to something other than `/tmp`, since files in `/tmp` can disappear after a reboot or system crash.

Note VSM also keeps a global log file named `/tmp/hsm.log`, but this name is not configurable (see “Checking and Managing the Logs” on page 307).

The table below shows the HSMDEV log file paths for bunny at the example site.

Table 4. Example HSMDEV Log File Paths

VSM Name	File System Pathname	Log File
hsm1	/home1	/var/logs/hsm1.log
hsm2	/home2	/var/logs/hsm2.log
hsm3	/dbrecs	/var/logs/hsm3.log
hsm4	/sd3	/var/logs/hsm4.log

Define the Management State for File Systems

The *state* parameter in the /usr/var/openv/hsm/database/migconfg file allows you to either enable or disable automatic space management and caching of migrated files. See "Disk-Space Management" on page 24 for more information on the commands mentioned here.

Enabling Automatic Space Management and Caches

Setting state to Active (1) enables automatic space management and caching:

- If file-system free space is less than the high-water mark percentage and a user process writes to the file system, the following occurs:
  - **a.** The UNIX kernel informs the migration daemon migd that the file system needs more space.
  - **b.** migd starts the mignospace command.
  - **c.** mignospace removes premigrated files, increasing free space to the purge mark percentage. If there are no premigrated files, then mignospace migrates files to secondary storage and purges the premigrated copies until the percentage of free space increases to the purge mark percentage. Files are selected for migration according to their size and how much time has elapsed since they were last accessed.
- If a user accesses a migrated file VSM automatically caches it back to its original directory.

Disabling Automatic Space Management and Caching

Setting state to Inactive (0) disables both automatic space management and caching. However, you can still manually manage disk space by periodically running migbatch to premigrate files and then mignospace to purge the files from the disk.

You can also use the `miglow` command. If file-system space is above the high-water mark and you execute `miglow`, this command calls `migbatch` to premigrate files and then `mignospace` to remove those files. This approach is desirable if, for some reason, you want to disable automatic caching but continue migrations by periodically running `miglow`.

For kernel-based implementations (Solaris *ufs* file systems), another option is to allow automatic caching but manually manage disk space. To accomplish this, you leave the state parameter set to Active then you use the `migthreshold` command to change the kernel's high-water mark threshold to 0 or some other low percentage. This leaves automatic caches enabled (because state is Active) but changing the kernel's threshold ensures that the kernel does not detect a low-space condition. This approach works because `miglow` and the kernel actually obtain their high-water mark values from different places.

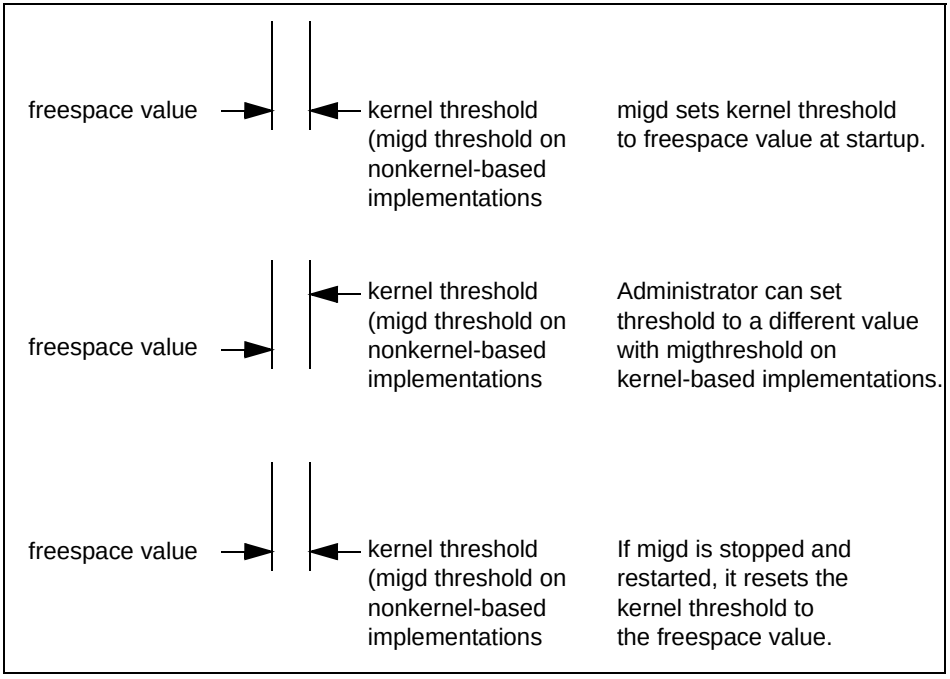
For nonkernel-based implementations, the migration daemon `migd`, not the kernel, detects low space conditions:

- `miglow` always uses the high-water mark percentage specified by the `freespace` parameter in the `migconf` configuration file.
- For kernel-based implementations (Solaris *ufs* file systems), the kernel uses a different percentage, called the *threshold* value, for its high-water mark. Whenever `migd` starts, it sets the kernel `threshold` to the same percentage as `freespace` and at this point the kernel and `miglow` have the same high-water mark. However, `migthreshold` allows you to change the kernel high-water threshold if you want the kernel and `miglow` to initiate migration at different points (see Figure 25).

  A subsequent stop and then restart of `migd` (as during a reboot), resets the kernel threshold to the `freespace` value. This also happens when you use the Java-based or Motif-based interfaces to change global configuration values and it sends a KILL-INT to `migd`. After saving `migconfg`, therefore, you need to reissue the `migthreshold` command. The one exception is when freespace is 0, in which case the kernel threshold remains unchanged.

  See the `migthreshold` man page for more information on options for kernel-based implementations (Solaris *ufs* file systems).

Figure 25. Threshold Value



Define Criteria for Migrating Files

After deciding on which file systems to manage, you can determine and specify criteria for migrating files from those file systems. The values that you choose must be such that VSM is able to keep space available on the disk at all times.

The migration criteria consists of:

- High-water mark (when to start migration); also expressed as free space
- Low-water mark (when to stop migration)
- Purge mark (when to stop purging premigrated files)
- File-selection criteria (which files to select for migration)

If you are configuring VSM for the first time, the best approach to defining the migration criteria is as follows:

1. Read the topics in this chapter.
2. Leave all values at the defaults unless the default will obviously cause problems. For example, you can change the low-water mark (explained later) to provide a limit for the number of files that VSM premigrates or removes.

3. Evaluate VSM operation at the default values to see how well it satisfies your migration and caching requirements. If you decide to vary your settings from the defaults, know the following before making any changes:

   - Total file-system space and how much is subject to migration. This is necessary in order to know how much actual space is represented by the high-water mark, low-water mark, and purge mark percentages.

Note VSM does not select files listed in the user's `.migrate` files until after the low-water mark is reached. Therefore, the percentage of free space after a migration can be larger than expected.

   - Average number and size of files that users create each day. This indicates how fast your file system can fill up. It also indicates how many files to premigrate in order to provide users enough space for their daily work.
   - Approximate size and age distribution of the files you migrate.
   - Selection of slice value to determine how much disk space will be occupied by migrated files even after they have been purged.
   - When and how often you must execute migrate operations to ensure that disk space is available for users. If possible, migrate files daily. See "Scheduling Migrations" on page 107 for information on scheduling.
   - Any special site requirements for retaining certain files in premigration.

4. Adjust the configuration to achieve the desired performance by following the guidelines in this chapter and using information from step 3.

The planning worksheet on Figure 26 on page 66 has areas in which you can record the data you need to set the migration thresholds and file selection criteria. for each managed file system. The lower part of the worksheet applies to the method for storing the migrated files and is explained under "Choose Storage Methods for Migrating Files" on page 82.

Figure 26. Managed-File-System Planning Worksheet

Managed-File-System Planning Worksheet

VSM Name _____ File System Name (mount point)_____

File System Thresholds for Migration

High-water Mark for this file system: _____%

Purge Mark for this file system: _____%

Low-water Mark for this file system: _____%

File Selection Criteria for Migration	File selection Criteria for Purging
Migrate files with Badness ≥ _____	Purge files with Badness ≥ _____
Age weight: _____	Age weight: _____
Size weight: _____	Size weight: _____
Weight operator: _____	Weight operator: _____
Min. age to migrate (days): _____	Min. age to purge (days): _____
Min. size to migrate (kbytes): _____	Min. size to purge (kbytes): _____

Best Storage Methods for File System

Copies: _____ (1 or 2) METHODn: `op.1.library/ct.3.vault/...`

METHOD1: _____

METHOD2: _____

METHOD3: _____

METHOD4: _____

METHOD5: _____

METHOD6: _____

METHOD7: _____

METHOD8: _____

High-Water Mark

The purpose of the high-water mark is to inform VSM that the file system is getting too full to satisfy user storage requirements. You express the high-water mark in terms of the percentage of free space remaining in the file system. the default is 10 percent.

A general guideline for setting the high-water mark percentage is to set it high enough so that if space used is at the high-water mark, there is still enough free space left for the largest file that you expect users to cache or create. If a user attempts to create a larger file, the kernel informs `migd`, which in turn starts the `mignospace` command to make space available.

For most systems, the default high-water mark of 10% free space remaining is adequate for satisfactory VSM operation. However, there are conditions that require you to change the percentage.

For example:

- You can increase the high-water mark percentage in a small file system where users cache or create large files. To illustrate this, 10% of `/home1` on `bunny` at the example site is 116 Mbytes and is adequate even for a 100 Mbyte file. However, 10% of `/home2` is only 45 Mbytes and a 100 Mbyte file is too large for the free space above the high-water mark.
- You can decrease the percentage if 10% is more than enough space. Decreasing the percentage has the advantage of allowing more files to reside on the disk.

Note On kernel-based implementations (Solaris *ufs* file systems), do not set the high-water mark to 0.

Low-Water Mark

You do not have to configure a low-water mark because VSM can operate satisfactorily without it under some conditions. However, without a low-water mark, VSM premigrates all files that meet the selection criteria (see “Define Criteria for Selecting Files” on page 70). This can result in purging files that you would rather keep on the disk.

The low-water mark provides a method for limiting the number of files that `migbatch` can premigrate. You express the low-water mark in terms of the percentage of free space remaining in the file system. The default is 0, interpreted by VSM as “no low-water mark.”

A general guideline for defining the low-water mark percentage is to set it high enough so that `migbatch` migrates enough files to last until the next `migbatch` session. If you execute *migbatch* during off-peak hours, this ensures that a potentially time-consuming migrate operation does not interfere with user activities. Provide enough space to allow a reasonable period of time before another `migbatch` operation is necessary.

VSM requires that the low-water mark percentage be equal to or higher than the high-water mark percentage. A value of 20% gives good results in most cases but there are other situations where you will want to vary it.

For example:

- You can increase the low-water mark percentage if the number of premigrated files is not enough to satisfy requirements between `migbatch` sessions.
- You can decrease the low-water mark percentage if VSM is removing more files than necessary to keep space available between `migbatch` sessions.

Figure 27 shows what can occur if the low-water mark is set properly.

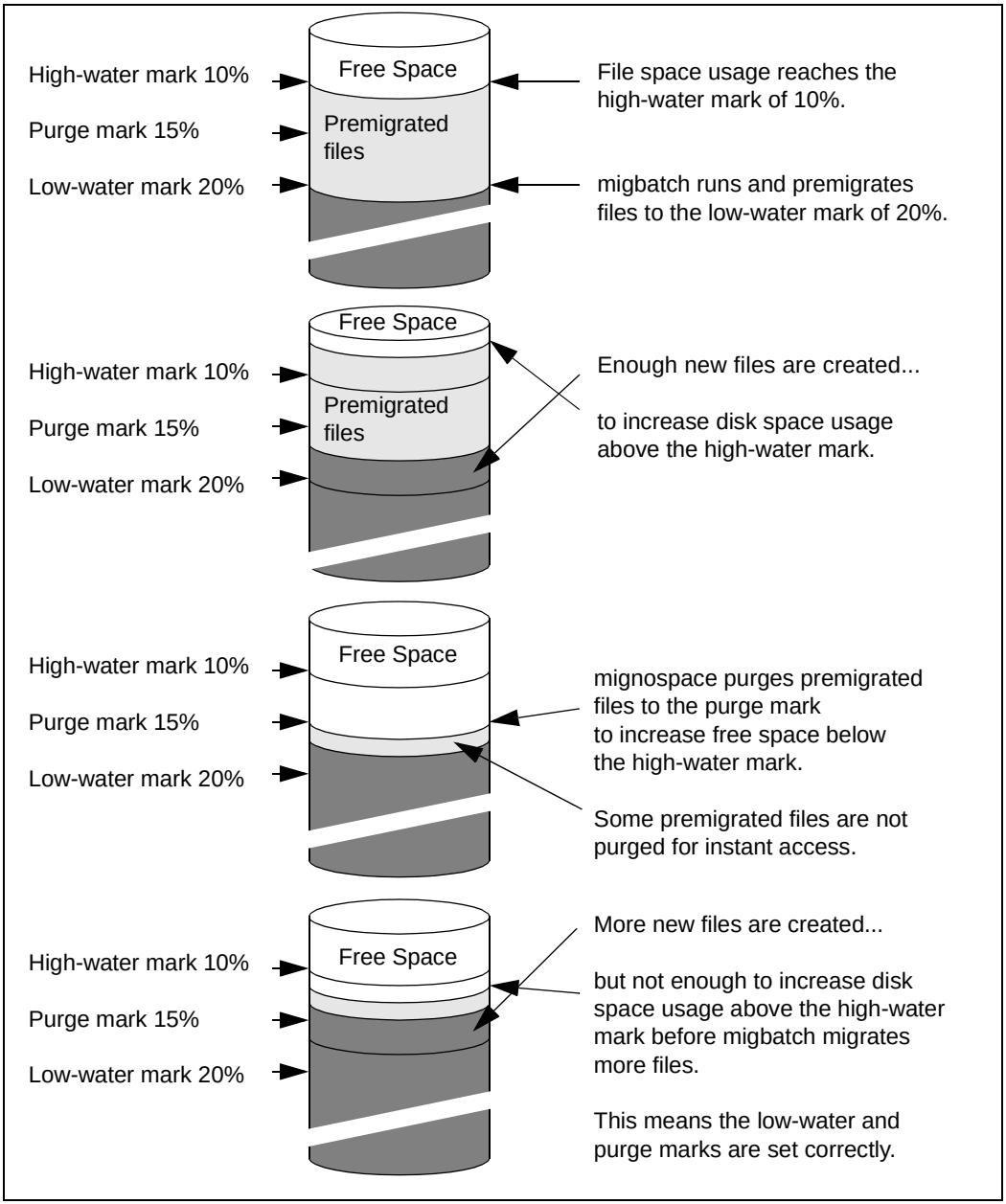
Purge Mark

You do not have to configure a purge mark because VSM can operate satisfactorily without it. However, without a purge mark, VSM purges all premigrated files that meet purging criteria when free space becomes less than the high-water mark percentage. This can result in removing files that you would rather keep on the disk. Access to premigrated files is instantaneous, while access to those same files after they have been purged imposes a caching delay to return those files from secondary storage. The purge mark, therefore, can minimize or postpone the removal of premigrated files from your file system, thereby avoiding unnecessary caching delays when accessing those files.

The purge mark provides a method for limiting the number of premigrated files `mignospace` will purge. You express the purge mark in terms of the percentage of free space remaining in the file system. The default is 0, interpreted by VSM as no purge mark.

A general guideline for defining the purge mark percentage is to set it low enough so that `mignospace` purges enough premigrated files to increase free space sufficiently while still retaining some premigrated files in the file system. See “Define Criteria for Purging Files” on page 78 for more information on how to configure which files `mignospace` removes first.

Figure 27. Low-Water Mark and Purge Mark Example



Although VSM will accept any value for purge mark, a value greater than the high-water mark percentage but less than the low-water percentage will give the best results. (The interfaces enforces this convention.) If the former is 10% and the latter 20%, a value of 15% gives good results in most cases.

Figure 27 shows what can occur if the low-water mark and purge mark are set properly.

Define Criteria for Selecting Files

So far we have discussed the points at which VSM starts and stops migration from a managed file system. Just as important are the files that VSM selects to migrate from that file system.

Administrators and users can list specific files -- those VSM will migrate or those VSM will prevent from migrating -- in special VSM control files. These take priority over all selection criteria described here. See the *Storage Migrator User's Guide* and "Global Migration Control" on page 290 for more information on how to create and use these VSM control files.

For VSM to consider a file for migration, the file must:

- Be a regular UNIX file (for example, not a device file or directory) without any new line (carriage return) characters, vertical bars (|), or null characters in its filename.
- Not already be premigrated or migrated.
- Reside within the VSM managed directory specified in `migconf`.
- Not reside within another file system, even if that file system is mounted in a VSM managed directory.
- Not be a symbolic link.
- Have a full pathname length less than or equal to 1023 characters.

Note A significant number of files with full pathname lengths greater than 1023 characters can eventually fill a file system because they can never be selected for migration (see the `migdbcheck(1M)` man page).

How VSM Selects Files to Migrate

From the set of files that meet the above criteria:

1. VSM eliminates files that are under the minimum age (in days since last accessed or modified) and size (in kilobytes). You set these minimums during configuration. The defaults are 7 days and 8 kilobytes.
2. Next, VSM selects from the remaining files according to a formula that assigns weighting factors to file age and size to derive what is called the file's *badness* value. If a file's calculated badness equals or exceeds the value that you configure for the file system, the file is eligible for migration. During configuration, you can change both the badness value that VSM allows and the weighting factors that VSM uses to calculate a file's badness. You can also define a site-specified badness formula in lieu of the VSM badness formula. This site-specified formula is also used for selecting premigrated files to purge. See "How to Customize File Migration Criteria" on page 78.

The standard formula that VSM uses to calculate a file's badness is as follows:

badness = *age* x *age_weight* (+ or x) *size* x *size_weight*

Where the following apply:

- *badness* is a value that you assign (default is 30). A file is migrated if its badness equals or exceeds this value.
- *age* is the number of days since the file was last accessed or last modified, whichever is most recent.
- *age_weight* is a value that you assign (default is 1).
- *size* is the size of the file in kilobytes.
- *size_weight* is a value that you assign (default is 1).
- + (add) or x (multiply) depends on the value you assign to *weight_operator*. The default is multiply.

Note Another option is to substitute a site-specified badness formula for the standard VSM badness formula. See "How to Customize File Migration Criteria" on page 78. There is one site-specified badness formula and, if configured, VSM also uses it to calculate a file's purge badness when selecting premigrated files to purge. See "Define Criteria for Purging Files" on page 78.

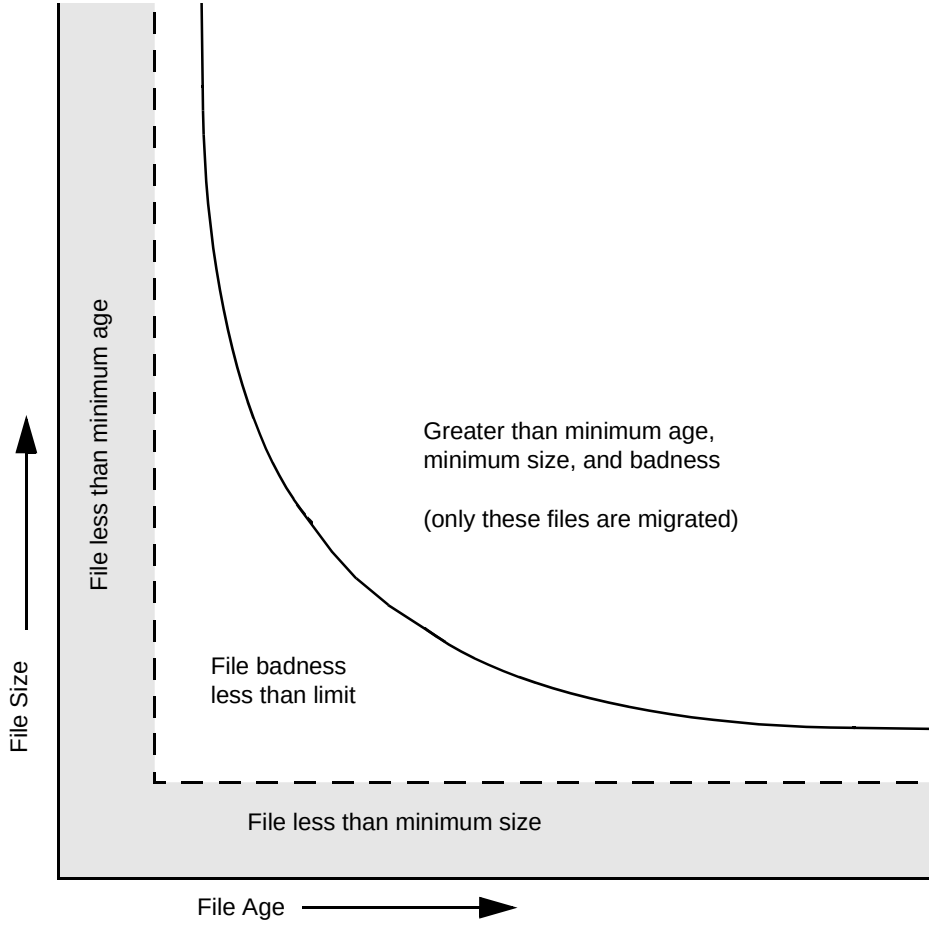
For information on how to configure a site-specified badness formula, see "Assign File System Attributes" on page 133.

By substituting age and size values in the standard badness formula, you can see in general that larger files are migrated before smaller files of equal age, and older files are migrated before younger files of equal size. For example, using a minimum age of 1, a 15 kilobyte file is migrated 2 days after the last access and a 30 kilobyte file 1 day after the last access.

Figure 28 illustrates the relationship between minimum age, minimum size, and badness. As this figure shows, a file must meet all of the following conditions for VSM to select it:

- Have not been accessed or modified within the minimum age period
- Be larger than or equal to the minimum size
- Have a badness value greater than or equal to the configured limit

Figure 28. Minimum age, minimum size, and badness



Procedure For Setting File Migration Criteria

The following procedure and examples are based on the standard VSM badness formula.

As a general rule, set minimum age, minimum size, and badness values so VSM can select at least enough files to reach the low-water mark when `migbatch` premigrates files. Also set these values so that VSM is not likely to migrate files you are going to need on a regular basis. You do not want to migrate a file that you create today and will update tomorrow. Instead, you migrate files that are not used regularly, especially if they are large.

The following suggests a method to use when determining the criteria for selecting files to migrate:

1. Determine the size and age range of files that you must migrate to increase file-system free space to the desired low-water mark.
2. Determine minimum age and size criteria for the files you want to migrate.
   - Set minimum age to at least 1. This prevents VSM from migrating files the same day you create them.
   - When setting minimum size, remember that VSM never automatically migrates files less than minimum size. Therefore, if you set too high a value, the file system can fill up even with automatic migration. Too low a value can cause problems by significantly increasing migration time relative to the amount of additional space provided.

     Unless you anticipate problems, leave minimum size at its default of 8 kilobytes. This allows VSM to migrate files larger than the default slice value (if they also equal or exceed the minimum age and badness factors); this is satisfactory for most sites.
3. Determine the point at which you want badness to start selecting files that equal or exceed the minimum age and size. Then, set badness accordingly.
4. Run `migtestbadness` to determine the amount of disk space your file-selection criteria would free, and adjust the criteria to suit your needs.
5. Monitor the file system and adjust your file-selection criteria. For example, change the values if your current settings allow the file system to gradually fill up.

Examples 1, 2, and 3 (starting on page page 75) illustrate how various badness settings affect file migration.

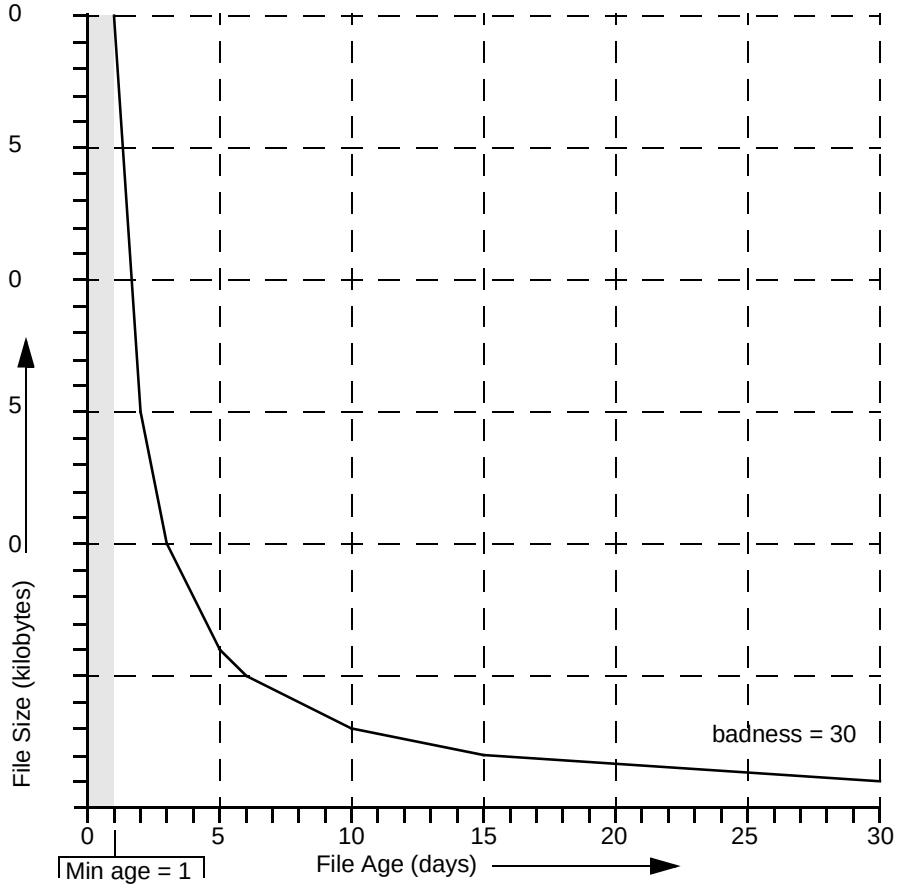
The example site uses the selection criteria of Example 1 for all file systems on the server `bunny`, except `/home2` which uses the selection criteria of Example 2.

Example 1:

Assume that for this managed file system, you can migrate enough files by setting minimum age to one day, minimum size to one kilobyte, and leaving badness at the default value of 30. VSM selects files for migration as follows:

- Never selects files that are less than one day old or less than one kilobyte in size.
- Initially selects all files that are at least one day old and 30 kilobytes in size.
- Selects files between 1 and 30 kilobytes when their age increases to the point that their badness factor equals or exceeds 30.

Figure 29. Badness Example 1 (badness = 30)

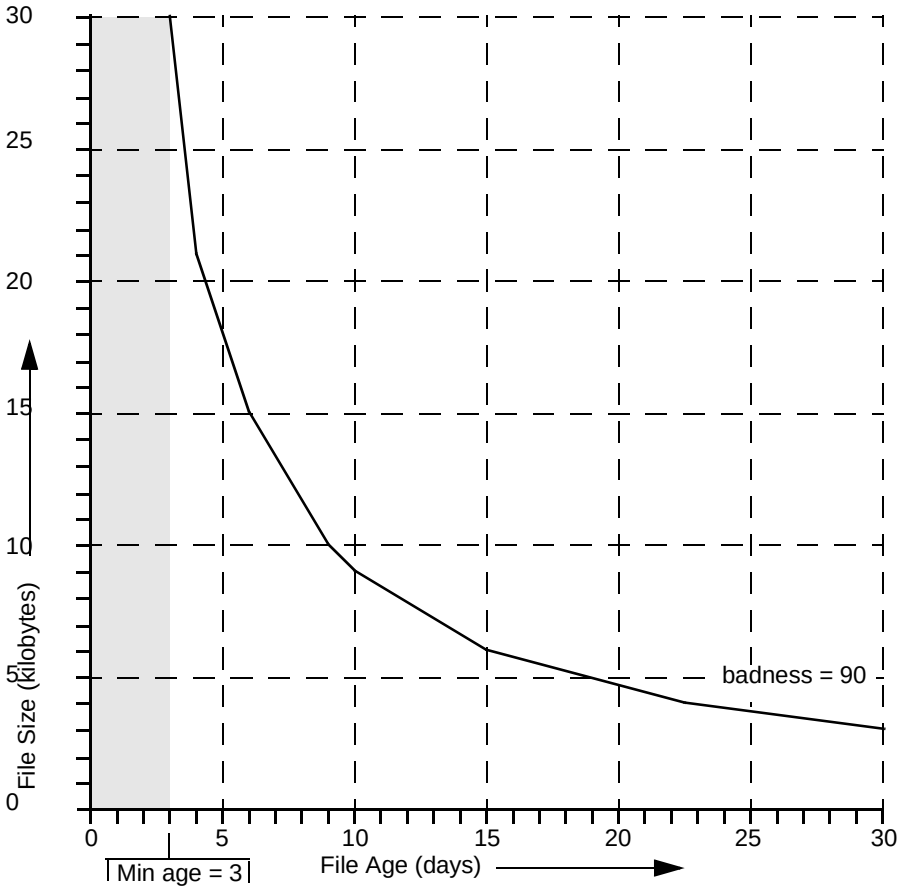


Example 2:

Assume that in this managed file system, you want to migrate all files that have not been accessed in at least three days and are at least 30 kilobytes in size. You set minimum age at three days and minimum size to one kilobyte. Set badness at 90. To arrive at this badness factor, multiply 3 by 30 (age in days by size in kilobytes) and leave the weighting factors at 1. VSM selects files for migration as follows:

- No files less than three days old or less than one kilobyte in size.
- All files that are at least 30 kilobytes and at least 3 days old.
- Files between 1 and 30 kilobytes when their age increases to the point that their computed badness factor equals or exceeds 90

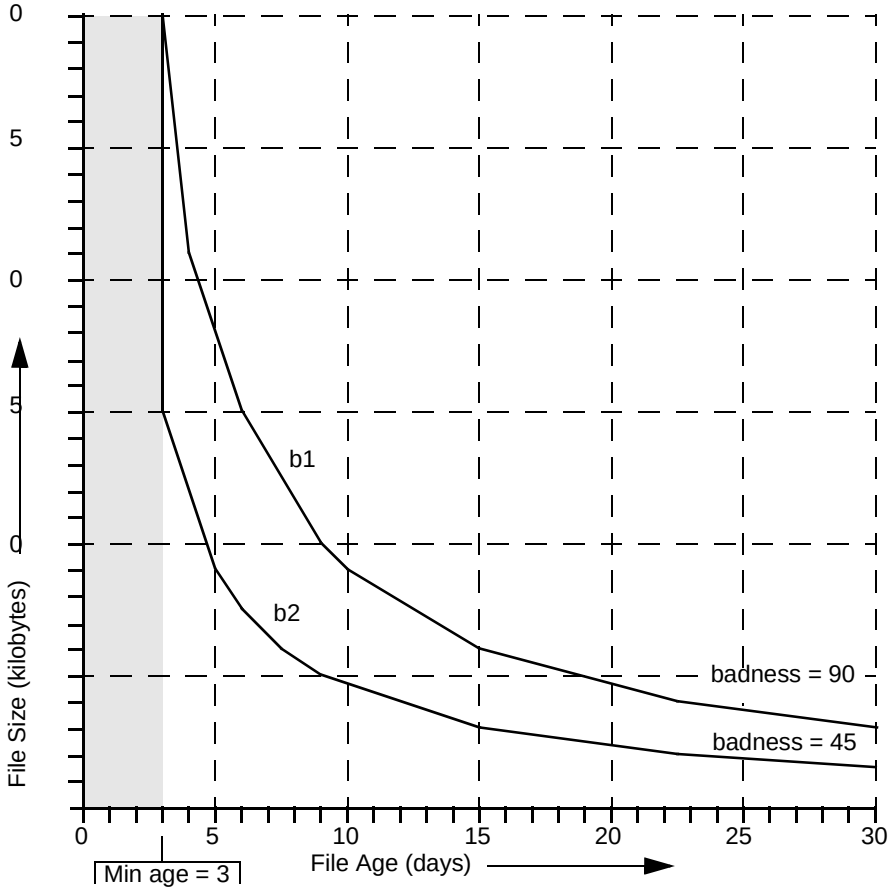
Figure 30. Badness Example 2 (badness=90)



Example 3:

This example shows the effect of cutting badness in half. The line labeled `b1` indicates the extent of file selection by `migbatch` when it selects files for premigration. The line labeled `b2` illustrates that many more files are selected when `mignospace` cuts badness in half, in this case from 90 to 45.

Figure 31. Badness Example 3 (badness=90, retry=2)



How to Customize File Migration Criteria

You can substitute a single site-specified badness formula to calculate file badness or purge badness or both. This program, `migsweep.site`, is located in the VSM database directory as shown in Figure 19 on page 47. VSM calls `migsweep.site` for each file that exceeds the minimum age and size attributes.

Input parameters are file name, age (in days) since the last access or last modification, whichever is most recent, size (in kilobytes), and configured badness or purge badness criteria.

Note For Solaris *ufs* file systems, the file name is the name of the file in the `migration/data` directory for calls to `migsweep.site`.

`migsweep.site` can be any type of program or script that echoes a true/false migration flag to standard output. VSM will migrate or purge the file if output is 0, and will not migrate or purge the file if output is anything else. See "migsweep.site" on page 305.

Define Criteria for Purging Files

After specifying the criteria for migrating files, you can determine and specify other criteria for selecting the sequence by which VSM purges those files that are premigrated or partially cached (see "Partial File Caching" on page 17).

Note Setting purge attributes is an optional step in configuring VSM. The default purge attributes will purge the oldest files first, regardless of size. In many cases, this is satisfactory and need not be changed.

By retaining premigrated files on disk, you can defeat the ability of VSM to create disk space when needed. For example, if all files were migrated in the last three days and you configure the minimum purge age to be four days, VSM will not find any files to purge when additional disk space is needed.

VSM selects premigrated or partially cached files to purge similarly to the way it selects files to migrate.

1. From the premigrated files, VSM eliminates files that are under the purge minimum age (in days) and size (in kilobytes). You set these minimums during configuration. The default is 0 for both.
2. Next, VSM selects from the remaining files according to a formula that assigns weighting factors to file age and size to derive what is called the file's *purge badness* value. If a file's calculated purge badness equals or exceeds the value that you configure for the file system, the file is eligible for purging. During configuration, you can change both the purge badness value that VSM allows and the weighting factors that VSM uses to calculate a file's purge badness.

The standard formula that VSM uses to calculate a file's purge badness is as follows:

purge badness = *age* x *purge_age_weight* (+ or x) *size* x *purge_size_weight*

Where the following apply:

- *purge badness* is a value that you assign (default is 0). A file is purged if its purge badness equals or exceeds this value.
- *age* is the number of days since the file was either migrated or copied, whichever is less (Solaris *ufs* file systems), or since the file was either accessed or modified, whichever is less (nonkernel-based implementations).
- *purge_age_weight* is a value that you assign (default is 1).
- *size* is the size of the file in kilobytes.
- *purge_size_weight* is a value that you assign (default is 0).
- + (add) or x (multiply) depends on the value you assign to *weight_operator*. The default is + (add) .

Note Another option is to substitute a site-specified badness formula for the standard VSM purge badness formula. See "How to Customize File Migration Criteria" on page 78. There is one site-specified badness formula and, if configured, VSM also uses it to calculate a file's badness when selecting files to migrate. See "How VSM Selects Files to Migrate" on page 71.

For information on how to configure a site-specified badness formula, see "Assign File System Attributes" on page 133.

By substituting age and size values in the standard purge badness formula or by using a site-specified badness formula, you can postpone purging certain premigrated files you want to remain on disk as long as possible.

Define Criteria for Moving Migrated Files

Note VERITAS Storage Migrator Remote does not support multilevel migration.

This section is applicable if you are using the multilevel migration capabilities of VERITAS Storage Migrator. After specifying the criteria for migrating and purging files, you can determine and specify other criteria VERITAS Storage Migrator uses for selecting which migrated files to move to various migration levels.

VERITAS Storage Migrator selects migrated files to move similarly to the way it selects files to migrate.

1. From the files at the source migration level, VERITAS Storage Migrator eliminates files that are under the move minimum age (in days since migrated or moved to this level, or since last accessed) and size (in kilobytes). You set these minimums during configuration. The default minimum age is 7 days, and the default minimum size is 0.
2. Next, VERITAS Storage Migrator selects from the remaining files according to a formula that assigns weighting factors to file age and size to derive what is called the file's *move badness* value. If a file's calculated move badness equals or exceeds the value that you configure for the source migration level or method name, the file is eligible for moving.

Note Move criteria for migration levels, if specified, take precedence over move criteria for method names, if specified. See Figure 32.

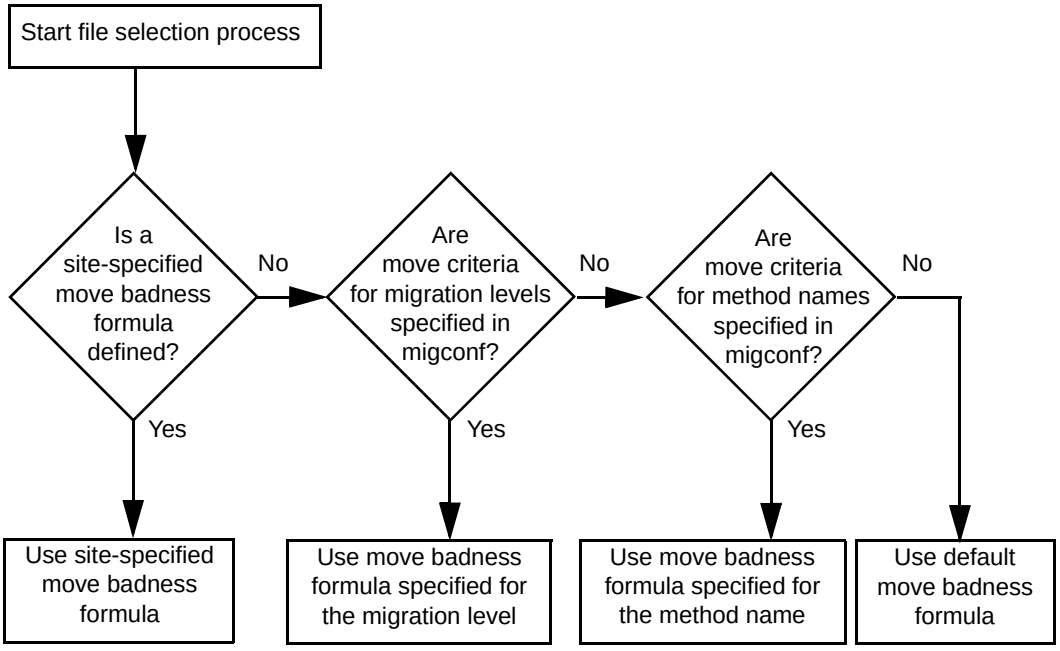
During configuration, you can change both the move badness value that VERITAS Storage Migrator allows and the weighting factors that VERITAS Storage Migrator uses to calculate a file's move badness.

You can also define a site-specified move badness formula in lieu of the standard VERITAS Storage Migrator move badness formula. This program, `migsweepm.site`, is located in the VSM database directory as shown in Figure 17 on page 43. VERITAS Storage Migrator calls `migsweepm.site` for each file that exceeds the minimum age and size attributes.

Input parameters are file name, size (in kilobytes), age (in days), and current move badness. `migsweepm.site` can be any type of program or script that echoes a true/false migration flag to standard output. VERITAS Storage Migrator will move the file if output is 0, and will not move the file if output is anything else. See "migsweepm.site" on page 305.

Site-specified move criteria, if defined, always take precedence over the standard move criteria for migration levels and for method names. See Figure 32.

Figure 32. Selection Criteria for Moving Files



The standard formula that VERITAS Storage Migrator uses to calculate a file's move badness is as follows:

move badness = *age* x *move_age_weight* (+ or x) *size* x *move_size_weight*

Where the following apply:

- *move badness* is a value that you assign (default is 30). A file is moved if its move badness equals or exceeds this value.
- *age* is the number of days since the file was migrated or moved to this level or since the file was last accessed, whichever is most recent.
- *move_age_weight* is a value that you assign (default is 1).
- *size* is the size of the file in kilobytes.
- *move_size_weight* is a value that you assign (default is 1).
- + (add) or x (multiply) depends on the value you assign to *weight_operator*. The default is multiply.

For information on how to configure a site-specified move badness formula, see "Configure and Edit Storage Method Names" on page 139 and "Configure and Edit Migration Levels" on page 145.

Choose Storage Methods for Migrating Files

After establishing the migration thresholds for each managed file system, you determine the best methods for storing the data that VSM migrates. Your choice of storage methods has a major impact on the caching performance of the system.

The first consideration when choosing storage methods is to decide whether to write one or two copies of migrated files and the type of media to use for each copy. For example, you can write one copy to magnetic tape and another copy to optical disc. It is also possible to record on several devices simultaneously in order to speed up the migration process. See “Concurrent Recording” on page 88.

The storage method for the first copy is a set of parameters called METHOD1 and the storage method for the second copy is called METHOD2. If you make only one copy, use only METHOD1.

You must configure at least as many copies as you specify with the `copies` parameter in the DEFAULTS section of the `migconf` file. For example, if `copies` has a value of 2, you must use both METHOD1 and METHOD2.

Caution The number and type of your storage drives must correspond to the storage methods you configure. For example, if you configure tape method names for both METHOD1 and METHOD2, you must have at least two tape drives configured.

The four individual parameters that you can set for METHOD1 and METHOD2 are as follows:

- Method Name (refers to device and media)
- Volume Set Number
- Volume Set Availability (hint)
- Volume Pool

The format in the `migconf` configuration file is as follows:

"*method name . volume set number . hint.volume pool*"

For example:

```
"ct.1.library.HSM"
```

Note The volume pool parameter is optional, and defaults to `HSM`.

This format is called a *stripe*. A storage method may consist of more than one stripe. For examples, see “Volume Set Number” on page 86 and “Concurrent Recording” on page 88.

The best storage method to use depends on the file system. If you have a file system with many large files, choose a method that employs media with a fast transfer rate. If you have many small files, then access time is more important.

The example worksheet, Figure 26 on page 66, has an area in which you can record your choice of storage methods for a file system.

The following topics on storage methods discuss each of the parameters for METHOD1 and METHOD2. There is also an explanation of concurrent recording. The final topic in this section provides guidelines on configuring the best method for your file systems.

Method Name

The METHOD section of the `migconf` file contains a set of parameters for each storage method that VSM supports. These parameters define the characteristics that VSM associates with each storage method. For example, the access time is considered 0 for a disk file and 30 minutes for an 8mm cartridge tape.

You use the method name to select the set of parameters for the intended storage method.

- `ad` (alternate magnetic disk)
- `ct` (tape - STK-9840 technology)
- `dt` (tape - DLT 70000 technology)
- `mt` (tape - Sony AIT-2 technology)
- `op` (optical disc as tape with random seek - rewritable)
- `ow` (optical disc as tape with random seek - write once, read many)
- `ft` (remote method using `ftp`)
- `nb` (NetBackup)
- `dk` (disk file) - used only for premigration

Note If desired, you can alter these default tape and optical methods. See “Configure and Edit Storage Method Names” on page 139.

Disk Methods

VERITAS Storage Migrator and VERITAS Storage Migrator Remote both support two disk storage methods. These methods are not intended for general purpose storage and you must use them only as described here.

`dk` (disk file)

- The disk file method is required for premigrating files. You must always configure the `dk` method during VSM installation (see VSM release notes). The premigration directory must be in the same file system as the files you are migrating. However, VSM does not migrate files from the premigration directory itself.

`ad` (alternate magnetic disk)

- The alternate magnetic disk method is similar to the disk file method in that it also uses disk volumes.
- One use of the alternate magnetic disk method is to migrate files to another file system mounted on the same managed server. It also serves as a remote method when used with NFS. If you NFS mount a remote file system and register it as a VSM volume, VSM can use it as secondary storage. You must mount the NFS file system before registering it in the Volume database.
- You also use the `ad` method with two-step tape or optical disc consolidation. See "VSM Volume Management" on page 272 for more information.

Tape Methods

VERITAS Storage Migrator supports three tape storage methods. VERITAS Storage Migrator Remote does not support these methods.

- `ct` (tape - STK-9840 technology)
- `dt` (tape - DLT 70000 technology)
- `mt` (tape - Sony AIT-2 technology)

Note If desired, you can alter these default methods for use with tapes currently not configured in `migconf`. See "Configure and Edit Storage Method Names" on page 139.

Your choice of tape method depends on the type of device you have on your site. If you have more than one type of device, you can analyze their characteristics as listed in the `migconf` file to determine the method to use for a specific file system.

VSM interfaces with the device-manager interface (`ltid`) just as it does for optical disc devices. Using the device-manager interface allows VSM to share a tape library with other applications.

You can remove tapes from a library and place them on the shelf or send them to an off-site storage location. When you physically remove a tape from the library device, Media Manager updates its database to indicate the volumes on that tape as offline. An access request for any file on the offline tape generates a message for the operator to mount the tape manually.

Optical Disc Methods

VERITAS Storage Migrator supports two optical disc storage methods. VERITAS Storage Migrator Remote does not support these methods.

- `op` (optical disc - rewritable)
- `ow` (optical disc - write once, read many)

Note If desired, you can alter these default methods by changing the configured density attribute in `migconf`. See "Configure and Edit Storage Method Names" on page 139.

Your choice of optical disc method depends on the type of media you have on your site.

The `op` and `ow` methods manage optical discs as logical tapes and treat each disk surface as a tape volume. VSM stores files in a format similar to that used for storing files on tape. However, the random seek ability of the optical-disk drive reduces access time.

Note When you register one side of an optical disc with `migreg`, VSM also automatically registers the other side with the same volume set number. This avoids deadlocks during volume consolidation and when moving files between migration levels.

VSM interfaces with the device-manager interface (`ltid`) just as it does for magnetic-tape devices. Using the device-manager interface allows VSM to share an optical-disk library with other applications.

Because VSM treats each optical disc as a tape volume, you can remove optical discs from a library and place them on the shelf or send them to an off-site storage location. When you physically remove an optical disc from the library device, Media Manager updates its database to indicate the volumes on that disk as offline. An access request for any file on the offline disk generates a message for the operator to mount the disk manually.

Remote Method

VERITAS Storage Migrator and VERITAS Storage Migrator Remote both support a method for storing migrated files on remote file systems.

`ft` (`ftp`)

The `ft` method uses standard `ftp` (File Transfer Protocol) for moving files.

A major difference between the ft method and tape or optical disc methods is that it transfers whole files without breaking them into granules.

The remote volume servers can be from a variety of different vendors and have different capacities. Files are migrated and retrieved by using standard file transfer protocols. The result is to combine a local UNIX file system with one or more remote file systems and create the impression of one large virtual file system.

Note If VSM transfers files greater than 2 gigabytes with the `ft` method, the remote volume server must be configured to accept and process files of this size.

If a file access causes a cache request, VSM attempts to cache the copy from the volume it can access in the shortest time, regardless of whether it is on remote or local media. See Figure 1 on page 3.

Note If VSM is running on the remote volume server, use the global stop file to prevent the migration of any *data_file*`.GLABEL` files from that remote server. See "Global Migration Control" on page 290. This will expedite the cleaning of remote file systems by removing obsolete files from `ft` volumes to reclaim filespace. See the `migmdclean(1M)` man page.

NetBackup Method

VERITAS Storage Migrator and VERITAS Storage Migrator Remote both support a method for using VERITAS NetBackup to store copies of migrated files.

`nb` (NetBackup)

The NetBackup method is similar to the remote `ft` method in that it transfers whole files without breaking them into granules.

Migrating files with the `nb` method causes a user backup of the files and their associated granule header files to a previously defined NetBackup class. See "To Define a NetBackup Class" on page 168.

If a file access causes a cache request, VSM attempts to cache remote (`ft` or `nb`) copies first (if they exist) first before attempting to cache copies from local media. See Figure 1 on page 3.

Note When using the `nb` method, your VSM server must be a master or slave server of NetBackup. See "To Define a NetBackup Class" on page 168.

Volume Set Number

The volume set number is assigned when media is registered (labeled). This registration process assigns a unique media ID to each volume which consists of a label, a method name, and a volume set number. Each numbered volume set contains one or more volumes. See "Register Media with VSM" on page 157.

The volume set number is incorporated when configuring storage method parameters.

METHODn = "*method_name.volume_set_number.hint.volume_pool*"

For example, this storage method consists of two stripes:

```
METHOD1="op.1.library.HSM"/"ct.1.vault.HSM"
```

Note The volume pool parameter is optional, and defaults to `HSM`.

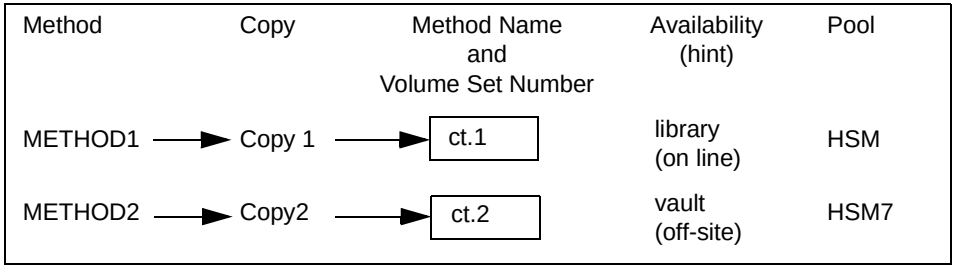
Stripes in any volume pool must contain unique volume sets, where a volume set is defined as a method name and its volume set number. If you configure a method name more than once, use a different volume set number each time, regardless of the volume pool. This prevents VSM from possibly migrating both copies to the same physical media.

For example:

```
METHOD1="ct.1.library.HSM"       METHOD2="ct.2.vault.HSM7"
```

This causes VSM to write the two copies to unique cartridge-tape volume sets. The first copy goes to method name `ct`, volume set number `1`, in the `HSM` volume pool. The second copy goes to method name `ct`, volume set number `2`, in the `HSM7` volume pool.

Figure 33. Multiple Volume Sets and Copies



This example includes `library` and `vault` as *hint* parameters. The following section provides more information on the hint parameter.

Volume Set Availability (hint)

If you create two copies of migrated files, VSM stores them on different volumes. When VSM caches a file from secondary storage, it attempts to use the volume with the shortest access time. To ensure that this occurs, each storage method includes a *hint* parameter that specifies where you intend to store the volume.

Depending on your choice, VSM adds an access-time bias to ensure that it always requests the most available volume first. If for some reason VSM cannot access the most available volume, it requests the other volume.

The three possible values for hint are as follows:

`library`

Volume is in an online library unit (an automounting, or robotic device). Access is immediate, so no access-time bias is added.

`operator`

Operator action is required to mount the volume. This choice adds 15 minutes to the access time bias.

`vault`

Volume is at a remote location. This choice adds 24 hours to the access time bias.

Note Cache requests are issued immediately, regardless of hint value.

When configuring the method, always choose a hint that maximizes caching performance. For example, assume METHOD1 goes to a high performance device and METHOD2 to a slower device. Here, you can assign `library` to METHOD1 and `vault` to METHOD2. This ensures that VSM selects the fastest device for a cache.

The same principals apply regardless of whether you are using local devices (such as tape) or remote volume servers. With remote volume servers, you can set up cooperative procedures with other administrators to ensure that you are made aware of changes that occur in the administration of the remote file systems. For example, if files that were formerly on disk with a hint value of `library` are now migrated to an operator controlled tape device, change your availability setting to `operator`.

Volume Pool

Use this optional parameter to designate a volume pool other than VSM from which to select volumes. If not specified, the default volume pool `HSM` is used.

Concurrent Recording

Concurrent recording allows VSM to migrate files to more than one device at the same time. When multiple devices are available, you can use this capability to speed up the migration process.

To take advantage of concurrent recording, you must organize your volumes into multiple stripes as described in “Volume Set Number” on page 86. You set up concurrent recording with the following three parameters from the `migconf` configuration file:

- `copies`
- `METHOD1`
- `METHOD2`

For example if you set these parameters to the following:

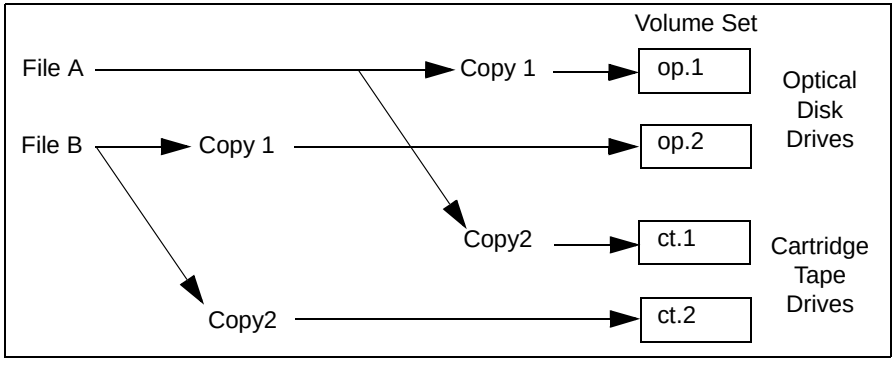
```
copies=2
METHOD1="op.1.library"/"op.2.library"
METHOD2="ct.1.vault"/"ct.2.vault"
```

In this example, the storage methods share the default volume pool (HSM) and each has two stripes. VSM makes two copies of each file and distributes them to four volume sets. Assuming there are enough devices configured and available, VSM writes to four different devices simultaneously (see Figure 34).

- Volume sets op.1 and op.2 contain copy 1. Every other file goes to the alternate volume set and device.
- Volume sets ct.1 and ct.2 contain copy 2. Again, every other file goes to the alternate volume set and device.

The number in the volume set has nothing to do with the copy being made. METHOD1 specifies the volumes for the first copy and METHOD2 specifies the volumes for the second copy.

Figure 34. Concurrent Recording Multiple Copies



You can take advantage of concurrent recording to speed up the migration process even if you make only one copy. To do this, you specify multiple stripes in the METHOD1 parameter. These volume sets each go to a different device. For example, if you have two cartridge tape drives you can set up the following:

```
copies=1
METHOD1="ct.1.library.HSM"/"ct.2.library.HSM4"
```

Note In this case the stripes happen to be in different volume pools, but the volume sets are unique regardless of the volume pool.

Here, VSM automatically distributes the migrated files between the two volume sets and writes each volume set on a separate device (see Figure 35).

VSM is able to perform any number of concurrent migrations within the constraints defined below:

- Number of secondary storage devices that are available.
- Server speed. Too many concurrent migrations interfere with the performance of the server. The actual number that VSM can support efficiently depends on the hardware, operating system, and applications that are running.

Figure 35. Concurrent Recording Single Copy

You can extend each storage method to more than two stripes.

```
METHOD1="ct.1.library.HSM"/"op.1.library.HSM2"/"mt.6.vault.pool_3"...
```

VSM has been tested with as many as five stripes.

Note The preceding section describes the default action of VSM for concurrent recording as defined in `migpolicy.script`. You can modify or replace this standard script in order to migrate files in different ways as described below.

Choosing the Best Method

The best method for storing your files is always the one that optimizes the migration and caching performance of your system. The two characteristics of your file system that have the greatest effect on caching are as follows:

- Size of files
- Frequency with which they are accessed

Consider the following device and media characteristics when making your choice are as follows:

- Access time
- Transfer rate
- Capacity

For example, if you have many small files that are frequently accessed, choose a device and media with a fast access time. Transfer rate is not as important if files are small. Optical generally has faster access time than tape.

However, as file size increases the importance of the transfer rate also increases and at very large file sizes becomes the most significant factor to consider. In most cases tape has a faster transfer rate than optical.

For example, if you want smaller files migrated to optical and larger files migrated to tape, you can set up the following stripes:

```
METHOD1= "ct.1.library"/"op.2.library"
```

To distinguish between smaller and larger files, you would also need to replace the standard `migpolicy.script` in `/usr/openv/hsm/bin/admincmd` with one like the example provided in `/usr/var/openv/hsm/example/database/sample.migpolicy.script`. See “Customize the VSM Policy and Method for Migrating Files” on page 295 and the `migpolicy(1M)` man page.

Media cost can also affect your choice of methods. This is true, for example, if a site is unable to afford fast enough devices or enough online capacity to handle all the requests. If cost constraints prevent you from achieving satisfactory cache times, set up special procedures for users. One example is to require that users request migrated files a day before they are needed.

The example site has both tape libraries and optical-disk robots. In general, optical disc has a shorter access time than tape but at a much lower transfer rate. Therefore, optical disc is better for smaller files and tape is better for the larger files. Based on this we decided that the following were the best methods for the file systems on `bunny` at the example site:

Table 5. Example Storage Methods

File System	Average File Size	Storage Method
`/home1` and `/home2`	18 Kbytes	METHOD1 = op.x.library METHOD2 = ct.x.vault
`/dbrecs`	1.8 Mbytes	METHOD1 = ct.x.library METHOD2 = ct.x.vault
`/sd3`	1.5 Kbytes	METHOD1 = op.x.library METHOD2 = op.x.vault

The storage methods in this table do not show specific volume set numbers because you assign those numbers and determine whether to use concurrent recording after assigning databases (see “Completing the Example Configuration Plan” on page 110).

The average file size for the /home1, /home2, and /sd3 file systems is relatively small (less than 100 kilobytes). Therefore, transfer time is not a factor and optical disc is the choice for METHOD1 (first copy) for all these file systems. The tape library provides storage for METHOD2 (second copy) for all except the /sd3 file system, which uses optical. The site wants to keep both copies of files from /sd3 on optical because of its greater reliability.

The files on /dbrecs are about 1.8 Mbytes. They are also accessed relatively infrequently (every 30 days). Therefore, the tape library is used for both copies of files migrated from this file system.

For all four of the file systems on bunny at the example site, volume set availability (hint) is always library for the first copy and vault for the second copy. This ensures that VSM always chooses copy 1 first for cache operations.

Figure 36 on page 93 through Figure 39 on page 96 show the example worksheets for the managed file systems on bunny. In "Choose Directories for Databases" on page 99, we evaluate and compare the storage methods on these worksheets in order to determine which file systems can share a database and which of them must have separate databases.

Note In this example, all storage methods use the default volume pool, HSM.

Figure 36. Managed-File-System Example Worksheets (1 of 4)

Managed-File-System Planning Worksheet

VSM Name hsm1 File System Name (mount point) /home1

File System Thresholds for Migration

High-water Mark for this file system: 10 %

Purge Mark for this file system: 15 %

Low-water Mark for this file system: 20 %

File Selection Criteria for Migration	File selection Criteria for Purging
Migrate files with Badness ≥ 30	Purge files with Badness ≥ 0
Age weight: 1	Age weight: 1
Size weight: 1	Size weight: 0
Weight operator: *	Weight operator: +
Min. age to migrate (days): 1	Min. age to purge (days): 0
Min. size to migrate (kbytes): 1	Min. size to purge (kbytes): 0

Best Storage Methods for File System

Copies: 2 (1 or 2) METHODn: op.1.library/ct.3.vault/...

METHOD1: op.1.library/ct.3.vault

METHOD2:

METHOD3:

METHOD4:

METHOD5:

METHOD6:

METHOD7:

METHOD8:

Figure 37. Managed-File-System Example Worksheets (2 of 4)

Managed-File-System Planning Worksheet

VSM Name hsm2 File System Name (mount point) /home2

File System Thresholds for Migration

High-water Mark for this file system: 10 %

Purge Mark for this file system: 15 %

Low-water Mark for this file system: 20 %

File Selection Criteria for Migration	File selection Criteria for Purging
Migrate files with Badness ≥ 90	Purge files with Badness ≥ 0
Age weight: 1	Age weight: 1
Size weight: 1	Size weight: 0
Weight operator: *	Weight operator: +
Min. age to migrate (days): 3	Min. age to purge (days): 8
Min. size to migrate (kbytes): 1	Min. size to purge (kbytes): 0

Best Storage Methods for File System

Copies: 2 (1 or 2) METHODn: op.1.library/ct.3.vault/...

METHOD1: op.1.library/op.2.library

METHOD2: ct.1.vault/ct.2.vault

METHOD3: _____

METHOD4: _____

METHOD5: _____

METHOD6: _____

METHOD7: _____

METHOD8: _____

Figure 38. Managed-File-System Example Worksheets (3 of 4)

Managed-File-System Planning Worksheet

VSM Name hsm3 File System Name (mount point) /dbrecs

File System Thresholds for Migration

High-water Mark for this file system: 10 %

Purge Mark for this file system: 15 %

Low-water Mark for this file system: 20 %

File Selection Criteria for Migration	File selection Criteria for Purging
Migrate files with Badness ≥ 30	Purge files with Badness ≥ 0
Age weight: 1	Age weight: 1
Size weight: 1	Size weight: 0
Weight operator: *	Weight operator: +
Min. age to migrate (days): 1	Min. age to purge (days): 0
Min. size to migrate (kbytes): 1	Min. size to purge (kbytes): 0

Best Storage Methods for File System

Copies: 2 (1 or 2) METHODn: op.1.library/ct.3.vault/...

METHOD1: ct.1.library/ct.2.library

METHOD2: ct.3.library/ct.4.vault

METHOD3: _____

METHOD4: _____

METHOD5: _____

METHOD6: _____

METHOD7: _____

METHOD8: _____

Figure 39. Managed-File-System Example Worksheets (4 of 4)

Managed-File-System Planning Worksheet

VSM Name hsm4 File System Name (mount point) /sd3

File System Thresholds for Migration

High-water Mark for this file system: 10 %

Purge Mark for this file system: 15 %

Low-water Mark for this file system: 20 %

File Selection Criteria for Migration	File selection Criteria for Purging
Migrate files with Badness ≥ 30	Purge files with Badness ≥ 0
Age weight: 1	Age weight: 1
Size weight: 1	Size weight: 0
Weight operator: *	Weight operator: +
Min. age to migrate (days): 1	Min. age to purge (days): 0
Min. size to migrate (kbytes): 1	Min. size to purge (kbytes): 0

Best Storage Methods for File System

Copies: 2 (1 or 2) METHODn: op.1.library/ct.3.vault/...

METHOD1: op.1.library/op.2.library

METHOD2: op.3.vault/op.4.vault

METHOD3:

METHOD4:

METHOD5:

METHOD6:

METHOD7:

METHOD8:

Quota on Migration Stop Files

Users are able specify files they want to prevent from migrating by listing them in special VSM control files named `.migstop`. See the *Storage Migrator User's Guide* for more information on how to create and use these VSM control files.

You can limit the number of bytes that each user in the file systems for a database can restrict from migration in their `.migstop` files. The default is 10,000,000 bytes. If there are many users, the restricted space can become a significant consideration. For example, the default allows 100 users to restrict a total of 1 gigabyte.

Once a user exceeds the quota, the files listed in their `.migstop` files will no longer be prevented from migration.

Choose Storage Methods for Moving Files

If you are configuring multilevel migration for VERITAS Storage Migrator, you determine the best methods for storing the data that VERITAS Storage Migrator moves from one migration level to another level. Your choice of storage methods has a major impact on the caching performance of the system.

You can configure migration levels up to a maximum of eight. The default configuration for moving files with VERITAS Storage Migrator has four migration tiers, with the odd-numbered migration levels (1, 3, 5, 7) associated with the first copy of migrated files and even-numbered migration levels (2, 4, 6, 8) associated with the second copy of migrated files. See Figure 13 on page 37.

The storage methods for moving files are similar to storage methods for migrating files. See "Choose Storage Methods for Migrating Files" on page 82. METHOD3 through METHOD8 correspond, respectively, to migration level 3 through migration level 8. In the default configuration METHOD3, METHOD5 and METHOD7 are associated with copy 1 of a migrated file, and METHOD4, METHOD6 and METHOD8 are associated with copy 2 of a migrated file.

The four individual parameters that you can set for METHOD3 through METHOD8 are as follows:

- Method Name (refers to device and media)
- Volume Set Number
- Volume Set Availability (hint)
- Volume Pool

The format in the `migconf` configuration file is as follows:

"*method name . volume set number . hint.volume pool*"

For example, this storage method consists of two stripes:

```
"op.1.library.HSM"/"ct.1.vault.HSM"
```

Note The volume pool parameter is optional, and defaults to **HSM**.

The best storage method to use depends on how you want to handle aging files. For example, you have migrated files initially to a robotic optical disc library:

```
METHOD1="op.1.library"
```

As these files grow older and caching delays become less important, you can decide to move the files to a standalone tape drive at a higher migration level. If you obsolete the FHDB entries for the files at source migration level 1 using the `migcons` command, this can eventually free up space on the optical robot for newer file. In this case, you can configure the following storage method:

```
METHOD3="ct.8.operator"
```

Still later, you can decide to archive these files off-site. In this case, you can configure yet another storage method at a different migration level:

```
METHOD5="mt.2.vault"
```

The example worksheet in Figure 26 on page 66 has an area in which you can record your choice of storage methods.

The following topics on storage methods briefly discuss each of the parameters for METHOD3 through METHOD8.

Method Name

The METHOD section of the `migconf` file contains a set of parameters for each storage method that VSM supports. These parameters define the characteristics that VSM associates with each storage method. For example, the access time is considered 0 for a disk file and 30 minutes for an 8mm cartridge tape.

You use the method name to select the set of parameters for the intended storage method.

The available choices for VERITAS Storage Migrator when moving files are as follows:

- `ad` (alternate magnetic disk)
- `ct` (tape - STK-9840 technology)
- `dt` (tape - DLT 70000 technology)
- `mt` (tape - Sony AIT-2 technology)
- `op` (optical disc as tape with random seek - rewritable)
- `ow` (optical disc as tape with random seek - write once, read many)

See “Method Name” on page 83 for a description of these method names.

Note If desired, you can alter these default tape an optical methods. See “Configure and Edit Storage Method Names” on page 139.

Volume Set Number

The volume set number is assigned when media is registered (labeled). This registration process assigns a unique media ID to each volume which consists of a label, a method name, and a volume set number. Each numbered volume set contains one or more volumes. See “Register Media with VSM” on page 157.

Volume Set Availability (hint)

If you create more than one copy of migrated files, VSM stores them on different volumes. When VSM caches a file from local secondary storage, it attempts to use the volume with the shortest access time. To ensure that this occurs, each storage method includes a hint parameter that specifies where you intend to store the volume. See “Volume Set Availability (hint)” on page 87.

Volume Pool

Use this optional parameter to designate a volume pool other than VSM from which to select volumes. If not specified, the default volume pool `VSM` is used.

Choose Directories for Databases

You define the database directories in the HSMDEV entries during global configuration. When VSM migrates files from a file system, it stores migration information in the database files located in the database directory specified in its HSMDEV entry.

Database files include the file-handle database (FHDB) and VSM volume database (VOLDB). Nonkernel-based implementations also have an inode to handle file (*hsmname*`.IHAND`). The database directory contains the `migconf` configuration file that defines the migration thresholds and storage methods that VSM uses for the file systems that this database supports.

Caution Always create the database directory in a local file system that VSM does not manage. This eliminates the possibility of migrating files from the database or workdir directories.

When defining database directories, the best approach is to use a separate directory for each file system. Using the same set of databases for more than one file system can result in the databases becoming large enough to degrade performance.

If you do combine file systems under a common database, remember:

- The database's `migconf` file must contain a separate set of migration-threshold parameters for each file system that the database supports.
- The storage-method parameters in `migconf` apply to all the supported file systems.
- Once you assign and start using a common database for file systems, you cannot easily reconfigure VSM to use separate databases for those file systems at a later date.

The steps to follow when assigning the same database for more than one file system are as follows:

1. Separate the managed file systems into groups that require the same storage methods. This is necessary because a database's `migconf` file contains a single set of METHOD1 and METHOD2 parameters that apply to all file systems using the database. "Choose Storage Methods for Migrating Files" on page 82 explained how to determine the best storage methods to use for each file system.

   The size of the file-handle database (FHDB) file and the amount of space it requires are other important considerations when assigning database directories. See "Number of Entries in the File-Handle Database" on page 105 and "File-Handle Database Space Requirement" on page 106. On nonkernel-based implementations, also consider the *hsmname*`.IHAND` file. See "Inode-to-Handle File Requirement (nonkernel only)" on page 106.
2. Specify the same database directory for each separate group.
3. Specify, within the database's `migconf` file, a separate set of migration-threshold parameters for each file system that the database supports.

Figure 40, Figure 41, and Figure 42 show one way to record planning information about each database's `migconf` file.

- Database Pathname. Indicates the path to the database.
- File-System-Migration Thresholds. Defines the migration thresholds for each file system that this database supports. You can get this information from the Managed-File-System Planning Worksheets. See Figure 26 on page 66.
- Migration Parameters. These include MachineID, Quota, Copies, Unmount Delay, and Checksum.
- Storage Methods for File Systems Using this Database. Methods 1 and 2 direct the migration of file copies 1 and 2, respectively, to secondary storage. Methods 3 through 8 (optional) direct the multilevel migration of these same files to other storage media at a later time.

- Move Attributes by Method Name, and Move Attributes by Migration Level. Move criteria for migration levels, if specified, take precedence over move criteria for method names, if specified.

  If a database supports more than one filesystem, each of the eight storage methods pertains to all supported filesystems.

Figure 40. migconf Planning Worksheet (1 of 3)

Managed-File_System Configuration Planning Worksheet (1 of 3)

Database Pathname: _____

File System Migration Thresholds

VSM Name: _____ _____ _____ _____

File System Name: _____ _____ _____ _____

High-Water Mark (free space): _____ _____ _____ _____

Purge Mark: _____ _____ _____ _____

Low-Water Mark: _____ _____ _____ _____

Badness: _____ _____ _____ _____

Minimum Age (days): _____ _____ _____ _____

Minimum Size (kbytes): _____ _____ _____ _____

Age Weight: _____ _____ _____ _____

Size Weight: _____ _____ _____ _____

Weight Operator: _____ _____ _____ _____

Purge Badness: _____ _____ _____ _____

Purge Minimum Age (days): _____ _____ _____ _____

Purge Minimum Size (kbytes): _____ _____ _____ _____

Purge Age Weight: _____ _____ _____ _____

Purge Size Weight: _____ _____ _____ _____

Purge Weight Operator: _____ _____ _____ _____

Migration Parameters for all File Systems Using this Database

Machine ID: _____ User-Space-Restriction Quota (bytes): _____

Copies: _____ Unmount Delay (seconds): _____ Checksum (y/n): _____

Figure 41. migconf Planning Worksheet (2 of 3)

Managed-File_System Configuration Planning Worksheet (2 of 3)

Database Pathname: _____

Storage Methods for All File Systems Using this Database

Copies: _____ (1 or 2)

METHOD1: _____

METHOD1: _____

METHOD1: _____

METHOD1: _____

METHOD1: _____

METHOD1: _____

METHOD1: _____

METHOD1: _____

Method Names for All File Systems Using this Database

Method Name:	ct	dt	mt	op	ow	ad	ft	nb
Flags:								
Access Time:							na	na
Capacity:				na	na		na	na
Speed:							na	na
Granule Size:							na	na
Block Size:				na	na	na	na	na
Density:						na	na	na
DeadmanTimeout:						na		
Demand Delay:						na	na	na

Figure 42. migconf Planning Worksheet (3 of 3)

Managed-File-System Configuration Planning Worksheet (3 of 3)

Database Pathname: _____

Move Attributes for all File Systems Using this Database by Method Name

Method Name:	ct	dt	mt	op	ow	ad
Move Badness:						
Move Minimum Age:						
Move Minimum Size:						
Move Age Weight:						
Move Size Weight:						
Move Weight Operator:						

Move Attributes for all File Systems Using this Database by Migration Level

Storage Method (level):	METH1	METH2	METH3	METH4	METH5	METH6	METH7	METH8
Move Badness:								
Move Minimum Age:								
Move Minimum Size:								
Move Age Weight:								
Move Size Weight:								
Move Weight Operator:								

The next four topics in this section describe factors to consider when assigning databases. Those topics are as follows:

- Number of Entries in the File-Handle Database
- File-Handle Database Space Requirement
- Total Database Space Requirement
- Example Filesystem-to-Database Assignments

The last topic in this chapter (“Completing the Example Configuration Plan” on page 110) shows examples of completed `migconf` planning worksheets.

Number of Entries in the File-Handle Database

The FHDB (file-handle database) file is one of the files that reside in the database directory. The number of entries in the FHDB file affects the performance of those VSM operations that require database lookup or update.

Note Although VSM currently allows more than one VSM managed file system to share the same database files, specifying this configuration can adversely affect performance if the FHDB becomes too large.

It is very important to anticipate system growth, and partition your file system accordingly. Once you have initially partitioned a file system, you cannot subdivide those partitions at a later time if performance deteriorates. Estimate FHDB size with care during VSM configuration.

VSM includes FHDB compression to reduce disk space utilization and improve processing time. Data compression occurs automatically on initial installations of VSM, and is included in the procedure for upgrading existing VSM managed file systems from earlier releases that is found in the Release Notes.

The FHDB contains at least one entry for each copy of a migrated file. If VSM divides a larger file into granules, the FHDB contains at least two entries for each file copy unless the file is a multiple of the granule size and does not span volumes. Additional FHDB entries are needed if a file copy spans more than one volume. The *maximum* number of FHDB entries per file copy is twice the number of volumes spanned for that copy. In the example above, assuming that 5% of the migrated files span two volumes, you can anticipate a FHDB of 84,000 entries.

(20,000 files) x (2 copies / file) x (2 entries / copy) x 1.05 = 84,000 entries

File-Handle Database Space Requirement

The amount of disk space you can reserve for a file-handle database depends on the number of entries you expect to be in the database.

The size of a typical FHDB entry is about 200 bytes (including full path name and file information). Multiplying the number of file entries by 200 provides the total bytes you reserve for the database.

For the hypothetical file system described above, therefore, the FHDB requires an estimated disk space of 8.4 megabytes:

(200 bytes / entry) x (84,000 entries) = 16.8 Mbytes

Inode-to-Handle File Requirement (nonkernel only)

The maximum space needed for the *hsmname* . `IHAND` file is 64 bytes times the number of inodes in the file system. This figure is platform-dependent.

Total Database Space Requirement

You must also reserve space for the other databases (for example, VOLDB and the workfiles located in *dwpath*`/workdir`). The total amount of space you need for all VSM databases, including the file-handle database, is about three times what you calculated for the file-handle database. In the above example, this totals 50 megabytes.

Depending on how certain you are about the number of files you expect to migrate, you can monitor the growth of the databases as you use VSM and adjust the space allocation if necessary.

Example Filesystem-to-Database Assignments

After evaluating the requirements of each managed file system, you can decide whether they have their own separate database or can share a database with another managed file system.

You can create a VSM database directory in any local file system that VSM does not manage. On `bunny` at the example site, the database path for the hsm3 configuration is as follows:

```
/usr/var/openv/hsm3
```

The default path for the database directory is as follows:

`/usr/var/openv/hsm.`*hsmname*

Where *hsmname* is the name you assign to the HSMDEV entry during global configuration.

Applying the guidelines discussed in this and previous topics to the example site results in database assignments as follows:

- `/home1` and `/home2` have the same storage methods. It is therefore permissible for them to share a database. This combined database will not get large enough to affect performance.
- `/dbrecs` sends both copies of migrated files to tape. Therefore, it must have a different database.
- `/sd3` also must have a separate database because it uses different storage methods (both copies to optical disc).

The following lists the HSMDEV names, file systems, and paths to the databases that we configure for each HSMDEV entry on `bunny` at the example site.

Table 6. Example Database Paths

HSMDEV	File System	Path to Database
hsm1, hsm2	/home1, /home2	/usr/var/openv/hsm1_2
hsm3	/dbrecs	/usr/var/openv/hsm3
hsm4	/sd3	/usr/var/openv/hsm4

Note that the configured paths do not include the term database because VSM adds this when it creates the database files. For example, the full path to the file-handle database for hsm3 is as follows:

```
/usr/var/openv/hsm3/database/FHDB
```

Scheduling Migrations

VSM allows you to schedule automatic, unattended migration of files either by using the scheduling feature of VSM (see “Scheduling Migrations” on page 292), or by using `cron` to invoke the `migbatch` command (see “Backup and Migrate Script” on page 155). When setting up schedules, the main considerations are as follows:

- Best times for migrating files
- How often to migrate files
- Time required to complete migration

The guidelines here also apply to using `miglow`.

Best Times for Migrating Files

If a user or process writes to a file system and the free space is less than the high-water mark percentage, VSM attempts to make space by removing premigrated files that you have previously created with `migbatch`. If there are no premigrated files, VSM automatically performs a migration by selecting files and copying them to secondary storage.

For large file systems, migrating files can take a long time, depending on factors such as:

- Size of the file system
- Number of files selected
- Availability of drives
- Transfer rates from disk to secondary storage
- Level of activity on the system.

The result can be reduced system performance and delays for users while VSM migrates files to secondary storage. Therefore, you need to anticipate requirements and execute migrations before space is needed and at times when system activity is low.

The best time to execute `migbatch` is at night or on weekends, when user and network activity is lowest. You can also schedule around restrictions such as large jobs that run only at night and on weekends, or departments that work in shifts to fully utilize the equipment.

To avoid performing extra backups, synchronize your migration and system backup schedules.

How Often to Migrate Files

In addition to knowing the best times during the week to migrate files, you need to know how frequently to perform migrations. The main factor here is the amount of new space users need every day. This determines how much time it takes for the file system to fill from the low-water mark to the high-water mark level.

You determined percentages for the high-water mark and low-water mark earlier in this chapter. Based on those values and the amount of space used each day, you can determine how often you need to migrate files. A good approach is to migrate files at least twice as often as it takes to fill the file system from the low-water mark to the high-water mark. For example, if you determine that the file system fills in two days, then migrate to the low-water mark every day.

After configuring an existing file system for VSM management, you can migrate files to the low-water mark before allowing users to resume normal operations with the file system. Perform this initial migration at night or on a weekend to minimize the effect on users.

Specify daily migrations in your initial schedule if possible. You can then assess the results and determine whether to adjust the frequency. A benefit to scheduling the operations daily is that if a system problem prevents migration on one day, `cron` executes the operation again on the following day. However, if you schedule migration only once a week and then miss a migration, `cron` waits another week before executing the operation again.

Time Required to Complete Migration

The time to complete a migration is also an important factor and depends primarily on the amount of data that you move. As the amount of data increases, so does the time required to complete the operation. Therefore, consider the amount of data and total time when determining when and how often to execute file migration.

One approach to dealing with large amounts of data is to back up and migrate different file systems on different schedules. This enables you to spread the migrations over several days. Another approach is to migrate files more often, providing it does not violate the guidelines in the previous two topics. If you have enough devices, concurrent recording also reduces migration time.

Schedule for Example Site

On `bunny` at the example site, the best time for backup and migration is between 10 pm and 6 am on weekdays and between 6 pm and 6 am on weekends. Based on this, schedule backups to start every night at 10 pm, followed by migration with `migbatch` at 1 am.

Reversing the schedule by starting file migration at 10 pm, followed by backups at 3 am, would not affect the time needed to complete file migration but would reduce backup time. When backup precedes migration, entire files are backed up. When migration precedes backup, however, only the inodes of any migrated files are backed up.

Completing the Example Configuration Plan

The following procedure summarizes the steps in the planning process by completing the configuration for bunny at the example site. The previous topics provide more information on each of these steps. Figure 43 on page 113 through Figure 49 show the completed worksheets for the example site.

1. Complete the global configuration (see Figure 43):
   a. Choose the managed file systems and specify the HSMDEV entry to which they belong. Table 2 on page 56 lists the file systems on bunny at the example site. Table 3 on page 61 lists only those file systems on bunny to be managed by VSM, and assigns their respective HSMDEV names.
   b. Specify the name for each HSMDEV entry log file. All VSM configurations on bunny use the form:

      `/var/log/`*`hsmname`*`.log`

      For example, the log for hsm1 is

      `/var/log/hsm1.log`

      Table 4 on page 62 lists the log file paths for each HSMDEV name on bunny at the example site.
   c. Specify the management state for each HSMDEV entry. On bunny all states are 1 (Active) so automatic space management and caching is enabled.

      Selecting the database directory for each VSM and its file system is also part of global configuration. However, we cannot complete that step until after we establish:

      - Migration criteria and preferred storage methods for each file system (steps 2 and 3).
      - Which managed file systems have common storage requirements for migrated files and therefore can share the same database (step 4).
2. Establish the migration thresholds for each managed file system:
   a. Specify start point (high-water mark) and stop point (low-water mark) for migration of files from managed file systems. Also specify the disk space to make available (purge mark) whenever free space drops below the high-water mark.

      The server bunny at the example site uses the same values for all managed file systems. These values are 10% free space for the high-water mark (default), 15% free space for the purge mark, and 20% free space for the low-water mark. Specifying a value for the low-water mark provides a limit on file selection.
   b. Specify criteria for migrating files:
      - Minimum age file to migrate (default is 7)

- Minimum size file to migrate (default is 8)
- Badness (default is 30)

The example site uses the standard badness formula to calculate file badness. See "How VSM Selects Files to Migrate" on page 71.

All managed file systems on `bunny`, except `/home2`, use 1 for minimum size, 1 for minimum age, and the default badness of 30. On `/home2`, most working files happen to be less than 3 days old and 30 Kbytes in size. Using 3 days for minimum age and a badness of 90 (3 days * 30 kbytes) prevents VSM from migrating the files for at least 3 days. In most cases this is enough to finish work on them.

Although VSM caches a file back in the file system when a user accesses it, there is a delay involved. Keeping frequently accessed files on the disk eliminates the delay and also reduces load on the system due to transferring them back from tape or optical disc.

c. Specify criteria for purging premigrated files:

- Minimum age file to purge (default is 0)
- Minimum size file to purge (default is 0)
- Purge badness (default is 0)

The example site uses the standard purge badness formula to calculate file purge badness. See "Define Criteria for Purging Files" on page 78.

All managed file systems on `bunny`, except `/home2`, use the default values for minimum size, minimum age, and purge badness. On `/home2`, however, the administrator wants to retain all migrated files in premigration for at least 5 days.

Note For kernel-based implementations (Solaris *ufs* file systems), minimum age is in days since migrated. For nonkernel-based implementations, minimum age is in days since either accessed or modified, whichever is less.

For kernel-based implementations (Solaris *ufs* file systems), using a minimum age of 5, a minimum size of 0, and a purge badness of 0 prevents VSM from purging premigrated files for at least 5 days.

For nonkernel-based implementations, using a minimum age 5 days greater than the migrate minimum age (3 + 5 = 8 for this example on `/home2`), a minimum size of 0, and a purge badness of 0 prevents VSM from purging premigrated files for at least 5 days.

At the example site, files are accessed infrequently after this time so caching delays are minimal.

These thresholds and selection criteria for the managed file systems on bunny at the example site are recorded on Figure 36 on page 93 through Figure 39.

3. Include stop file information suitable for all file systems using the database (see "Quota on Migration Stop Files" on page 97). The user-space-restriction quota in this example is 400,000 bytes.

4. Determine the best storage methods to use for each managed file system. Table 5 on page 91 lists the best storage methods for each managed file system on bunny at the example site. These are recorded on Figure 36 on page 93 through Figure 39.

   As a part of completing the storage method definitions, you assign volume sets and determine whether you want to use concurrent recording. The example site has two tape libraries and two optical disc libraries. Therefore, the VSM configurations on bunny use concurrent recording for all migrations in order to reduce migration times.

5. Compare the storage-method requirements of the managed file systems and determine which file systems can share a database.

   Then, specify a database directory for each VSM and its managed file system. Table 6 on page 107 lists the database directories for each managed file system on bunny at the example site.

   When you know the database directories, you can complete the global configuration worksheet. See Figure 43 for the example site. You can also complete the **migconf** planning worksheet for each database. Figure 44 on page 114 through Figure 49 show the completed migconf worksheets for the example site. (Move attributes are not shown in these figures.)

Note After entering data from the planning worksheets using the xhsmadm GUI, make sure that you save migconfg and migconf manually or the changes will not be entered.

Note In this example, all storage methods use the default volume pool, HSM.

6. Although not shown on the worksheets, the final step in the planning process is to define migration schedules.The example site uses crontab entries to start backup at 10:00 pm every night and migbatch at 1:00 am every morning for all managed file systems. These times do not interfere with normal user activities (see "Best Times for Migrating Files" on page 108).

Figure 43. Worksheet for Example Global Configuration

Global Configuration Worksheet

VSM Name: hsm1
Mount Point: /home1
Database Pathname /usr/var/openv/hsm1_2
Logfile Pathname: /var/log/hsm1.log
State: 1 (active)

VSM Name: hsm2
Mount Point: /home2
Database Pathname /usr/var/openv/hsm1_2
Logfile Pathname: /var/log/hsm2.log
State: 1 (active)

VSM Name: hsm3
Mount Point: /dbrecs
Database Pathname /usr/var/openv/hsm3
Logfile Pathname: /var/log/hsm3.log
State: 1 (active)

VSM Name: hsm4
Mount Point: /sd3
Database Pathname /usr/var/openv/hsm4
Logfile Pathname: /var/log/hsm4.log
State: 1 (active)

Figure 44. Worksheet for Example 1 migconf (1 of 2)

Managed-File-System Configuration Planning Worksheet (1 of 3)

Database Pathname: /usr/var/openv/hsm1_2

File System Migration Thresholds

VSM Name:	hsm1	hsm2		
File System Name:	/home1	/home2		
High-Water Mark (free space):	10	10		
Purge Mark:	15	15		
Low-Water Mark:	20	20		
Badness:	30	30		
Minimum Age (days):	1	3		
Minimum Size (kbytes):	1	1		
Age Weight:	1	1		
Size Weight:	1	1		
Weight Operator:	*	*		
Purge Badness:	0	0		
Purge Minimum Age (days):	0	5		
Purge Minimum Size (kbytes):	0	0		
Purge Age Weight:	1	1		
Purge Size Weight:	0	0		
Purge Weight Operator:	+	+		

Migration Parameters for all File Systems Using this Database

Machine ID: _____ User-Space-Restriction Quota (bytes): 400000

Copies: _____ Unmount Delay (seconds): _____ Checksum (y/n): _____

Figure 45. Worksheet for Example 1 migconf (2 of 2)

Managed-File-System Configuration Planning Worksheet (2 of 3)

Database Pathname: /usr/var/openv/hsm1_2

Storage Methods for All File Systems Using this Database

Copies: 2 (1 or 2)

METHOD1: op.1.library/op.2.library

METHOD1: ct.1.vault/ct.2.vault

METHOD1:

METHOD1:

METHOD1:

METHOD1:

METHOD1:

METHOD1:

Method Names for All File Systems Using this Database

Method Name:	ct	dt	mt	op	ow	ad	ft	nb
Flags:								
Access Time:							na	na
Capacity:				na	na		na	na
Speed:							na	na
Granule Size:							na	na
Block Size:				na	na	na	na	na
Density:						na	na	na
DeadmanTimeout:						na		
Demand Delay:						na	na	na

Figure 46. Worksheet for Example 2 migconf (1 of 2)

Managed-File-System Configuration Planning Worksheet (1 of 3)

Database Pathname: /usr/var/openv/hsm3

File System Migration Thresholds

VSM Name:	hsm3			
File System Name:	/dbrecs			
High-Water Mark (free space):	10			
Purge Mark:	15			
Low-Water Mark:	20			
Badness:	30			
Minimum Age (days):	1			
Minimum Size (kbytes):	1			
Age Weight:	1			
Size Weight:	1			
Weight Operator:	*			
Purge Badness:	0			
Purge Minimum Age (days):	0			
Purge Minimum Size (kbytes):	0			
Purge Age Weight:	1			
Purge Size Weight:	0			
Purge Weight Operator:	+			

Migration Parameters for all File Systems Using this Database

Machine ID: _____ User-Space-Restriction Quota (bytes): 400000

Copies: _____ Unmount Delay (seconds): _____ Checksum (y/n): _____

Figure 47. Worksheet for Example 2 migconf (2 of 2)

Managed-File-System Configuration Planning Worksheet (2 of 3)

Database Pathname: /usr/var/openv/hsm3

Storage Methods for All File Systems Using this Database

Copies: 2 (1 or 2)

METHOD1: ct.1.library/ct.2.library

METHOD1: ct.3.vault/ct.4.vault

METHOD1:

METHOD1:

METHOD1:

METHOD1:

METHOD1:

METHOD1:

Method Names for All File Systems Using this Database

Method Name:	ct	dt	mt	op	ow	ad	ft	nb
Flags:								
Access Time:							na	na
Capacity:				na	na		na	na
Speed:							na	na
Granule Size:							na	na
Block Size:				na	na	na	na	na
Density:						na	na	na
DeadmanTimeout:						na		
Demand Delay:						na	na	na

Figure 48. Worksheet for Example 3 migconf (1 of 2)

Managed-File-System Configuration Planning Worksheet (1 of 3)

Database Pathname: /usr/var/openv/hsm4

File System Migration Thresholds

VSM Name:	hsm4			
File System Name:	/sd3			
High-Water Mark (free space):	10			
Purge Mark:	15			
Low-Water Mark:	20			
Badness:	30			
Minimum Age (days):	1			
Minimum Size (kbytes):	1			
Age Weight:	1			
Size Weight:	1			
Weight Operator:	*			
Purge Badness:	0			
Purge Minimum Age (days):	0			
Purge Minimum Size (kbytes):	0			
Purge Age Weight:	1			
Purge Size Weight:	0			
Purge Weight Operator:	+			

Migration Parameters for all File Systems Using this Database

Machine ID: _____ User-Space-Restriction Quota (bytes): 400000

Copies: _____ Unmount Delay (seconds): _____ Checksum (y/n): _____

Figure 49. Worksheet for Example 3 migconf (2 of 3)

Managed-File-System Configuration Planning Worksheet (2 of 3)

Database Pathname: /usr/var/openv/hsm4

Storage Methods for All File Systems Using this Database

Copies: 2 (1 or 2)

METHOD1: op.1.library/op.2.library

METHOD1: op.3.vault/op.4.vault

METHOD1:

METHOD1:

METHOD1:

METHOD1:

METHOD1:

METHOD1:

Method Names for All File Systems Using this Database

Method Name:	ct	dt	mt	op	ow	ad	ft	nb
Flags:								
Access Time:							na	na
Capacity:				na	na		na	na
Speed:							na	na
Granule Size:							na	na
Block Size:				na	na	na	na	na
Density:						na	na	na
DeadmanTimeout:						na		
Demand Delay:						na	na	na

Configuring VSM (Motif-based GUI) 3

This chapter explains how to configure VSM with the Motif-based GUI, `xhsmadm`. You can configure VSM with either `xhsmadm` or VSM-Java. Refer to "Configuring VSM (Java-based GUI)" on page 179 for information on how to configure VSM using the Java-based GUI, VSM-Java.

Caution VERITAS strongly recommends that you use only one of the two GUIs, VSM-Java or `xhsmadm`, to reconfigure your system. Do not use both GUIs simultaneously on your system; doing so can produce unpredictable results.

The major topics discussed in this chapter are as follows:

- Install VSM Software
- Configure Tape and Optical Storage Devices
- Perform Global Configuration
- Configure Servers
- Configure Migration Parameters
  - Assign Default Values
  - Assign Storage Methods
  - Assign File System Attributes
  - Configure and Edit Storage Method Names
  - Configure and Edit Migration Levels
- Administer Parallel Inode Files
- Administer Inode-to-Handle Files
- Update fstab or vfstab File
- Update Scripts and Create crontab Entries
- Link Global Log to Non-tmp File
- Control Migration of Specific Files

- Enable User Permissions
- Register Media with VSM
- Initial Startup and Testing

The examples use information from the planning worksheets developed in chapter 2 of this manual. Familiarize yourself with the information in chapter 2 before starting to configure VSM. The background and planning information in that chapter will make configuration much easier.

Install VSM Software

To use VERITAS Storage Migrator or VERITAS Storage Migrator Remote, you must first install the software by following the instructions in the *VERITAS Storage Migrator for UNIX Installation Guide*:

1. Install Media Manager if you plan to use optical disc or tape secondary storage devices. This is not necessary if you only intend to use VERITAS Storage Migrator Remote for secondary storage on local or remote magnetic disks. The examples in this chapter assume a Tape Library robotic device. The *Media Manager System Administrator's Guide* lists other available robotic devices.
2. Install VERITAS Storage Migrator or VERITAS Storage Migrator Remote.

Caution For kernel-based implementations (Solaris *ufs* file systems), do not use a VSM managed file system as a regular (ufs) file system on Sun platforms. Changes will corrupt the file system in that configuration, and it will no longer provide proper access to migrated files. Therefore, an *hsm* file system must always be mounted and used as an *hsm* file system except where otherwise indicated in supporting documentation.

Configure Tape and Optical Storage Devices

If you have installed VSM and VERITAS Media Manager, you will also need to configure tape and optical devices and start the Media Manager daemon.

Refer to the *Media Manager System Administrator's Guide* for a detailed explanation of how to configure secondary storage devices correctly, create the necessary VSM volume pools, and configure media within Media Manager.

First, create device (`/dev`) entries for the robots, tape drives and other devices used for storage. Once this is done, you will need to define these robots and their devices to Media Manager.

After configuring the robots and drives, you start Media Manager with the `ltid` command which, in turn, initiates the robot daemons. Also enable automatic tape head cleaning for robotic tape devices.

Use one of the Media Manager administration utilities (`vmadm` or `xvmadm`) to configure one or more volume pools for VSM. This allows you to designate specific volumes for use by VSM.

When you have created a VSM volume pool, add the volumes in that pool to the Media Manager volume database.

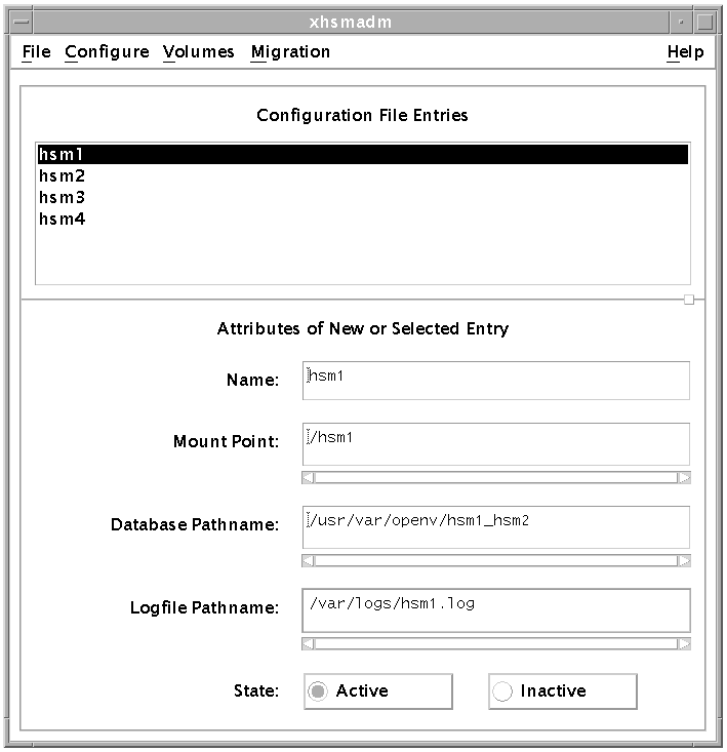
Perform Global Configuration

VSM has two graphical user interfaces (GUI), either of which lets you easily configure your system. This chapter describes the Motif-based GUI, `xhsmadm`. This GUI requires a display server compatible with X11.4 or later. It conforms to OSF / Motif conventions featuring icons and pull-down menus, and includes full mouse support for greater ease of use. Refer to "Configuring VSM (Java-based GUI)" on page 179 for information on how to configure VSM using the Java-based GUI, VSM-Java.

Note VERITAS recommends that you choose which administrative GUI to use, and then refrain from alternating between the Motif-based and Java-based GUIs for your ongoing system administrative activities.

Figure 50, shows the main `xhsmadm` screen you use to perform global configuration. Here you define each file system directory that you intend to manage by specifying it in an HSMDEV entry in the VSM global-configuration file, `/usr/var/openv/hsm/database/migconfg`. Figure 50, displays the `migconfg` file for hsm1 on `bunny` at the example site described in chapter 2 of this manual. This information corresponds to Figure 43 on page 113.

Figure 50. xhsmadm_main



For global configuration, perform the following steps to define each managed file system:

1. Start `xhsmadm` (after defining the *DISPLAY* variable):

   ```
   /usr/openv/hsm/bin/xhsmadm &
   ```

2. Create an HSMDEV entry for each file system directory you are going to manage. Selecting each configuration file entry automatically displays the attributes assigned to that entry. You can add entries and edit attributes easily through the `xhsmadm` interface.

   Attributes that you include in each configuration file entry are as follows:

   Name

   Name (*hsmname*) that you assign to this HSMDEV entry. This name must be a unique alphanumeric value such as: `hsm1`

   You can use only letters or a combination of letters and numbers with no trailing whitespace. Avoid using only numbers because this can cause some utilities to work incorrectly. Maximum name length is 32 characters. The default name is `hsm`. You can configure up to 64 *hsmnames*.

Mount Point

Path name (*fspath*) of the file system for this HSMDEV entry. It is the mount point for the file system. The Mount Point parameter is required. The full path *fspath* (plus any managed directories) must be 1023 characters or less (see "File Systems" on page 128 for more information).

Caution Always create the database directory in a local file system that VSM does not manage. This eliminates the possibility of migrating files from the database or workdir directories.

Note If another file system is mounted in a VSM managed file system or managed directory, it will not be managed by HSM.

Database Pathname

Path name (*dwpath*) of the directory containing the database and workdir directories. This contains information about the files migrated from this HSMDEV entry's file system. One of the files in *dwpath*`/database` is called `migconf`, and it contains the migration parameters for the file system.

The full paths *dwpath*`/database/`*filename* and *dwpath*`/workdir/`*filename* must be 1023 characters or less. The maximum *filename* recognized in these paths is 64 characters. The default *dwpath* is `/usr/var/openv/hsm.`*hsmname*.

Logfile Pathname

Path (*lgpath*) to the file that will contain log messages for operations pertaining to the file system and databases. The full path *lgpath* must be 1023 characters or less. The default value is `/tmp/hsm.`*hsmname*`.log`. Always change the log path to something other than `/tmp`, since files in `/tmp` can disappear after a reboot or system crash.

State

Specifies whether automatic disk-space management and access to migrated files is Active (1) or Inactive (0) for this file system. The default state is Active.

Note If the state is Inactive (0) on kernel-based implementations (Solaris *ufs* file systems), VSM blocks all file system activity for users other than root, and sends an appropriate error message. If the state is Inactive (0) on non-kernel-based implementations, VSM denies access to migrated files for all users but allows some other file system activity.

Pull down the Configure menu of the main `xhsmadm` screen and select Add, then pull down the File menu of the main `xhsmadm` screen and select Save global configuration file.

Configure Servers

To make database directories, workdir, and database files (including the `.PAIN` file on required platforms) for each HSMDEV entry you have created, click on an entry in the Configuration File Entries field of the main `xhsmadm` screen. Pull down the Configure menu, and select Execute setuphsm command. Repeat for each *hsmname* in your global configuration.

Note The `setuphsm` command will unmount the managed file system. This is possible only if the file system is not in use. Use the UNIX `fuser` command to determine if the file system is in use. If it is, `setuphsm` will fail and issue an error message. Follow the procedure in an appendix of the Release Notes to do manually what `setuphsm` does automatically.

This defines where the database files will reside and where VSM will store working files. It also copies the initial database files into the database directory. The database pathname specified in the global configuration file defines where the database and workdir directories will reside.

Configure Migration Parameters

Each database pathname that you specify during global configuration contains a set of database files and also a configuration file called `migconf`. The `migconf` file controls file migration and storage for those file systems that use the database.

In Figure 44 on page 114, `hsm1` and `hsm2` both use the database in the `/usr/var/openv/hsm1_2` directory. Therefore, the `migconf` file in that directory contains the storage methods and migration parameters for the `/home1` and `/home2` file systems.

Figure 51, shows the `xhsmadm` edit screen you use to create and configure `migconf` files for your VSM databases. It displays the defaults and storage methods for the `/usr/var/openv/hsm1_2/database/migconf` file on `bunny` at the example site described in chapter 2 of this manual. This information corresponds to Figure 44 on page 114.

Pull down the Dialog menu of the `xhsmadm` edit screen and select Save configuration file after entering any configuration changes.

Figure 51. xhsmadm_edit

Assign Default Values

The following procedure explains how to create and configure `migconf` files for your VSM databases:

1. From the main `xhsmadm` screen, double click on an entry in the Configuration File Entries field, or pull down the Edit menu and select Edit configuration file. This opens the `xhsmadm` edit screen for the database used by that file system (see Figure 51).
2. In the upper section of the screen, change the defaults to the desired values. You specify one set of defaults in `migconf` and they apply to all file systems using this database. The fields are as follows:

Machine ID

Integer (nonzero) identifier for each VSM managed file system. All file systems exporting or importing files between them must be distinct; each must have a unique Machine ID. VERITAS recommends a unique Machine ID for each *hsmname*. Valid values range from 1 to FFFFFF. The default value is 1000.

Quota

Maximum number of bytes that each user can restrict from migration with their `.migstop` files. The default value is 10 million bytes (10000000). The example on Figure 51 uses 400000.

Copies

Number of copies of each file that VSM migrates to secondary storage. The maximum value is 2 and The default value is 2. You must configure a METHOD1 for the first copy and a METHOD2 for the second copy (see "Assign Storage Methods" on page 129).

Unmount Delay

Time in seconds a volume that is mounted in read mode remains mounted pending another read request. If no read request arrives prior to the expiration of this time delay, VSM unmounts the volume (see "Demand Delay" on page 144). The default value is 3 minutes. Applicable to `ct`, `dt`, `mt`, `op`, and `ow` methods.

Checksum

Specifies whether VSM calculates a check sum as it writes files to secondary storage. If Checksum is Yes (1), VSM calculates a check sum for each granule it writes. If No (0), VSM does not calculate a check sum. The default value is Yes.

During a read, VSM checks the checksum only if the granule was written with a check sum.

Storage Methods

Storage methods assigned to the file systems using this `migconf` (see "Assign Storage Methods" on page 129).

File Systems

Managed file systems or managed directories using this `migconf`. Managed file systems must be at the mount point designated by *fspath*. Managed directories (*dirm*) in a mounted file system must be below the mount point. Enter managed directories in the form *fspath*/*dirm*. This expression must not contain a trailing slash (see "Mount Point" on page 125). For examples, see "File Systems to Manage" on page 58.

Method Names

Method names used in the assigned storage methods in this `migconf`. Most sites will not have to change this section other than to disable superfluous entries, which cause additional VSM processing.

Note Never disable the `dk` method. This method is necessary to support premigration.

If the assigned storage methods (METHOD 1 through METHOD8) specify only method names `op` and `ct`, for example, you must include `dk`, `op`, and `ct` in this `migconf`, and methods `dt`, `mt`, `ow`, `ad`, `nb`, and `ft` become superfluous.

Levels

Migration levels with assigned move criteria (see "Configure and Edit Migration Levels" on page 145). You cannot edit this field directly; instead, add and delete levels using the GUI as shown in Figure 57 on page 146.

Pull down the Dialog menu of the `xhsmadm` edit screen and select Save configuration file after entering any configuration changes.

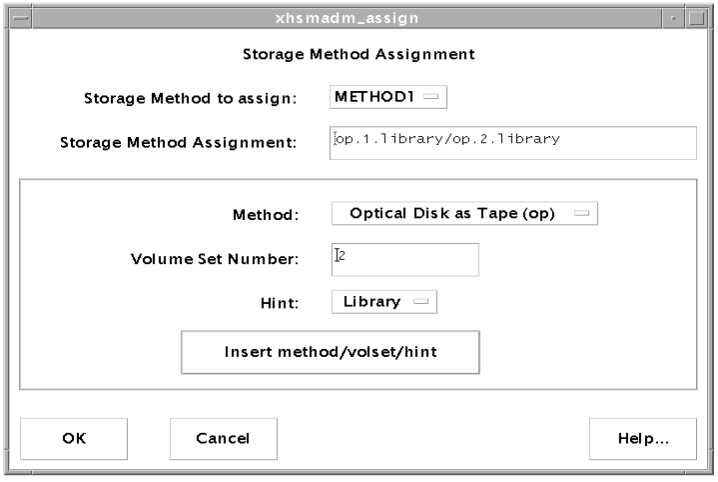
Assign Storage Methods

The following procedure explains how to assign the storage method(s) for all of the file systems using this VSM database.

Caution If you are editing an existing configuration, always execute `migrc -R` before changing the METHOD1 through METHOD8 parameters. This ensures that all previous migration processes are complete.

1. To add a new storage method, pull down the StorageMethods menu on the `xhsmadm` edit screen and select Add storage method. To delete an existing storage method, click on an entry in the Storage Methods field, pull down the StorageMethods menu, and select Delete storage method. To edit an existing storage method, double click on an entry in the Storage Methods field, or pull down the StorageMethods menu and select Edit storage method. This brings up the `xhsmadm` assign screen (see Figure 52).

Figure 52. xhsmadm_assign



2. Use METHOD1 and METHOD2 parameters to define where VSM writes each copy of migrated files. (See Copies in "Assign Default Values" on page 127.) Use METHOD3 through METHOD8 to define where VSM moves file copies after they are migrated. (See "Choose Storage Methods for Moving Files" on page 97.) You assign one set of storage methods in `migconf` and they apply to all file systems using this database.

Caution The number and type of your storage drives must correspond to the storage methods you configure. For example, if you configure tape method names for both METHOD1 and METHOD2, you must have at least two tape drives configured.

- Add a METHOD1 to define where to write the first copy.
- Add a METHOD2 to define where to write the second copy (if used).
- Add METHOD3 through METHOD8 to define where to move migrated files from one volume set to another (if used). Each of these storage methods is associated with its own migration level, 3 through 8 respectively.

The format of METHOD1 through METHOD8 is as follows:

method_name.volume_set_number.hint.volume_pool

This format is called a *stripe*. To define concurrent recording for a copy or to control where a copy is written, specify multiple stripes, separating the stripes by slashes:

method_name . *volume_set_number* . *hint.volume_pool* /
method_name . *volume_set_number* . *hint.volume_pool*

Note The volume pool parameter is optional, and defaults to `HSM`.

You can edit these text fields directly with keyboard and mouse, or you can add new stripes to the default volume pool by using the option menus and text fields on the `xhsmadm` assign screen, and clicking the Insert Method/Volset/Hint pushbutton.

Observe the following rules when configuring these parameters:

- Stripes in any volume pool must contain unique volume sets, where a volume set is defined as a method name and its volume set number. If you configure a method name more than once, use a different volume set number each time, regardless of the volume pool. This prevents VSM from possibly migrating both copies to the same physical media.
- Choose a method name from the following list (see "Method Name" on page 83 for more information on each method):
  - `ad` (alternate magnetic disk)
  - `ct` (tape - STK-9840 technology)
  - `dt` (tape - DLT 7000 technology)

- `mt` (tape - Sony AIT-2 technology)
- `op` (optical disc as tape with random seek - rewritable)
- `ow` (optical disc as tape with random seek - write once, read many)
- `ft` (remote method using `ftp`) Method name `ft` is valid only for METHOD1 and METHOD2.
- `nb` (NetBackup) Method name `nb` is valid only for METHOD1 and METHOD2.

The descriptions above correspond to the default attributes in the example `migconf` file. Tape method names can be used for other technologies and optical disc method names can be used for other technologies if you use the GUI to modify the method attributes accordingly in `migconf` (see "Configure and Edit Storage Method Names" on page 139).

- Ensure that each method name matches one in the Methods section of the `migconf` file. (See Methods in "Assign Default Values" on page 127.) Note that `dk` (disk file) is used only for premigration.
- Choose a volume set number. Two basic rules for these numbers are:
  - For METHOD1 and METHOD2, either use different method names or different volume set numbers if using the same method name. This ensures that each copy goes to a different volume, thus reducing the chances of losing both copies of a file.
  - Use different volume set numbers for each portion of a copy when using concurrent recording. This prevents VSM from writing concurrently to the same volume.

Note When you register one side of an optical disc with `migreg`, VSM also automatically registers the other side with the same volume set number to avoid this problem.

See "Volume Set Number" on page 86 and "Concurrent Recording" on page 88 for more information on setting up volume sets.

- Use a hint appropriate for the availability of the volume set. When caching files and two copies exist, VSM always chooses the most available copy, which is the one on the volume it can access in the least time. If that volume is not available, VSM requests another volume. The valid hint values are as follows:

  Library

  Volume is in an online library unit (an automounting or robotic device). Access is immediate, so no access-time bias is added.

Operator

Operator action is required to mount the volume. This choice adds 15 minutes to the access time.

Vault

Volume is stored at a remote location. This choice adds 24 hours to the access time.

Note Cache requests are issued immediately, regardless of hint value.

See "Volume Set Availability (hint)" on page 87 for more information on the hint parameter.

- Designate a volume pool if different than the default `HSM`. Edit the text field on the `xhsmadm` assign screen directly with keyboard and mouse.

  Always configure the same volume pool for a given volume set, where a volume set is defined as a method name and its volume set number. The same volume set cannot exist in more than one volume pool.

The example on Figure 52, shows the METHOD1 parameters for the `/usr/var/openv/hsm1_2` database on `bunny` at the example site (see Figure 45 on page 115). These settings cause VSM to use concurrent recording.

- Copy 1 goes to optical disc and is distributed between volume sets `op.1` and `op.2`. These volume sets reside in the online library.
- Copy 2 goes to cartridge tape and is distributed between volume sets `ct.1` and `ct.2`. These volume sets reside off site.

Note You must always configure METHOD1. However, if you are creating only one copy, METHOD2 can remain undefined.

Note See "Configure and Edit Storage Method Names" on page 139 for additional information on changing storage methods.

Pull down the Dialog menu of the `xhsmadm` edit screen and select Save configuration file after entering any configuration changes.

Assign File System Attributes

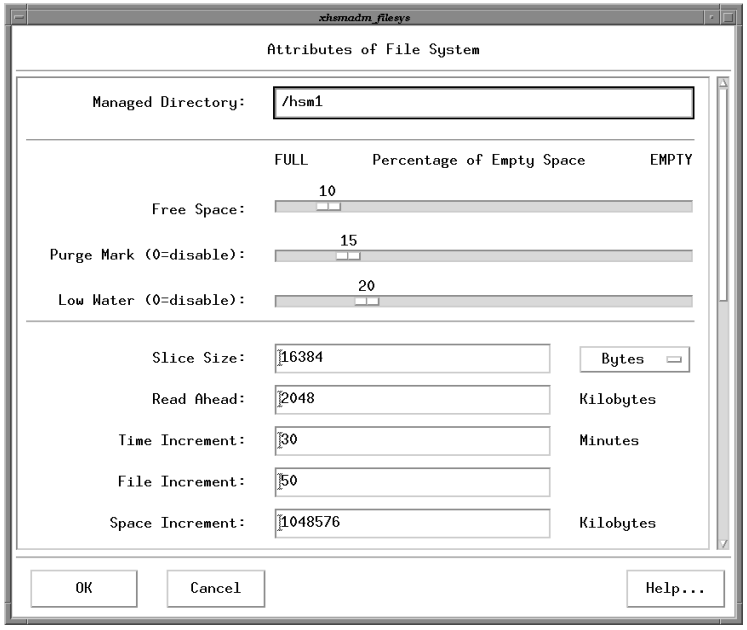
The following procedure explains how to assign the file system attributes for each of the managed file systems or managed directories using this VSM database.

1. To add a new file system, pull down the FileSystems menu on the `xhsmadm` edit screen and select Add file system.

Note If more than one file system uses the database, you must create a FILESYS entry for each of them.

To delete an existing file system, click on an entry in the File Systems field, pull down the FileSystems menu, and select Delete file system. To edit an existing file system, double click on an entry in the File Systems field, or pull down the FileSystems menu and select Edit file system (see Figure 51). This brings up the `xhsmadm` filesys screen. (see Figure 53).

Figure 53. xhsmadm_filesys (part 1 of 2)



2. Change the file system attributes to the desired values. The attributes in each entry are defined as follows:

Managed Directory

Name of the managed file system or managed directory to which this entry applies.

Free Space

Referred to as high-water mark in chapter 2. When the percentage of free space falls below this amount, the `mignospace` and `miglow` commands make more space available by initiating migration operations. The default value is 10 (percent).

Attempting to migrate files when the managed file system is 100% full (ENOSPC) can cause error conditions. It is important, therefore, to configure adequate free space and migrate files often enough to avoid this condition.

Note To avoid error conditions, configure free space to be less than 100%. For kernel-based implementations (Solaris *ufs* file systems), do not set free space to 0.

Purge Mark (0 = disable)

Referred to as purge mark in chapter 2. `mignospace` either purges all premigrated files or stops purging when free space reaches the purge mark percentage, whichever comes first. (`mignospace -i` purges all premigrated files, ignoring the purge mark.) If specified, purge mark must be greater than or equal to the `freespace` setting, and less than or equal to the `lowwater` setting. If no `purgemark` is specified, `mignospace` purges all premigrated files. The default value is 0 (interpreted by VSM as no purge mark).

Low Water (0 = disable)

Referred to as low-water mark in chapter 2. This is the percentage of free space at which VSM stops selecting files for migration. If specified, low water must be greater than or equal to the free space setting. If no low-water mark is specified, VSM continues to select files until there are no more files that meet the selection criteria (see the three parameters minimum age, minimum size, and badness described below). The default value is 0 (interpreted by VSM as no low-water mark).

Note The GUI enforces rules governing the relative values of free space, purge mark and low water after you change one of them using a mouse.

Note Configuring a smaller value for Low Water results in faster sweeps of the file managed file system or directory, but these sweeps can terminate before selecting some files with large badness values. Configuring a larger value for Low Water results in slower but more extensive sweeps. HSM's round robin sweeping, however, eventually scans all of the files in the managed file system or directory.

Figure 53 displays the file system attributes for the `/hsm1` file system on `bunny` at the example site described in chapter 2 of this manual (see Figure 44 on page 114).

Slice Size

Number of bytes at the front of the file that VSM leaves stored on disk for migrated files. These bytes are also migrated but VSM keeps a copy of them in the file system even when it migrates the associated file to secondary media, thus allowing this number of bytes to be read without caching the file. In kernel-based implementations (Solaris *ufs* file systems), VSM needs additional disk space in the managed file system to hold the slice. You express the slice value in bytes and it can range from 0 to 65536 (64 kilobytes). On DMAPI implementations the upper limit to the slice is 2147483648 (2 gigabytes). The value 0 implies that no bytes will be kept in the file system. The default value is 8192 (8 kilobytes).

Read Ahead

Minimum number of kilobytes beyond the current `read` request that VSM partially caches to disk. Applies only to DMAPI implementations of HSM. Setting the value to -1 disables partial file caching (default).

Time Increment

Maximum time in minutes `migcopy` runs before stopping an accelerated file space availability operation. The default value is 60. A value of 0 signifies no limit.

File Increment

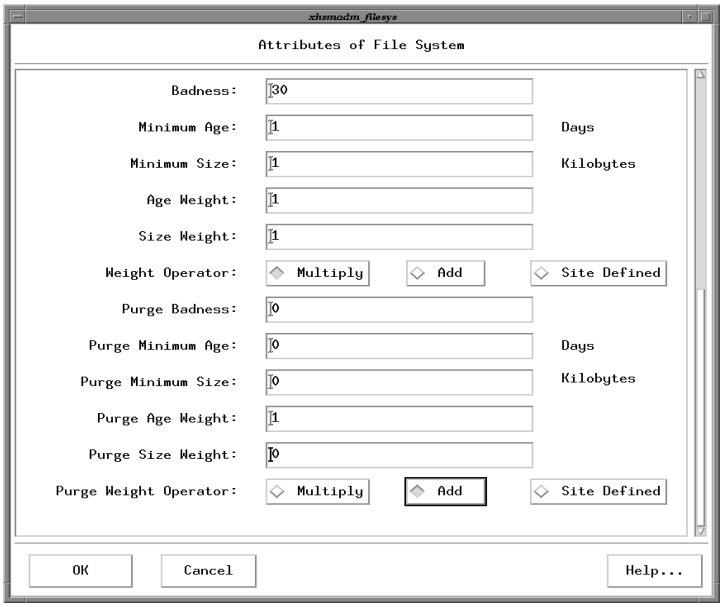
Maximum number of files processed by `migcopy` and `migsweep` before stopping an accelerated file space availability operation. A value of 0 signifies no limit (default).

Space Increment

Minimum amount of disk space (in kilobytes) freed by `migcopy` and `migsweep` before stopping an accelerated file space availability operation. The default value is 1,048,576. A value of 0 signifies no limit.

3. Use the vertical scroll bar on the `xhsmadm` edit screen to find the migration and purge attributes for each of the file systems using this VSM database (see Figure 54).

Figure 54. xhsmadm_filesys (part 2 of 2)



Badness

The criteria that VSM uses to select files to migrate after skipping those that are less than the minimum age and size. VSM computes each file's badness and selects those whose badness equals or exceeds the configured value. The VSM badness formula is as follows:

`age * age_weight (+ or *) size * size_weight`

The default value is 0.

File age in the VSM badness formula is the time in days since the file was last accessed or last modified, whichever is most recent. The term *Badness* in this context is equivalent to the term *Thresholds* defined on page 228.

In lieu of the VSM badness formula, administrators have the option to define a site-specified badness formula. (See Weight Operator.)

Minimum Age

Age of file (in days since last access or modification). VSM does not select files for migration that have been either accessed or modified within this time. Set this to a value greater than 0 to prevent VSM from migrating files on the same day you create them. The default value is 7.

Minimum Size

Size of file (in kilobytes). VSM does not select files for migration that are smaller than this. The default value is 8.

Note No disk space will be made available if a file smaller than the slice size is migrated. The recommended minimum size, therefore, equals or exceeds the configured slice size.

Age Weight

Weighting factor that VSM uses for age in the badness formula. The default value is 1.

Size Weight

Weighting factor that VSM uses for size in the badness formula. The default value is 1.

Weight Operator

Arithmetic operator used in the VSM badness formula (Multiply or Add). The default value is Multiply.

To define a site-specified badness formula, select Site Defined.

Note If a site configures a site-specified badness formula for calculating both file badness and file purge badness, the formulas are identical (see "How to Customize File Migration Criteria" on page 78).

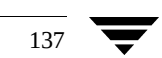
After you have configured these migration attributes, you can run `migtestbadness` to check what effect changing the badness computation parameters has on migration operations. The test results can help you modify the file system configuration for optimum migration performance. See the `migtestbadness`(1M) man page for more information on how to use this test.

Figure 54 displays the file system attributes for the `/hsm1` file system on `bunny` at the example site described in chapter 2 of this manual (see Figure 44 on page 114).

4. Change the purge attributes to the desired values. The attributes in each entry are defined as follows:

Purge Badness

The criteria that VSM uses to purge premigrated files after skipping those that are less than the minimum purge age and size. VSM computes each premigrated file's purge badness and selects those whose purge badness equals or exceeds the configured value.



The VSM purge badness formula is as follows:

```
age * purge_age_weight (+ or *) size * purge_size_weight
```

The default purge badness value is 0.

File age in the VSM purge badness formula is the time in days since the file was either migrated or copied, whichever is less (kernel-based implementations, Solaris *ufs* file systems), or the time in days since the file was either either accessed or modified, whichever is less (non-kernel-based implementations).

In lieu of the VSM purge badness formula, administrators have the option to define a site-specified badness formula. (See Purge Weight Operator.)

Purge Minimum Age

Age of premigrated file (in days). VSM does not purge premigrated files within this period. The default value is 0.

Note For kernel-based implementations (Solaris *ufs* file systems), minimum age is in days since migrated or copied. For non-kernel-based implementations, minimum age is in days since either accessed or modified, whichever is less.

Purge Minimum Size

Size of premigrated file (in kilobytes). VSM does not purge premigrated files smaller than this. The default value is 0.

Purge Age Weight

Weighting factor that VSM uses for age in the purge badness formula. The default value is 1.

Purge Size Weight

Weighting factor that VSM uses for size in the purge badness formula. The default value is 0.

Purge Weight Operator

Arithmetic operator used in the VSM purge badness formula (Multiply or Add). The default value is Add.

Note Because the default purge size weight is **0**, the default purge weight operator must be **Add**.

To define a site-specified badness formula, select Site Defined.

Note If a site configures a site-specified badness formula for calculating both file badness and file purge badness, the formulas are identical (see "How to Customize File Migration Criteria" on page 78). Figure 54 displays the default purge attributes.

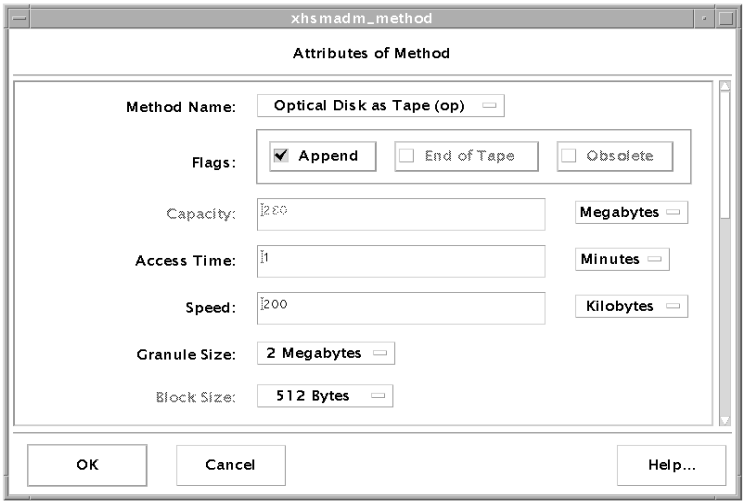
Pull down the Dialog menu of the xhsmadm edit screen and select Save configuration file after entering any configuration changes.

Configure and Edit Storage Method Names

The following procedure explains how to configure and edit storage method names for all of the file systems using this VSM database:

1. To configure a storage method name, pull down the MethodNames menu on the xhsmadm edit screen and select Add method name. To delete an existing method name, click on an entry in the Method Names field, pull down the MethodNames menu, and select Delete method name. To edit an existing method name, double click on an entry in the Method Names field, or pull down the MethodNames menu and select Edit method name (see Figure 51). This opens the xhsmadm method screen for the database used by that file system (see Figure 55).

Figure 55. xhsmadm_method (part 1 of 2)



2. Change the method name attributes to the desired values. Attributes that do not apply to the specified method name are shown in gray and cannot be changed. The attributes in each entry are defined as follows:

   Method Name

   The name of the storage method.

Note Method name dk is required, but dk attributes are not configurable except for the Obsolete flag.

Flags

The flags parameter settings depend on the device and media and are as follows:

- Obsolete

  Media supports granule obsoleting. (See man page migmdclean(1M) for more information on the Obsolete flag. Use this flag only for methods `ad`, `ft`, and `dk`).

- Append

  This flag allows VSM to place multiple migrations on the same volume by appending them to that volume until it is full to its configured limit. When writing to optical, VSM continues to append to the disc side until the side is full and then writes on the next empty disc side. With either tape or optical, the result is to allow smaller migrations without wasting space. Applies to methods `ct`, `dt`, `mt`, `op`, and `ow`.

  When the Append flag is not present, each migration always starts on an empty volume.

  When the Append flag is present, VSM performs the following two steps to select a volume for a new migration:

  1. Checks the dwpath`/database/VOLDB` file for:

  • A volume that belongs to the correct volume set.

  • A volume that is currently being written. When VSM selects an empty volume for migration, it adds WRITING 0x0040 to the entry for that volume in the *dwpath*`/database/VOLDB` file. This indicates that VSM is using that volume for writing.

  2. Based on the step 1 check, VSM selects a volume as follows:

  • If VSM finds a volume that is in the correct volume set and is also being written, it extends the volume with the new migration.

  • If VSM does not find a volume that meets the requirements of step 1, it selects an empty volume for the migration.

- End of Tape

  This flag allows VSM to write to the media until end of tape (EOT) is encountered instead of using the `capacity` value. It is still necessary to specify `capacity` because VSM uses that value for calculating volume requirements during consolidation. Applies only to `ct`, `dt` and `mt` methods.

Access Time

Time to access the media in seconds. VSM combines the `access` value with hint, speed, and file size to determine the relative time required to cache a copy of a file. The formula is as follows:

`Relative cache time = access + hint + file size/speed`

Where:

- Relative cache time is a value that VSM uses to determine which method to use to cache first.
- Access is the value of the `access` parameter.
- Hint is the value of the hint parameter (see "Volume Set Availability (hint)" on page 99). Hint values in this formula are as follows: `library` = 0, `operator` = 900, `vault` = 86400.
- File size is the total size of the file in bytes.
- Speed is the value of the `speed` parameter.

If there is more than one copy of a file on local secondary storage, VSM uses the relative cache time to determine which volume to select for a cache.

Note VSM attempts to cache remote copies first (if they exist) before attempting to cache copies from local media (see Figure 1 on page 3).

When VSM caches a file from local secondary storage, it attempts to use the volume with the shortest computed relative cache time. If that volume is not available, VSM requests another volume.

Capacity

Capacity of the method in bytes. Applies only to `ct`, `dt`, `mt`, and `ad` methods. During labeling, VSM records this value on the tape volume.

Note The capacity of an optical disc volume (`op` or `ow` method) is determined automatically by VSM when the volume is labeled. The capacity of the `ft` and `nb` methods is specified when you register volumes using the `migreg` command.

Speed

Relative speed with which the device can transfer data. As mentioned in the description for the `access` parameter, VSM uses speed when determining which copy of a file to cache.

Caution Do not change the granule size value for a method as long as any migrated files remain stored under that method. Otherwise, VSM cannot read the media and you can lose the data.

Caution Do not change the granule size value for a method to be less then the `dk` method granule size (256K default). Doing so may cause loss of data. The loss will not be noticed until the file is cached.

Granule Size

VSM divides files into granules. Each granule must fit on one volume. The granule size parameter specifies the number of bytes in each granule that VSM writes to the device. The allowable granule sizes are: 128K, 256K, 512K, 1M, 2M, 4M, 8M, 16M, 32M, and 64M. Granule size is a power of 2 and an integral multiple of block size. Does not apply to methods `ft` or `nb`.

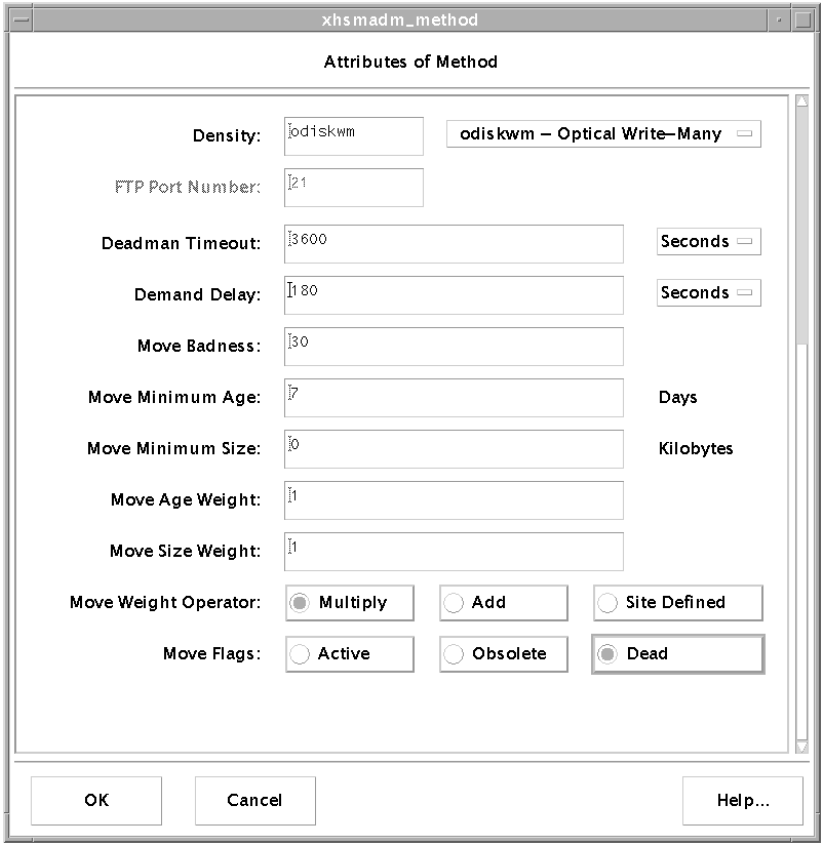
Block Size

Block size in bytes to use when writing to the device. The allowable block sizes are: 512, 1K, 2K, 4K, 8K, 16K, 32K, 64K, 128K, and 256K. The value must be a power of 2. Do not change this parameter after the initial configuration. Applies only to `ct`, `dt` and `mt` methods. It is not necessary to configure block size for methods `op` or `ow` because VSM determines the actual physical block size of optical volumes each time they are mounted or opened.

Caution Do not change this parameter after the initial configuration.

3. Use the vertical scroll bar on the `xhsmadm` edit screen to find the remaining attributes for this method (see Figure 56).

Figure 56. xhsmadm_method (part 2 of 2)



Density

Density of the tape or optical disc medium. Pull down the menu and select the type of storage medium configured for this storage name. This is used only for the `ct`, `dt`, `mt`, `op` and `ow` methods.

FTP Port Number

ftp port number. This is used only for the `ft` method. The default value is 21.

Deadman Timeout

The maximum period of time in seconds that VSM waits for an ftp or tape request to complete. The default value is 3600 seconds (one hour). This is used only for the `ct`, `dt`, `mt`, `op`, `ow`, `ft`, and `nb` methods.

Demand Delay

Time in seconds a mount request waits before VSM unmounts a similar unused volume. If VSM identifies a mounted but unused volume of the same density whose unmount delay has not yet expired, it unmounts that volume as soon as the demand delay occurs (see "Unmount Delay" on page 128). Otherwise, the mount request remains active until a drive becomes available. This is used only for the `ct`, `dt`, `mt`, `op`, and `ow` methods, and the default varies.

4. Configure and edit the criteria associated with this method name for moving migrated files to another migration level.

Note Move criteria for migration levels, if specified, take precedence over move criteria for method names, if specified (see Figure 32 on page 81).

Move Badness

The criteria that VSM uses to move files from one migration level to another after skipping those that are less than the minimum move age and size. VSM computes each migrated file's move badness and selects those whose move badness equals or exceeds the configured value. The VSM move badness formula is as follows:

```
age * move_age_weight (+ or *) size * move_size_weight
```

The default value is 30.

File age in the VSM move badness formula is the time in days since the file was migrated or moved to this level or since the file was last accessed, whichever is most recent.

In lieu of the VSM move badness formula, administrators have the option to define a site-specified move badness formula. (See Move Weight Operator.)

Move Minimum Age

Age of file (in days since migrated or moved to this level, or since accessed). VSM does not move files migrated, moved, or accessed within this time. The default value is 7.

Move Minimum Size

Size of file (in kilobytes). VSM does not move files smaller than this. The default value is 0.

Move Age Weight

Weighting factor that VSM uses for age in the move badness formula. The default value is 1.

Move Size Weight

Weighting factor that VSM uses for size in the move badness formula. The default value is 1.

Move Weight Operator

Arithmetic operator used in the VSM move badness formula (Multiply or Add). The default value is Multiply.

To define a site-specified move badness formula, select Site Defined.

Move Flags

Mark FHDB entries for file copies at the source migration level either Dead, or Active, or Obsolete. The default value is Dead for methods `ct`, `dt`, `mt`, `op`, and `ow`. The default value is Obsolete for method `ad`. Move flags are not applicable to the `ft` and `nb` methods.

Note Figure 56 displays the default move attributes for the `op` method name.

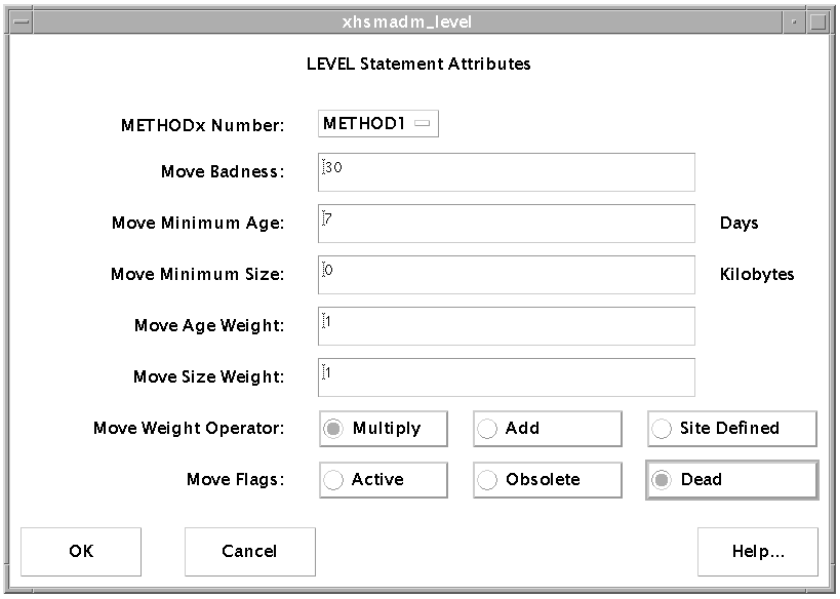
Pull down the Dialog menu of the `xhsmadm` edit screen and select Save configuration file after entering any configuration changes.

Configure and Edit Migration Levels

The following procedure explains how to configure and edit the criteria associated with each migration level for moving migrated files to another migration level.

1. To configure a migration level, pull down the Levels menu on the `xhsmadm` edit screen and select Add level. To delete an existing migration level, click on an entry in the Levels field, pull down the Levels menu, and select Delete level. To edit an existing migration level, double click on an entry in the Levels field, or pull down the Levels menu and select Edit level (see Figure 51). This opens the `xhsmadm` level screen for the database used by that file system (see Figure 57).

Figure 57. xhsmadm_level



2. Configure and edit the criteria associated with this migration level for moving migrated files to another migration level.

Note Move criteria for migration levels, if specified, take precedence over move criteria for method names, if specified (see Figure 32 on page 81).

METHODx Number

The storage method associated with this migration level. (METHOD1 for level 1, METHOD2 for level 2, etc.)

Move Badness

The criteria that VSM uses to move files from one migration level to another after skipping those that are less than the minimum move age and size. VSM computes each migrated file's move badness and selects those whose move badness equals or exceeds the configured value. The VSM move badness formula is as follows:

```
age * move_age_weight (+ or *) size * move_size_weight
```

The default value is 30.

File age in the VSM move badness formula is the time in days since the file was migrated or moved to this level or since the file was last accessed, whichever is most recent.

In lieu of the VSM move badness formula, administrators have the option to define a site-specified move badness formula. (See Move Weight Operator.)

Move Minimum Age

Age of file (in days since migrated or moved to this level, or since accessed). VSM does not move files migrated, moved or accessed within this time. The default value is 7.

Move Minimum Size

Size of file (in kilobytes). VSM does not move files smaller than this. The default value is 0.

Move Age Weight

Weighting factor that VSM uses for age in the move badness formula. The default value is 1.

Move Size Weight

Weighting factor that VSM uses for size in the move badness formula. The default value is 1.

Move Weight Operator

Arithmetic operator used in the VSM move badness formula (Multiply or Add). The default value is Multiply.

To define a site-specified move badness formula, select Site Defined.

Move Flags

Mark FHDB entries for file copies at the source migration level either Dead, or Active, or Obsolete. The default value is Dead for methods `ct`, `dt`, `mt`, `op`, and `ow`. The default value is Obsolete for method `ad`. Move flags are not applicable to the `ft` and `nb` methods.

Figure 57 displays the default move attributes for the `op` method name.

Pull down the Dialog menu of the `xhsmadm` edit screen and select Save configuration file after entering any configuration changes.

Administer Parallel Inode Files

Note Applicable only to kernel-based implementations (Solaris *ufs* file systems).

VSM keeps the status of migrated files in a file called the `.PAIN` (Parallel Inode) file rather than in the inode itself. You created a `.PAIN` file for each HSMDEV entry during the configuration process when you executed the `setuphsm` command (see “Configure Servers” on page 126).

The `setuphsm` command creates a `.PAIN` file in each VSM managed file system before mounting or using the file system under HSM. VSM provides the `pfinit` utility for this purpose which is called by `setuphsm`. The path to the `.PAIN` file is *fspath*`/.PAIN`, where *fspath* is the path to the managed file system as configured with the *fspath* parameter in the `/usr/var/openv/hsm/database/migconfg` file.

At the time of creation, the `.PAIN` file is preallocated and divided into fixed sized entries for each inode in the file system. Each entry in the file is indexed by the inode number and is viewed as an extension to the inode. The `.PAIN` entries show the migration status of each file that VSM migrates from the file system and contain the following information:

- version

  `.PAIN` file version number

- inumber

  Inode number for this entry

- flags

  Migration flags

- slice

  Slice when file was migrated

- size

  Actual size of the file

- mtime

  Modification time of the file

- fhandle

  File handle

The size of each `.PAIN` file entry is 40 bytes. Therefore, the total size of a `.PAIN` file for a file system with *N* inodes would be as follows:

***N* x 40 bytes**

During migration and cache operations, the kernel makes changes to the `.PAIN` file as necessary to record the migration status of files. For example, it sets the migration flags and adds a file handle when it migrates a file.

You must never make any changes to this file outside of those made with `pfinit` (to extend the file) and `pfcheck` (to verify and fix the file). Any differences in `.PAIN` format that may occur between software releases will be accommodated automatically when an existing VSM site upgrades to the new release level.

Creating .PAIN Files

You created a `.PAIN` file for each HSMDEV entry during the configuration process when you executed the `setuphsm` command (see "Configure Servers" on page 126).

Perform the following steps to create a `.PAIN` file manually only if you do not use `setuphsm`:

1. On Solaris, mount the file system you want to manage as a *ufs* file system. For example:

   ```
   mount -F ufs /beta
   ```

2. Execute the `pfinit` command to create a `.PAIN` file for the file system:

   ```
   pfinit -c /beta
   ```

3. Unmount the file system:

   ```
   umount /beta
   ```

4. On Solaris, change the file system type in `/etc/vfstab` to *hsm* by using vi or another editor:

   ```
   vi /etc/vfstab
   ```

   (make necessary changes)

5. Repeat the above for all file systems that you are going to manage with HSM.

6. Remount the file system, but this time mount it as an *hsm* type file system. On Solaris, for example, issue this command:

   ```
   mount -F hsm /beta
   ```

Extending a .PAIN File after Adding Inodes to the File System

You can expand the size of an existing file system by increasing the number of available inodes. If you do add inodes to a managed file system, you must also extend the `.PAIN` file for that file system. The `pfinit -e` option allows you to do this as follows:

1. Unmount the file system.

   ```
   umount /beta
   ```

2. On Solaris, change the file system type in `/etc/vfstab` to *ufs* by using vi or another editor:

   ```
   vi /etc/vfstab
   ```

   (make necessary changes)

3. Remount the file system as a *ufs* file system. For example:

   ```
   mount -F ufs /beta
   ```


4. Execute the `pfinit` command to extend the `.PAIN` file to include entries for the additional inodes:

   ```
   pfinit -e /beta
   ```

5. Unmount the file system.

   ```
   umount /beta
   ```

6. On Solaris, change the file system type in `/etc/vfstab` to *hsm* by using vi or another editor:

   ```
   vi /etc/vfstab
   ```

   (make necessary changes)

7. Remount the file system as an *hsm* type file system.

   On Solaris, for example, issue this command:

   ```
   mount -F hsm /beta
   ```

Checking the .PAIN File

The VSM `pfcheck` command allows you to compare the current file system against the `.PAIN` file and correct any invalid `.PAIN` entries. There are two conditions under which you have to verify the `.PAIN` file:

- If you use `fsck` to fix any file system problems.
- If you do a dump and partial restore of a managed file system. In this case, any migrations or caches that occurred in the interim between the dump and restore can obsolete the `.PAIN` file.

  You do not have to use `pfcheck` if you dump and restore the entire file system.

The file system must not be mounted as an *hsm* file system when you issue a `pfcheck` command.

1. Unmount the file system.

   ```
   umount /beta
   ```

2. On Solaris, change the file system type in `/etc/vfstab` to *ufs* by using `vi` or another editor:

   ```
   vi /etc/vfstab
   ```

   (make necessary changes)

3. Remount the file system as a *ufs* file system. For example:

   ```
   mount -F ufs /beta
   ```

4. Execute the `pfcheck` command to check consistency of the `.PAIN` file:

   ```
   pfcheck /beta
   ```

5. Unmount the file system.

   ```
   umount /beta
   ```

6. On Solaris, change the file system type in `/etc/vfstab` to *hsm* by using vi or another editor:

   ```
   vi /etc/vfstab
   ```

   (make necessary changes)

7. Remount the file system as an *hsm* type file system.

   On Solaris, for example, issue this command:

   ```
   mount -F hsm /beta
   ```

See the man page for more information on how to use the `pfcheck` command.

Standard File System Utilities

Because VSM does not modify inodes, it is possible to use the standard utilities provided with your operating system:

`fsck`

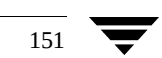
Checks and fixes the file system including any migrated files. Be sure to run `pfcheck` after fixing any file system problems with `fsck`.

As mentioned in the previous topic, always run `pfcheck` to verify and fix the `.PAIN` file after a partial restore of a managed file system.

Listing Files in a Managed Directory

The VSM `fls` command shows whether a file has been migrated, purged, cached, or modified since the last cache. The `fls` command is the VSM version of `ls`.

`fls` displays an `m` in the left column of the mode bit field as well as the machine ID and file handle on the right for all migrated files. If a migrated file is purged from premigration, a *t* also appears in the mode bit field. `fls` removes the `m` if a migrated file is cached back to disk, and removes the *t*, machine ID, and file handle if a cached file is modified. See the `fls(1)` man page or the *Storage Migrator User's Guide* for a more detailed explanation.



Administer Inode-to-Handle Files

Note Applicable only to non-kernel-based implementations.

VSM keeps the state of migrated files in a file called the `.IHAND` (Inode-to-Handle) file. The `.IHAND` file is created by `migd` if the file does not exist. The `.IHAND` file is a sparse file that grows as new inodes are assigned to migrated files.

The full path of the `.IHAND` file is *dwpath*`/database/`*hsmname*`.IHAND`. Each entry being a fixed size. The file is indexed by inode number. Entries show the migration state of each file VSM migrates from the file system and contain the following information:

- machid

  Machine ID

- handle

  VSM file handle assigned

- flags

  State flags (in hexadecimal)

  ```
  01  reloading
  02  removing
  04  partial cache
  08  cached, unmodified
  ```

- slice

  Size of slice in bytes at the time of migration. For DMAPI implementations, the size of the configured slice is replaced by the size of the effective slice during partial file caching.

- DM_handle

  DMAPI handle

- DM_length

  Total size of the DMAPI handle in bytes

- highest_read

  Offset plus length in bytes from the `read` event during partial file caching

The size of each entry is 48 bytes for DMAPI implementations, so the total size of the `.IHAND` file becomes the product of the entry size and the number of inodes in the file system.

You must never make any changes to this file outside of those made with ihprint (to fix the file) and rebuild_ihand (to recover the file). Any differences in .IHAND format that may occur between software releases will be accommodated automatically when an existing VSM site upgrades to the new release level.

You can expand the size of an existing file system by increasing the number of available inodes. If you do add inodes to a managed file system, the .IHAND file will grow as needed to accommodate the larger file system.

Checking or Correcting the .IHAND File

Caution This file is used by HSM, and changing it incorrectly with the ihprint command or any other way can hang the system or produce inconsistent results.

The VSM command ihprint will allow you to display and alter entries in the .IHAND file. Only use ihprint to fix the file if you are certain the .IHAND entry is incorrect.

Recovering the .IHAND File

The .IHAND file must not be backed up with the FHDB and the VOLDB. If you are restoring the FHDB from a previous backup, you must recreate the .IHAND file using the rebuild_ihand command based on information in the FHDB and the file system.

Typical error messages indicating that rebuild_ihand needs to be run:

```
05/15 22:17:45 migd[343]: ERROR: Handle mis-match for inode 1440
05/15 22:17:45 migd[343]: ERROR: could not convert DM to mig
```

See the man page for more information on how to use the rebuild_ihand command.

Update fstab or vfstab File

Note Applicable only to kernel-based implementations (Solaris *ufs* file systems).

Update the fstab file (vfstab file on Solaris) to include all VSM managed file systems. This ensures that the system mounts these file systems at start up. Otherwise, you must mount them individually or with a special script.

The vfstab entries for the example server (Solaris) are:

```
/dev/sd1g    /home1      hsm     rw  1 1
/dev/sd2g    /home2      hsm     rw  1 1
/dev/sd4g    /dbrecs     hsm     rw  1 1
/dev/sd3a    /sd3        hsm     rw  1 1
```


Update Scripts and Create crontab Entries

Prior to starting HSM, update your system's startup files to automate startup. Also create other scripts and `crontab` entries to ease the task of administrating HSM.

The following provides guidelines.

Startup Script

For kernel-based implementations (Solaris *ufs* file systems), if you want to automatically `fsck` and mount a VSM managed file system at boot time, you can use the following procedure and add appropriate `startmigd` and `stopmigd` commands:

```
cp /usr/openv/hsm/bin/goodies/S73HSM.mount
/etc/rc2.d/S73HSM.mount
```

The `setuphsm` command used later assumes this has been done, and changes `/etc/vfstab` appropriately.

Note If the sequence number 73, which is part of the name of this startup script, is already used, you can use another number as necessary.

Note `S73HSM.mount` starts the Media Manager daemon, `ltid`, if it is installed on the system and you are using tape and or optical methods for HSM.

For non-kernel-based implementations on Solaris using the DMAPI interface, the following suggest startup script will install the VSM attribute driver, start the daemon, and mount the *vxfs* file system:

```
cp /usr/openv/hsm/bin/goodies/S78hsmveritas
/etc/rc2.d/S78hsmveritas
```

Note If the sequence number 78, which is part of the name of this startup script, is already used, you may use another number as necessary.

Note `S78hsmveritas` starts the Media Manager daemon, `ltid`, if it is installed on the system and you are using tape and or optical methods for HSM.

The following suggested startup script for IRIX will start the daemon and mount the *xfs* file system:

```
/usr/openv/hsm/bin/goodies/irixrc.sh
```

To use this script, copy it to the `/etc` directory as follows.

```
cp /usr/openv/hsm/bin/goodies/irixrc.sh
/etc/rc2.d/irixrc.sh
```

On HP-UX, you can use the following script:

```
/usr/openv/hsm/bin/goodies/hpuxrc.sh
```

Descriptions of these latter two files can be found in:

```
/usr/openv/hsm/bin/goodies/README
```

Note These unmodified startup scripts do not automatically start the `migrd` daemon. Each VSM server host which is to accept the VSM-Java GUI must be executing the `migrd` daemon as described in "Install VSM Software" on page 180.

Backup and Migrate Script

You can provide automatic control of backups and migrations by creating a script with the necessary commands and then executing the script with `cron`.

Before you attempt your first file migration, make sure you have backed up the managed file systems and their associated databases using VERITAS NetBackup. See "Backing Up VSM Databases and Managed File Systems" on page 260 for information on how to do this.

Note Once you have completed your initial full backup, establish a schedule for processing migrations *before* backups in order to shorten the backup cycle.

Migrate files by executing `migbatch` on managed file systems. Depending on the parameters that you use in this script, you can set up separate jobs for each VSM configuration or file system. See the `migbatch(1M)` man page for more information.

After creating the script, you can use a `crontab` entry to execute it. See "Scheduling Migrations" on page 107 for advice on when to perform backups and migrations.

Media Usage Script

Another task to automate is a periodic check for available space on the migration media. The `miggetvol` command provides the type of report you need. You can use the information from this report to determine when to add new tapes to the library and also when to consolidate tapes. The following is an example entry that provides a `miggetvol` report for `hsm1` on at the example site:

```
* 5 * * * /usr/openv/hsm/bin/miggetvol hsm1
```

By adding the above entry to a file and then executing `crontab` on that file, the administrator can execute `miggetvol` each morning at 5 am. By varying the parameters, you can create entries to provide reports based on specific HSMs or storage methods. See the `miggetvol(1M)` man page for more information.

Link Global Log to Non-tmp File

The VSM global log file contains messages that pertain to all VSM configurations on the server. This log is named `/tmp/hsm.log`.

You cannot reconfigure the path or log name. However, to ensure that you do not lose log messages after a system reboot or crash, link this log to a file that is in a directory other than `/tmp`.

The example site links the global log to a file in the `/var/logs` directory as follows:

```
ln -s /var/logs/hsm.log /tmp/hsm.log
```

See “Checking and Managing the Logs” on page 307 for information on using and managing this log file.

Control Migration of Specific Files

You can control the migration of specific files by including them in one of two global control files:

`/usr/var/openv/hsm/database/migrate`

The global migrate file, containing a list of the files or directories of files that VSM will migrate each time automatic migration occurs.

`/usr/var/openv/hsm/database/migstop`

The global stop file, containing a list of the files that VSM will not migrate.

These files apply to all VSM managed file systems. The specifications only affect files that reside in a VSM managed directory. See “Global Migration Control” on page 290 for more information on how to create and use these global VSM control files. The relationship of global control files to local control files and the circumstances under which control files are overridden are explained in the *Storage Migrator User’s Guide*.

Enable User Permissions

The administrator must set the appropriate permissions before users can use certain commands and files.

Set permissions, if desired, for the `migcat`, `migrate`, `migpurge`, `migtie`, and `miggrouup` user commands.

- The `migrate` command allows users to force migrate specific files. Enter this command to give users permission to execute `migrate`:

  ```
  chmod 4511 /usr/openv/hsm/bin/cmd/migrate
  ```

- The `migpurge` command allows users to force purge specific files. Enter this command to give users permission to execute `migpurge`:

  `chmod 4511 /usr/openv/hsm/bin/cmd/migpurge`

  On Solaris platforms the administrator must also enter the following command to allow users to purge their own files:

  `chmod 4511 /usr/openv/hsm/bin/admincmd/migmkspace`

- The `migtie` command allows users to cache groups of related files together. Enter this command to give users permission to execute `migtie`:

  `chmod 4511 /usr/openv/hsm/bin/migtie`

- The `miggroup` command allows users to premigrate files in a grouped directory together. Enter this command to give users permission to execute `miggroup`:

  `chmod 4511 /usr/openv/hsm/bin/miggroup`

Set permissions, if desired, for the `migtarhelp` command.

- The `migtarhelp` command makes it possible to restore a migrated and purged file using VERITAS NetBackup.

Note This restores only the inode information about the file. The slice is restored after the file is cached and remigrated.

Enter this command to give users permission to execute `migtarhelp`:

`chmod 4511 /usr/openv/hsm/bin/admincmd/migtarhelp`

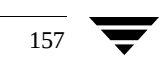
`chmod 4511 /usr/openv/hsm/bin/migsetdb`

Register Media with VSM

Note It is not necessary to explicitly register tape and optical disc volumes. If VSM runs out of registered volumes, it will automatically select additional new volumes from the designated volume pool, which defaults to the `HSM` volume pool, or from the scratch pool. See “Keeping a Supply of Unused Volumes” on page 273 for more information.

The media registration process varies depending on the type of media you are using. The following topics explain how to register media for any of the methods VSM supports for secondary storage. There is also a topic on using volume set 0. For complex configurations, you will have to use more than one of the methods explained here.

Note VSM adds media to its VSM volume database (VOLDB) in the order that you label them. It also requests them in that order.



In addition, when you label media remember that a label must be unique not only to the database in which you register it, but also among all of the databases in the VSM managed file system. Although you cannot use the same label to register media in more than one database, you can use the same method and volume set.

For example:

- You *can* have `hs0001.ct.1` in one database and `hs0002.ct.1` in another because the labels are different.
- You *cannot* have `hs0001.ct.1` in one database and `hs0001.ct.2` or `hs0001.mt.1` in another because the labels are the same. This would result in VSM requesting the same tape from Media Manager in all three cases.

You can register media either with `xhsmadm`, or by using the `migreg` command. Using `xhsmadm`, you can only register media in the default volume pool, `HSM`. See the `xhsmadm` man page for more information on the `migreg` command. The sections that follow describe how to register media with `xhsmadm`.

Prior to using this command you must perform the following steps (see "Configure Tape and Optical Storage Devices" on page 122).

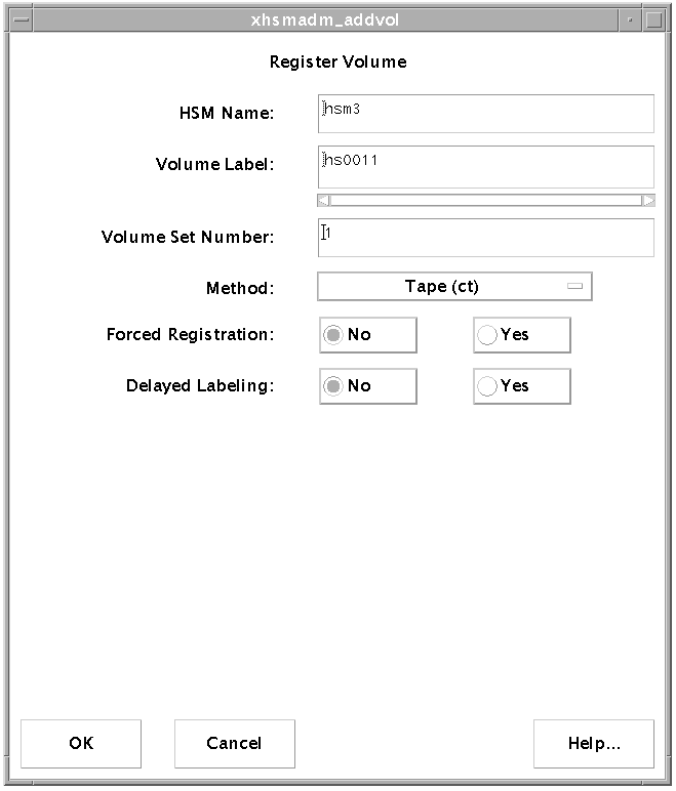
Note If registering only `ad` or `ft` volumes, perform only step 3.

1. Configure an `HSM` Volume Pool.
2. Configure the media within Media Manager, and place them in the `HSM`. volume pool.
3. Complete VSM installation and configuration.
4. Start the device manager daemon, `ltid`.
5. When those steps are complete you can label and register the media as in the following examples.

Registering Tape Media

1. From the main `xhsmadm` screen, select a file system that uses the database pathname you want to configure, pull down the Volumes menu and select Volume registration and reports. This opens the `xhsmadm` volume registration screen used by that file system. Pull down the Edit menu and select Add. This opens the `xhsmadm` addvol screen (see Figure 58).

Figure 58. xhsmadm_addvol for Tapes



2. Change the volume registration attributes to the desired values. The attributes in each entry are defined as follows:

   HSM Name

   The name of the selected file system.

Volume Label

The unique name of the volume to be recorded on the volume and in the VSM volume database VOLDB. VSM restricts volume names to an alphabetic character followed by up to five alphanumeric characters (for a maximum of six total characters), and converts all lower case input to upper case.

Volume Set Number

The integer number of the volume set of which this volume is a part.

Method

The method name under which this volume will be recorded. Valid values are `ct`, `dt` and `mt` for tape media. These must be defined in the *dwpath*`/database/migconf` configuration file for *hsmname* (see Methods in "Assign Default Values" on page 127).

Forced Registration

Select Yes to force the registration of a previously labeled volume. The volume cannot currently be registered in any VSM volume database. Select No to not force registration.

Delayed Labeling

Select Yes to delay labeling of media until needed. Select No to label the media immediately.

The tapes in this example are for use with METHOD1 and METHOD2 for the `/usr/var/openv/hsm3` database on `bunny` at the example site described in Chapter 2 (see Figure 47 on page 117).

```
METHOD1 = "ct.1.library"/"ct.2.library"

METHOD2 = "ct.3.vault"/"ct.4.vault"
```

These storage methods use the example site's two 8 mm tape libraries.

The first two tapes are for `ct` volume sets 1 and 2. The second two tapes are for `ct` volume sets 3 and 4. Each has a unique volume name. There is no forced registration of previously labeled volumes, nor is labeling delayed.

The four tapes registered in this example are as follows:

```
hsm3 hs0011 ct 1

hsm3 hs0012 ct 2

hsm3 hs0013 ct 3

hsm3 hs0014 ct 4
```

Only the first of these four volumes is shown in Figure 58.

Use the same approach to register more tapes.

Because there is both a METHOD1 and METHOD2 in this example, VSM writes two copies of every file that it migrates for this database. VSM also uses concurrent recording when making its copies. For example, when writing the first copy (METHOD1), it writes one file to a volume set 1 tape and the next to a volume set 2 tape.

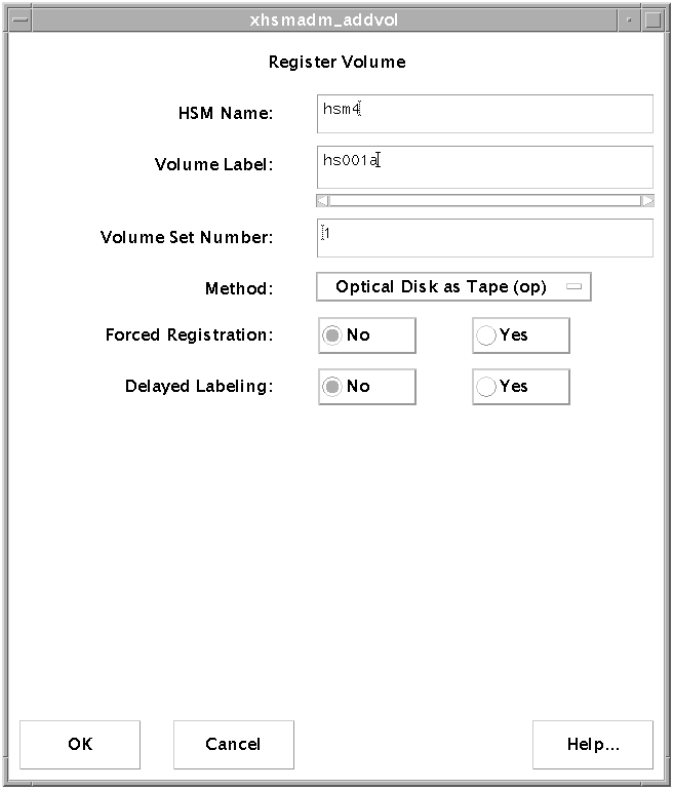
Using `xhsmadm`, you can only register media in the default volume pool, `HSM`. You must use the `migreg` command to specify a different volume pool. (see the `migreg(1M)` man page).

Registering Optical Disc Media

Note On all platforms, you must use `tpformat` to format optical disc media unless the disks are preformatted. On Solaris and IRIX platforms, you must also use `tpformat` to label optical discs before they can be registered to HSM. Labeling is recommended but not mandatory on other platforms. Use the Media Manager GUI `xvmadm` to write a volume label and RVSN (recorded volume serial number) on the optical disc. See the Media Manager `tpformat(1M)` man page for more information, including server configuration restrictions.

Registering optical disc media using method name `op` or `ow` is very similar to the process described above for tape media (see Figure 59).

Figure 59. xhsmadm_addvol for Optical Discs



Change the volume registration attributes to the desired values. Set the Method name to `op` or `ow`. This must be defined in the *dwpath*`/database/migconf` configuration file for *hsmname* (see Methods in "Assign Default Values" on page 127; see "Registering Tape Media" on page 159 for other attribute definitions).

The disks are for use with METHOD1 and METHOD2 for the `/usr/var/openv/hsm4` database on `bunny` at the example site described in Chapter 2 (See Figure 49 on page 119).

```
METHOD1 = "op.1.library"/"op.2.library"
METHOD2 = "op.3.vault"/"op.4.vault"
```

These storage methods use the example site's single optical-disk library with two drives.

Because there is both a METHOD1 and METHOD2, VSM writes two copies of every file that it migrates for this database.

Note When you register one side of an optical disc with `migreg`, VSM also automatically registers the other side with the same volume set number to prevent both copies from going to the same disc.

The first two discs are for op volume sets 1 and 2. The second two discs are for op volume sets 3 and 4. Each side of the discs has a unique volume name. There is no forced registration of previously labeled volumes, nor is labeling delayed.

The four optical discs registered in this example are as follows:

```
hsm4 hs001a op 1
hsm4 hs001b op 1
hsm4 hs002a op 2
hsm4 hs002b op 2
hsm4 hs003a op 3
hsm4 hs003b op 3
hsm4 hs004a op 4
hsm4 hs004b op 4
```

Only the first of these eight volumes is shown in Figure 59.

For METHOD1 and METHOD2 in the above example:

- hs001a and hs001b are the two sides of one disc and belong to volume set 1. hs002a and hs002b are the two sides of another disc and belong to volume set 2.
- hs003a and hs003b are the two sides of one disc and belong to volume set 3. hs004a and hs004b are the two sides of another disc and belong to volume set 4.

When VSM writes to these volumes, the first copy goes to discs from volume sets 1 and 2. The second copy to discs from volume sets 3 and 4.

Note Using xhsmadm, you can only register media in the default volume pool, HSM.

Registering Alternate Magnetic Disks

VSM supports the ad (alternate magnetic disk) storage method name for migrating files between two UNIX file systems. If file migration is local, both file systems are mounted on the same managed server. If file migration is remote, the file system to which you are migrating files is NFS mounted from a remote volume server. In this case you must mount the NFS file system before registering it in the Volume database. The managed server becomes a client to the remote volume server.

Registering an alternate magnetic disk using method name ad is very similar to the process described above for tape or optical media, except that there is no Delayed Labeling attribute (see Figure 60).

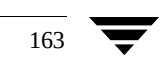
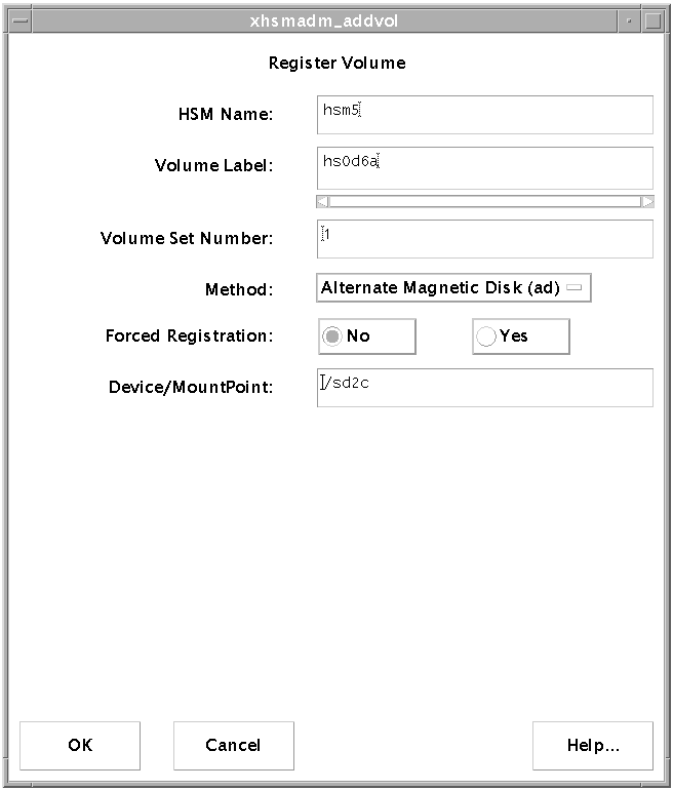


Figure 60. xhsmadm_addvol for Alternate Magnetic Disks



Change the volume registration attributes to the desired values. Set the Method name to `ad`. This must be defined in the *dwpath*`/database/migconf` configuration file for *hsmname* (see Methods in "Assign Default Values" on page 127). The attributes in each entry are defined either in "Registering Tape Media" on page 159 or as follows:

Device Name/Mount Point

The device name or the filesystem mount point required when registering a volume for use with method name `ad`. Make sure the specified device name or filesystem is mounted before registering the volume.

Caution Do not register a disk partition for a directory below the mount point because the entire capacity of the file system at the mount point is assumed.

In the following example, assume that `bunny` (the example managed server) and `mercury` at the example site each have an extra disk partition and those extra partitions are being registered for use with HSM. Because VSM does not use Media Manager for `ad` volumes, the only prerequisite is to install and configure HSM.

This disk partition is for ad volume set 1. There is no forced registration of previously labeled volumes.

The disk partition on bunny shown in Figure 60, is registered as follows:

```
hsm5 hs0d6a ad 1
```

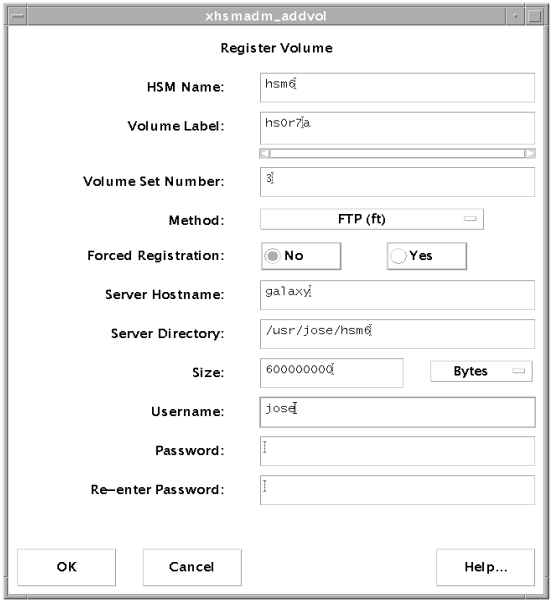
Because the disk on mercury is not directly connected to bunny, you must first NFS mount that file system on bunny. When the file system is mounted, you can use xhsmadm to register it in the same manner as shown above. During migrations, VSM uses NFS to send files to the remote file system. The file system must remain mounted during migrations and caches.

Registering Volumes for ft Remote Method

VSM supports the ft storage method name for migrating files to file systems located on a remote volume server. In this configuration, the managed server becomes a client to the remote volume server. A major difference between the ft and tape or optical disc method names is that VSM transfers whole files with ft without breaking them into granules.

1. From the main xhsmadm screen, select a file system that uses the database pathname you want to configure, pull down the Volumes menu and select Volume registration and reports. This opens the xhsmadm volume registration screen used by that file system. Pull down the Edit menu and select Add. This opens the xhsmadm addvol screen (see Figure 61).

Figure 61. xhsmadm_addvol for ft Remote Method



2. Change the volume registration attributes to the desired values. Set the Method name to `ft`. This must be defined in the *dwpath*`/database/migconf` configuration file for *hsmname* (see Methods in "Assign Default Values" on page 127). The attributes in each entry are defined either in "Registering Tape Media" on page 159 or as follows:

   Server Hostname

   The name of the remote volume server. This can be the internet id or number of the server. VSM uses this name on the ftp `open` command as the host parameter. It must be a valid `ftp` host.

   Server Directory

   The full pathname of the file system directory or subdirectory on the remote volume server. The VSM username on the managed server must have read and write permissions to this remote directory. This can be any directory on the remote volume server that is not already registered for HSM.

   Size

   The capacity in bytes of the remote file system available for storing migrated files. This value can either be all or just part of the total capacity of the remote file system. For example, you can choose to use only 600 megabytes of a file system that has a total capacity of 1.2 gigabytes. It is important to remember that VSM does not migrate data beyond the space that you allocate. A value of 0 is interpreted as unlimited storage capacity.

Note If VSM transfers files greater than 2 gigabytes with the `ft` method, the remote volume server must be configured to accept and process files of this size.

   Username and Password

   The username and password VSM uses when accessing the remote volume server from the managed server through `ftp`. This name and password must be valid on the remote volume server and also must have read and write access to the remote file system that you are using for migration.

   Keystrokes into the password field are echoed as asterisks. Since the password cannot be seen, it must be entered twice.

Note To change a user name and password, see "Changing Usernames and Passwords for ft Remote Method" on page 167.

3. Ensure that you store the data from each managed file system on the managed server in a separate directory within the remote file system. For example, data from managed file systems `A` and `B` can be stored in directories `RA` and `RB` within remote file system `FTA`. You can also use a separate remote file system for each of the managed file systems. In addition, you can use concurrent recording to send the files from one managed file system to multiple remote file system directories.

In the following example, assume that bunny (the example managed server) is connected to a remote volume server named galaxy. The file system directory on galaxy being registered for use with VSM is /usr/jose/hsm6. Because VSM does not use Media Manager for ft volumes, the only prerequisite is to install and configure HSM.

This file system directory is for ft volume set 3. There is no forced registration of previously labeled volumes. The aggregate size of the files migrated to this file system cannot exceed 600 megabytes.

The file system directory on galaxy shown in Figure 61, is registered as follows:

```
hsm6 hs0r7a ft 3
```

Upon successful registration, VSM creates and stores a file named ID_LABEL on the remote volume server. This file contains the name of the client and the volume label in a single line of text. When the remote file system is *mounted* for migration and cache operations, VSM uses the ID_LABEL file to verify that the remote file system is actually registered for use by the client that has it mounted.

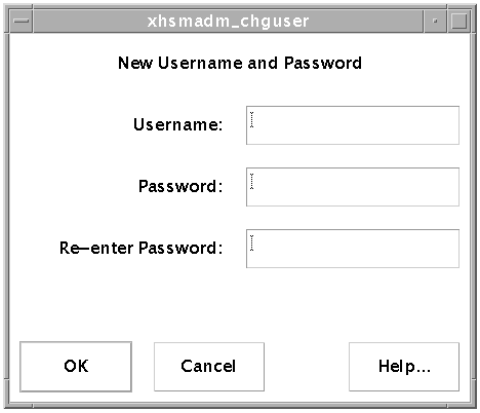
VSM also uses the ID_LABEL file to prevent registration of remote file systems with multiple clients. A remote file system must be registered with one and only one client. Clients attempt to validate that this condition exists by using ftp commands. However, the administrators of all systems must also cooperate to ensure multiple registration does not occur

After you successfully register the volume, VSM can use the ft method to migrate and cache files from the file systems that use this method and volume set. The remote volume server must be up and running during VSM operation. If the remote volume server is down, ftp requests timeout, causing unnecessary delays and errors. The administrators of all systems must cooperate to minimize the possibility of these problems occurring.

Changing Usernames and Passwords for ft Remote Method

From the main xhsmadm screen, select a file system that uses the database pathname (*dwpath*) you want to configure, pull down the Volumes menu and select Volume registration and reports. This opens the xhsmadm volume registration screen used by that file system. If you want to change all registered ft volumes, pull down the Edit menu and select Change user/password for all volumes. If you want to change only one volume, select it on the xhsmadm_volreg, pull down the Edit menu and select Change user/password for selected volumes. Either selection opens the xhsmadm change user screen (see Figure 62).

Figure 62. xhsmadm_chguser



Enter the username and new password. Reenter the password a second time in the designated field.

Registering Volumes for nb NetBackup Method

VSM supports the `nb` method name for using VERITAS NetBackup to make copies of migrated files. A major difference between the `nb` and tape or optical disc method names is that VSM transfers whole files with `nb` without breaking them into granules.

You do not register individual media as volumes to VSM for the `nb` method. Instead, you first define one or more NetBackup classes and then register each class as a volume. NetBackup selects the storage media from the volume pool associated with the NetBackup class.

Caution Configure these NetBackup classes with an infinite retention level so migrated images will not expire (see “Retention Level: (9) Infinity” on page 173).

Migrating files with the `nb` method causes a user backup of the files and their associated granule header files to a previously defined NetBackup class.

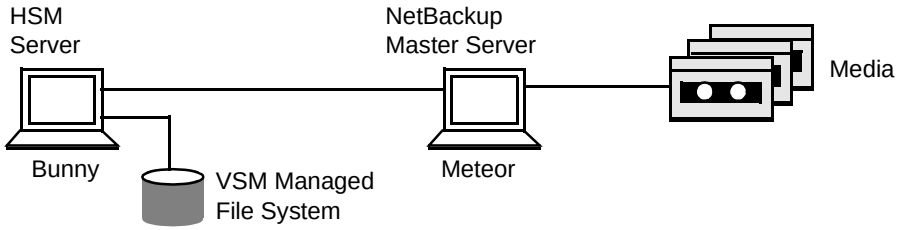
To Define a NetBackup Class

Note For a more detailed description of how to configure NetBackup classes, see chapter 4 of the *VERITAS NetBackup System Administrator’s Guide*.

In the following example, assume that bunny, the VSM managed server, is connected to a NetBackup master server named meteor. The NetBackup class on meteor being registered for use with VSM is class nbu009hsm. Because VSM does not use Media Manager to manage nb volumes, it is not necessary to install and configure Media manager for the nb method.

Note For more information on NetBackup class examples, see /usr/openv/hsm/bin/goodies/cltemplates.sh.

Figure 63. Example Configuration for nb Method



Notes: Bunny is a media server of the NetBackup master server meteor.
Bunny is a client of meteor.

1. The VSM server must be a NetBackup server. Change the /usr/openv/netbackup/bp.conf file on the VSM server, bunny, so the first line reads as follows:

   SERVER = *machine name of NetBackup master server*

   In this example, bunny is a media server of the NetBackup master server meteor, so the bp.conf file on bunny could read as follows:

   ```
   SERVER = meteor.min.ov.com
   ```

2. The VSM server must be a client of the NetBackup master server. Change the /usr/openv/netbackup/bp.conf file on the VSM server bunny to include the following line:

   CLIENT_NAME = *machine name of VSM server*

   In this example, bunny is a client of meteor, so the bp.conf file on bunny could read as follows:

   ```
   CLIENT_NAME = bunny
   ```

Note It is possible to configure the NetBackup master server, the VSM server, and the NetBackup client all on the same platform.

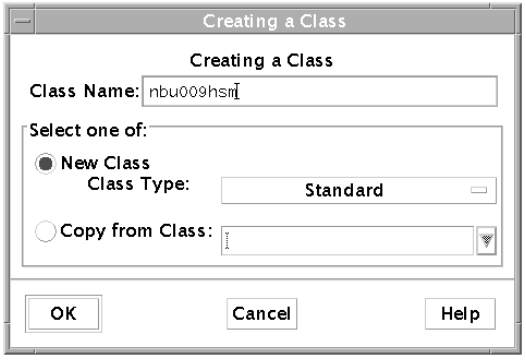
3. Set your DISPLAY variable, and bring up the NetBackup GUI on the NetBackup master server.

   ```
   /usr/openv/netbackup/bin/xbpadm
   ```

4. In the NetBackup Administration window, select a class from the Classes list and then click New class on the Actions menu. (If no items happen to be highlighted in any lists, click New on the Actions menu and then Class on the resulting submenu.)

   The dialog box on Figure 64 appears.

Figure 64. Creating a New Class



5. Enter a unique name in the Class Name box.

   Use numeric, alphabetic, plus, minus, underscore, and period characters in the name. Do not leave any spaces between characters.

   In this example, the NetBackup class on `meteor` being created for use with VSM is `nbu009hsm`.

6. Choose whether you want to start with standard class defaults or with the same properties as an existing class.

   - To use the standard defaults, leave the Copy from Class box blank.
   - To start with the same properties as an existing class, type in the name of the class you are copying or use the arrow to drop down a selection list.

     Using an existing class is convenient if it has many of the properties you need. If you copy a class, NetBackup duplicates the following:

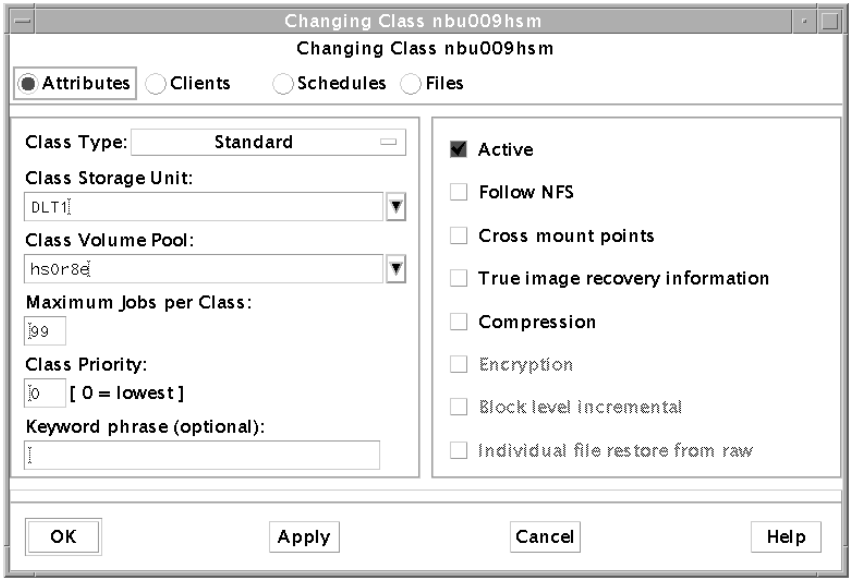
     - Attributes
     - Clients
     - Schedules
     - Files

     You can then change any items desired.

7. Click OK to confirm your choices.

This adds the class to the configuration and opens a Changing Class dialog box where you can change the class properties from their initial settings. The name of the new class appears at the top of the dialog as in Figure 65.

Figure 65. Class Attributes



When the dialog box opens, the middle section shows the Attributes. You can change it to show Clients, Schedules, or Files, by clicking the respective button. Move between these views as necessary to change the class properties.

8. Configure class Attributes as follows:

Class Type: Standard (default)

Class Storage Unit: Choose a specific storage unit, (*not* the default, Any available).

Class Volume Pool: Enter a unique name or select an existing pool. Do not enter a volume pool used by VSM for backing up files. In this example, the name of the volume pool is `hs0r8a`.

If you define more than one NetBackup class, assign a different volume pool to each class. This prevents VSM from possibly migrating both copies of a file to the same physical media. Assign at least one piece of media to the volume pool.

Note To maintain performance, it is better to define a larger number of classes, each with a smaller volume pool, than it is to define a smaller number of classes, each with a larger volume pool.

Maximum jobs per Class: 99 (default)

Class Priority: 0 (default)

Active? Yes (default)

Follow NFS? No (default)

Cross mount points? No (default)

Collect true image recovery information? No (default)

Compression? No (default)

9. Click OK to confirm your choices for the class attributes you select.

10. Click Clients to add a client to this class.

Click New to create a new client.

Specify the hardware and operating system of the VSM server. Since the VSM server must be either a NetBackup master or media server, do not install client software. Enter the name of the VSM server as the client name.

In this example, the client name is `bunny`.

Click OK to confirm your choices.

11. Click Schedules to add a schedule to this class.

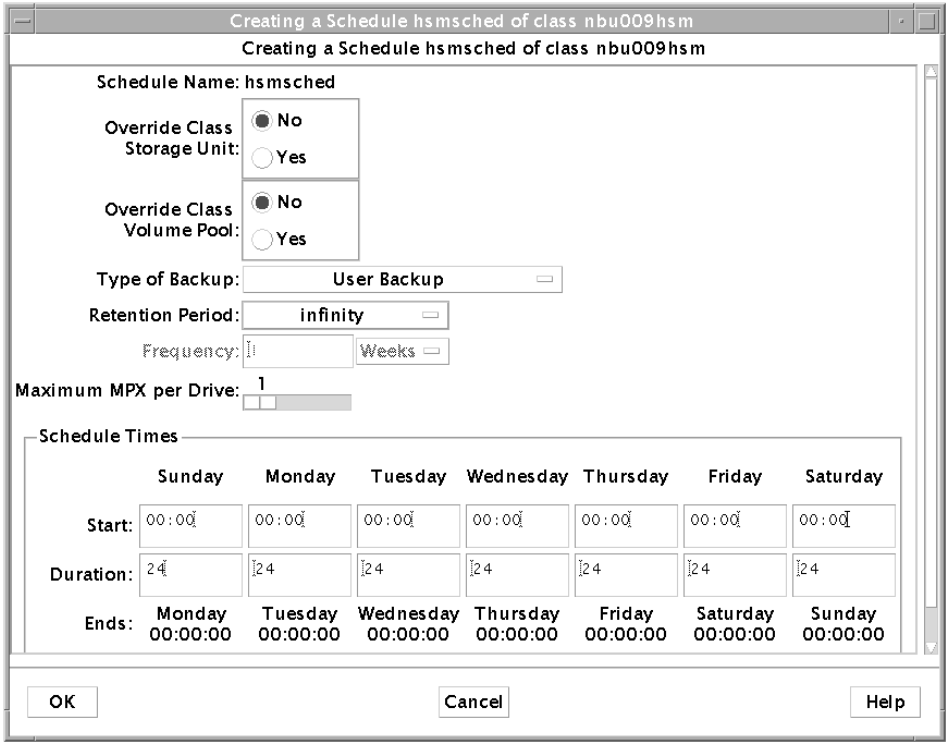
Click New to create a new schedule.

Name of Schedule: Enter a unique name.

In this example, the schedule name is `hsmsched`.

Click OK to open the schedule dialog box. The name of the new schedule appears at the top of the dialog as shown in Figure 66.

Figure 66. New Schedule Dialog Box



Override Class Storage Unit: No (default)

Override Class Volume Pool: No (default)

Type of Backup: User Backup

Note User Backup is the only schedule type supported for this purpose.

Caution Configure these NetBackup classes with an infinite retention level so migrated images will not expire.

Retention Level: (9) Infinity

Maximum MPX per Drive: 1 (default)

Schedule Times: The start time is 00:00 (hh:mm) and duration is 24 hours for each day in the week. Key in in Start and Duration for Sunday, and then click the Duplicate button to complete the remaining six days of the week.

Click OK to confirm your choices.

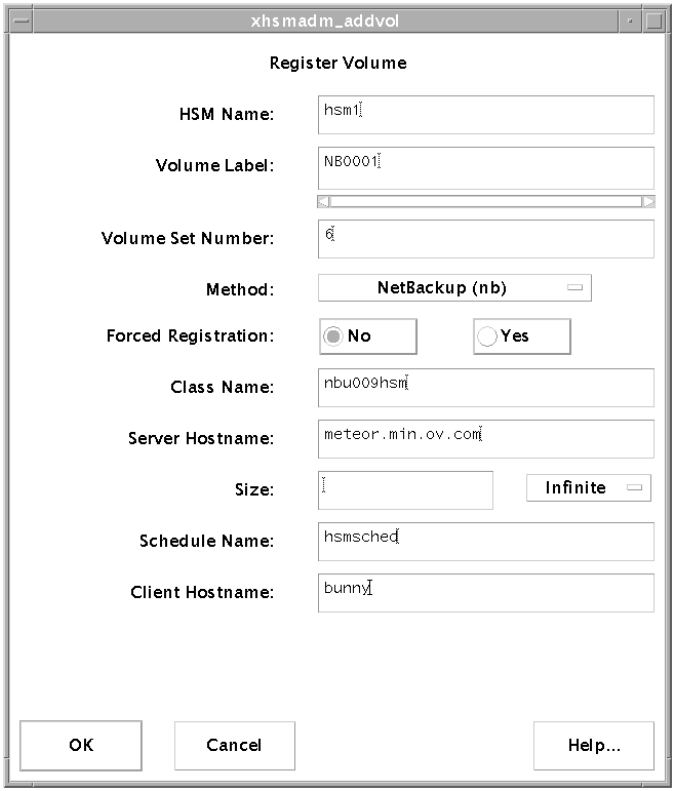
12. Click Files and make sure the file list is empty.

Click OK to confirm your choices.

To Register Volumes for nb NetBackup Method

1. From the main `xhsmadm` screen, select a file system that uses the database pathname (*dwpath*) you want to configure, pull down the Volumes menu and select Volume registration and reports. This opens the `xhsmadm` volume registration screen used by that file system. Pull down the Edit menu and select Add. This opens the `xhsmadm` addvol screen (see Figure 67).

Figure 67. xhsmadm_addvol for nb NetBackup Method



2. Change the volume registration attributes to the desired values. Set the Method name to `nb`. This must be defined in the *dwpath*`/database/migconf` configuration file for *hsmname* (see Methods in "Assign Default Values" on page 127). The attributes in each entry are defined either in "Registering Tape Media" on page 159 or as follows:

Volume Label

The unique name of the volume to be recorded on the volume and in the VSM volume database VOLDB. VSM restricts volume names to an alphabetic character followed by up to five alphanumeric characters, and converts all lower case input to upper case. (The volume label and the volume pool shown in Figure 65 on page 171 may but need not be the same.)

In this example, the volume label for volume pool `hsOr8a` is `NB0001`.

Class Name

The name of the NetBackup class to be registered as an `nb` volume (see "To Define a NetBackup Class", step 5, on page 170).

Note In this example, the class name is `nbu0009hsm`.

Server Hostname

The name of the NetBackup master server.

In this example, the NetBackup master server is `meteor.min.ov.com`.

Size

The capacity in bytes of the NetBackup system available for storing migrated files. This value can be either all or just part of the total capacity. It is important to remember that VSM does not migrate data beyond the space that you allocate. A value of 0 is interpreted as unlimited storage capacity.

Schedule Name

The name of the schedule defined for the NetBackup class. The backup window for this schedule must be 24 hours per day, seven days per week (see "To Define a NetBackup Class", step 11, on page 172).

Note In this example, the schedule name is `hsmsched`.

Client Hostname

The name of the NetBackup client for the NetBackup class (see "To Define a NetBackup Class", step 10, on page 1720.

Note In this example, the client hostname is `bunny`.

Make sure that the NetBackup class is defined and registered as an `nb` volume to be used exclusively for migration purposes. If you define more than one NetBackup class, assign a different volume pool to each class. This prevents VSM from possibly migrating both copies of a file to the same physical media.

This example shows `nb` volume set 6. The aggregate size of the files migrated to this volume is unlimited. The NetBackup volume shown in Figure 67, is registered as follows:

```
hsm1 NB0001 nb 6
```

After you successfully register the volume, VSM can use the `nb` method to migrate and cache files from the file systems that use this method and volume set. Because the NetBackup server must be up and running during VSM operation, the backup window for the class schedule is defined to be 24 hours per day, seven days per week. If the NetBackup server is down, backup requests timeout, causing unnecessary delays and errors. The administrators of all systems must cooperate to minimize the possibility of these problems occurring.

Registering Extra Volumes

You can use a volume set 0 to create extra volumes for any of the VSM storage methods. The following example registers three tapes to `ct` volume set 0:

For example, you can use `xhsmadm` to register three tapes to `ct` volume set 0 as follows:

```
hsm3 hs0015 ct 0
hsm3 hs0016 ct 0
hsm3 hs0017 ct 0
```

VSM assigns these tapes to other `ct` volume sets as they need tapes. For example:

- If VSM needs another tape for a write to `ct.1`, it reregisters `hs0015 ct.0` as `hs0015 ct.1`.
- If another volume is then required for `ct.2`, VSM reregisters `hs0016 ct.0` as `hs0016 ct.2`.

Likewise, you can register two optical discs to `op` volume set 0:

```
hsm4 hs018a op 0
hsm4 hs018b op 0
hsm4 hs019a op 0
hsm4 hs019b op 0
```

Here `hs018a` and `hs018b` are the two sides of one disc, and `hs019a` and `hs019b` are the two sides of the second disc.

VSM assigns these optical discs to other op volume sets as they need additional discs. For example:

- If VSM needs another optical disc for a write to `op.1`, it reregisters `hs018a op.0` as `hs018a op.1`, and `hs018b op.0` as `hs018b op.1`. (Note that both sides of the disc are assigned to the same volume set.)
- If another volume is then required for `op.2`, VSM reregisters `hs019a op.0` as `hs019a op.2`, and `hs019b op.0` as `hs019b op.2`.

Initial Startup and Testing

When configuration is complete, you can start VSM and test it as follows:

1. Reboot the system to test both VSM and your script changes. Make sure the device manager daemon, `ltid`, is running if you are migrating to tape or optical media.
2. Label and register enough secondary storage media to test the configuration (see "Register Media with VSM" on page 157 for instructions).
3. Change your working directory to one that resides in a managed file system. (See "File Systems" on page 128.)

   cd *fspath*/dirm
4. Create a `test` directory, and change to that directory:

   mkdir test

   cd test
5. Write a test file such as the ASCII release notes to the `test` directory:

   cp /usr/openv/hsm/release_notes testfile

   Determine if `testfile` is greater than the configured slice, and make it so if it is not.
6. Premigrate `testfile`:

   /usr/openv/hsm/bin/migrate testfile

   This premigrates the file.
7. Copy the file to secondary storage:

   /usr/openv/hsm/bin/migrc -R *hsmname*

   `migrc` interprets the files in premigration to be part of an uncompleted migration operation, and finishes migrating them to secondary storage. (Wait for this background job to complete before continuing.)
8. Check that the file is migrated by using the `migloc` command:

   /usr/openv/hsm/bin/migloc testfile

The resulting display gives the status of test file `testfile` as Migrated, and shows the media to which VSM has copied the file. If a copy of the file remains on disk in premigration, one of the Medium lines will show a dk method name. See the `migloc(1)` man page for information on the fields in the display.

9. Execute `migpurge` to purge the premigrated file.

```
/usr/openv/hsm/bin/migpurge testfile
```

This frees diskspace. The next time `testfile` is accessed, VSM will cache it back to disk.

10. Check that the file is purged by using `the migloc` command:

```
/usr/openv/hsm/bin/migloc testfile
```

The resulting display gives the status of test file `testfile` as Migrated, and no Medium lines will show a dk method name.

11. Access the purged file to ensure that VSM caches it back to your directory:

```
cat testfile
```

This caches the purged file back to disk.

12. Check that the file is cached by using `the migloc` command:

```
/usr/openv/hsm/bin/migloc testfile
```

The resulting display gives the status of `testfile` as Cached.

Note If you get any error messages following this procedure, examine the VSM error log files to determine the exact cause.

---

13. After concluding your verification, delete the `test` directory containing `testfile`.

Configuring VSM (Java-based GUI) 4

This chapter explains how to configure VSM with the Java-based GUI, VSM-Java. You can configure VSM with either `xhsmadm` or VSM-Java. Refer to "Configuring VSM (Motif-based GUI)" on page 121 for information on how to configure VSM using the Motif-based GUI, `xhsmadm`.

Caution VERITAS strongly recommends that you use only one of the two GUIs, VSM-Java or `xhsmadm`, to reconfigure your system. Do not use both GUIs simultaneously on your system; doing so can produce unpredictable results.

The major topics discussed in this chapter are as follows:

- Install VSM Software
- Configure Tape and Optical Storage Devices
- Login Procedure
- Main Screen Layout
- Configuring VSM
  - Using the Basic Initial Configuration Wizard
  - Using the Advanced Initial Configuration Wizard
  - Manually Changing an Existing Configuration

The VSM-Java graphical user interface (GUI) is a Java application which allows system administrators to configure VSM on a server and to perform a variety of operational commands. VSM-Java is fully documented by its online help system.

Two techniques are available for configuring HSM:

- To perform an initial VSM configuration, use either of the two Initial Configuration Wizards. This guides you through the configuration process. Wizard dialogs contain step-by-step instructions, and additional online help is available from those dialogs. See "Using the Basic Initial Configuration Wizard" on page 198 or "Using the Advanced Initial Configuration Wizard" on page 206.
- To change an existing configuration, use the Edit menu as described in "Manually Changing an Existing Configuration" on page 232.

Install VSM Software

To use VERITAS Storage Migrator or VERITAS Storage Migrator Remote, you must first install the software by following the instructions in the *VERITAS Storage Migrator for UNIX Installation Guide*. A prerequisite for installing VSM software is to have VERITAS NetBackup already installed.

1. Install Media Manager if you plan to use optical disc or tape secondary storage devices. This is not necessary if you only intend to use VERITAS Storage Migrator Remote for secondary storage on local or remote magnetic disks. The examples in this chapter assume a Tape Library robotic device. The *Media Manager System Administrator's Guide* lists other available robotic devices.
2. Install VERITAS Storage Migrator or VERITAS Storage Migrator Remote.

Caution For kernel-based implementations (Solaris ufs file systems), do not use a VSM managed file system as a regular (*ufs*) file system on Sun platforms. Changes will corrupt the file system in that configuration, and it will no longer provide proper access to migrated files. Therefore, an *hsm* file system must always be mounted and used as an *hsm* file system except where otherwise indicated in supporting documentation.

This installs VSM-Java on a HP-UX or Solaris system. VSM-Java files are installed to `/usr/openv/java` directory. Also install NetBackup-Java from the NetBackup release media; this is needed for the installation of the Java Runtime Environment.

To install VSM-Java on a PC, use the InstallShield on the VSM release media. VSM-Java files are installed to a directory of your choice. The default pathname is `C:\Veritas\java`.

Note Attention SGI Users: You must run the Java GUI from NT or Windows 2000, or you must use the Motif-based GUI (`xhsmadm`) from the server.

Once VSM-Java is installed you can use it with its default settings without any reconfiguration. The following reconfiguration notes are optional.

Port Usage (optional)

Once VSM-Java is installed it defaults to using port number 13699 to communicate with the VSM software. If you want to use another port number, you must edit two files. Edit the `/etc/services` file to add a new entry for service `hsm` with protocol `tcp` and the port number of your choice. The `migrd` daemon will listen on this port number the next time it is restarted. Also edit VSM-Java's properties file, `/usr/openv/java/HSMApplet.properties`, to add a new line of the form:

`port=`*port_number*

where *port_number* is the port number you choose in place of 13699. This port number will be used the next time you restart VSM-Java.

Screen Size (optional)

The HSMApplet.properties file also defines the WIDTH and HEIGHT of the applet display window. The default values are 572 and 480 pixels, respectively, and these are the internally enforced minimum dimensions. You can provide larger values if you have a suitably large display and want to be less dependent on scrollbars.

Status Display Script (optional)

VSM-Java offers a System Status command. This produces an activity entry in the lower window, and clicking upon this entry displays output from the script `/usr/openv/hsm/bin/goodies/status_display.sh` in the upper-right window.

The `status_display.sh` script expects to be used either by the Motif-based GUI `xhsmadm` or by VSM-Java. When used by `xhsmadm`, there is no command-line argument and the script is expected to produce one full screen of information and complete, as `xhsmadm` has its own mechanism to poll and repeat. When used by VSM-Java, the `-j` command-line argument is provided and any polling and repetition must be provided by the script itself. As released, the script detects which mode is in effect, and for VSM-Java it repeats 100 times on a polling delay of slightly more than 30 seconds, thus it persists for approximately an hour.

Note Customers are free to modify this script to meet their own site requirements.

Operational Environment

VSM-Java provides a management station which executes on one computer and can login to either the same host or another host for the actual VSM server. Each VSM server host which is to accept VSM-Java must be executing the `migrd` daemon. Start `migrd` by executing the following command:

```
/usr/openv/hsm/bin/admincmd/migrd
```

VSM-Java is available on the following platforms: Solaris 2.5.1, 2.6 or 7, HP-UX 10.20 or 11.0, and PC's running Windows NT or Windows 2000. Heterogeneous networks with VSM installed on supported UNIX systems *other than* HP-UX or Solaris can still use VSM-Java if the native VSM server platform is connected to an HP-UX, Solaris or PC management station with VSM-Java installed.

Note VSM-Java installed on a PC is an interactive station for managing VSM servers on any of the supported UNIX platforms. VSM-Java on a PC is unrelated to VERITAS Storage Migrator Remote for Windows NT.

Note The release level of VSM-Java on the interactive management station must match the release level of VSM on the server being managed.

Note Refer to the chapter on NB-Java in the *NetBackup Release Notes - UNIX* document for a description of deficiencies and limitations that are inherently common to both NetBackup and HSM.

Configure Tape and Optical Storage Devices

If you have installed VSM and Media Manager, you will also need to configure tape and optical devices and start the Media Manager daemon.

Refer to the *Media Manager System Administrator's Guide* for a detailed explanation of how to configure secondary storage devices correctly, create the necessary VSM volume pools, and configure media within Media Manager.

First, create device (`/dev`) entries for the robots, tape drives and other devices used for storage. Once this is done, you will need to define these robots and their devices to Media Manager.

After configuring the robots and drives, you start Media Manager with the `ltid` command which, in turn, initiates the robot daemons. Also enable automatic tape head cleaning for robotic tape devices.

Use one of the Media Manager administration utilities (`vmadm` or `xvmadm`) to configure one or more volume pools for HSM. This allows you to designate specific volumes for use by HSM.

When you have created a VSM volume pool, add the volumes in that pool to the Media Manager volume database.

Login Procedure

1. Type in the following fields on the login screen:

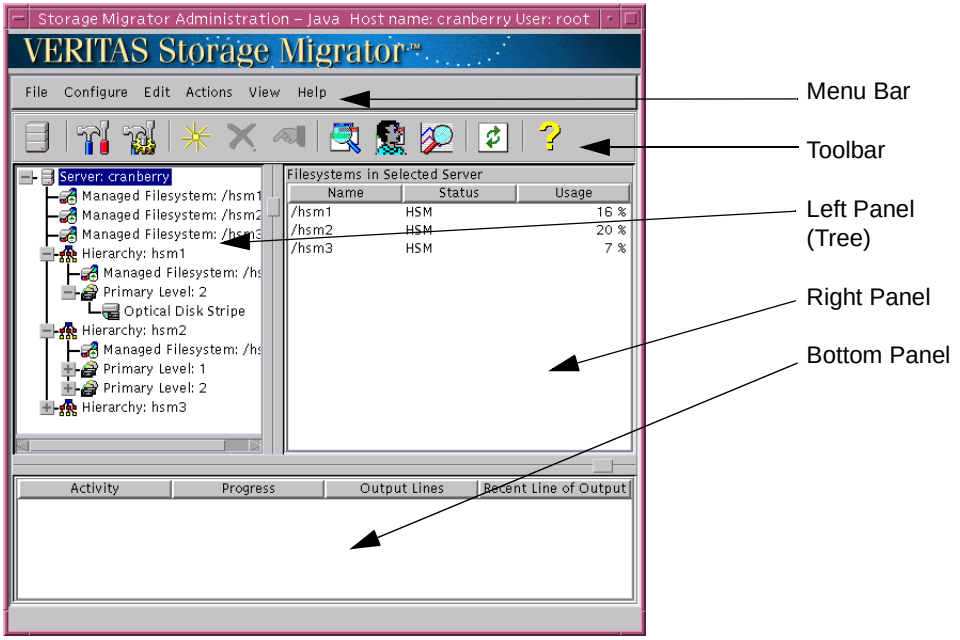
- Host: The hostname of the server on which VSM is installed
- User: Either root or another username with superuser privileges
- Password: The root password or username password

2. Click the Login button, or type <CR> within the Password field.

Main Screen Layout

The main screen layout of VSM-Java contains five elements.

Figure 68. VSM-Java Main Dialog



For further information, refer to the following sections.

- “Menu Bar (Pull-Down Menus)” on page 184
- “Toolbar” on page 195
- “Left Panel” on page 197
- “Right Panel” on page 197
- “Bottom Panel” on page 197

Menu Bar (Pull-Down Menus)

The menu bar at the top of the main screen offers six pull-down menus.

For further information, refer to the following sections.

- “File Menu” on page 185
- “Configure Menu” on page 185
- “Edit Menu” on page 185

- “Actions Menu” on page 189
- “Preferences...” on page 194
- “Help Menu” on page 195

File Menu

Use the File pull-down menu to change servers, or exit VSM-Java.

Change Server...

- To logout from this server and login to another server, select Change Server... from the File pull-down menu and confirm your decision.

Exit

- To exit the VSM-Java interface, select Exit from the File pull-down menu.

Configure Menu

Use the Configure pull-down menu to activate an Initial Configuration Wizard. For more information see “Basic Configure Tool” on page 195 or “Advanced Configure Tool” on page 196.

Edit Menu

Use the Edit pull-down menu to configure VSM manually or to customize the configuration after using an Initial Configuration Wizard.

Actions Menu

Use the Actions pull-down menu to initiate many operations commonly used to manage HSM. For more information, see “Actions Menu” on page 189.

View Menu

Use the View pull-down menu to manage VSM job activity. For more information, see “Preferences...” on page 194.

Help Menu

Use the Help pull-down menu to bring up the Help window.

Note Refer the “Glossary” on page 575 for term definitions.

Adding Configuration Elements

- To *add a hierarchy* to the server:

  a. Highlight the Server node in the tree panel on the left.

  b. Select New Hierarchy from the Edit pull-down menu, or click the New tool.

  When the hierarchy is created, it appears on the tree display as a child under the Server node. If you want to manage more than one file system, you may either define a separate hierarchy for each one or assign multiple file systems to a single hierarchy. See "Advanced Wizard- Hierarchy Properties" on page 208 for more information.

- To *assign a managed file system* to a hierarchy:

  a. Highlight the Filesystem node to be managed in the tree panel on the left.

  b. Select New HSM Management from the Edit pull-down menu, or click the New tool.

  Choose the hierarchy from the pop-up menu. When the file system is assigned, it appears on the tree display as a Managed Filesystem, both as a child of the Server node and as a child of the Hierarchy node. Repeat for each file system to be managed.

  You can also use the Edit pull-down menu to change managed file system properties. An advisory label appears if changing a property on one file system might affect other managed file systems in the same hierarchy. See "Advanced Wizard- Filesystem Properties" on page 223 for more information.

- To *define storage levels* for each hierarchy:

  a. Highlight a Hierarchy node in the tree panel on the left.

  b. Select New Level from the Edit pull-down menu, or click the New tool.

  Define at least one Primary level for the selected hierarchy. If you intend to activate dual migration copies, also define at least one Alternate level. If you intend to use the multilevel migration feature of HSM, define more than one Primary or Alternate level for the selected hierarchy. When a storage level is defined, it appears on the tree display as a child of the Hierarchy. Repeat for each hierarchy.

  You can also use the Edit pull-down menu to change storage level properties. An advisory label appears if changing a property on one storage level might affect other storage levels in the same hierarchy. See "Manually Change Level Properties" on page 232 for more information.

- To *assign stripes* for each storage level:

  a. Highlight a storage Level node in the tree panel on the left.

  b. Select New Stripe from the Edit pull-down menu, or click the New tool.

  Assign one or more recording methods (stripes) to the selected storage level. If you assign multiple stripes to a level, VSM will migrate files in a round-robin manner, using multiple storage devices simultaneously for faster performance. This is known as concurrent recording. When a stripe is assigned it appears on the tree display as a child of the storage Level. Repeat for each storage level.

  You can also use the Edit pull-down menu to change stripe properties. An advisory label appears if changing a property on one stripe might affect other stripes in the same storage level. See “Advanced Wizard- Stripe Properties” on page 211 for more information.

- To *implicitly assign tape or optical disc volumes*:

  a. Highlight a Tape, Optical Disc or WORM Disc stripe node in the tree panel on the left.

  b. Select Change Stripe from the Edit pull-down menu, or click the New tool.

  c. Name a unique Media Manager volume pool.

  d. Use the Media Manager interface to assign individual tapes or optical discs to the same volume pool.

  e. Repeat for each tape and optical disc stripe.

- To explicitly assign Alternate Disk, FTP (file transfer protocol) or NetBackup volumes:

  a. Highlight an Alternate Disk, FTP, or NetBackup stripe node in the tree panel on the left.

  b. Select New Volume from the Edit pull-down menu, or click the New tool.

  c. Add volumes.

  d. Repeat for each Alternate Disk, FTP, or NetBackup stripe.

Deleting Configuration Elements

- To *delete a configured hierarchy*
  - **a.** Highlight a Hierarchy node in the tree panel on the left.
  - **b.** Select Delete Hierarchy... from the Edit pull-down menu, or click the Delete tool.

  - **c.** Confirm your command.
- To *delete a configured managed filesystem*:
  - **a.** Highlight a Managed Filesystem node in the tree panel on the left.
  - **b.** Select Delete Filesystem... from the Edit pull-down menu, or click the Delete tool.

  - **c.** Confirm your command.
- To *delete a configured level*:
  - **a.** Highlight a Level node in the tree panel on the left.
  - **b.** Select Delete Level... from the Edit pull-down menu, or click the Delete tool.

  - **c.** Confirm your command.
- To *delete a configured stripe*:
  - **a.** Highlight a Stripe node in the tree panel on the left.
  - **b.** Select Delete Stripe... from the Edit pull-down menu, or click the Delete tool.

  - **c.** Confirm your command.

Changing Configuration Elements

- To *change a configured* hierarchy:
  - **a.** Highlight a Hierarchy node in the tree panel on the left.
  - **b.** Select Change Hierarchy Properties... from the Edit pull-down menu, or click the Properties tool.

- To *change a configured managed filesystem*:
  - **a.** Highlight a Managed Filesystem node in the tree panel on the left.
  - **b.** Select Change Filesystem Properties... from the Edit pull-down menu, or click the Properties tool.

- To *change a configured level*:
  - **a.** Highlight a Level node in the tree panel on the left.
  - **b.** Select Change Level Properties... from the Edit pull-down menu, or click the Properties tool.

- To *change a configured stripe*:
  - **a.** Highlight a Stripe node in the tree panel on the left.
  - **b.** Select Change Stripe Properties... from the Edit pull-down menu, or click the Properties tool.

Actions Menu

Use the Actions pull-down menu to initiate many operations commonly used to manage HSM. Inappropriate selections are grayed out and cannot be selected.

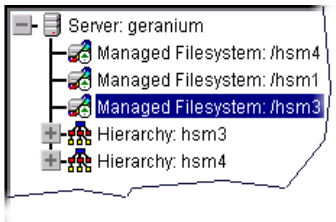
Server

In the tree panel on the left, highlight the Server node.

- To set logging level, select Server > Set Logging Level... from the Actions pull-down menu. Valid values are 1 through 8. Default is 3.
- To start VSM daemons, select Server > Start Daemons from the Actions pull-down menu.
- To stop VSM daemons, select Server > Stop Daemons... from the Actions pull-down menu.
- To kill VSM processes on a server, select Server > Kill All HSM Processes on Server... from the Actions pull-down menu.
- To restart migration and move processes on a server, select Server > Restart Migrations and Moves for Server... from the Actions pull-down menu.
- To clean VSM database entries, select Server > Clean DB Entries for Server... from the Actions pull-down menu.
- To clear VSM database locks on a server, select Server > Clear Old Locks... from the Actions pull-down menu.
- To migrate files on a server to secondary storage, select Server > Batch Migrate from the Actions pull-down menu.
- To move files on a server to another migration level, select Server > Move from the Actions pull-down menu.
- To add a new or upgraded license key, select Server > Add License Key... from the Actions pull-down menu.

- To view your license key information, select Server > Show License Key from the Actions pull-down menu.
- To view the usage capacity of your file system, select Server > Show Capacity Usage from the Actions pull-down menu.

Figure 69. Example Server Node



Filesystem

In the tree panel on the left, either highlight a particular Managed Filesystem node as it appears under a Hierarchy, or highlight a particular Hierarchy node as it appears under the Server node and then highlight the Filesystem in the panel on the right.

- To kill VSM processes in a managed file system, select Filesystem > Kill HSM Processes for Filesystem... from the Actions pull-down menu.
- To restart migration and move processes in a managed file system, select Filesystem > Restart Migrations and Moves for Filesystem... from the Actions pull-down menu.
- To clean VSM database entries, select Filesystem > Clean DB Entries for Filesystem... from the Actions pull-down menu.
- To check VSM database consistency in a managed file system, select Filesystem > Check DB Consistency for Filesystem... from the Actions pull-down menu.
- To correct VSM database inconsistencies in a managed file system, select Filesystem >Fix DB for Filesystem... from the Actions pull-down menu.
- To clear VSM database locks in a managed file system, select Filesystem > Clear Old Locks... from the Actions pull-down menu.
- To migrate files in a managed file system to secondary storage, select Filesystem > Batch Migrate from the Actions pull-down menu.
- To purge files from disk in a managed file system after they are migrated to secondary storage, select Filesystem > Batch Purge... from the Actions pull-down menu.
- To move files in a managed file system to another migration level, select Filesystem > Move... from the Actions pull-down menu.
- To activate users, select Filesystem > Activate... from the Actions pull-down menu. This changes the state of the managed file system to active.

- To deactivate users, select Filesystem > Deactivate... from the Actions pull-down menu. This changes the state of the managed file system to inactive.
- To change a user's password, select Filesystem > Change Password... from the Actions pull-down menu.

Level

In the tree panel on the left, either highlight a particular Level node as it appears under a Hierarchy, or highlight a particular Hierarchy node as it appears under the Server node and then highlight the Level in the panel on the right.

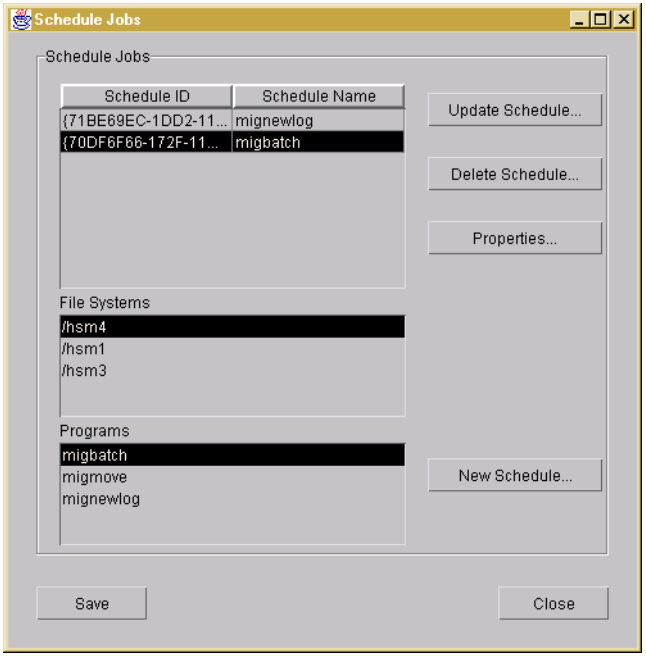
- To move files to the next higher configured migration level and remove the source level copy, select Level > Move This Level to Next... from the Actions pull-down menu.
- To move files to a specific migration level and remove the source level copy, select Level > Move Between Specified Levels... from the Actions pull-down menu.
- To copy files to a specific migration level and retain the source level copy, select Level > Copy Between Specified Levels... from the Actions pull-down menu.

Volume

In the tree panel on the left, highlight a particular Stripe node as it appears under a Level. Then select a volume from the panel on the right.

- To clean obsolete entries from a volume, select Volume > Clean Obsolete Entries... from the Actions pull-down menu.
- To delete a volume from the VSM volume database, select Volume > Delete Volume... from the Actions pull-down menu.
- To consolidate a volume, select Volume > Consolidate Volume... from the Actions pull-down menu.
- To recycle a volume, select Volume > Recycle Volume... from the Actions pull-down menu.
- To scan a volume, select Volume > Scan Volume... from the Actions pull-down menu.
- To set a full volume to be read only, select Volume > Set Volume Readonly... from the Actions pull-down menu.
- To change a password, select Volume > Change Password... from the Actions pull-down menu.

Figure 70. Schedule Jobs Dialog



Schedule

- To access the VSM Scheduling interface, select Schedule... from the Actions pull-down menu. Use this interface to add, revise, or delete a scheduled job for your file system.

Note If you use `mignewlog` with the scheduler tool, VSM automatically creates a new log file and moves the old log file. For example, you have a log file hsm1 and run `mignewlog`, you would see hsm1, which is the new log, and hsm.*date*, which is the old log file. Keep old log files to help troubleshoot VSM at a later time.

- To *schedule a job* within the Scheduling interface (on the Schedule Jobs screen):

  a. Highlight the file system for which the job will be scheduled in the File Systems list.

  b. Highlight the job in the Programs field

  c. Click on New Schedule… to bring up the Schedule Component Configuration dialog.

d. Choose the appropriate settings for this job schedule within the Schedule Component Configuration dialog and click OK.

e. Click Close or Save to save your job schedule.

f. Click Close to exit the Schedule Jobs screen.

- To *change an existing schedule* within the Scheduling interface (on the Schedule Jobs screen):

  a. Highlight the schedule you want to revise from the Schedules list.

  b. Click on Update Schedule… to bring up the Schedule Component Configuration dialog.

  c. Choose the appropriate settings for this job schedule within the Schedule Component Configuration dialog and click OK.

  d. Click Close or Save to save your job schedule.

  e. Click Close to exit the Schedule Jobs screen.

- To *delete a schedule* within the Scheduling interface (on the Schedule Jobs screen):

  a. Highlight the schedule you want to revise from the Schedules list.

  b. Click on Delete Schedule… and verify that you want to delete the schedule.

  c. Click OK to delete the schedule.

  d. Click Close to exit the Schedule Jobs screen.

- To *view properties for a schedule* within the Scheduling interface (on the Schedule Jobs screen):

  a. Highlight the schedule from the Schedules list that has the properties you want to view.

  b. Click on Properties… to view the Schedule Properties dialog. This screen displays information specific to this job, such as file system name, program name, and a description of the schedule.

  c. If you want to revise the properties, click Edit, make the changes within the Schedule Component Configuration dialog, and click OK.

  d. Click OK to close the Schedule Properties window.

  e. Click Close to exit the Schedule Jobs screen.

Activity Monitor

- To access the VSM Activity Monitor interface, select Activity Monitor from the Actions pull-down menu. Use this interface to monitor the job activity on your file system.

File Browser

- To access the VSM File Browser interface, select File Browser from the Actions pull-down menu. Use this interface to execute the most common VSM user commands on your file system.

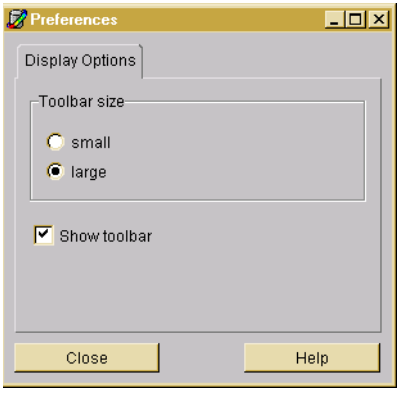
File System Analyzer

- To scan your file system via the VSM File System Analyzer interface, select File System Analyzer from the Actions pull-down menu. Use this interface to visualize how the size of your file system grows over time.

View Menu

Use the View pull-down menu to manage VSM job activity.

Figure 71. Preferences Dialog



Preferences...

- To change toolbar display options, select Preferences from the View pull-down menu.

Volume Preferences

- To display volume characteristics while the granule count runs in the background, select Volume > Delayed Granule Count from the Actions pull-down menu.
- To display volume characteristics only after the granule count is complete, select Volume > Wait for Granule Count from the Actions pull-down menu.
- To display volume characteristics while suppressing the granule count, select Volume > No Granule Count from the Actions pull-down menu.

Refresh

- To refresh the screen, select Refresh from the View pull-down menu.

System Status

- To view the status of your files system, select System Status from the View pull-down menu. The status information displays in the bottom panel.

Managing Activity Table Tasks

- To manage jobs in the activity table, displayed in the panel at the bottom of the main screen, select one of the three items from the View pull-down menu:
  - Clear Completed Jobs from Activity Table
  - Detach Selected Job from Activity Table
    In the activity table, highlight the job you want to detach and then select this item from the View pull-down menu.
  - Cancel Selected Job from Activity Table
    In the activity table, highlight the job you want to cancel and then select this item from the View pull-down menu.

Help Menu

Use the Help pull-down menu to bring up the HELP window and to display VSM Administration GUI version information.

Toolbar

The toolbar at the top of the main screen offers eleven tool options.

To change toolbar display options, select Preferences from the View pull-down menu.

Change Server Tool

Click the Change Server tool to logout from this server and login to another server.

Basic Configure Tool

Click the Basic Configure tool to activate the Basic Initial Configuration Wizard.

Advanced Configure Tool

Click the Advanced Configure tool to activate the Advanced Initial Configuration Wizard.

New Tool

Click the New tool to add an element to the configuration. See “Adding Configuration Elements” on page 186 for more information.

Delete Tool

Click the Delete tool to remove an element from the configuration. See “Deleting Configuration Elements” on page 188 for more information.

Properties Tool

Click the Properties tool to change the properties of an element in the configuration. See “Changing Configuration Elements” on page 188 for more information.

Activity Monitor Tool

Click the Activity Monitor tool to open the VSM Activity Monitor.

File Browser Tool

Click the File Browser tool to open the VSM File Browser.

File System Analyzer Tool

Click the File System Analyzer tool to open the File System Analyzer interface.

Refresh Tool

Click the Refresh tool to refresh the current window display.

Help Topics Tool

Click the Help Topics tool to bring up the HELP window.

Note To change toolbar display options, select Preferences from the View pull-down menu.

Left Panel

The main screen displays a tree structure of the VSM Server, its configured hierarchies, and the directory structure for those configured hierarchies in the panel on the left. To expand or contract the information displayed in this panel, click on selected nodes in the tree.

Right Panel

When you highlight a particular element in the left (tree) panel, the panels on the right display information about that element. Clicking the column headings on the right panel sorts the information in the panel, toggling from ascending to descending sorts.

Bottom Panel

Job activity is displayed in the panel at the bottom of the main screen. Most of the commands invoked from the Actions pull-down menu execute asynchronously, and their progress is summarized in this panel. See “Actions Menu” on page 189 for more information.

- To see more job detail, highlight a job in the bottom panel. The complete output for that job is displayed in the right panel.
- To manage job activity, use the View pull-down menu. See “Preferences...” on page 194 for more information.

Configuring VSM

VSM has two graphical user interfaces (GUI), either of which lets you easily configure your system. This chapter describes the Java-based GUI, VSM-Java. Refer to “Configuring VSM (Motif-based GUI)” on page 121 for information on how to configure VSM using the Motif-based GUI, `xhsmadm`.

To perform an initial VSM configuration, use one of the two Initial Configuration Wizards. Wizard dialogs contain step-by-step instructions, and additional online help is available from those dialogs.

- The Basic Initial Configuration Wizard guides you through the basic configuration process, accepting many default values. See “Basic Configure Tool” on page 195 for more information.
- The Advanced Initial Configuration Wizard offers greater flexibility for customizing your initial VSM configuration. See “Advanced Configure Tool” on page 196 for more information.
- Optionally, you can use a manual procedure to configure hierarchies and stripes before using an Initial Configuration Wizard. See “Manual Configuration Procedure” on page 230.
- To change an existing configuration, use the Edit menu as described in “Manually Changing an Existing Configuration” on page 232.

Using the Basic Initial Configuration Wizard

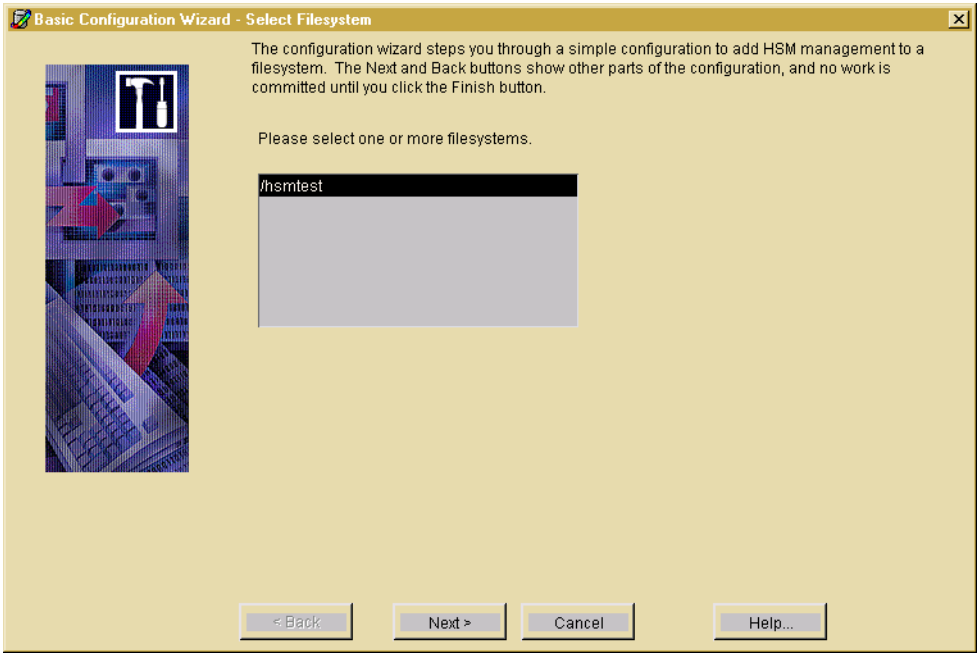
The Basic Initial Configuration Wizard steps through a series of dialogs to configure an unmanaged filesystem into a VSM managed file system, accepting many default values. The wizard will ask which filesystem to manage and which storage method to use. The last wizard screen summarizes all configured and default values for the managed file system. The wizard will not commit completion of work until the last screen, when the Next button is redefined as the Finish button.

Use the Configure pull-down menu or click the Basic Configure tool to activate the Basic Initial Configuration Wizard.

Basic Wizard- Select Filesystem

The first step in the configuration wizard is to select a filesystem you want to manage with HSM.

Figure 72. Select Filesystem Dialog



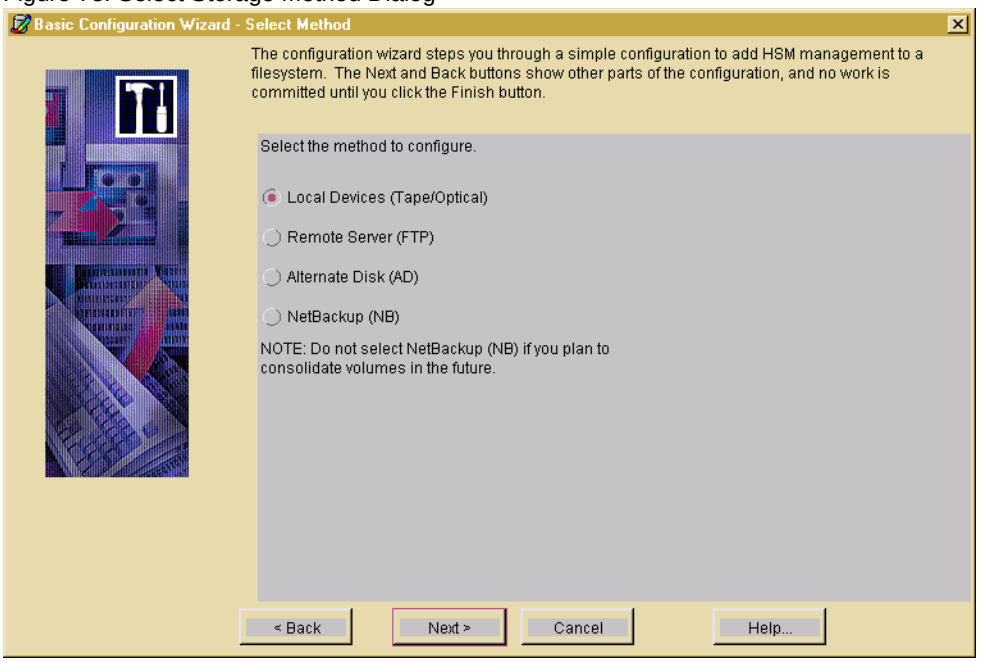
You can select one or more file systems to manage.

Click the Next button when done.

Basic Wizard- Select Method

Select a method for storing data migrated from the VSM managed filesystem.

Figure 73. Select Storage Method Dialog



- Local Devices (Tape Optical): Local tape or optical disc methods. Optical disc methods may be to either rewritable or write once, read many (WORM) media.
- Remote Server (FTP): Remote method using File Transfer Protocol.
- Alternate Disk (AD): This alternate magnetic method is used in the two-step consolidation of tape or optical disc volumes, for migrating files to another file system mounted on the same managed server, and as a remote method when used with NFS.
- NetBackup (NB): Remote method using NetBackup.

Note Do not use the NetBackup method if you intend to consolidate volumes.

Click the Next button when done.

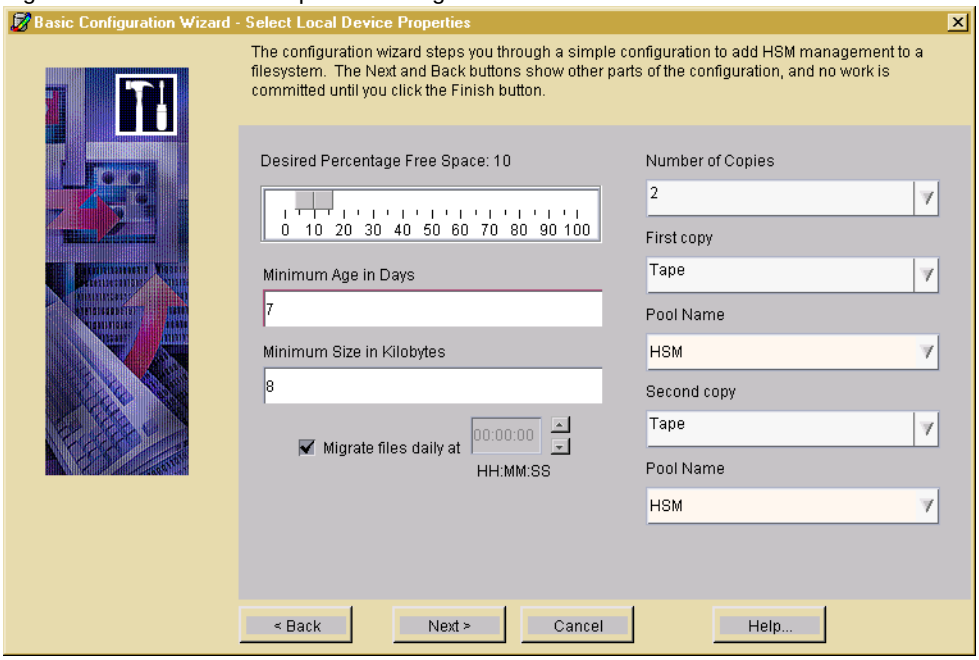
Basic Wizard- Select Local Device Properties

Select the properties for the VSM managed filesystem and storage method being configured. Configurable values vary, depending on the method chosen.

Local Devices

Select the properties for the VSM managed file system and remote server storage method being configured.

Figure 74. Local Device Properties Dialog



Desired Percentage Free Space: the percentage of free space below which VSM makes more space available by initiating migration operations. The default value is 10 (percent).

Minimum Age in Days: Do not migrate files accessed or modified within this time. Set this to a value greater than 0 to prevent files from migrating the same day they are created. Default is 7 days.

Minimum Size in Kilobytes: Do not migrate files smaller than this. Default is 8 kilobytes.

Migrate Files Daily...: If you want to schedule migration operations daily, click the Migrate Files Daily... check box and choose the start time from the list box.

Number of Copies: Migrate one or two (default) copies to secondary storage.

Note If only one tape or optical device with a single drive is attached, the default is one copy.

First copy: Select the storage medium for the first copy.

Pool Name: Select the volume pool name for the first copy.

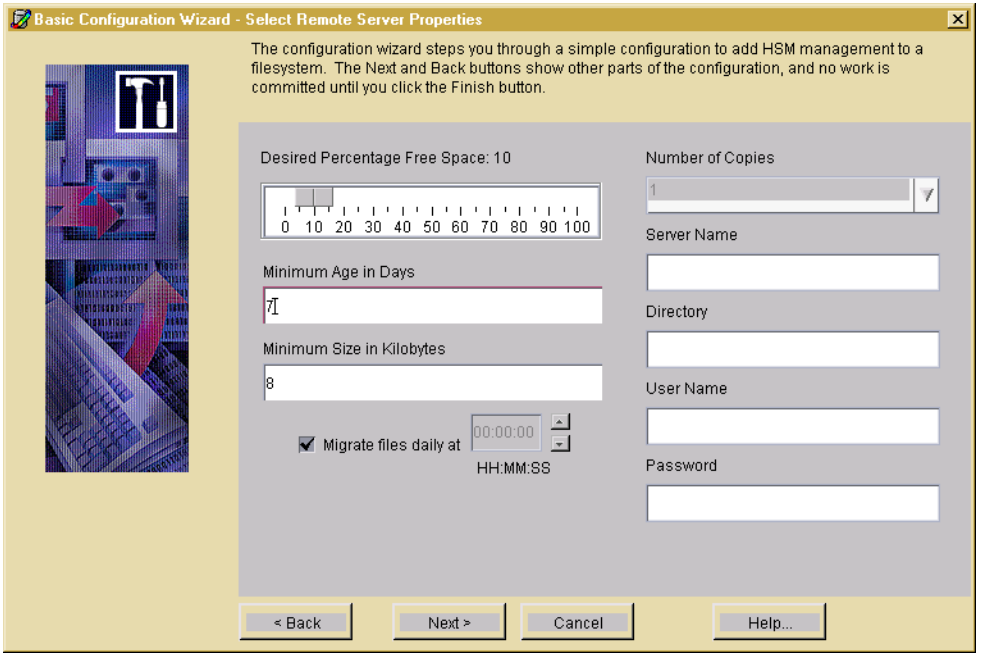
Second Copy: Select the storage medium for the second copy (if configured).

Pool Name: Select the volume pool name for the second copy (if configured).

Click the Next button when done.

Basic Wizard- Select Remote Server

Figure 75. Remote Server Properties Dialog



Desired Percentage Free Space: the percentage of free space below which VSM makes more space available by initiating migration operations. The default value is 10 (percent).

Minimum Age in Days: Do not migrate files accessed or modified within this time. Set this to a value greater than 0 to prevent files from migrating the same day they are created. Default is 7 days.

Minimum Size in Kilobytes: Do not migrate files smaller than this. Default is 8 kilobytes.

Migrate Files Daily...: If you want to schedule migration operations daily, click the Migrate Files Daily... check box and choose the start time from the list box.

Number of Copies: Migrate one (default) copy to secondary storage.

Server Name: The name of the remote server. This can be the internet id or number of the server. VSM uses this name on the `ftp open` command as the host parameter. It must be a valid FTP host.

Directory:The full pathname of the file system directory or subdirectory on the remote server. The VSM username on the managed server must have read and write permissions to this remote directory. This can be any directory on the remote server that is not already registered for HSM.

User Name: The username (and password) VSM uses when accessing the remote server from the managed server through FTP. This name (and password) must be valid on the remote server and also must have read and write access to the remote file system that you are using for migration.

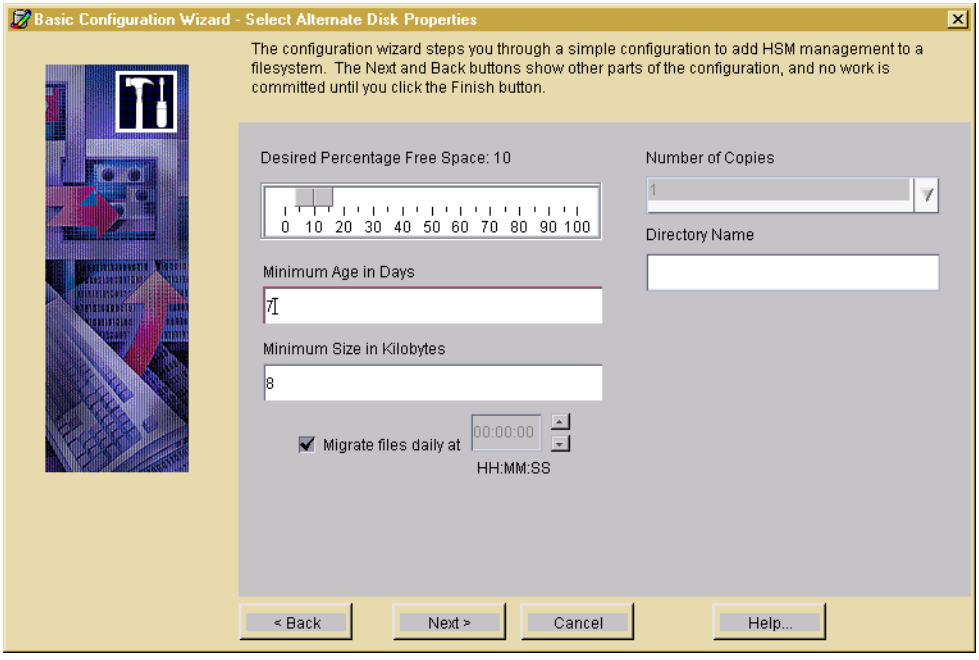
Password: See User Name.

Click the Next button when done.

Basic Wizard- Select Alternate Disk Properties

Select the properties for the VSM managed files system and alternate storage method being configured.

Figure 76. Alternate Disk Properties Dialog



Desired Percentage Free Space: the percentage of free space below which VSM makes more space available by initiating migration operations. The default value is 10 (percent).

Minimum Age in Days: Do not migrate files accessed or modified within this time. Set this to a value greater than 0 to prevent files from migrating the same day they are created. Default is 7 days.

Minimum Size in Kilobytes: Do not migrate files smaller than this. Default is 8 kilobytes.

Migrate Files Daily...: If you want to schedule migration operations daily, click the Migrate Files Daily... check box and choose the start time from the list box.

Number of Copies: Migrate one (default) copy to secondary storage.

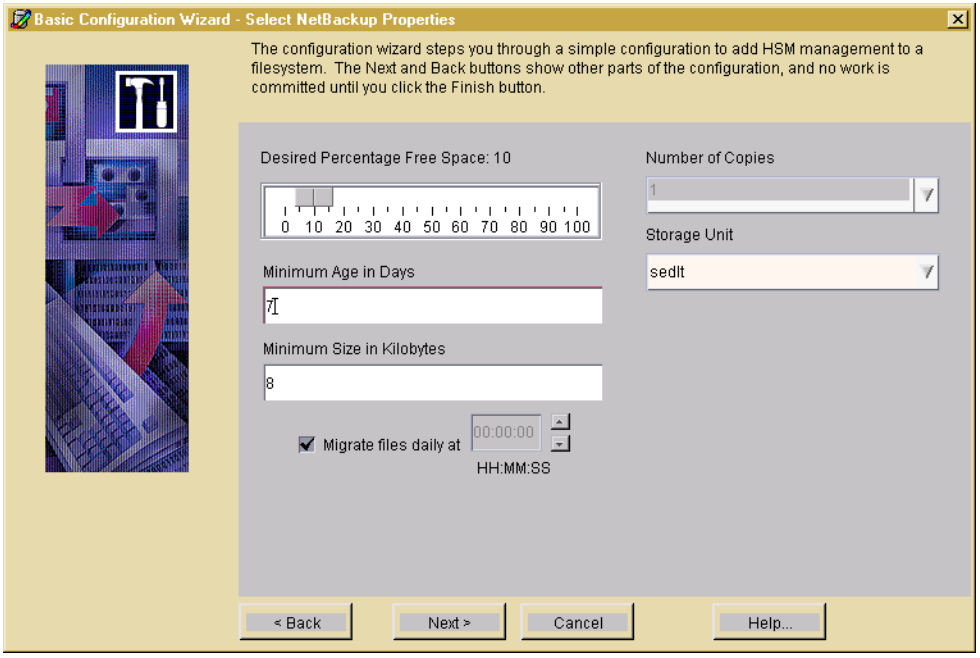
Directory Name: The device name or the filesystem mount point required when registering a volume. Make sure the specified device name or filesystem is mounted before registering the volume. Do not register a disk partition for a directory below the mount point because the entire capacity of the file system at the mount point is assumed.

Click the Next button when done.

Basic Wizard- Select NetBackup Properties

Select the properties for the VSM managed file system and NetBackup storage method.

Figure 77. NetBackup Properties Dialog



Desired Percentage Free Space: The percentage of free space below which VSM makes more space available by initiating migration operations. The default value is 10 (percent).

Minimum Age in Days: Do not migrate files accessed or modified within this time. Set this to a value greater than 0 to prevent files from migrating the same day they are created. Default is 7 days.

Minimum Size in Kilobytes: Do not migrate files smaller than this. Default is 8 kilobytes.

Migrate Files Daily...: If you want to schedule migration operations daily, click the Migrate Files Daily... check box and choose the start time from the list box.

Number of Copies: Migrate one (default) copy to secondary storage.

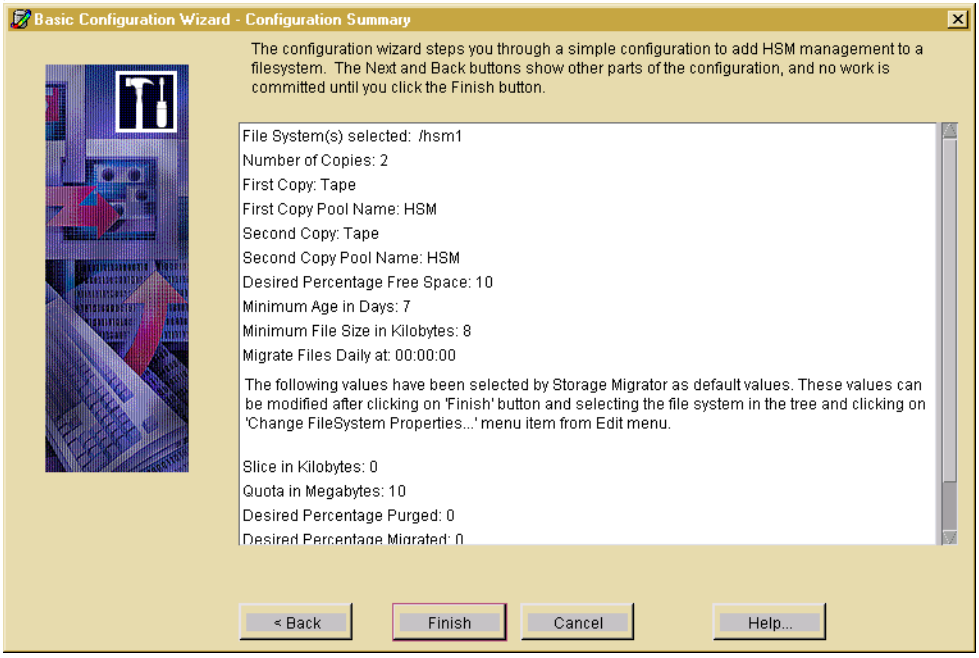
Storage Unit: Select the storage unit from the list provided by NetBackup.

Click the Next button when done.

Basic Wizard- Configuration Summary

The last screen summarizes all configured and default values for the managed file system. Properties vary, depending on the method chosen.

Figure 78. Configuration Summary Dialog



Review this summary and make sure you are satisfied with the configuration.

Click the Finish button when done. The wizard will not commit completion of work until until you click Finish.

To change an existing configuration, use the Edit menu as described in “Manually Changing an Existing Configuration” on page 232.

Using the Advanced Initial Configuration Wizard

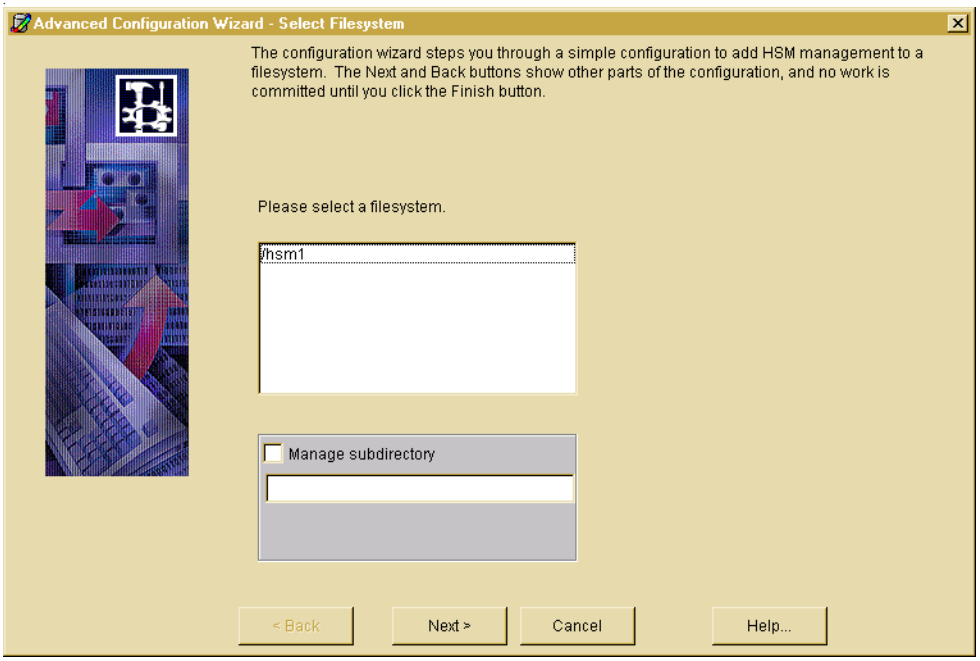
The Advanced Initial Configuration Wizard steps through a series of dialogs to configure an unmanaged filesystem into a VSM managed file system. The wizard will ask which filesystem to manage, then whether or not to use an existing hierarchy. If you do not have any hierarchies or do not want to use any previously defined hierarchies, the wizard will step through the creation of a new hierarchy. The wizard will not commit completion of work until the last screen, when the Next button is redefined as the Finish button.

Use the Configure pull-down menu or click the Advanced Configure tool to activate the Advanced Initial Configuration Wizard.

Advanced Wizard- Select Filesystem

The first step in the configuration wizard is to select a filesystem you want to manage with HSM.

Figure 79. Select Filesystem Dialog



If you want to manage a subdirectory in a mounted file system instead of managing the entire file system, click the Manage subdirectory check box and type the path to the managed directory in the field provided. This expression must not contain a trailing slash.

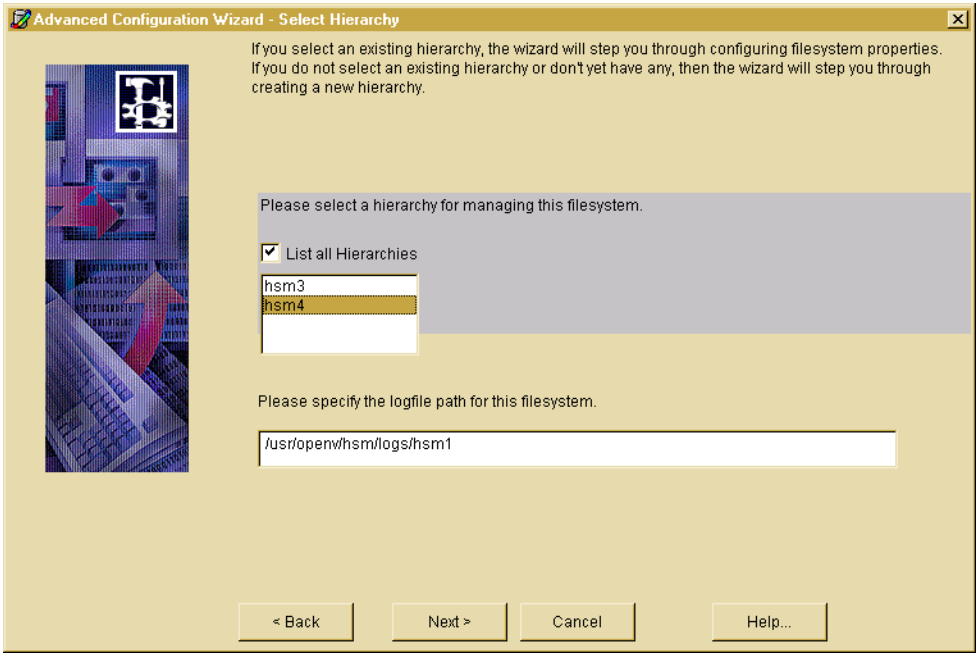
Click the Next button when done.

Advanced Wizard- Select Hierarchy

Note If no hierarchies have been configured, this dialog is omitted.

Use this screen to assign the filesystem you want to manage to a hierarchy.

Figure 80. Select Hierarchy Dialog



1. If assigning to an existing hierarchy, select that hierarchy from the list provided, and click the Next button.

   OR-- If not assigning to an existing hierarchy, simply click the Next button.

2. Specify the logfile path for this filesystem or accept the default path shown. This is the pathname of the file that will contain log messages for operations pertaining to the file system and databases. The full path must be less than or equal to 1023 characters.

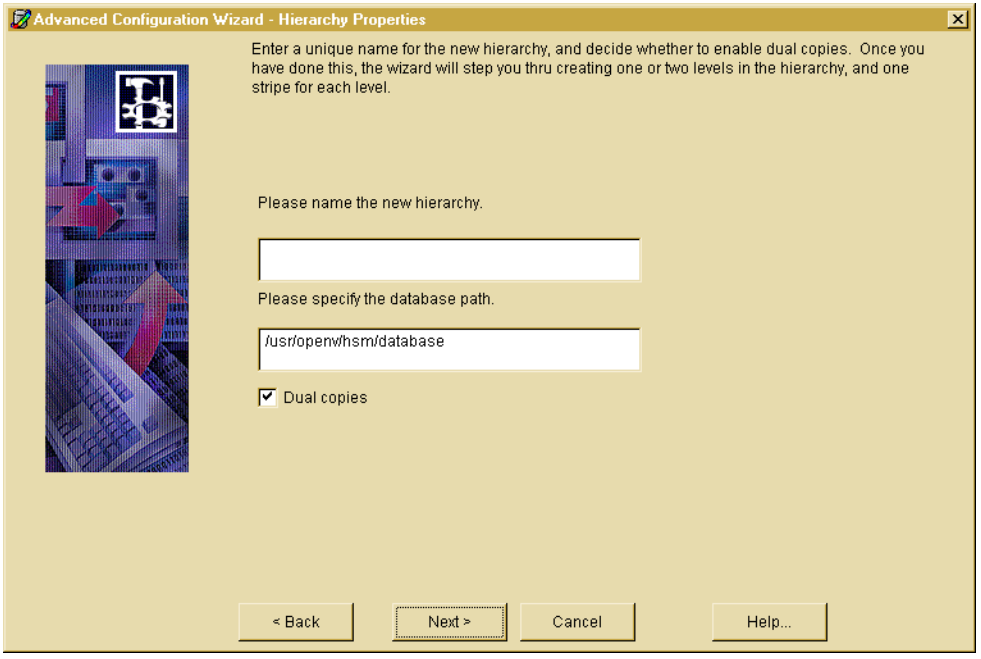
3. Click the Next button when done.

Advanced Wizard- Hierarchy Properties

Note If a previously configured hierarchy was selected, this dialog is omitted.

Use this screen to create a name for the new VSM hierarchy.

Figure 81. Hierarchy Properties Dialog



1. Specify the name of the hierarchy in alphanumeric characters.
2. Specify the database path for this hierarchy or accept the default path shown. This is the path name of the directory containing the database and workdir directories. The full path must be less than or equal to 1023 characters.
3. Dual Copies: Migrate two copies to secondary storage (default). Deselect only if you want VSM to migrate just one copy to secondary storage.
4. Define at least one Primary Level for the selected hierarchy. If you select Dual Copies, also define at least one Alternate Level.
5. Click the Next button when done.

Advanced Wizard- New Level

Use this screen to define at least one Primary level for this hierarchy.

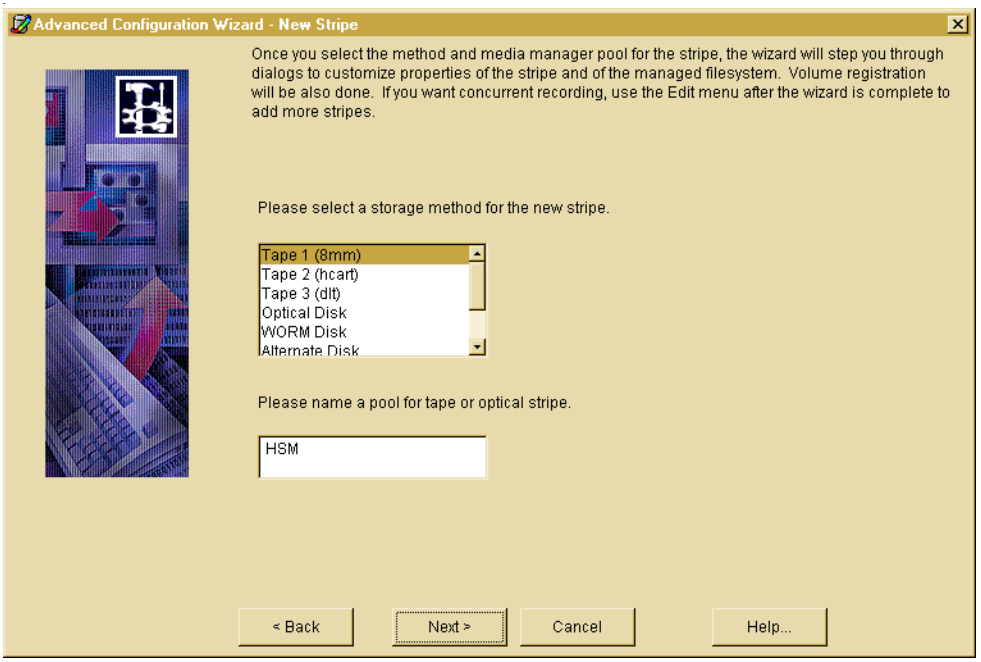
Figure 82. New Level Dialog

1. Define at least one Primary level for the selected hierarchy. If you intend to activate dual migration copies, also define at least one Alternate level. If you intend to use the multilevel migration feature of HSM, define more than one Primary or Alternate level for the selected hierarchy. When a storage level is defined it appears on the tree display as a child of the Hierarchy. Repeat for each hierarchy defined.
2. An advisory label appears if changing a property on one storage level might affect other storage levels in the same hierarchy.
3. Click the Next button when done.

Advanced Wizard- New Stripe

Use this screen to assign the stripe properties for the storage method used with this filesystem.

Figure 83. New Stripe Dialog



1. Method: The name of the storage method.
   - Tape 1, Tape 2, Tape 3: Your choice of tape method depends on the type of device you have on your site. If you have more than one type of device, you can analyze their characteristics to determine the method to use for a specific file system.
   - Optical Disc: Rewritable optical disc method.
   - WORM Disc: Write once, read many optical disc method.
   - Alternate Disk: This alternate magnetic method is used in the two-step consolidation of tape or optical disc volumes, for migrating files to another file system mounted on the same managed server, and as a remote method when used with NFS.
   - FTP: Remote method using File Transfer Protocol.
   - NetBackup: Remote method using NetBackup.
2. Pool: The media manager pool for tape and optical volumes.
3. Click the Next button when done.

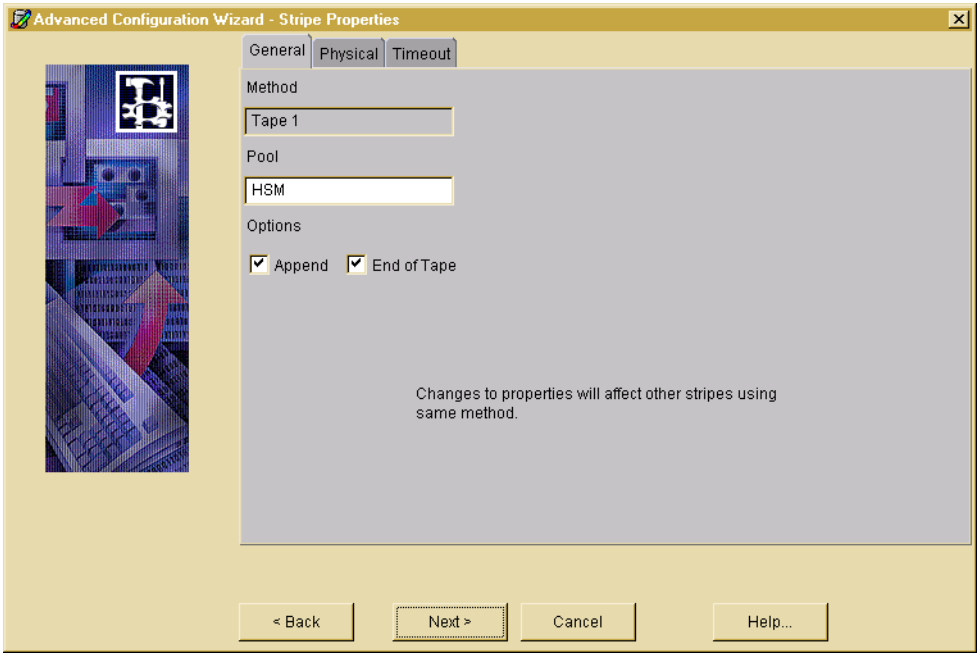
Advanced Wizard- Stripe Properties

Use this screen to assign the stripe properties for Tape, Optical Disc, or WORM Disc methods. These methods are shown on three tabbed pages in this dialog: “General Tab (Tape and Optical Media),” “Physical Tab (Tape and Optical Media),” and “Timeout Tab (Tape and Optical Media).” Click each tabbed page in the dialog to reveal the properties.

Stripe properties for all other storage methods are shown on a single page in this dialog: “Stripe Properties - for an Alternate Disk,” “Stripe Properties - for FTP,” and “Stripe Properties - for NetBackup.”

General Tab (Tape and Optical Media)

Figure 84. General Stripe Properties Dialog



Method: The name of the storage method.

- Tape 1, Tape 2, Tape 3: Your choice of tape method depends on the type of device you have on your site. If you have more than one type of device, you can analyze their characteristics to determine the method to use for a specific file system.
- Optical Disk: Rewritable optical disc method.
- WORM Disk: Write once, read many optical disc method.

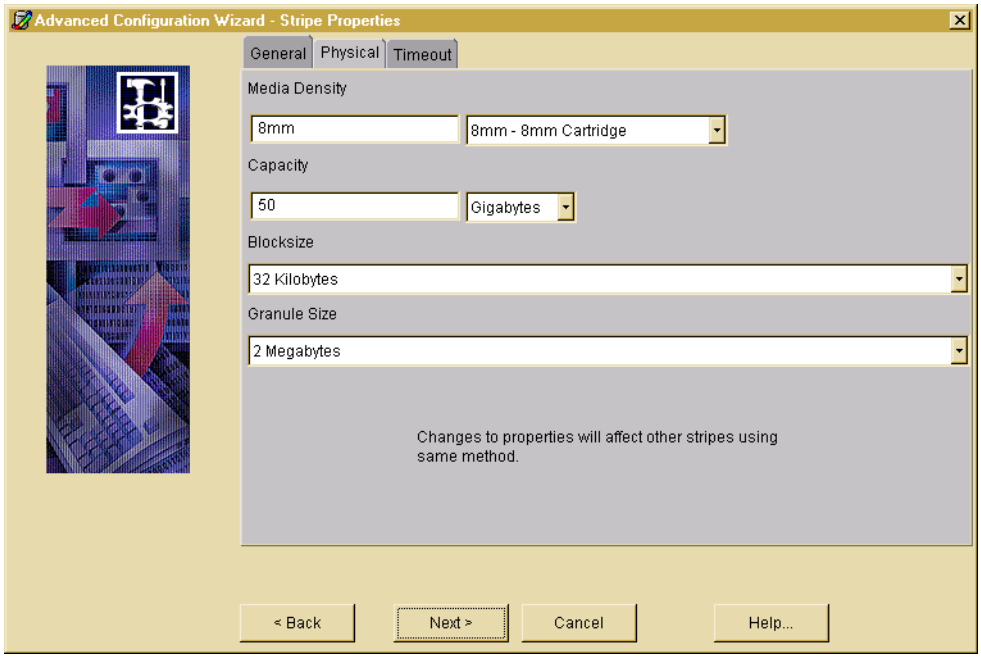
Pool: The media manager pool for tape and optical volumes.

Options:

- Append: Place multiple migrations on the same volume until it reaches full capacity. If not checked, each migration always starts on an empty volume. Applies to the Tape, Optical Disc, and WORM Disc methods.
- End of Tape: Place multiple migrations on the same volume until end of tape (EOT) is encountered instead of full capacity. Applies to the Tape methods.

Physical Tab (Tape and Optical Media)

Figure 85. Physical Stripe Properties Dialog



Media Density: Density of the tape or optical disc medium. Pull down the menu and select the type of storage medium configured for this storage name. This is used only for the Tape, Optical Disc, and WORM Disc methods.

Capacity: Capacity of the method in bytes. Applies only to Tape and Alternate Disk methods. During labeling, VSM records this value on the tape volume. The capacity of an optical disc volume (Optical Disc or WORM Disc method) is determined automatically by VSM when the volume is labeled. The capacity of the FTP and NetBackup methods is specified when you register volumes.

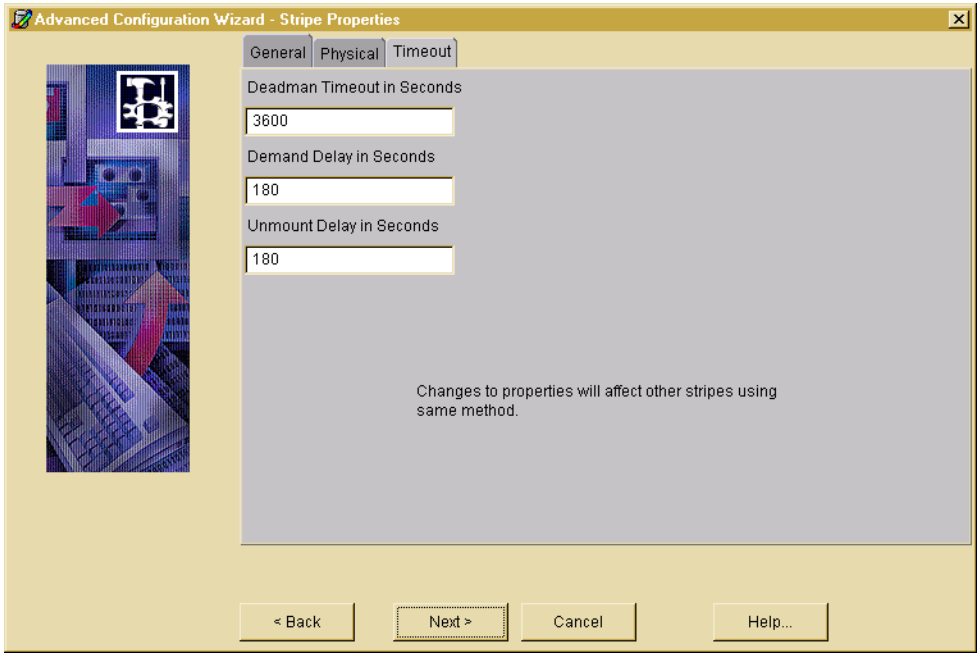
Block Size: Block size in bytes to use when writing to the device. The allowable block sizes are: 512, 1K, 2K, 4K, 8K, 16K, 32K, 64K, 128K, and 256K. The value must be a power of 2. Do not change this parameter after the initial configuration. Applies only to the Tape

methods. It is not necessary to configure block size for the Optical Disc or WORM Disc methods because VSM determines the actual physical block size of optical volumes each time they are mounted or opened.

Granule Size: VSM divides files into granules. Each granule must fit on one volume. The granule size parameter specifies the number of bytes in each granule that VSM writes to the device. The allowable granule sizes are: 128K, 256K, 512K, 1M, 2M, 4M, 8M, 16M, 32M, and 64M. Granule size is a power of 2 and an integral multiple of block size.

Timeout Tab (Tape and Optical Media)

Figure 86. Timeout Stripe Properties Dialog

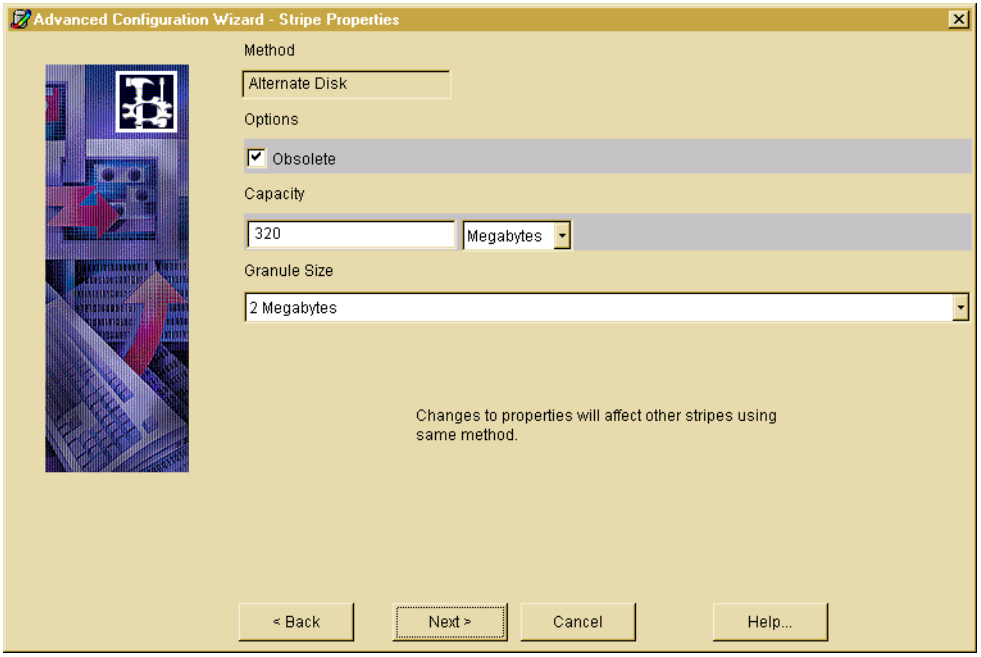


Deadman Timeout: Maximum time in seconds that VSM waits for an FTP or tape request to complete. The default is 3600 seconds (one hour). This is used only for the Tape, Optical Disc, WORM Disc, FTP, and NetBackup methods.

Demand Delay: Time in seconds a mount request waits before VSM unmounts a similar unused volume. If VSM identifies a mounted but unused volume of the same density whose unmount delay has not yet expired, it unmounts that volume as soon as the demand delay occurs. Otherwise, the mount request remains active until a drive becomes available. This is used only for the Tape, Optical Disc, and WORM Disc methods, and the default varies.

Unmount Delay: Time in seconds a volume that is mounted in read mode remains mounted pending another read request. If no read request arrives prior to the expiration of this time delay, VSM unmounts the volume. The default value is 3 minutes. A single unmount delay value pertains to all stripes in this configuration using the Tape, Optical Disc, or WORM Disc methods.

Figure 87. Alternate Disk Stripe Properties Dialog



Stripe Properties - for an Alternate Disk

Method: The name of the storage method.

- Alternate Disk: This alternate magnetic method is used in the two-step consolidation of tape or optical disc volumes, for migrating files to another file system mounted on the same managed server, and as a remote method when used with NFS.

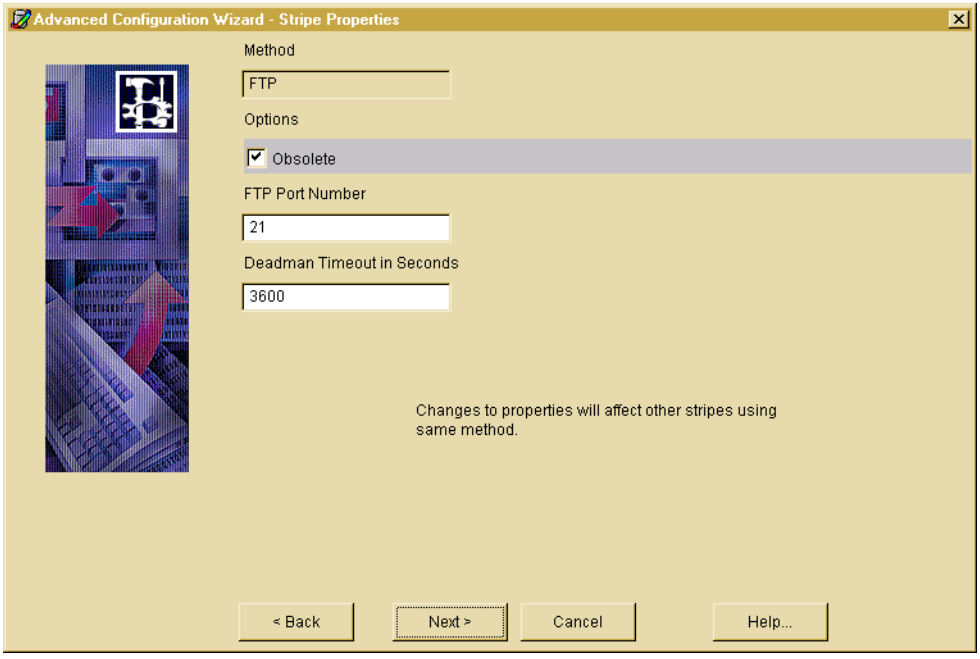
Options:

- Obsolete: Media supports granule obsoleting.

Capacity: Capacity of the method in bytes.

Granule Size: VSM divides files into granules. Each granule must fit on one volume. The granule size parameter specifies the number of bytes in each granule that VSM writes to the device. Granule size is a power of 2. The allowable granule sizes are: 128K, 256K, 512K, 1M, 2M, 4M, 8M, 16M, 32M, and 64M.

Figure 88. FTP Stripe Properties Dialog



Stripe Properties - for FTP

Method: The name of the storage method.

- FTP: Remote method using File Transfer Protocol.

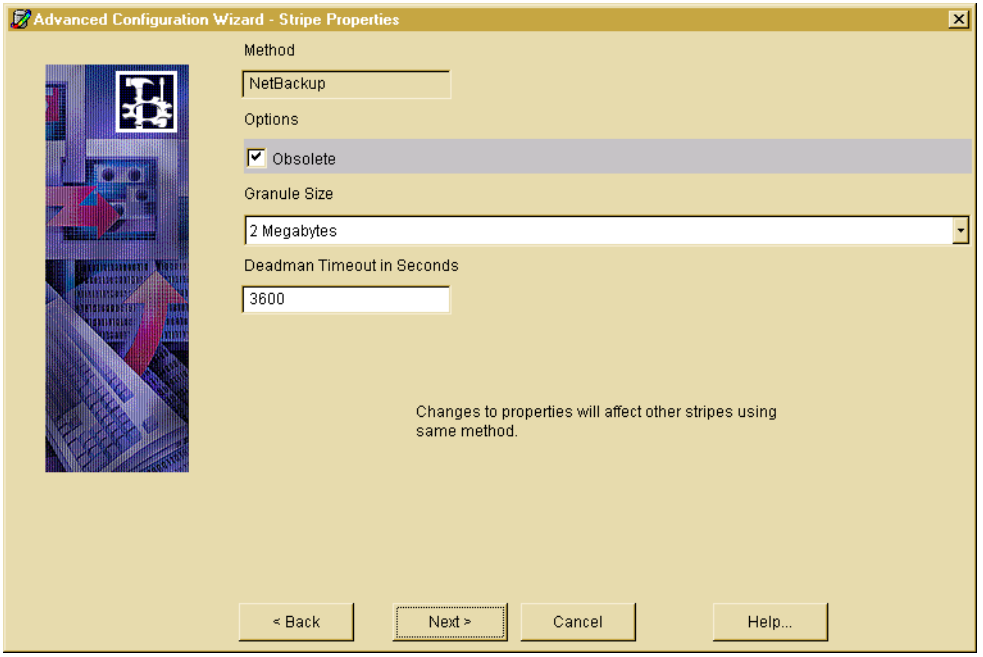
Options:

- Obsolete: Media supports granule obsoleting.

FTP Port Number: The default value is 21.

Deadman Timeout: Maximum time in seconds that VSM waits for an FTP request to complete. The default is one hour (3600 seconds).

Figure 89. NetbackUp Strip Properties Dialog



Stripe Properties - for NetBackup

Method: The name of the storage method.

- NetBackup: Remote method using NetBackup.

Options:

- Obsolete: Media supports granule obsoleting.

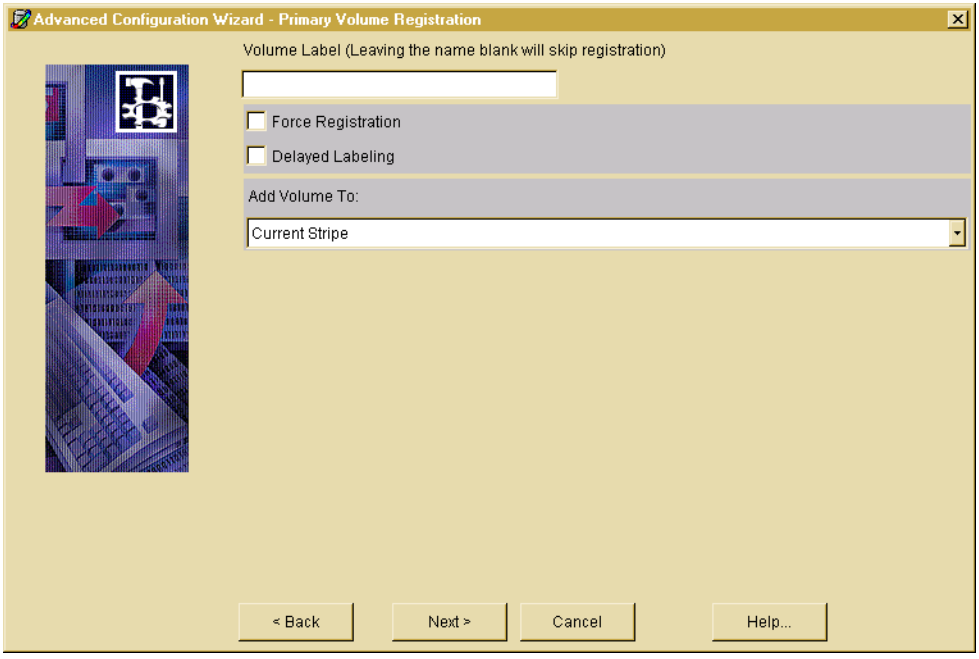
Granule Size: VSM divides files into granules. Each granule must fit on one volume. The granule size parameter specifies the number of bytes in each granule that VSM writes to the device. Granule size is a power of 2. The allowable granule sizes are: 128K, 256K, 512K, 1M, 2M, 4M, 8M, 16M, 32M, and 64M.

Deadman Timeout: Maximum time in seconds that VSM waits for an FTP or tape request to complete. The default is one hour (3600 seconds).

Advanced Wizard- Volume Registration (Primary or Alternate)

Use this screen to setup volume registration for the filesystem you wish to manage.

Figure 90. Primary Volume Registration Dialog - Tape Media



Volume registration information varies from one media type to another.

For more information, refer to one of the following sections:

- “Volume Properties (Primary or Alternate)- for Tape Media” on page 218
- “Volume Properties (Primary or Alternate)- for Optical Disc Media” on page 219
- “Volume Properties (Primary or Alternate)- for Alternate Disk” on page 220
- “Volume Properties (Primary or Alternate)- for FTP” on page 221
- “Volume Properties (Primary or Alternate)- for NetBackup” on page 222

Figure 91. Primary Volume Registration Dialog- Tape Media

Volume Properties (Primary or Alternate)- for Tape Media

Volume Label: The unique name of the volume to be recorded on the volume and in the VSM volume database VOLDB. VSM restricts volume names to an alphabetic character followed by up to five alphanumeric characters, and converts all lower case input to upper case.

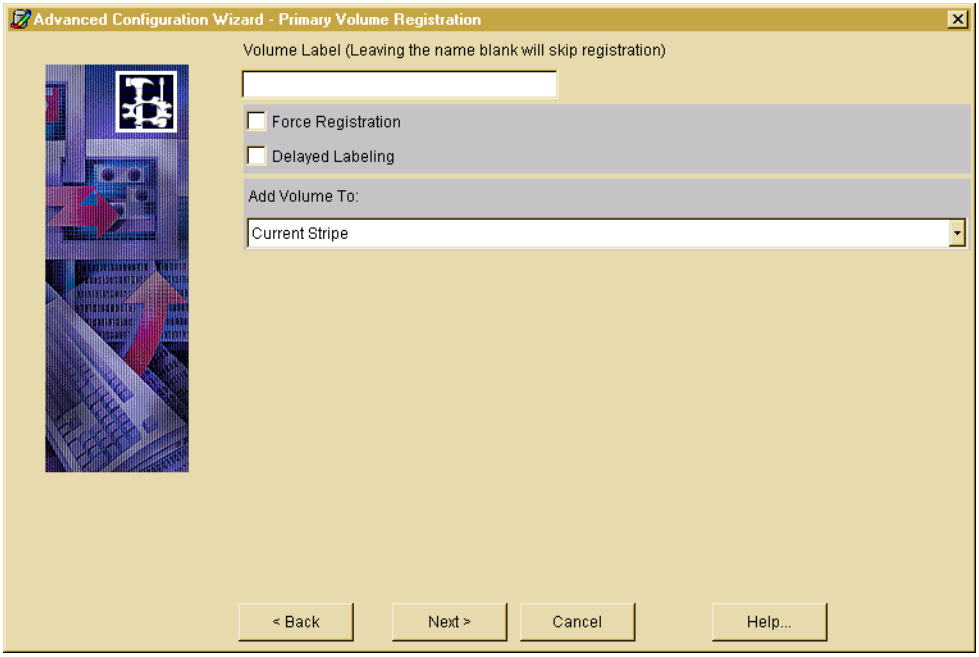
Force Registration: Check this box to force the registration of a previously labeled volume. If not checked, previously labeled volumes are not reregistered.

Caution This action forces the registration of a previously labeled volume and destroys all information that may be on the volume. The volume cannot currently be registered in any VSM volume database.

Delayed Labeling: Check this box to delay labeling of media until needed. If not checked, the media is labeled immediately.

Add Volume To: Select either Current Stripe or Volume Set 0.

Figure 92. Primary Volume Registration Dialog- Optical Disk Media



Volume Properties (Primary or Alternate)- for Optical Disc Media

Volume Label: The unique name of the volume to be recorded on the volume and in the VSM volume database VOLDB. VSM restricts volume names to an alphabetic character followed by up to five alphanumeric characters, and converts all lower case input to upper case.

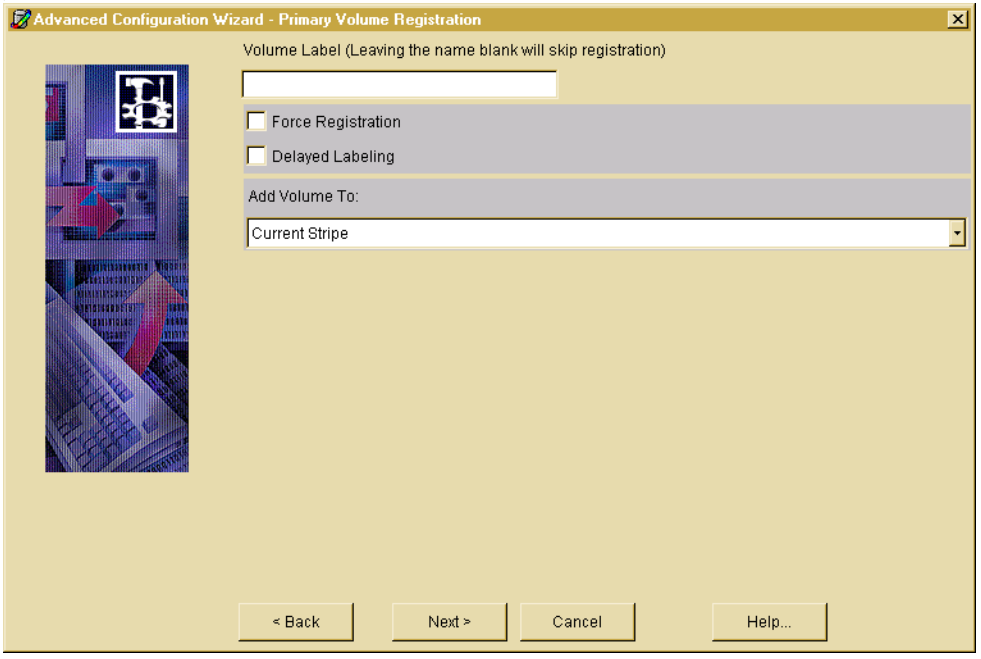
Force Registration: Check this box to force the registration of a previously labeled volume. If not checked, previously labeled volumes are not reregistered.

Caution This action forces the registration of a previously labeled volume and destroys all information that may be on the volume. The volume cannot currently be registered in any VSM volume database.

Delayed Labeling: Check this box to delay labeling of media until needed. If not checked, the media is labeled immediately.

Add Volume To: Select either Current Stripe or Volume Set 0.

Figure 93. Primary Volume Registration Dialog- Optical Disk Media



Volume Properties (Primary or Alternate)- for Alternate Disk

Volume Label: The unique name of the volume to be recorded on the volume and in the VSM volume database VOLDB. VSM restricts volume names to an alphabetic character followed by up to five alphanumeric characters, and converts all lower case input to upper case.

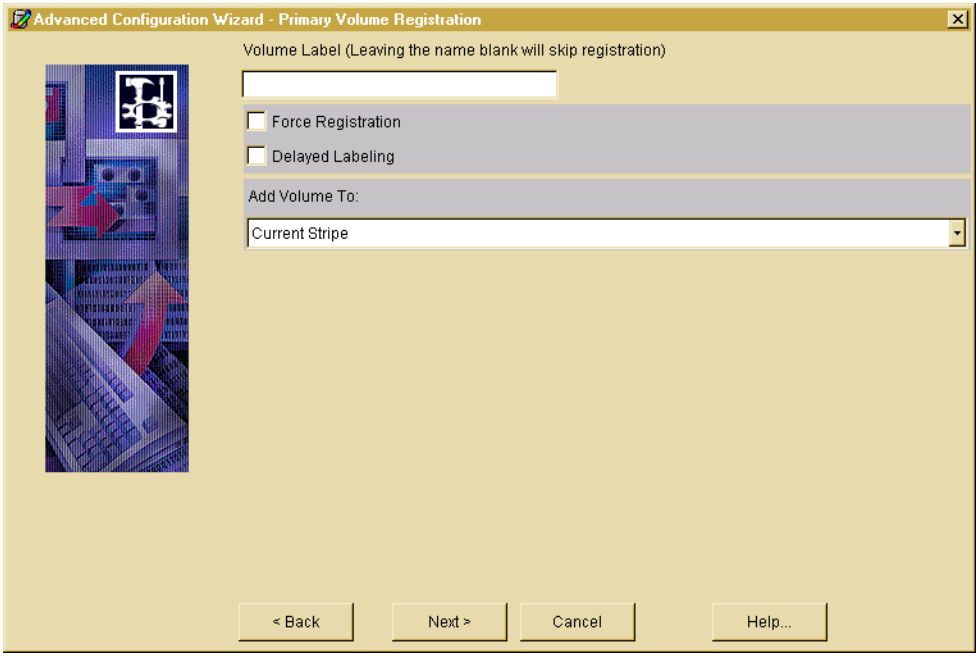
Directory: The device name or the filesystem mount point required when registering a volume. Make sure the specified device name or filesystem is mounted before registering the volume. Do not register a disk partition for a directory below the mount point because the entire capacity of the file system at the mount point is assumed.

Force Registration: Check this box to force the registration of a previously labeled volume.

Caution This action forces the registration of a previously labeled volume and destroys all information that may be on the volume. The volume cannot currently be registered in any VSM volume database.

Add Volume To: Select either Current Stripe or Volume Set 0.

Figure 94. Primary Volume Registration Dialog- FTP



Volume Properties (Primary or Alternate)- for FTP

Volume Label: The unique name of the volume to be recorded on the volume and in the VSM volume database VOLDB. VSM restricts volume names to an alphabetic character followed by up to five alphanumeric characters, and converts all lower case input to upper case.

Server: The name of the remote server. This can be the internet id or number of the server. VSM uses this name on the `ftp open` command as the host parameter. It must be a valid FTP host.

Directory: The full pathname of the file system directory or subdirectory on the remote server. The VSM username on the managed server must have read and write permissions to this remote directory. This can be any directory on the remote server that is not already registered for HSM.

Username and Password: The username and password VSM uses when accessing the remote server from the managed server through FTP. This name and password must be valid on the remote server and also must have read and write access to the remote file system that you are using for migration.

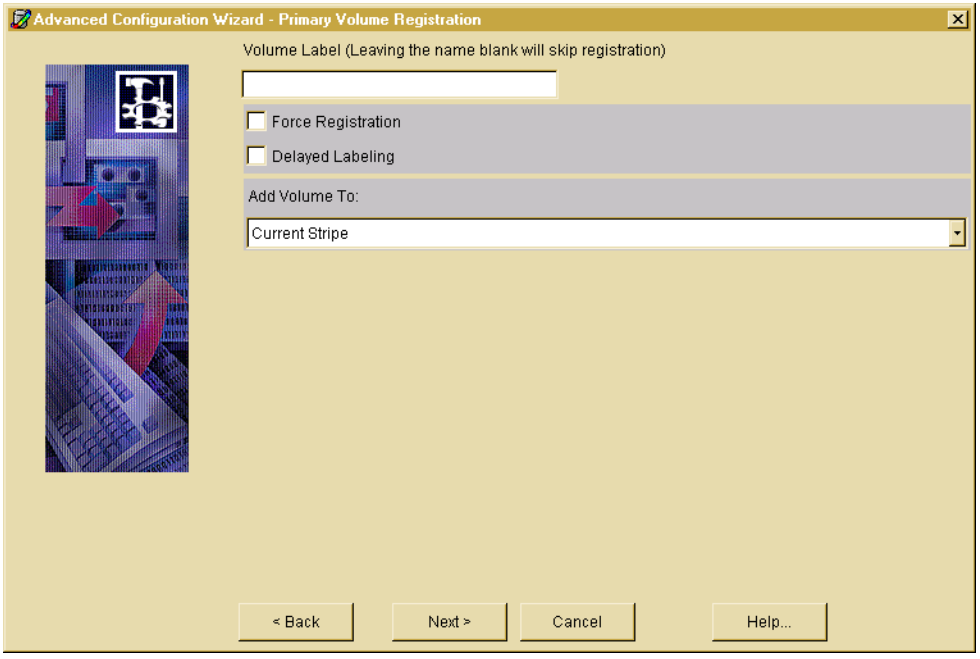
Keystrokes into the password field are echoed as asterisks. Since the password cannot be seen, it must be entered twice.

Force Registration: Check this box to force the registration of a previously labeled volume.

Caution This action forces the registration of a previously labeled volume and destroys all information that may be on the volume. The volume cannot currently be registered in any VSM volume database.

Add Volume To: Select either Current Stripe or Volume Set 0.

Figure 95. Primary Volume Registration Dialog- NetBackup



Volume Properties (Primary or Alternate)- for NetBackup

Volume Label: The unique name of the volume to be recorded on the volume and in the VSM volume database VOLDB. VSM restricts volume names to an alphabetic character followed by up to five alphanumeric characters, and converts all lower case input to upper case.

Server: The name of the NetBackup master server.

Class: The name of the NetBackup class to be registered as a NetBackup volume.

Schedule: The name of the schedule defined for the NetBackup class. The backup window for this schedule must be 24 hours per day, seven days per week.

Force Registration: Check this box to force the registration of a previously labeled volume.

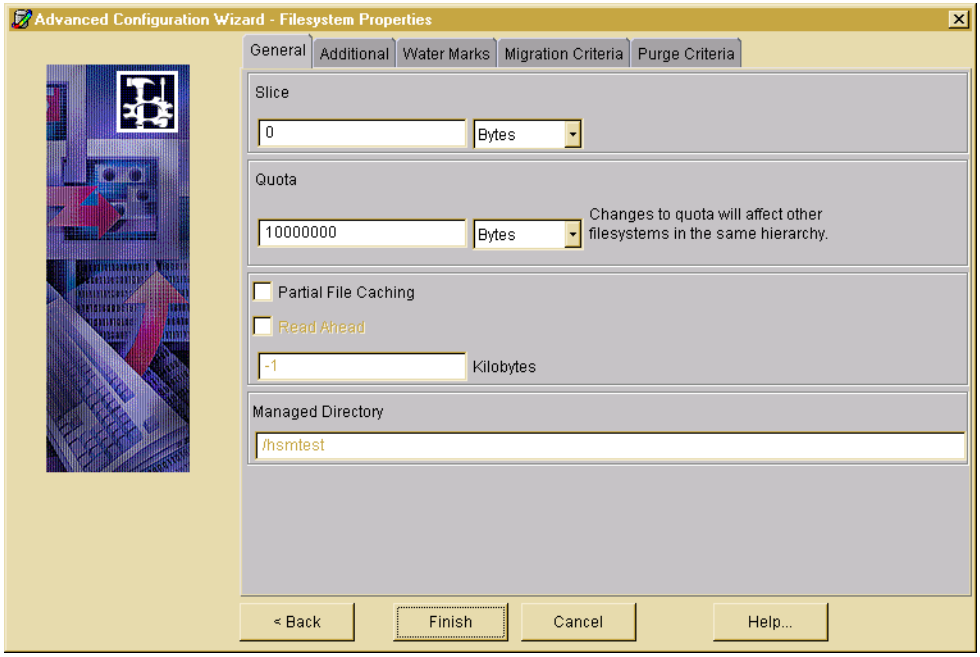
Caution This action forces the registration of a previously labeled volume and destroys all information that may be on the volume. The volume cannot currently be registered in any VSM volume database.

Add Volume To: Select either Current Stripe or Volume Set 0.

Advanced Wizard- Filesystem Properties

Use this screen to setup the filesystem properties for the filesystem you wish to manage. Managed file system properties are shown on five tabbed pages in this dialog. Click each tabbed page to reveal the properties.

Figure 96. Filesystem Properties Dialog



For more information, refer to one of the following sections:

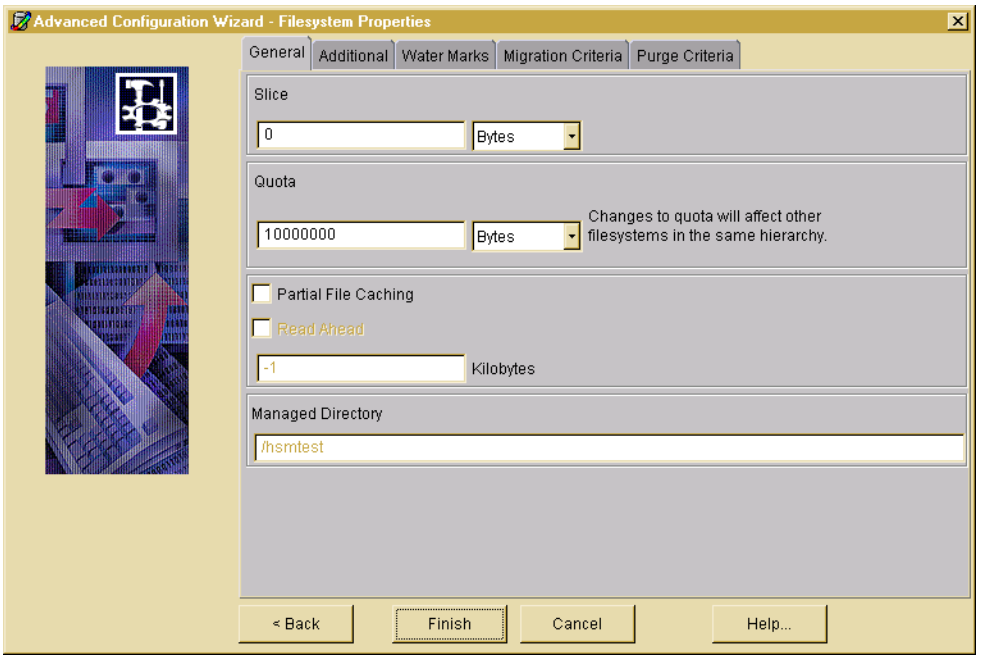
- “Filesystem Properties- General Tab” on page 224
- “Filesystem Properties Dialog- Additional Tab” on page 225
- “Filesystem Properties Dialog- Water Marks Tab” on page 226
- “Filesystem Properties Dialog- Migration Criteria Tab” on page 227
- “Filesystem Properties Dialog- Purge Criteria Tab” on page 228

Click the Finish button when done. The wizard will not commit completion of work until until you click Finish.

To change an existing configuration, use the Edit menu as described in “Manually Changing an Existing Configuration” on page 232.

Filesystem Properties- General Tab

Figure 97. Filesystem Properties Dialog- General Tab



Slice: Number of bytes at the front of the file that VSM leaves stored on disk for migrated files. These bytes are also migrated but VSM keeps a copy of them in the file system even when it migrates the associated file to secondary media, thus allowing this number of bytes to be read without caching the file. The value 0 implies that no bytes will be kept in the file system (default).

Quota: Maximum number of bytes that each user can restrict from migration. The default is 10 Mbytes (10000000). A single quota value pertains to all managed file systems in this configuration.

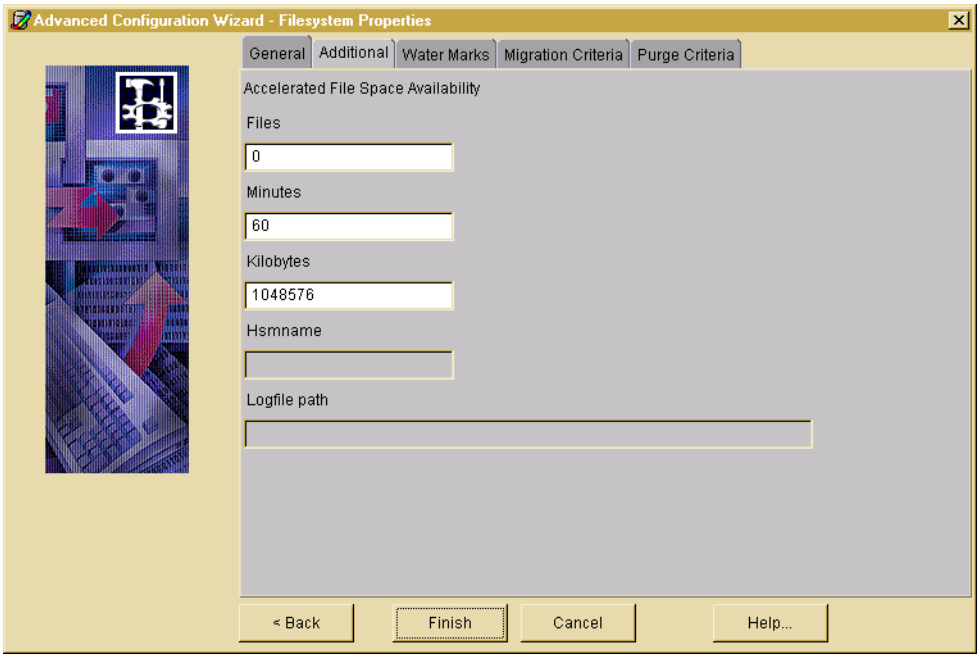
Partial File Caching: A process which allows `read` access to a migrated file without caching the entire file. See Read Ahead.

Read Ahead: Minimum number of kilobytes beyond the current read request that VSM partially caches to disk. Applies only to DMAPI implementations of HSM. Setting the value to -1 disables partial file caching (default).

Manage Subdirectory: Manage a subdirectory in a mounted file system instead of managing the entire file system. Type the path to the managed directory in the field provided. This expression must not contain a trailing slash.

Filesystem Properties Dialog- Additional Tab

Figure 98. Filesystem Properties Dialog- Additional Tab



Accelerated File Space Availability: Make file space available sooner by selecting, migrating, and purging files incrementally.

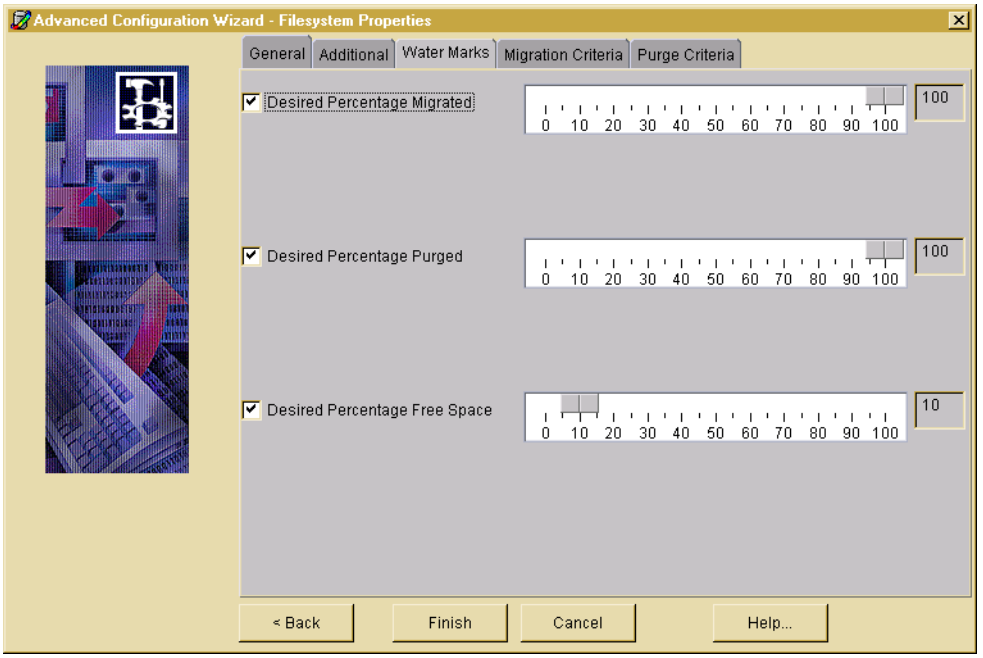
Files: Maximum number of files processed with Accelerated File Space Availability enabled before users can resume accessing available free space. A value of 0 signifies no limit (default).

Minutes: Maximum time increment with Accelerated File Space Availability enabled before users can resume accessing available free space. The default is 60. A value of 0 signifies no limit.

Kilobytes: Minimum amount of disk space freed with Accelerated File Space Availability enabled before users can resume accessing available free space. The default is 1,048,576. A value of 0 signifies no limit.

Filesystem Properties Dialog- Water Marks Tab

Figure 99. Filesystem Properties Dialog- Water Marks Tab



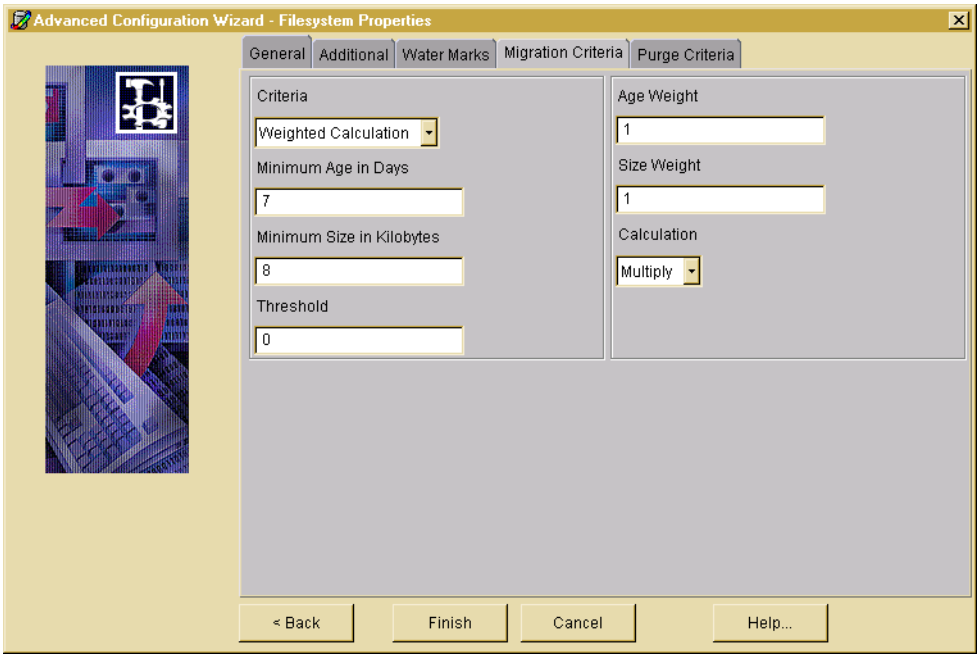
Desired Percentage Migrated: (Sometimes called low-water mark.) Percentage of free disk space at which VSM stops selecting files for migration. The default is 100, which is interpreted as no low-water mark. If Desired Percentage Migrated is not selected, the slider is not shown.

Desired Percentage Purged: (Sometimes called purge mark.) Percentage of free disk space at which VSM stops purging premigrated files. The default is 100, which is interpreted as no purge mark. Purge mark must be equal to or between high-water mark and low-water mark. If Desired Percentage Purged is not selected, the slider is not shown.

Desired Percentage Free Space: (Sometimes called high-water mark.) Minimum percentage of free disk space at which VSM initiates migration operations. The default value is 10. Desired Percentage Free Space must be selected and configured.

Filesystem Properties Dialog- Migration Criteria Tab

Figure 100. Filesystem Properties Dialog- Migration Criteria Tab



Criteria:

- Default: Select files to migrate based on the default weighted algorithm which factors both file size and file age.
- Large Files: Select larger files to migrate first, regardless of age.
- Old Files: Select older files to migrate first, regardless of size.
- Weighted Calculation: Select files to migrate based on a modified weighted algorithm which factors both file size and file age.

  threshold = (age)(age weight) [x or +] (size)(size weight)

  where age is in days and size is in kilobytes.
- Site-defined Script: Select files to migrate based on an algorithm specified by the system administrator.

Minimum Age: Do not migrate files accessed or modified within this time. Set this to a value greater than 0 to prevent files from migrating the same day they are created. Default is 7 days.

Minimum Size: Do not migrate files smaller than this. Default is 8 kilobytes.

Threshold: Select files to migrate if the weighted algorithm output is equal to or greater than this threshold. Default is 0. The term *Threshold* in this context is equivalent to the term *Badness* as defined on page 136.

Age Weight: Weighting factor for file age used in the algorithm. Default is 1.

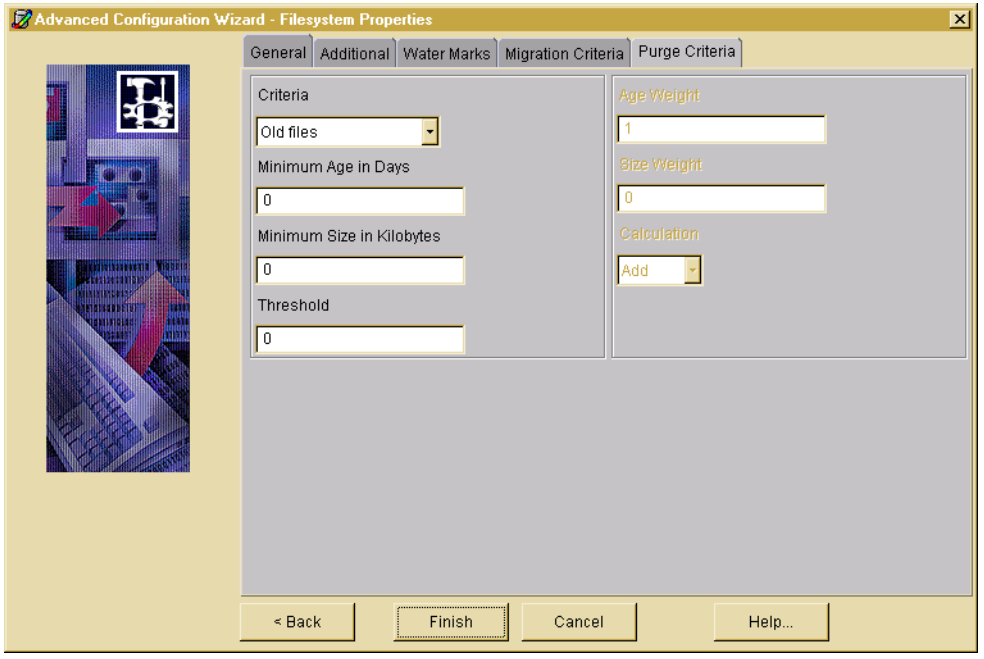
Size Weight: Weighting factor for file size used in the algorithm. Default is 1.

Calculation:

- Multiply or Add: Arithmetic operator between file size and file age in the weighted algorithm. Default is Multiply.
- Site-Defined: In lieu of using the weighted algorithm, define any type of program or script that echoes a true / false migration flag to standard output for each file checked. VSM will migrate the file if output is 0, and will not migrate the file if output is anything else. The same site-defined algorithm applies to both migrating and purging files.

Filesystem Properties Dialog- Purge Criteria Tab

Figure 101. Filesystem Properties Dialog- Purge Criteria Tab



Criteria:

- Default: Select files to purge based on the default weighted algorithm which factors both file size and file age.
- Large Files: Select larger files to purge first, regardless of age.
- Old Files: Select older files to purge first, regardless of size.
- Weighted Calculation: Select files to purge based on a modified weighted algorithm which factors both file size and file age.

  Threshold = (age)(age weight) [x or +] (size)(size weight)

  where age is in days and size is in kilobytes.
- Site-defined Script: Select files to purge based on an algorithm specified by the system administrator.

Minimum Age: Do not purge files processed within this time. Set this to a value greater than 0 to prevent files from purging the same day they are processed. Default is 0 days.

Note For kernel-based implementations (Solaris *ufs* file systems), minimum age is in days since migrated. For nonkernel-based implementations, minimum age is in days since either accessed or modified, whichever is less.

Minimum Size: Do not purge files smaller than this. Default is 0 kilobytes.

Threshold: Select files to purge if the weighted algorithm output is equal to or greater than this threshold. Default is 0. The term *Threshold* in this context is equivalent to the term *Purge Badness* as defined on page 137.

Age Weight: Weighting factor for file size used in the purge algorithm. Default is 1.

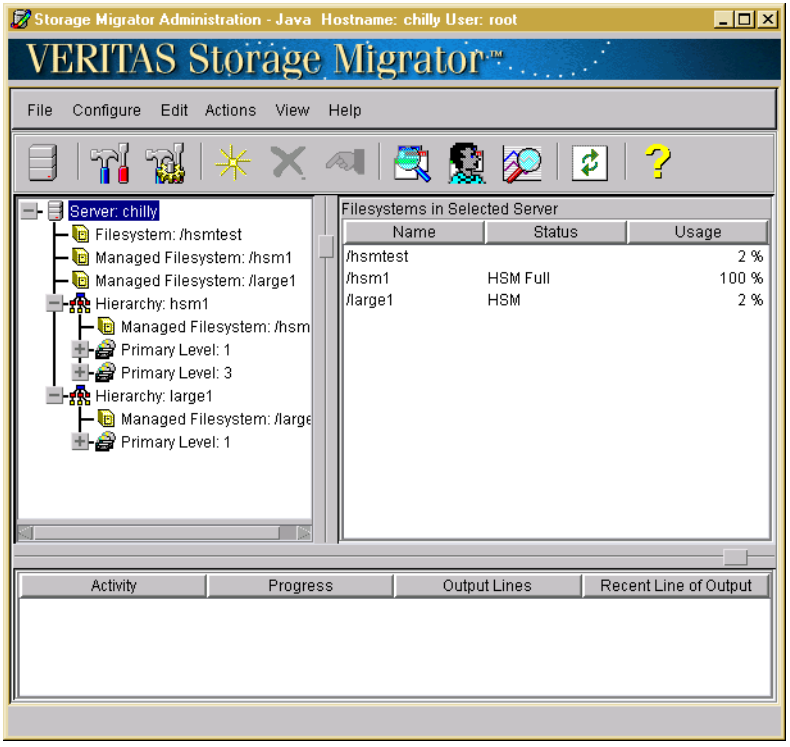
Size Weight: Weighting factor for file size used in the purge algorithm. Default is 0.

Calculation:

- Multiply or Add: Arithmetic operator between file size and file age in the weighted purge algorithm. Default is Add.
- Site-Defined: In lieu of using the weighted algorithm, define any type of program or script that echoes a true/false migration flag to standard output for each file checked. VSM will purge the file if output is 0, and will not purge the file if output is anything else. The same site-defined algorithm applies to both migrating and purging files.

Hierarchy Structure of a Managed File System

Figure 102. Managed File System Structure



When configuration is complete, the hierarchy for the managed file system is displayed in the left panel of the main screen (see illustration).

Manual Configuration Procedure

Manual Initial Configuration

Follow these steps to configure hierarchies and stripes manually:

1. Defining hierarchies.
   a. Highlight the Server node in the tree panel on the left.
   b. Select New Hierarchy from the Edit pull-down menu, or click the New tool.

This brings up the Advanced Initial Configuration Wizard and steps you through the following actions:

- Define the hierarchy.
- Define storage levels for the hierarchy.

  Define at least one Primary level for the selected hierarchy. If you intend to activate dual migration copies, also define at least one Alternate level. If you intend to use the multilevel migration feature of HSM, define more than one Primary or Alternate level for the selected hierarchy. When a storage level is defined it appears on the tree display as a child of the hierarchy.
- Assign stripes for each storage level in the hierarchy.

  Assign one or more recording methods (stripes) to the selected storage level. If you assign multiple stripes to a level, VSM will migrate files in a round-robin manner, using multiple storage devices simultaneously for faster performance. When a stripe is assigned it appears on the tree display as a child of the storage Level.

Note This variation of the configuration wizard does not include volume registration.

When the hierarchy is created, it appears on the tree display as a child under the Server node. If you want to manage more than one file system, you may either define a separate hierarchy for each one or assign multiple file systems to a single hierarchy.

2. Assign managed file systems to a hierarchy.
   a. Highlight the Filesystem node to be managed in the tree panel on the left.
   b. Select New HSM Management from the Edit pull-down menu, or click the New tool.

   Choose the hierarchy from the pop-up menu. When the file system is assigned, it appears on the tree display as a Managed Filesystem, both as a child of the Server node and as a child of the Hierarchy node. Repeat for each file system to be managed.

   You can also use the Edit pull-down menu or Properties tool to change managed file system properties. An advisory label appears if changing a property on one file system might affect other managed file systems in the same hierarchy.

3. Add volumes for VSM use.
   a. Highlight a stripe node in the tree panel on the left.
   b. Select New Volume from the Edit pull-down menu, or click the New tool.

   c. Add volumes.

d. Specify volume label to register the volumes (optional).

e. Repeat for each Alternate Disk, FTP, or NetBackup stripe assigned in Step 1.

Manually Changing an Existing Configuration

Use the Edit pull-down menu to customize an existing configuration after using an Initial Configuration Wizard. See “Edit Menu” on page 185.

Manually Change Hierarchy Properties

1. Highlight a Hierarchy node in the tree panel on the left.
2. Select Change Hierarchy Properties... from the Edit pull-down menu, or click the Properties tool.

Manually Change Managed Filesystem Properties

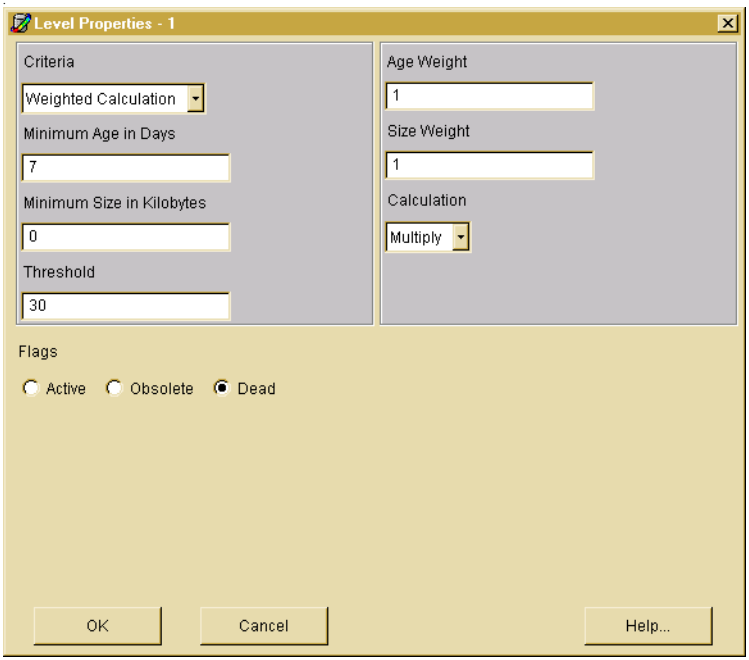
1. Highlight a Managed Filesystem node in the tree panel on the left.
2. Select Change Filesystem Properties... from the Edit pull-down menu, or click the Properties tool.

   See “Advanced Wizard- Filesystem Properties” on page 223 for more information.

Manually Change Level Properties

1. Highlight a Level node in the tree panel on the left.
2. Select Change Level Properties... from the Edit pull-down menu, or click the Properties tool.

Figure 103. Level Properties Dialog- Move Criteria



Criteria:

- Default: Select files to move based on the default weighted algorithm which factors both file size and file age.
- Large Files: Select larger files to move first, regardless of age.
- Old Files: Select older files to move first, regardless of size.
- Weighted Calculation: Select files to move based on a modified weighted algorithm which factors both file size and file age.

  Threshold = (age)(age weight) [x or +] (size)(size weight)

  where age is in days and size is in kilobytes.
- Site-defined Script: Select files to move based on an algorithm specified by the system administrator.

Minimum Age: Do not move files migrated or moved to this level within this time, or accessed within this time, whichever is most recent. Set this to a value greater than 0 to prevent files from moving the same day they are migrated. Default is 7 days.

Minimum Size: Do not move files smaller than this to a different level. Default is 0 kilobytes.

Threshold: Select files to move if the weighted move algorithm output is equal to or greater than this threshold. Default is 30. The term *Threshold* in this context is equivalent to the term *Move Badness* as defined on page 144.

Age Weight: Weighting factor for file age used in the move algorithm. Default is 1.

Size Weight: Weighting factor for file size used in the move algorithm. Default is 1.

Calculation:

- Multiply or Add: Arithmetic operator between file size and file age in the weighted move algorithm. Default is Multiply.
- Site-Defined: In lieu of using the weighted algorithm, define any type of program or script that echoes a true / false migration flag to standard output for each file checked. VSM will move the file if output is 0, and will not move the file if output is anything else.

Flags:

- Dead: Mark FHDB entries for file copies at the source migration level Dead.
- Active: Mark FHDB entries for file copies at the source migration level Active.
- Obsolete: Mark FHDB entries for file copies at the source migration level Obsolete.

Default is Dead for the Tape, Optical Disc, and WORM Disc methods. Default is Obsolete for the Alternate Disk method. Move flags are not applicable to the FTP and NetBackup methods.

Manually Change Stripe Properties

1. Highlight a Stripe node in the tree panel on the left.
2. Select Change Stripe Properties... from the Edit pull-down menu, or click the Properties tool.

See “Advanced Wizard- Stripe Properties” on page 211 for more information.

The VSM Activity Monitor 5

This chapter describes the job activity monitor, VSM Activity Monitor. This feature allows administrators and users to view VSM jobs in progress, observe and control the status for jobs, and view a job activity log.

The following VSM jobs are tracked by the VSM Activity Monitor:

`migadscan` - Provide information on contents of archive disk volumes

`migbatch` - Premigrate files and copy to secondary storage

`migcons` - Consolidate VSM tape and optical volumes

`migconsweep` - Enable constant file system sweeping

`migftscan` - Scan an ft remote volume and reconstruct database entries

`migin` - Cache a file from secondary storage to disk

`miglow` - Start `mignospace` if file system is above high-water mark

`migmdclean` - Remove obsolete entries from media

`migmove` - Move migrated files from one migration level to another level

`mignbexport` - Export VSM files

`mignbimport` - Import VSM files

`mignbscan` - Scan a NetBackup volume and reconstruct database entries

`mignewlog` - Copy or delete global or individual VSM log files

`mignospace` - Purge or migrate files to make disk space available

`migopscan` - Provide information on contents of optical volumes

`migpurge` - Purge a specific file or files

`migrate` - Premigrate a specific file or files

`migrc` - Clear locks, remove defunct lock files, restart migrations

`migreconstruct` - Reconstruct damaged or deleted migrated files

`migrecycle` - Reregister an empty volume

`migstage` - Pre-cache files to avoid caching delays

`migtscan` - Provide information on contents of tape volumes

Refer to the respective man pages for additional information about these commands.

VSM Activity Monitor displays all currently active VSM jobs along with some of their parameters, such as time started and job owner. It also allows administrators to kill selected jobs. Users without root privilege can kill their own jobs, but not those of other users.

VSM Activity Monitor tracks jobs initiated from either an administrative interface, (VSM-Java or `xhsmadm`), the Java-based VSM File Browser, or a command line.

The following topics in this chapter explain how to view VSM jobs in more detail:

- Launching the VSM Activity Monitor
- Main Screen Layout
- Using the VSM Activity Monitor

Launching the VSM Activity Monitor

VSM Activity Monitor is invoked from the administrative interface (VSM-Java) or as a stand-alone program.

Starting the VSM Activity Monitor

You can start the VSM Activity Monitor interface in two ways:

- from the Administrative Interface
- as a stand-alone program

Administrative Interface

To launch the VSM Activity Monitor from the administrative interface, VSM-Java, you have two options:

- Click the Activity Monitor tool on the toolbar.
- Or, select Actions and Activity Monitor from the main menu.

Stand-Alone Program

You can launch the VSM Activity Monitor as a stand-alone program from either a UNIX or Windows NT environment.

- To launch Activity Monitor from a *UNIX* environment (Solaris or HP-UX), do the following:

1. Login to the host you are using as a management station.
2. Define the *DISPLAY* environment variable for your X11 screen.
3. Execute the command: `/usr/openv/java/migam`

- To launch Activity Monitor from a *Windows NT* environment (Windows NT or Windows 2000), do the following:

1. Click Start on the taskbar of your Windows desktop.
2. Point to Programs within the Start menu.
3. Point to VERITAS Storage Migrator in the Programs submenu.
4. Point to the Unix Activity Monitor within the VERITAS Storage Migrator submenu.

Login to the VSM Activity Monitor

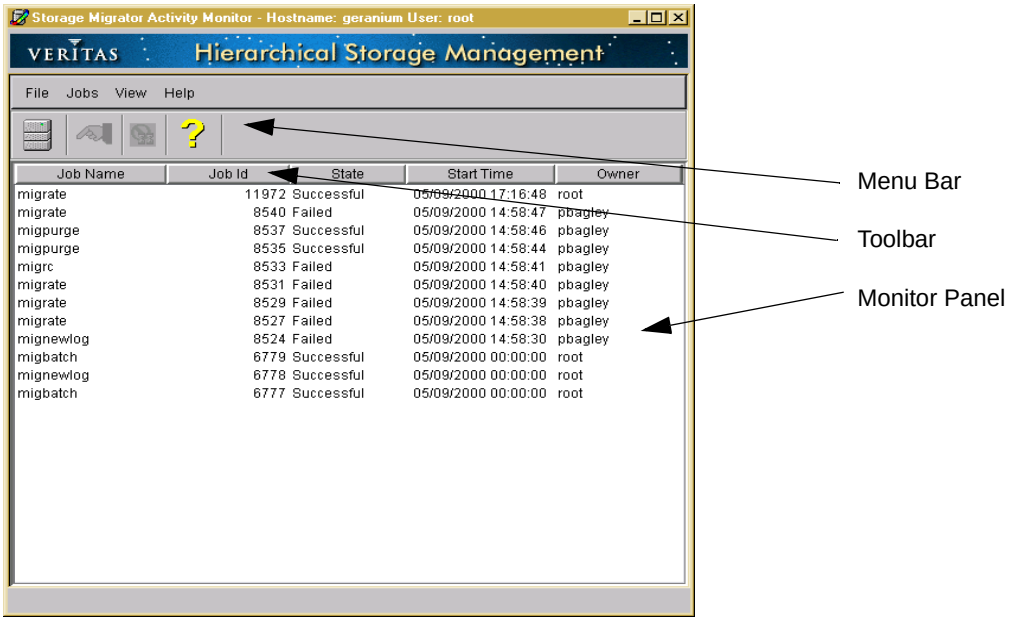
After you start the VSM Activity Monitor, you will see a login window. To login, do the following:

1. Type in the following fields on the login window:
   - Host name: The hostname of the server on which VSM is installed
   - User: Either root or another username
   - Password: The password
2. To cancel the login, press <*Esc*>.
3. To login, Click the Login button or press <*CR*>.

Main Screen Layout

The main screen layout of the VSM Activity Monitor consists of three elements.

Figure 104. VSM Activity Monitor Main Window



For further information, refer to the following sections.

- “Menu Bar” on page 238
- “Toolbar” on page 240
- “Monitor Panel” on page 241

Menu Bar

The menu bar at the top of the main screen offers four pull-down menus.

For further information, refer to the following sections.

- “File” on page 239
- “Jobs” on page 239
- “View” on page 239
- “Help” on page 239

File

Use the File pull-down menu to change servers or to exit the VSM Activity Monitor.

- To change servers, select Change Server... from the File pull-down menu, or click the Change Server tool. When you do this, you will see a login dialog. For details on how to login, see “Login to the VSM Activity Monitor” on page 237.
- To exit the VSM Activity Monitor, select Exit from the File pull-down menu. This terminates this instance of the VSM Activity Monitor.

Jobs

Use the Jobs pull-down menu to view details or kill a selected job displayed in the Activity Monitor.

- To view the details for a job, highlight one job from the monitor panel. Then, select Details for Selected Job from the Jobs pull-down menu.

Note If you select multiple jobs, this feature is disabled.

- To end (kill) one or more jobs, highlight one or more jobs from the monitor panel. Then, select Kill Selected Job(s) from the Jobs pull-down menu.

View

Use the View pull-down menu to change the toolbar settings for VSM Activity Monitor.

- To change toolbar settings, select Preferences... from the View pull-down menu. See “Using the Preferences Dialog” on page 247 for more information.

Help

Use the Help pull-down menu to find out more information about this product.

- To bring up the HELP window, select Help topics... from the Help pull-down menu. See “Using Help” on page 247 for more information.
- To display the About box, select About Job Activity Monitor... from the Help pull-down menu. See “Using Help” on page 247 for more information.

Toolbar

The toolbar at the top of the main screen offers command buttons, which allow you to quickly complete several common VSM Activity Monitor tasks.

For further information, refer to the following sections.

- "Job Details" on page 240
- "Kill Jobs" on page 240
- "Change Server" on page 240
- "Help" on page 240

Job Details

- Use the Job Details command button to display information on a job selected in the monitor panel.
- To view the details for a specific job (or jobs), select the job (or jobs) you want to view, then click the Job Details command button.
- See "Displaying Job Details for a VSM Job" on page 243 or "Using the Job Details Dialog" on page 245 for complete information on this topic.

Kill Jobs

- Use the Kill Job button to kill any job (or jobs) selected in the monitor panel.
- To kill a specific job (or jobs), select the job (or jobs) you want to kill, then click the Kill Jobs button.
- See "Killing VSM Jobs" on page 244 for complete information on this topic.

Change Server

Use the Change Server button to change which server VSM Activity Monitor displays.

- To change to another server, click the Change Server button and login to the server you want to view.

Help

Use the Help button to find out more information about this product.

- To bring up the HELP window, click the Help button.
- See "Using Help" on page 247 for complete information on this topic.

Monitor Panel

The monitor panel displays certain basic parameters about each active VSM job. Any job listed in the monitor panel displays information in several columns, which are described below.

Job Name

This column displays the name of the job assigned by `migrd`.

Job Id

This column displays the job identifier assigned by `migrd`.

State

This column displays the current state of the job: active, successful, or failed. *Active* denotes a job that is still running on VSM. *Successful* denotes a job that successfully finished running on VSM. *Terminated* denotes a job that failed running on VSM.

Start Time

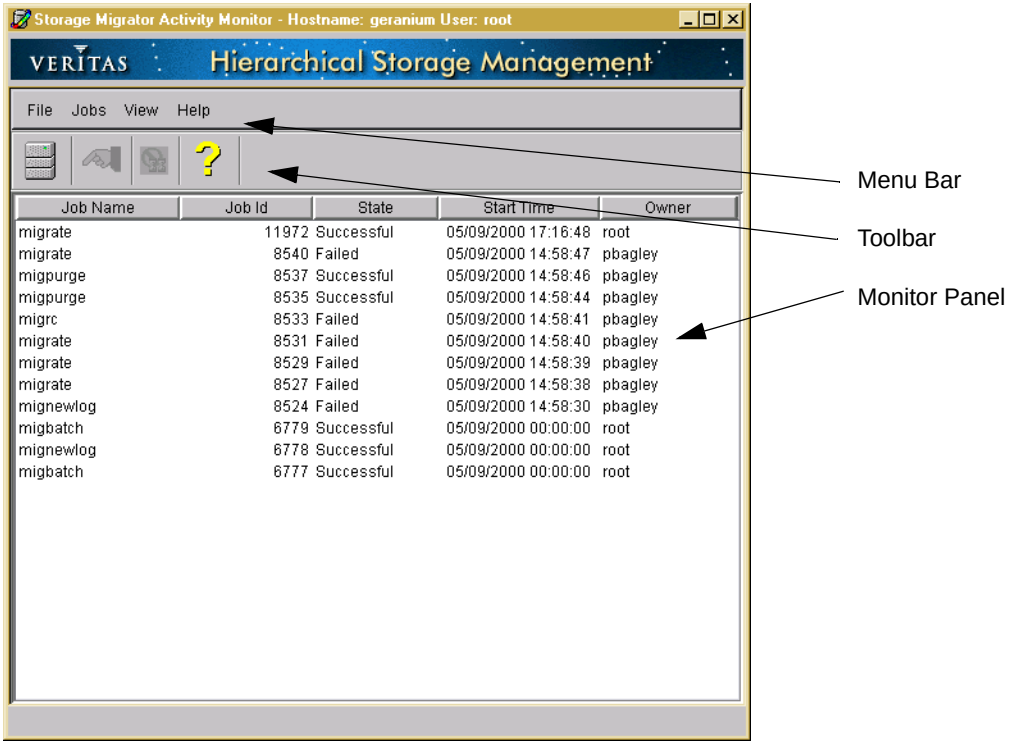
This column displays the time the job started, in the format MM/DD/YYYY hh:mm:ss AM/PM

Owner

This column displays the user name of the person who owns the job.

Using the VSM Activity Monitor

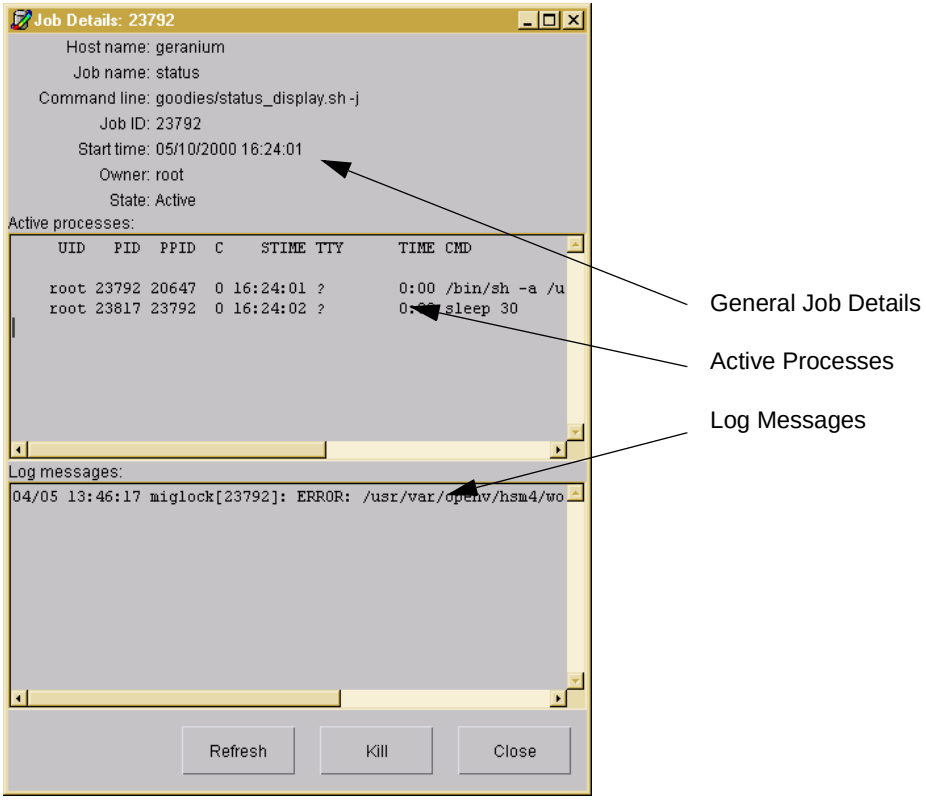
Figure 105. VSM Activity Monitor Main Window



The VSM Activity Monitor main window lists jobs and details for each job. From here, you can perform a number of commands, including viewing the specific job details in a separate Job Detail dialog.

- display job details (see “Displaying Job Details for a VSM Job” on page 243 for complete details).
- kill jobs (see “Killing VSM Jobs” on page 244 for complete details).
- getting help (see “Using Help” on page 247 for complete details).

Figure 106. Job Detail Dialog



Displaying Job Details for a VSM Job

You have several options available to you for displaying job details:

Note For complete information on this, see "Using the Job Details Dialog" on page 245 for more information.

◆ To view job details with the monitor panel:

1. Select a job from the monitor panel on the main window, if you want to view job details on that specific job. You can only select one job at a time.
2. Select Job details... from the Jobs pull-down menu. The Job Details dialog appears.
3. To see only *active* jobs in Active Processes in the Job Details dialog, click the Refresh button.

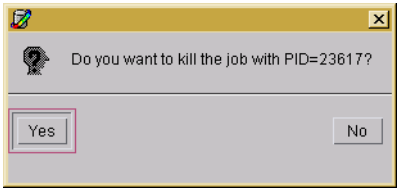
4. To kill the job, click the Kill button.
5. To close the Job Details dialog, click the Close button.

Note For complete information on this, see "Using the Job Details Dialog" on page 245 for more information.

- View job details directly from the Job Details dialog:

1. Double-click on a job in Active Processes to immediately view the Job Details dialog.
2. To update (refresh) the active jobs shown in Active Processes box in the Job Details dialog, either wait 5 seconds for VSM to automatically do this or click the Refresh button to do this more quickly.
3. To kill the job, click the Kill button.
4. To close the Job Details dialog, click the Close button.

Figure 107. Kill Job Confirmation Dialog



Killing VSM Jobs

It is possible to kill active jobs either from the main window or from the Job Details dialog.

Killing Jobs from the Main Window

- To kill a job from the main window:

1. On the main window, select one or more jobs to kill from the monitor panel.
2. Select Kill job from the Jobs pull-down menu. You will see a confirmation window for each job you kill.
3. Click Y on each confirmation window to confirm killing the job.
   Or, click N on each confirmation window to cancel killing the job. (See Figure 107.)

Note Users without root privilege can kill only their own jobs.

Killing Jobs from Job Details Dialog

Follow one of the following processes to kill jobs.

- To kill a job with the Kill button:

1. Select one or more jobs from the monitor panel on the main window.
2. To kill the job, click the Kill button. You will see a confirmation window for each job you kill.
3. Select Kill job from the Jobs pull-down menu. You will see a confirmation window for each job you kill.
4. Click Y on each confirmation window to confirm killing the job.
   Or, click N on each confirmation window to cancel killing the job. (See Figure 107 on page 244.)
5. To kill other jobs, repeat the above steps for each job.

- To kill a job directly from the Job Details dialog:

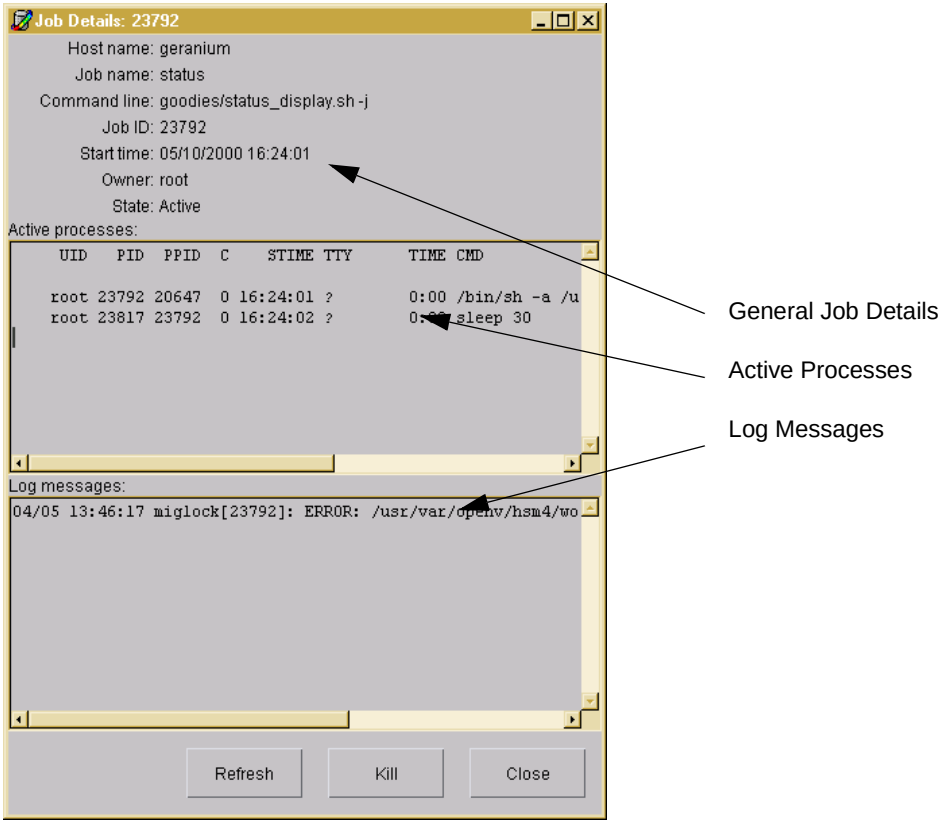
1. Double-click on an active job in Active Processes (in the main screen) to immediately view the Job Details dialog.
2. To kill the job, click the Kill button. You will see a confirmation window for each job you kill.
3. Click Y on each confirmation window to confirm killing the job.
   Or, click N on each confirmation window to cancel killing the job. (See Figure 107 on page 244.)
4. To kill other jobs, repeat the above steps for each job.

Using the Job Details Dialog

The Job Details dialog repeats all of the job parameters displayed in the VSM Activity Monitor main window, plus the Host for this VSM managed file system and the command line input that initiated the job. The Job Details dialog also displays job processes and log messages for the job. If you want to simultaneously view job details for more than one job, you must open a separate Job Details dialog for each job.

Note If you select multiple jobs, this feature is disabled. You can only select one job to view at a time.

Figure 108. Job Detail Dialog



General Job Details

The General Job Details box, in the Job Details dialog, shows general parameters for the jobs that are displayed in this window.

Active Processes

The Active Processes box, in the Job Details dialog, shows the main processes for the selected job (parent job) and the processes this job initiates (child processes). The information within this screen is updated and refreshed every five seconds.

Log Messages

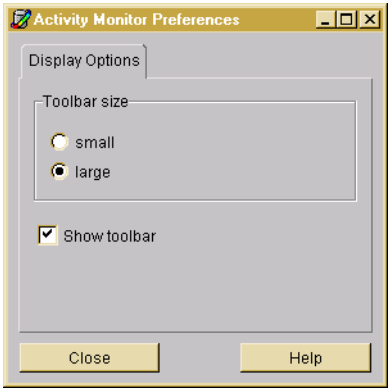
The Log Messages box, in the Job Details dialog, displays a dynamic list of VSM log messages created by any processes running on HSM. These log messages are displayed in the order they are run; new messages are added to the bottom of this list.

Note This screen is not refreshed when you click on the Refresh button.

Using the Preferences Dialog

The Preferences Dialog allows you to change your toolbar display options.

Figure 109. Preferences Dialog



Display Options- Toolbar Size

The Toolbar Size box of the Preferences dialog shows all the toolbar display sizes available for the VSM Activity Monitor. Select the appropriate radio button for the setting you prefer.

Display Options- Show Toolbar

The Show Toolbar box of the Preferences dialog shows whether you want the VSM Activity Monitor toolbar to display on the main screen. Select the appropriate setting you prefer.

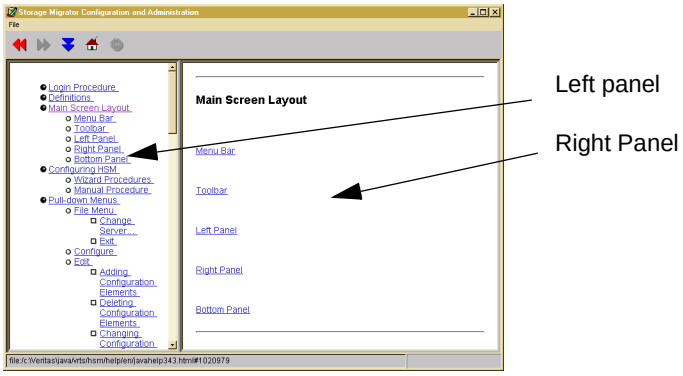
Using Help

The Help pull-down menu allows you to get more information on VSM Activity Monitor.

Help Topics...

The Help Topics... submenu provides you with additional instructions on how to do specific tasks in VSM Activity Monitor.

Figure 110. Help Dialog



- To view help with the Help button:

1. Click on the Help button on the toolbar.
2. Double-click on the topic that interests you in the left panel of the Help dialog. This topic will then display in the right panel of the Help dialog.

- To view help with the Help pull-down menu:

1. Select Help Topics... from the Help pull-down menu.
2. Double-click on the topic that interests you in the left panel of the Help dialog. This topic will then display in the right panel of the Help dialog.

About Storage Migrator

The About Storage Migrator, under the Help pull-down menu, provides you with software version and copyright information. See Figure 111, "About Storage Migrator Dialog," below.

Figure 111. About Storage Migrator Dialog

The VSM File Browser 6

This chapter explains how to execute the most common VSM user commands using the Java-based interface, VSM File Browser, instead of using a command line interface. These VSM Operations are described in detail in Chapter 2.

The VSM File Browser is a Java application which allows users without root privilege to perform a variety of operational commands.

The following topics in this chapter explain these user-directed VSM operations in more detail:

- Migrating Files
- Migrating Directories (Grouping Directory)
- Purging Files
- Accessing Migrated Files
- Accessing Migrated Files
  - Identifying Migrated Files
  - Pre-caching Files
  - Caching a Group of Associated Files

Note VSM File Browser is fully documented by its online help system.

Installation

To use VERITAS Storage Migrator or VERITAS Storage Migrator Remote, the system administrator must first install the software by following the instructions in the *VERITAS Storage Migrator for UNIX Installation Guide*. This includes VSM File Browser as well as the Java-based administrator's interface, VSM-Java.

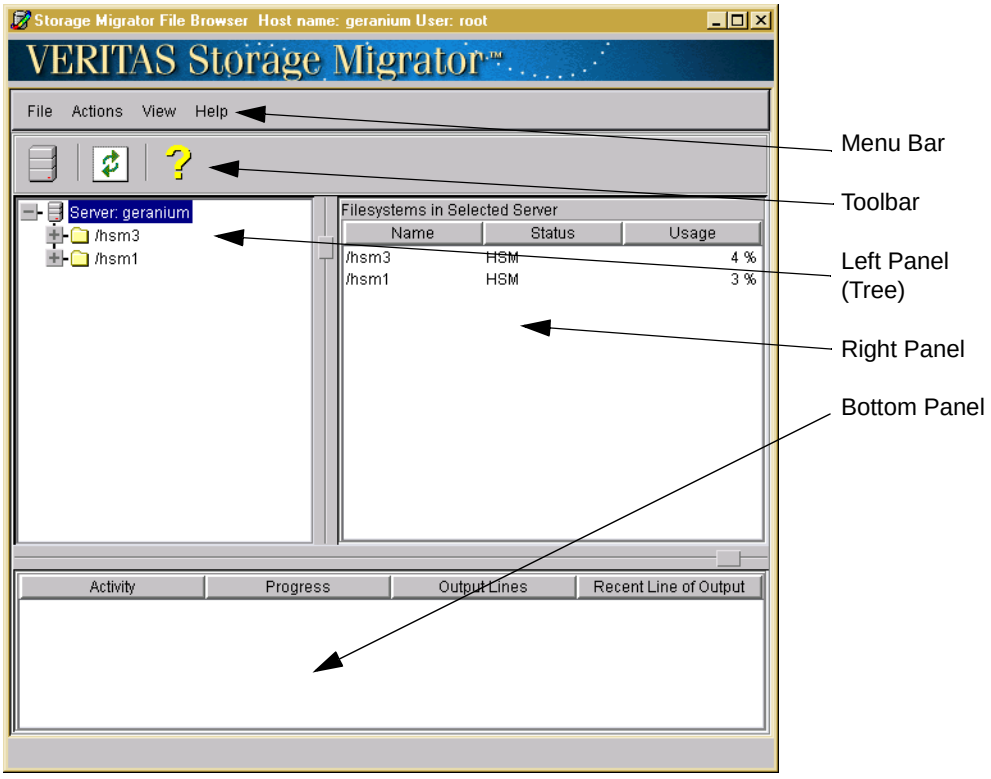
Login Procedure

1. Type in the following fields on the login screen:
   - Host name: The hostname of the server on which VSM is installed.
   - User: Either root or another username with superuser privileges.
   - Password: The root password or username password.
2. Click the Login button, or type <CR> within the Password field.

Main Screen Layout

The main screen layout of VSM File Browser contains five elements (see Figure 112, ”VSM File Browser Main Screen”).

Figure 112. VSM File Browser Main Screen



For further information on this screen, refer to the following sections:

- “Menu Bar” on page 251
- “Toolbar” on page 253
- “Left Panel” on page 254
- “Right Panel” on page 254
- “Bottom Panel” on page 255

Menu Bar

The menu bar at the top of the main screen offers four pull-down menus. For further information, refer to the following sections:

- “File Menu” on page 251
- “Actions Menu” on page 251
- “View Menu” on page 253
- “Help Menu” on page 253

File Menu

Use the File pull-down menu to change Servers or Exit from the File Browser.

Change Server...

- To logout from this server and login to another server, select Change Server... from the File pull-down menu and confirm your decision.

Exit

- To end your VSM File Browser session, select Exit from the File pull-down menu.

Actions Menu

Use the Actions pull-down menu to initiate VSM operations commonly used to control your files and directories.

Note The system administrator must provide the proper permissions before users can perform certain operations on VSM File Browser. If users are not granted permission, VSM menu items are not available to the user. These items appear in gray (grayed out).

Directory Groups (Available in DMAPI only)

In the tree panel on the left, highlight a particular Directory. Then, you can complete a number of different tasks, which are fully explained below:

Note The Directory Groups menu is only available to DMAPI environments.

- To group, but not defragment a directory, so its files and those in its subdirectories will be premigrated (and cached) together, select Directory Groups > Group from the Actions pull-down menu.
- To defragment a directory before grouping so its files and those in its subdirectories will be premigrated (and cached) together, select Directory Groups >Defragment and Group from the Actions pull-down menu.
- To ungroup a previously grouped directory so its files and those in its subdirectories will *not* be premigrated (and cached) together, select Directory Groups > Ungroup from the Actions pull-down menu.
- To list information about a grouped or ungrouped directory, select Directory Groups > List Information from the Actions pull-down menu.
- To list more detailed information about a grouped directory including the volume(s) on which the migrated data resides, select Directory Groups > Verbose Information from the Actions pull-down menu.
- To copy but not purge all of the files in a directory group, select Directory Groups > Copy from the Actions pull-down menu.
- To copy and purge all of the files in a directory group, select Directory Groups > Copy and Purge from the Actions pull-down menu.

Migration...

In the tree panel on the left, highlight the Directory node in the managed file system. Files in that directory are displayed in the right panel. Highlight one or more files in the right panel:

- To premigrate highlighted files, select Migration > Migrate from the Actions pull-down menu.
- To purge highlighted files that are premigrated and copied, select Migration > Purge from the Actions pull-down menu.
- To stage (precache) highlighted files that are purged, select Migration > Stage from the Actions pull-down menu.
- To locate highlighted files, select Migration > Locate from the Actions pull-down menu.

View Menu

In the activity table at the bottom, highlight the job. Activity from that job is displayed in the right panel.

Preferences...

- To change your VSM File Browser settings, select Preferences... from the View pull-down menu.

Refresh

- To refresh the display in the current VSM File Browser window, select Refresh from the View pull-down menu.

Clear Completed Jobs from Activity Table

- To clear all completed jobs, select Clear Completed Jobs from Activity Table from the View pull-down menu.

Cancel Selected Job in Activity Table

- To cancel highlighted jobs, select Cancel Selected Job in Activity Table... from the View pull-down menu.

Help Menu

Use the Help pull-down menu to bring up the HELP window or see version information for the VSM File Browser.

Toolbar

The toolbar at the top of the main screen offers three tool options that duplicate some of the operations in the menu bar.

To change toolbar display options, select Preferences from the View pull-down menu.

Change Server Tool

Click the Change Server tool to logout from this server and login to another server.

Refresh Tool

Click the Refresh tool to refresh the current window display.

Help Topics Tool

Click the Help Topics tool to bring up the Help window.

Left Panel

The main screen displays the tree directory structure for the managed file system in the panel on the left. To expand or contract the information displayed in this panel, click on selected nodes in the tree. :

Table 7. Icon Descriptions for VSM (found in the Left Panel)

Icon	Icon Description
	Directory: a normal directory that has not been grouped.
	Grouped Directory: a directory that has been grouped so its files and those in its subdirectories will be premigrated (and cached) as a group.

Right Panel

When you highlight a particular element in the left (tree) panel, the panels on the right display information about that element. Clicking the column headings on the right panel sorts the information in the panel, toggling from ascending to descending sorts.

When you highlight a particular job in the activity table displayed in the bottom panel, the panel on the right displays information about that job.

Table 8. Icon Descriptions for VSM (found in the Right Panel)

Icon	Icon Description
	Regular File: a file that resides on disk and is neither premigrated nor migrated.

Table 8. Icon Descriptions for VSM (found in the Right Panel)

Icon	Icon Description
	Migrated File: a premigrated file that has been copied to secondary storage.
	Purged File: a migrated file that has been purged from disk.

Bottom Panel

Job activity is displayed in the panel at the bottom of the main screen. Most of the commands invoked from the Actions pull-down menu execute asynchronously, and their progress is summarized in this panel. See “Actions Menu” on page 251 for more information.

- To see more job detail, highlight a job in the bottom panel. The complete output for that job is displayed in the right panel.
- To manage job activity, use the Jobs pull-down menu. See “View Menu” on page 253 for more information.

Migrating Files

Note The system administrator must provide the proper permissions before users can perform this operation on VSM File Browser. If users are not granted permission, VSM menu items are not available to the user. These items appear in gray (grayed out).

Users can premigrate their own files through a menu selection on VSM File Browser. VSM copies premigrated files to secondary storage during its next migration operation:

1. In the tree panel on the left, highlight the Directory node in the managed file system.
2. Highlight one or more files in the right panel that you want to premigrate.
3. Select Migration > Migrate from the Actions pull-down menu.

No files listed in the global stop file or a local stop file will be premigrated unless the stop file is overridden. See “Controlling Automatic File Migration” on page 6 for information about these migration control files. See “Migrating Files” on page 4, or the `migrate(1)` man page for more information.

Migrating Directories (Grouping Directory)

Note The system administrator must provide the proper permissions before users can perform this operation on VSM File Browser. If users are not granted permission, VSM menu items are not available to the user. These items appear in gray (grayed out).

Users can premigrate their own directories and subdirectories through a menu selection on VSM File Browser. VSM groups all owned files in an owned directory and all of its owned subdirectories and copies the premigrated files to secondary storage as a group during its next migration operation. Users with root privilege can group directories owned by any user. When any file in the group is cached, all of the files in the group are cached.

The following operation does *not* cache migrated files in the directory prior to grouping the directory, but does include those files in the directory group:

1. In the tree panel on the left, highlight the Directory node in the managed file system.
2. Select Migration > Group from the Actions pull-down menu.

It is also possible to defragment the specified directory by first caching all previously migrated files in the directory before grouping the directory. All files in the group are then premigrated together and copied to a minimum number of sequential storage media.

1. In the tree panel on the left, highlight the Directory node in the managed file system.
2. Select Migration > Defragment and Group from the Actions pull-down menu.

Once VSM has copied all of the files in a grouped directory to secondary storage media, users can purge the grouped files through a menu selection on VSM File Browser.

1. In the tree panel on the left, highlight the Directory node in the managed file system.
2. Select Migration > Purge from the Actions pull-down menu.

Users with root privilege can also copy their premigrated grouped files to secondary storage media by selecting either Migration > Copy or Migration > Copy and Purge from the Actions pull-down menu.

Note See “Migrating Directories” on page 4, or man page `miggroup(1)` for more information.

Purging Files

Note The system administrator must provide the proper permissions before users can perform this operation on VSM File Browser. If users are not granted permission, VSM will not display the associated menu item.

Users can purge their own premigrated and copied files through a menu selection on VSM File Browser. This makes disk space available without waiting for normal purging operations to occur. If the slice value is nonzero, VSM will leave that much data on disk for each file purged:

1. In the tree panel on the left, highlight the Directory node in the managed file system.
2. Highlight one or more files in the right panel that you want to purge.
3. Select Migration > Purge from the Actions pull-down menu.

Note See "Purging Files" on page 6, or man page migpurge(1) for more information.

Accessing Migrated Files

You access migrated files the same way you access other files. However, delays in caching files from secondary storage and the extra disk space required to cache those files can require you to cache some files differently.

Identifying Migrated Files

Users can identify information about files through a menu selection on VSM File Browser. This operation is informational only, and does not change the location or migration status of the files:

1. In the tree panel on the left, highlight the Directory node in the managed file system.
2. Highlight one or more files in the right panel that you want to locate.
3. Select Migration > Locate from the Actions pull-down menu.

Pre-caching Files

Users can stage (precache) purged files through a menu selection on VSM File Browser. Caching files in anticipation of accessing them in the near future avoids caching delays at the time of access:

1. In the tree panel on the left, highlight the Directory node in the managed file system.
2. Highlight one or more files in the right panel that you want to stage.
3. Select Migration > Stage from the Actions pull-down menu.

Note See "Pre-caching Files" on page 11, or man page migstage(1) for more information.

Caching a Group of Associated Files

The `migtie` command described on page 11 cannot be executed from the user interface. Instead, users can cache a group of associated files by first placing them all in a common directory and then migrating the grouped directory. When any file in the group is cached, all of the files in the group are cached.

For more information about grouped directories, see "Migrating Directories" on page 4. To premigrate your directories using the VSM File Browser, see "Migrating Directories (Grouping Directory)" on page 256.

Managing VSM 7

This chapter contains topics related to VSM administration and management and includes the following:

- Backing Up VSM Databases and Managed File Systems
- Starting VSM
- Graceful and Emergency Shutdown and Startup
- Killing VSM Processes
- Starting and Stopping the VSM Daemons
- Powering Down Remote Volume Servers
- Special VSM Commands
- VSM Volume Management
- VSM Migration Management
- VSM Export/Import Management
- VSM Databases
- Problem Solving
- Frequently Used VSM Commands

Note Some of the procedures described in this chapter refer to the `xhsmadm` interface. Similar procedures are possible using the VSM-Java interface.

Backing Up VSM Databases and Managed File Systems

Migrating files is not a substitution for backup. VSM detaches the data from migrated files when they are purged from disk. This migrated data on secondary storage cannot be used to reconstruct the file paths and access modes of the original files. Only backup utilities such as VERITAS NetBackup can backup all files and the directory structures.

Caution Do not back up VSM-managed file systems to a NetBackup class used by VSM for migrating files with the `nb` method.

Caution Do not use the FlashBackup feature of NetBackup (if supported) to back up VSM-managed file systems.

You must always back up VSM databases and managed file systems on a regular schedule by following the guidelines given here. See Figure 17 on page 43. One way to do this is with NetBackup and a backup script executed by `cron`. See “Backup and Migrate Script” on page 155.

VSM Databases

Always perform database backups on the same schedule as the VSM-managed file systems. Otherwise, the databases will not match the contents of the file systems after a restore.

Note One way to do this is to copy the FHDB and VOLDB, with timestamp extensions, into the managed file system each time you begin to back up the managed file system. This makes it easier to restore the managed file system following a system crash.

For example, assume that you back up the file system (but not the databases) on Monday and then migrate files on Monday night. If a disk crash occurs on Tuesday, recovering the last backup results in restoring the file system to the state it was in on Monday. However, the databases now are not synchronized with the file system because they show the results of Monday night’s migrations. This can result in problems during subsequent migrations and caches. Backing up both the databases and file system at the same time prevents this type of problem.

Caution Restoring a previous VOLDB can cause the following problems:

- Loss of any volumes that were registered after the backup.
- If a tape or optical volume is written after the backup, the restored VOLDB will not be able to migrate to that volume.
- If an `ft` volume is written after the backup, the restored VOLDB will not have the correct available space for the volume.

Use VERITAS NetBackup to backup VSM files and the directory structures. The standard UNIX `tar` command must *not* be used to backup and restore VSM-managed file systems because it causes VSM to cache all of the migrated files in the directory referenced by the command.

Kernel-based Implementations

Note Kernel-based implementations in this release run on Solaris *ufs* file systems.

Caution If you are backing up an *hsm* file system you must exclude the `.PAIN` (Parallel Inode) file. Backing up the rest of the files in the file system can alter the `.PAIN` file, making it inconsistent with the state of the file system.

If you do back up the `.PAIN` file, you must never restore it. You must create a new `.PAIN` file when restoring an *hsm* file system previously backed up. See "Creating .PAIN Files" on page 149.

See "Administer Parallel Inode Files" on page 147 for an explanation of the `.PAIN` file.

Nonkernel-based Implementations

Caution If you are backing up an *hsm* file system you must exclude the `.IHAND` (Inode-to-Handle) file. Backing up the rest of the files in the file system can alter the `.IHAND` file, making it inconsistent with the state of the file system.

If you do back up the `.IHAND` file, you must never restore it. Restoring an *hsm* file system previously backed up automatically creates the correct `.IHAND` file.

See "Administer Inode-to-Handle Files" on page 152 for an explanation of the `.IHAND` file.

General VSM Backup Procedure

Use the following procedure to back up VSM databases and managed file systems:

1. Make sure the migration daemon (`migd`) is running and the VSM state is Active.
2. Run migbatch to write all premigrated files to volumes. Otherwise, files may get cached back from premigration by the backup operation.
3. Purge all premigrated and copied files. Run `mignospace` to purge premigrated files to the purgemark or `mignospace -i` to purge all premigrated files.

Note It is not necessary to clear premigration before backing up your file system. If you choose to back up the premigration directory you will be able to restore it. However, the fewer files there are in premigration, the faster the backup operation will be using NetBackup.

4. Backup the VSM-managed file system to tape or optical disc using VERITAS NetBackup, making sure to exclude the `.PAIN` file in kernel-based implementations (Solaris *ufs* file systems).

Caution Restoring a partial VSM-managed file system can cause problems because there is no way of partially restoring the corresponding FHDB.

The slice value for files restored by VERITAS NetBackup is lost and set to 0. This forces VSM to cache restored files the next time they are accessed.

If a file is already migrated, NetBackup backs up only the inode with the VSM file handle, not the data itself. Restoring such a file restores only the inode and VSM file handle, and VSM must then cache the data back to disk before you can access it.

Caution When you remove migrated files, VSM marks the files *obsolete* in the FHDB. Running `migrc -O FHDB` removes obsolete entries from the FHDB older than the specified age variable (default is 7 days). Because you cannot use NetBackup to restore files unless they have a valid FHDB entry, be sure to set the age variable for the `migrc` and `migmdclean` commands at a value higher than the longest NetBackup backup retention period on the managed filesystem. You can change the default age of 7 days for the `migrc` command by editing the script. The line to modify reads `AGE=7`.

See also “Restoring VSM-managed File Systems” on page 310.

Starting VSM

Perform the following steps to restart VSM after a normal shutdown and reboot of the system. You can either execute these steps manually or put them in your system startup script (`/etc/rc.local file` or equivalent) to automatically start VSM.

Caution If restarting VSM after a system crash or abnormal shutdown, see “Recovering from a System Crash” on page 309.

1. Start Media Manager and its associated device manager with the `ltid` command. This is not necessary if you are only using local or remote magnetic disks as the only devices for migration.
2. For nonkernel-based implementations, execute `startmigd` to start the VSM migration daemon (`migd`) and VSM volume daemon (`migvold`). If using the VSM-Java interface, also execute the `migrd` command to start the `migrd` daemon as described in "Install VSM Software" on page 180.
3. Mount the managed file systems.

Note Mounting the managed file system is different for each system. On most systems this is done automatically by the default startup scripts. See "Startup Script" on page 154.

Make sure all managed file systems for kernel-based implementations (Solaris *ufs* file systems) are mounted as *hsm* file systems. Nonkernel-based implementations must be mounted as *vxfs* file systems for Solaris and HP-UX, and as *xfs* with the `dmi` option for IRIX.

4. Execute the `migrc` command with the desired parameters. Two common choices are:
   - Execute `migrc` with only the `-L` option. For example, the following only clears locks:

     ```
     migrc -L hsm2
     ```

     This option is the fastest and is the best choice if you are starting the system during regular working hours.
   - Execute `migrc` with other options. For example, the following clears locks, removes obsolete entries, and restarts any incomplete migrations.

     ```
     migrc -LoR hsm2
     ```

     Because this can require additional system resources, avoid using this option during normal working hours or in a script that executes automatically upon startup.
5. Ensure that any remote volume servers that provide access to `ft` remote volumes are up. Ensure that the NetBackup daemons are running on the master server if using the `nb` method.
6. For nonkernel-based implementations, toggle the state to Inactive and then back to Active again for `migd` to begin accessing the managed file system. For kernel-based implementations (Solaris *ufs* file systems), execute `startmigd` to start the VSM migration and volume daemons.

Graceful and Emergency Shutdown and Startup

Graceful VSM Shutdown

1. Get the file system to a state such that it could be unmounted.

   `fuser -c` *managed_file_system*

   or

   `fuser -u` ***managed_file_system***

   If fuser shows any process ids for the file system, you will not be able to unmount the file system.

   To get to this state, do the following:

   a. Have all users `cd` out of the managed file system. This will prevent any new VSM activity caused by users.

   b. Prevent `cron` from initiating VSM activity.

   c. Unshare or unexport the file system so there can be no NFS access.

   d. If you want `migcopy` to terminate, use `KILL` *pid_migcopy*. This causes `migcopy` to terminate after it completes the file it is currently working on. You may have to do this more than once because there may be more than one `migcopy` running or a new one may start.

      Before doing a `KILL` *pid_migcopy* you must determine if `migcopy` is working for the managed file system you want to shut down by running:

      `ps -ef | grep migcopy | grep` ***hsmname***

      If this shows a `migcopy`, use `KILL` *pid_migcopy*.

Note Other background processes may still be running after `migcopy` has terminated. Wait for all VSM activity to finish before continuing.

2. If desired, you may now set the managed file system inactive.

3. If desired, you may now unmount the managed file system.

4. If desired, you may run `stopmigd`, provided you have shut down all managed file systems.

VSM Startup After a Graceful Shutdown

These steps apply even if the system was rebooted.

1. For Solaris *vxfs* file systems, make sure the `migattr` driver is still installed. After a reboot `migattr` may seem to be installed but may not be working. The following steps will reinstall `migattr`.

   ```
   echo vx_kdm_attrkeep/W 0 | adb -kw  /dev/ksyms /dev/mem
   rem_drv migattr
   add_drv migattr
   modinfo | grep migattr
   ```

2. Using the interface, change the state of any managed file systems you want to change to either active or inactive.
3. Execute `startmigd` to start both the `migd` and the `migvold` daemons. If using the VSM-Java interface, also execute the `migrd` command to start the `migrd` daemon as described in "Install VSM Software" on page 180.
4. Mount any managed file systems you want to be mounted.
5. VSM is ready to use.

Emergency VSM Shutdown

1. Execute `HSMKiller` to kill all active VSM processes and unmount secondary storage volumes that are left mounted by any killed processes.

   Kill processes in one managed file system by executing:

   `HSMKiller` ***hsmname***

   Kill processes in all managed file systems by executing:

   ```
   HSMKiller
   ```

2. Using the interface, change the state of each managed file system to inactive.

   This will let you do some cleanup later after VSM is restarted, but before users can resume caching files. This also prevents `mignospace` processing.

3. If there are still VSM routines running, you will have to do a `kill -9` ***pid*** on them.

   If only shutting down one managed file system you will have to find the correct processes to kill. Using `ps` may not find all of them.

   `ps -ef | grep mig | grep` ***hsmname***

   You could also try using `/usr/proc/bin/ptree` ***pid*** on any processes found by the `ps` command to find process that you think are part of VSM.

4. For *vxfs* managed file systems, run `/usr/openv/hsm/bin/migcleanup -k` to cleanup any DMAPI sessions that were left around as a result of killing VSM processes.
5. Touch file `/usr/var/openv/hsm/workdir/migd.pid`. This is to prevent automatic startup of VSM if the system is rebooted. Your VSM startup script must reference this file.

   See "VSM Startup After a Graceful Shutdown" on page 265.

This emergency shutdown procedure may cause the following problems.

- Tapes with no trailer label

  Run `/usr/openv/hsm/bin/migdbrpt -a` *hsmname* to show what tapes were left write locked following the emergency shutdown.

  The following example shows `<*>W` indicating this volume was locked and being written.

  ```
  00001007 <*>W  AD0001 dt.1 335544320 1 0 10000 0.00
  ```

  Volume `AD0001` will need to have a trailer label put back on it. See "VSM Startup After an Emergency Shutdown" on page 267.

- Locked VOLD, FHDB and COPYDB entries

  These will need to be corrected before VSM can be used again. See "VSM Startup After an Emergency Shutdown" on page 267.

- DVDB's that have not been merged into the FHDB

  These will need to be corrected before VSM can be used again. See "VSM Startup After an Emergency Shutdown" on page 267.

- `outdb` files may exist containing files that need to be added to COPYDB worklists.

  These will need to be corrected before VSM can be used again. See "VSM Startup After an Emergency Shutdown" on page 267.

- IHAND or PAIN file entries that do not match the file system

  These can happen because `migd` and `Tmigunlink` were terminated before the IHAND or PAIN file were updated. These may or may not need to be corrected. See "VSM Startup After an Emergency Shutdown" on page 267.

- COPYDB worklist entries that have not been completed for making copies of migrated files

  This work needs to be completed. See "VSM Startup After an Emergency Shutdown" on page 267.

- COPYDB worklist entries that have not been completed for moving copies of files, and `migmoveflags` worklist entries for files that have been moved but the previous FHDB entry has not been updated

  This work needs to be completed. See "VSM Startup After an Emergency Shutdown" on page 267.

- For vxfs managed file systems, VSM may leave DMAPI sessions and active DMAPI events around. These need to be cleaned up. See "VSM Startup After an Emergency Shutdown" on page 267.

VSM Startup After an Emergency Shutdown

Use this procedure for restarting VSM after an emergency shutdown of VSM or after a system panic.

The first problem is preventing VSM from being started automatically after a reboot. To prevent this, change your startup script to check for the existence of the file `/usr/var/openv/hsm/workdir/migd.pid`. If the file exists the script must not start the VSM daemons.

If `migd` was terminated by `stopmigd`, the `migd.pid` file will not exist. This would be the case after VSM was shut down gracefully. The emergency shutdown procedure said to touch `migd.pid`, so the file will exist in that case.

To recover VSM do the following:

1. For Solaris *vxfs* file systems, make sure the `migattr` driver is still installed. After a reboot `migattr` may seem to be installed but may not be working. The following steps will reinstall `migattr`.

   ```
   echo vx_kdm_attrkeep/W 0 | adb -kw  /dev/ksyms /dev/mem
   rem_drv migattr
   add_drv migattr
   modinfo | grep migattr
   ```

2. Using the interface, deactivate all managed file systems. This will prevent `mignospace` processing and the caching of files while cleanup is done.
3. If managed file systems are *vxfs*, run `migcleanup -k` to get rid of any possible VSM DMAPI sessions and events.
4. Execute `startmigd` to start both the `migd` and the `migvold` daemons. If using the VSM-Java interface, also execute the `migrd` command to start the `migrd` daemon as described in "Install VSM Software" on page 180.
5. Mount any managed file systems you want to be mounted.
6. Identify which tape volumes need to have new trailer labels.

Run `/usr/openv/hsm/bin/migdbrpt -a` *hsmname* to show you what tapes were write locked.

The following example shows `<*>W` indicating this volume was locked and being written.

```
00001007 <*>W  AD0001 dt.1 335544320 1 0 10000 0.00
```

Volume `AD0001` will need to have a trailer label put back on it. See step 15 on page 269.

7. Clear locks and flags for all *hsmname*s. If you only want to clean up one *hsmname* include the name when executing `migrc`.

Run `migrc -L` to clear all locks. This will clear locks in FHDB, VOLDB, COPYDB files and also get rid of any VSM files indicating that a part of VSM is currently running. For *vxf*s file systems, `migrc -L` will also clear flags in the IHAND file.

For kernel based systems (Solaris *ufs* file systems) run `/usr/openv/hsm/bin/pfclear` to clear flags in the `.PAIN` file.

8. If you want to remove COPYDB entries that have been completed, or for which the source file to be copied no longer exists, run:

```
migrc -O COPYDB
```

9. If you want to remove obsolete and dead entries from the FHDB, run:

```
migrc -a age_in_days -O FHDB
```

10. If you want to remove dead VOLDB entries, run:

```
migrc -O VOLDB
```

11. To start recover work, run the following:

```
migrc -RM
```

This causes the following things to occur:

- DVDB's will be merged into the FHDB
- `outdb` files will be added to COPYDB work lists
- *migcopy* will restart processing COPYDB work lists
- `migcopy` will restart processing COPYDB `workd` lists for `migmove`
- `migmoveflags` file will be processed to alter any FHDB entries that need altering for `migmove`

Note When the `migrc` command completes, other background processes may still be running. Wait for all recovery work to finish by checking to make sure `migrecover.sh` has completed.

12. Run **migdbcheck** to make sure there are no problems remaining. Correct any problems it finds. If migdbcheck asks you to run migrc -R, do so now.

    IHAND or PAIN file entries that do not match the file system can usually be corrected by using:

    - migalter
    - ihprint or pfprint

    Use migsetdb to modify the FHDB and VOLDB.

    These problem are sometimes found by and corrected by migdbcheck. Others may only be found when VSM sweeps looking for files to migrate or purge.

13. Using the interface, change the state of any managed file systems you want to change to either active or inactive.

14. VSM is ready to use.

15. Add trailer labels to any tape volumes that need them.

    Use the /usr/openv/hsm/bin/goodies/migadd_trailer.sh script to add the trailer label.

    If you want VSM to use this volume for writing again, use:

    migsetdb -V -w *hsmname volume_id*

Killing VSM Processes

The HSMKiller command allows you to kill running and hung processes for either all or just specific *hsmname*s. The daemons migd and migvold are not killed if HSMKiller is run for a specific *hsmname*. This command also unmounts any secondary storage media in use for those managed file systems. You must use HSMKiller if you power down the managed server or if you encounter problems that leave VSM processes in a hung condition.

Processes affected by HSMKiller are listed in two kill lists. If you specify one or more *hsmname*s, HSMKiller searches for processes listed in /usr/openv/hsm/bin/admincmd/HSMKiller.kill_list. If you do not specify any *hsmname*s, *HSMKiller* searches for processes listed in both /usr/openv/hsm/bin/admincmd/HSMKiller.kill_list and /usr/openv/hsm/bin/admincmd/HSMKiller.global.kill_list.

HSMKiller searches for processes to kill which are executing using the full pathname for VSM binaries, /usr/openv/hsm/bin. All VSM processes which call other VSM processes use this convention. For HSMKiller to work properly, all VSM processes called by customer defined scripts or cron schedules must follow this same convention instead of depending on PATH variables to allow shorthand command names.

Note Executing HSMKiller when migcopy is running can leave a volume without an end of file mark in some situations. This will make it impossible to append any more data to that volume. Data previously written to the volume, however, will remain accessible.

To kill the processes for designated VSM configurations:

1. Execute HSMKiller and specify *hsmname*(s):

   For example:

   ```
   HSMKiller hsm1 hsm2
   ```

   kills listed processes for hsm1 and hsm2. The VSM daemons migd and migvold are not killed.

2. Execute migrc -L to clear any locks left set by the killed processes.

   For example:

   ```
   migrc -L hsm1 hsm2
   ```

   clears locks for hsm1 and hsm2.

To kill all listed VSM processes:

1. Execute HSMKiller without any parameters:

   ```
   HSMKiller
   ```

   This kills all listed processes for all *hsmname*s. The daemons migd and migvold are also killed, thus preventing VSM from automatically removing or migrating files from any managed file system and also preventing users from caching migrated files.

2. Execute migrc -L to clear any locks left set by the killed processes.

   ```
   migrc -L
   ```

   This clears locks for all *hsmname*s.

3. Restart the VSM daemons. See “Starting VSM” on page 262 and “Recovering from a System Crash” on page 309.

You can edit the two default kill lists for HSMKiller. For example, if you never use the ft method for remote storage, you can delete migftdel.sh, migftget.sh, migftls.sh, migftput.sh, migftreg and migftscan.sh from /usr/openv/hsm/bin/admincmd/HSMKiller.kill_list. This would speed up HSMKiller processing.

Also, if you *always* use commands like `migbatch`, `migmove`, and `migrc` with an *hsmname* argument, you can add them to `/usr/openv/hsm/bin/admincmd/HSMKiller.kill_list` and delete them from `/usr/openv/hsm/bin/admincmd/HSMKiller.global.kill_list`. This would cause `HSMKiller` *hsmname* to search for these processes also. Use care when editing the default kill lists because this can result in a less comprehensive termination of active processes.

Starting and Stopping the VSM Daemons

The migration daemon (`migd`) must be running in order for any caches or automatic removal or migrations to occur. The migration daemon can be started automatically when you boot the system (see "Startup Script" on page 154). However, there are also times when you must start it manually. For example, stopping and then restarting the migration daemon enables VSM to pick up configuration changes made without using the interface. (VSM automatically captures configuration changes made using either interface.)

The volume daemon (`migvold`) unmounts tape and optical disc media mounted in read mode. It is not required for VSM operations involving local or remote magnetic disks, but it must be running if any VSM operations such as caching or multilevel migration (`migmove`) read from tape or optical media.

To start the `migd` and `migvold` daemons, see "Starting VSM" on page 262.

Execute `startmigd` to start both daemons. Use the `-m` option to start only `migd`, or the `-v` option to start only `migvold`.

To stop the daemons, execute `stopmigd`. Use the `-m` option to stop only `migd`, or the `-v` option to stop only `migvold`.

Stopping `migd` prevents caches and automatic removal or migrations from managed file systems. This is necessary before performing certain procedures. For example, always stop the daemon before powering down the managed system.

The Java daemon (`migrd`) must be executing on each VSM server host which is to accept the VSM-Java interface.

To start the `migrd` daemon, execute the following command:

```
/usr/openv/hsm/bin/admincmd/migrd
```

Powering Down Remote Volume Servers

Before powering down a remote volume server, the administrator must notify the administrators of all the managed servers that use the remote volume server. This allows the administrators of the managed servers to stop the migration daemon, thus preventing any further migrations or caches until the remote volume server is available again.

Special VSM Commands

The special command `/usr/openv/hsm/bin/fls` is a variation of the standard UNIX command `ls`. Use `fls` instead of `ls` to show migration status. See the *Storage Migrator User's Guide* or manpage migfind(1) for more information.

A troubleshooting command, `/usr/openv/hsm/bin/migfind`, is used to determine the full pathname of a file. See manpage migfind(1M) for instructions and examples on using this command.

VSM Volume Management

Volume management tasks that you perform as a VSM administrator include the following:

- Monitoring Volume Usage
- Keeping a Supply of Unused Volumes
- Cleaning nb Volumes
- Consolidating Volumes
- Removing Tape or Optical Volumes for Offline Storage
- Moving Files to a New Volume Set
- Removing Volume Database Entries
- Duplicating a Tape

Caution Never use media containing VSM migrated files in a device that is not under control of Media Manager. Doing so can result in loss of data due to the media being used by non-VSM processes.

Monitoring Volume Usage

It is important to monitor volume usage in order to ensure the most efficient use of your media and also to ensure that you have enough media for pending migrations. The `migdbrpt` and `miggetvol` commands provide reports that you can use to list your VSM volumes and determine how much space remains on each of them.

- Use `migdbrpt` to check the used and free space of volumes. See man page `migdbrpt(1M)` for more information.
- Use `miggetvol` to list volumes in order of space utilization. See man page `miggetvol(1M)` for more information.

When monitoring volume usage, look for volumes in the following categories:

- Unused volumes that VSM can use for future migrations.
- Volumes which can be consolidated and recycled because they contain many obsolete files.
- Damaged volumes no longer assigned for writing but from which VSM can still attempt to read migrated files.

Note The other topics in this section explain how to deal with each of these categories.

Keeping a Supply of Unused Volumes

Unused volumes are your reserve for future storage. These are volumes that you have already registered but VSM has yet to use them for storage.

Ensure that you always have enough unused volumes available to prevent VSM from running out of media during a migration. One way to accomplish this is by registering extra media in volume set 0 for each method (see “Registering Extra Volumes” on page 176). The extra media can be either new or media that you have reclaimed through the recycling process (see “Consolidating Volumes” on page 274).

After using up previously registered media, VERITAS Storage Migrator automatically registers additional tape and optical disc media as needed from one or more volume pools for writing the premigrated files on the work list to secondary storage. The volume pool is the one specified for the storage method in the configuration file, `migconf`. If this optional parameter is not specified for METHODX, the default volume pool, `HSM`, is used.

If there are no unassigned volumes in the specified pool, VERITAS Storage Migrator registers a volume from the Media Manager scratch pool and changes the pool name of that volume to match its own volume pool name. For this to occur, include the following statement in the Media Manager configuration file, `/usr/openv/volmgr/vm.conf`.

```
SCRATCH_POOL = scratch_pool_name
```

where *scratch_pool_name* is the pool name for all volumes currently unassigned in the Media Manager scratch pool.

See “Register Media with VSM” on page 157 for instructions and examples on registering and labeling media.

Cleaning nb Volumes

When VSM migrates files with the `nb` method, it attempts to identify the NetBackup imageID for the migrated files. This imageID takes the form of the client name and the Unix timestamp. If the attempt is unsuccessful, a default timestamp of 0 is entered in the

FHDB. When a file is cached from an `nb` volume, VSM informs NetBackup of the timestamp to help locate the file image on the volume. The restore operation is slower if the timestamp is 0 because no time range is indicated.

If all file images from a given timestamp are obsolete, the `migmdclean` command notifies NetBackup to delete the images from its database. The site administrator is responsible for determining when all the images on a NetBackup volume have been deleted, and then expiring the volume itself.

If the timestamp is 0, VSM cannot correctly inform NetBackup which images to delete, and `migmdclean` cannot clean the `nb` volume. To correct this, substitute the correct imageIDs for all files by executing the `mignbscan` command and merging the FHDB.*label* output file with the FHDB using `dbconstruct.sh`. Then run `migdbcheck -r` to eliminate FHDB entries for deleted files. The remaining FHDB entries have the proper timestamp, and `migmdclean` will correctly inform NetBackup which images to delete.

Consolidating Volumes

Note Consolidation is not applicable for `ft` or `nb` volumes.

When a cached file is modified or removed, VSM obsoletes the file in the FHDB. The space occupied by obsolete files on a volume cannot be reused as long as there are active files on the same volume. Consolidation makes it possible to reuse this wasted space.

Use the `migmdclean` or `migrc` commands to mark obsolete files dead before consolidating volumes. The age variable in these commands allows you to preserve obsolete files for a period, usually seven days, before marking them dead.

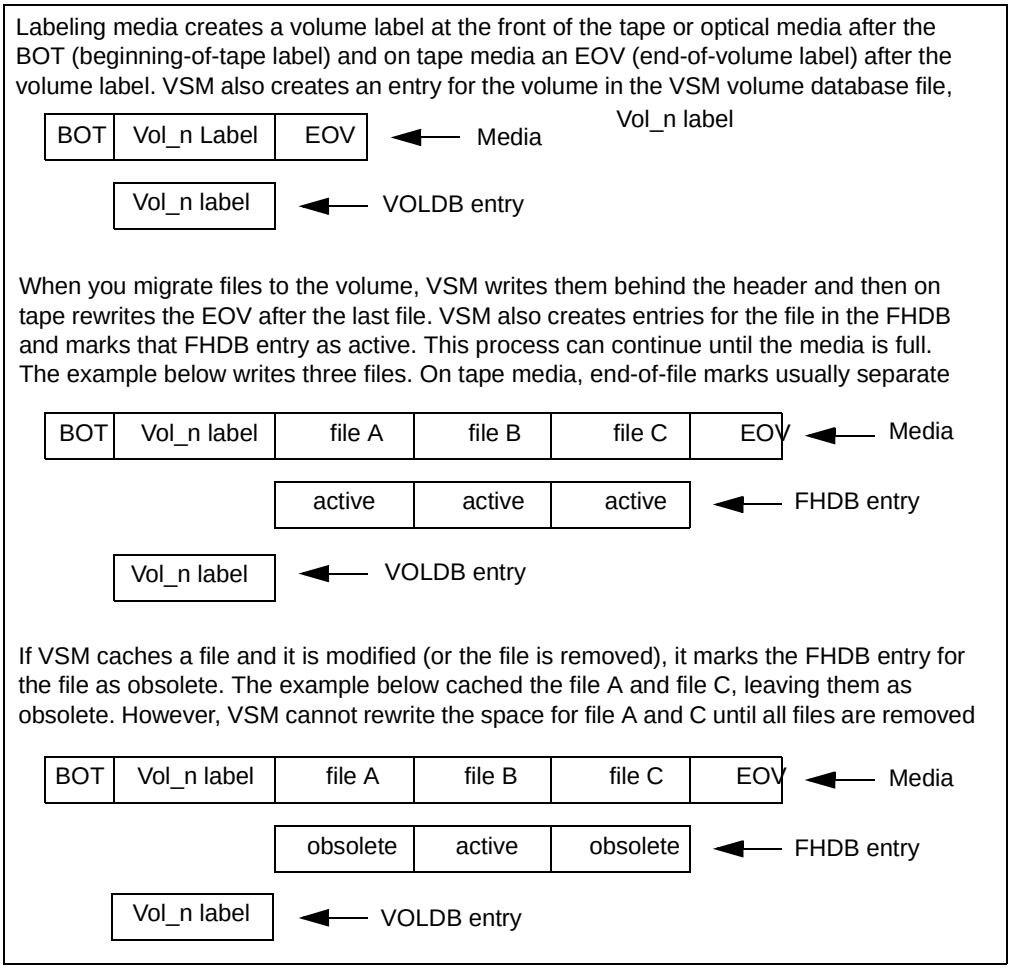
Consolidation reads active and obsolete data from a set of input volumes and writes it to a set of output volumes. No dead data is written to output. After consolidation, all files on the input volumes are marked dead, and the volume can be recycled.

A feasibility check allows consolidation only if the empty volumes for the output method have enough capacity to receive the files from the input volumes. If consolidation is forced and output volume capacity is inadequate, VERITAS Storage Migrator automatically registers additional tape or optical disc media as needed to provide adequate output volume capacity for writing the data from the input volumes. These output volumes are registered in the same volume pool as the first input volume being consolidated.

In addition to using consolidation to recover wasted space, you can use consolidation to copy data from one set of volumes to another set.

Figure 113 on page 275 shows what occurs as VSM writes and caches files to an optical (methods `op` and `ow`) or tape volume (methods `ct`, `dt`, and `mt`). The process is the same for alternate magnetic disk volumes (method `ad`) except that there is no EOV label to rewrite.

Figure 113. Writing and Obsoleting Files on Media



The recycling process allows you to reclaim space wasted due to dead files. You can recycle volumes through consolidation, which involves:

- Writing any remaining active and obsolete files to another volume, thus freeing all space on the volume you are recycling.

Note To write only active files to another volume, run `migmdclean` to set obsolete entries to dead before consolidation. See man page `migmdclean`(1M).

- Removing from the FHDB all entries for file granules on the volume you are recycling.
- Removing from the VOLDB the entry for the volume you are recycling
- Registering and labeling the volume for reuse by VSM.

The result is to consolidate active and obsolete files onto a smaller number of volumes, thus releasing some volumes for reuse. The frequency with which you must perform consolidation depends primarily on the rate at which you modify or remove migrated files.

If all of the files and granules on a volume are already dead there is nothing left to consolidate. In this case you can use the `migrecycle` command to reregister the volume as empty. See "Recycling Empty Volumes" on page 285. See also manpages `migmdclean(1M)` and `migrecycle(1M)`.

Note Although you can consolidate `ow` volumes (write once, read many), you can not recycle them.

The following procedures explain both a one-step and two-step approach to consolidation. There is also a third procedure that you can use if all files on the volume are dead. To implement these consolidation procedures using `xhsmadm`, see "Consolidation Using the interface" on page 281. It is also possible to consolidate volumes using the VSM-Java interface.

One-Step Consolidation

Use one-step consolidation if you have one drive to read input volumes (volumes you are consolidating) and a second drive to write output volumes. The input and output storage methods can be either the same or different. Use the two-step consolidation procedure if you do not have enough drives to support one-step consolidation.

The steps to perform in one-step consolidation are as follows. See the man pages for more information on the commands and scripts mentioned here.

1. If you only want to consolidate active data and not obsolete data, run `migmdclean` on the volumes to clean the media and databases by setting obsolete FHDB entries to dead and then removing the dead entries.

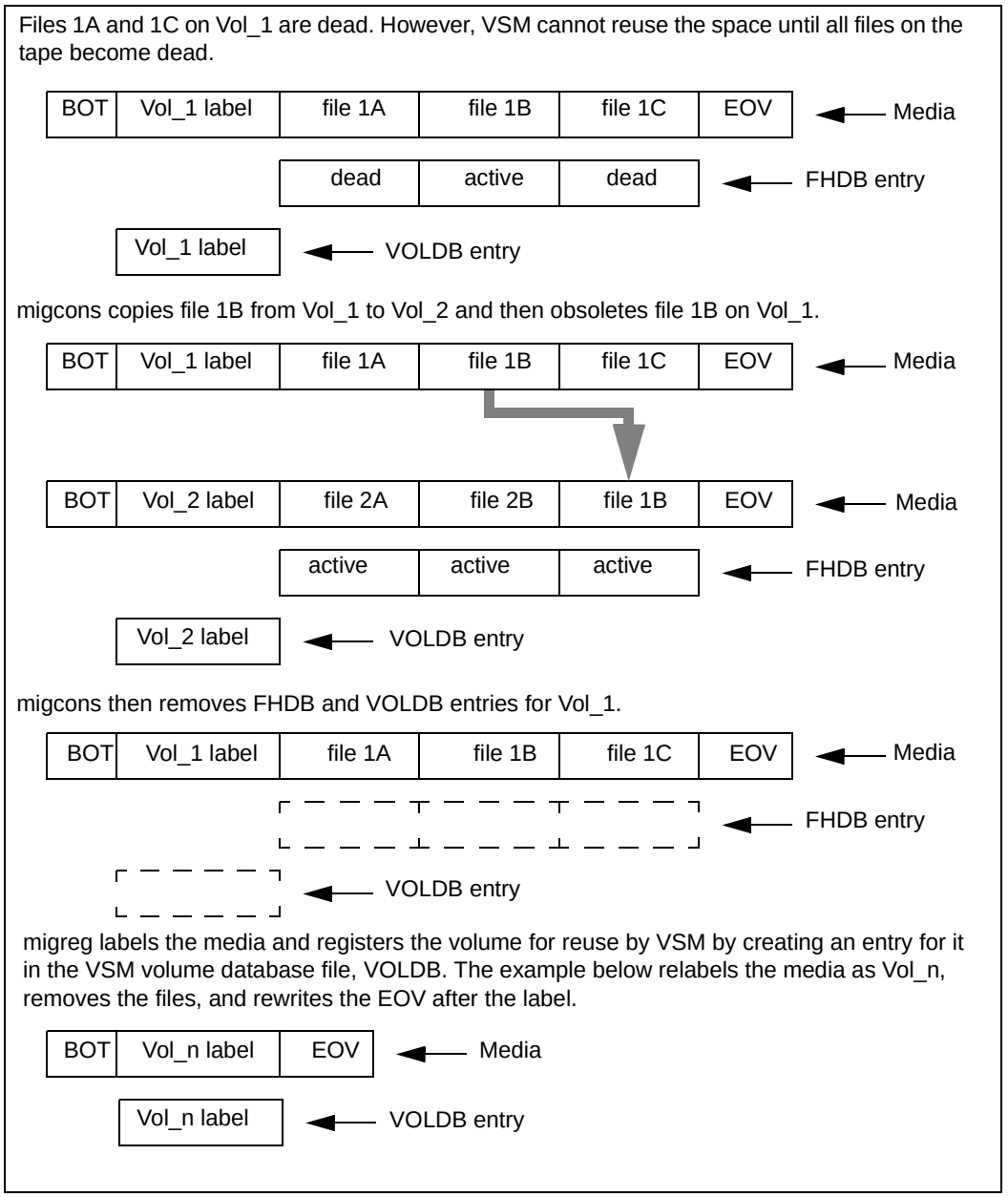
Note For the `ad` and `ft` methods, `migmdclean` attempts to remove files from the media when the `MFLAG_OBSOLETE` flag is specified in the configuration file for these methods. For the `ct`, `dt`, `mt`, `op`, and `ow` methods, `migmdclean` never attempts to remove files from the media, but makes the data inaccessible.

2. Consolidate volumes by executing `migcons` with the desired parameters. By using `migselect` in conjunction with `migcons`, you can consolidate volumes based on percentage of space utilization. The consolidation process does the following:
   - Copies active and obsolete files to the desired output volume.
   - Clears the FHDB and VOLDB databases of entries related to the input volume. You can now label and register the input volume for reuse (see step 3).

Figure 114 on page 278 shows what occurs during the consolidation of optical disc or tape volumes. The process is the same for alternate magnetic disk volumes (method `ad`) except that there is no EOV label to rewrite.

3. Register and label the input volume for future use with `migreg`.
4. This process also makes all data on the input volume inaccessible. See “Register Media with VSM” on page 157 for instructions and examples on registering and labeling media.

Figure 114. One-Step Consolidation



Two-Step Consolidation

Use two-step consolidation if you have only one drive available for reading and writing. With two-step consolidation, VSM first consolidates input volumes to an `ad` method volume and then from that intermediary volume to the final output volumes. You must register at least one volume to `ad` method, volume set 0.

The steps to perform in two-step consolidation are as follows. See the man pages for more information on the commands mentioned here.

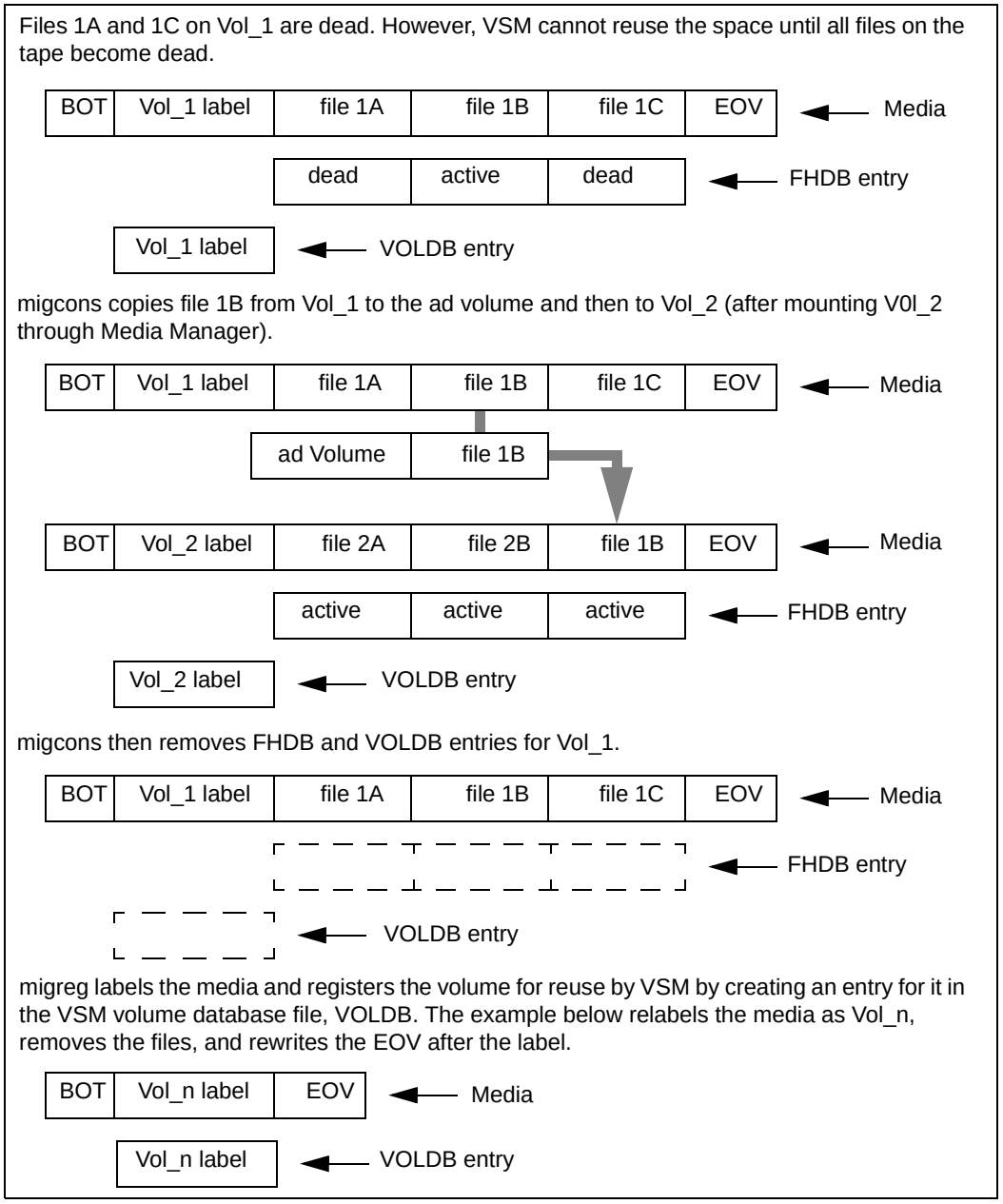
1. Run `migmdclean` on the input volumes if you want to mark obsolete files dead. If you do not run `migmdclean`, obsolete files will be consolidated to the output volumes along with the active files.

Note For the `ad` and `ft` methods, `migmdclean` attempts to remove files from the media when the `MFLAG_OBSOLETE` flag is specified in the configuration file for these methods. For the `ct`, `dt`, `mt`, `op`, and `ow` methods, `migmdclean` never attempts to remove files from the media, but makes the data inaccessible.

2. Register an alternate magnetic disk volume to volume set 0 by using the `migreg` command (see “Register Media with VSM” on page 157). The `ad` volumes you register must have enough space to hold the data being copied from the input volumes. You can estimate this by examining the output from `migdbrpt` for all input volumes.
3. Consolidate volumes by executing `migcons` with the desired parameters. The two-step consolidation process performs the following (see Figure 115 on page 280):
   - Copies active and obsolete files from the input volumes to the alternate disk volumes.
   - Copies the files from the alternate disk volumes to the final output volumes.
   - Removes FHDB entries for all input volumes.
   - Marks the input volumes dead in the VOLDB.
   - Makes active FHDB entries for active files copied to output volumes (and obsolete FHDB entries for any obsolete files copied).
   - Updates VOLDB entries for output volumes.

   You can now register the consolidated volumes for reuse.
4. Use `migreg` to register and label the volume for future use by VSM (see “Register Media with VSM” on page 157). This process also makes all data on the input volume inaccessible.

Figure 115. Two-Step Consolidation

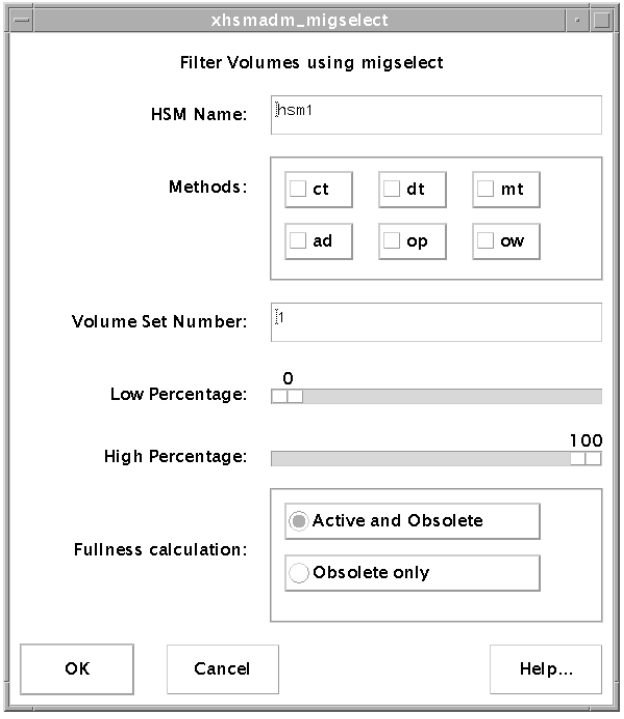


Consolidation Using the interface

The process of selecting and consolidating volumes explained above is made easier by using either interface. This example shows `xhsmadm`.

1. From the main `xhsmadm` screen, select an *hsmname* that uses the database pathname for the volumes you want to consolidate, pull down the Volumes menu and select Volume Registration and Reports. This opens the `xhsmadm` volume registration screen used by that file system.
2. You can select which volumes to consolidate in several ways:
   - To display candidates for consolidation by storage method name, pull down the `View` menu, select Filter by Method, and select your method of choice.
   - To display candidates for consolidation by percent of capacity that is obsolete, pull down the View menu and select Filter by `migselect`. This opens the `xhsmadm` volume select screen. See Figure 116 (below).

Figure 116. xhsmadm_migselect



Change the volume selection attributes to the desired values. The attributes in each entry are defined as follows:

VSM Name

The name of the selected file system.

Methods

The method name (or names) for the volumes to be selected.

Volume Set Number

The integer number of the volume set of which this volume is a part.

Low and High Percentage

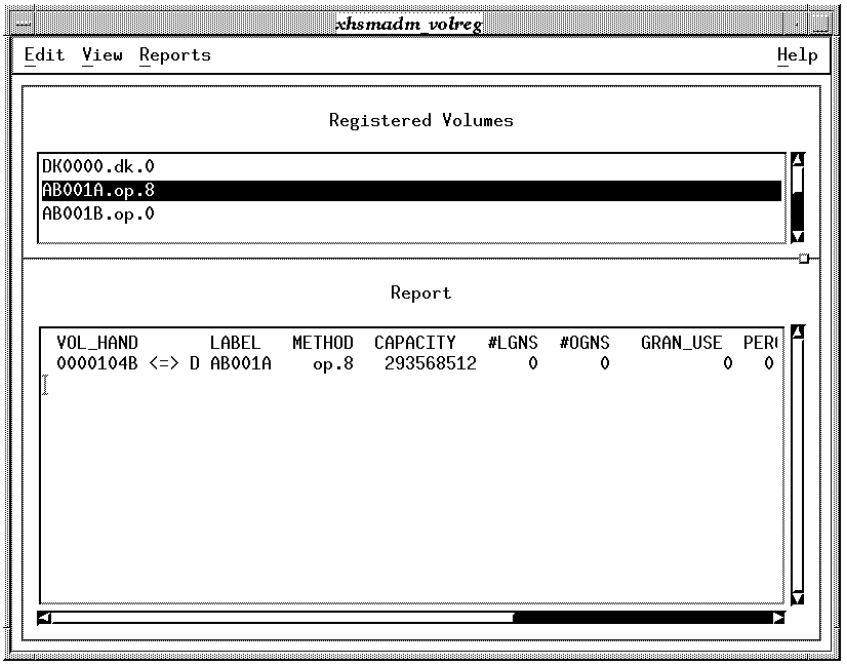
The lower and upper limits of space utilization for the volumes to be selected.

Fullness Calculation

- To select volumes based on the percentage of a full volume that is both active and obsolete, select the Active and Obsolete option. This is the default.
- To select volumes based on the percentage of a full volume that is just obsolete, select the Obsolete Only option.

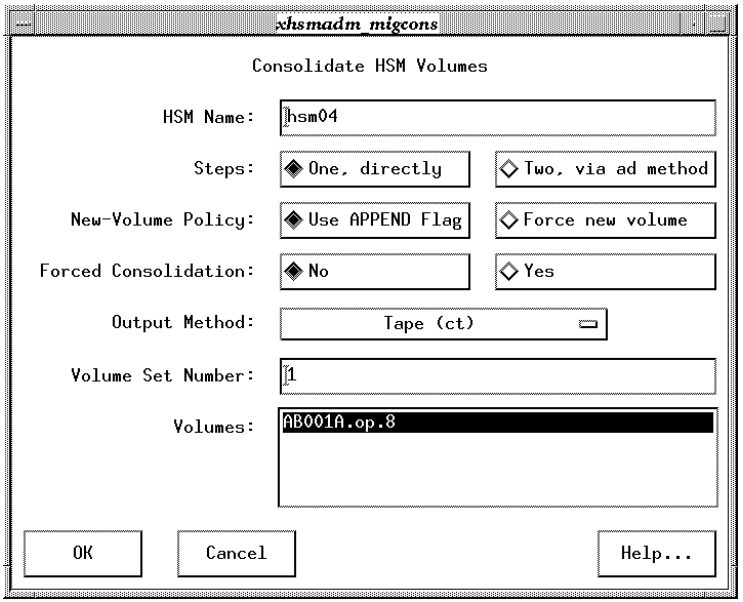
3. Click the OK pushbutton. The selected volumes are now displayed in the Registered Volumes field of the `xhsmadm` volume registration screen. See Figure 117 on page 283.

Figure 117. xhsmadm_volreg



4. Highlight the volumes you wish to consolidate, pull down the Edit menu and select Consolidate via migcons. This opens the `xhsmadm` volume consolidation screen. See Figure 118 on page 284.

Figure 118. xhsmadm_migcons



5. Change the volume consolidation attributes to the desired values. The attributes in each entry are defined as follows:

VSM Name

The name of the selected file system.

Steps

To copy files directly from the input media to empty volumes under the output method as described in "One-Step Consolidation" on page 276, select the One, Directly option. To copy from the input media to alternate magnetic disk media (method `ad`) and then to empty volumes under the specified output method as described in "Two-Step Consolidation" on page 279, select the Two, via ad method option.

New-Volume Policy

To append data to the volume currently being written, select the Use APPEND Flag option. To write data to a new volume, select the Force new volume option.

Forced Consolidation

Perform consolidation without feasibility check that allows consolidation only if the empty volumes for the output method have enough capacity to receive the files from the input volumes. If consolidation is forced and output volume capacity is inadequate, available tape or optical disc media in the same volume

pool as the first input volume being consolidated are registered automatically as needed to provide adequate output volume capacity for writing the data from the input volumes. If output volume capacity is still inadequate, the command will fail when output volumes become full. These output volumes are marked full in the VOLDB, but no FHDB changes occur for input volumes being consolidated.

Output Method

Output method name for consolidation.

Volume Set Number

The integer number of the volume set of which this volume is a part.

Volumes

The volumes selected on the `xhsmadm` volume registration screen to be consolidated. See Figure 117 on page 283.

6. Click the OK pushbutton. VSM now consolidates the selected volumes.

Recycling Empty Volumes

A volume is called empty if all of the files and granules on the volume are marked dead in the FHDB. This can occur when the migrated files on the volume have all been either modified or removed. VSM does not automatically reclaim this space, and the obsolete data remains available for possible reference at some future time. When the obsolete data is no longer of any value you can recycle the empty volumes.

The `migrecycle` command makes data on empty volumes inaccessible, and also removes the related entries from the FHDB and VOLDB database files. `migrecycle` then calls `migreg` to reregister and label the empty volume, thus reclaiming its storage space for future use.

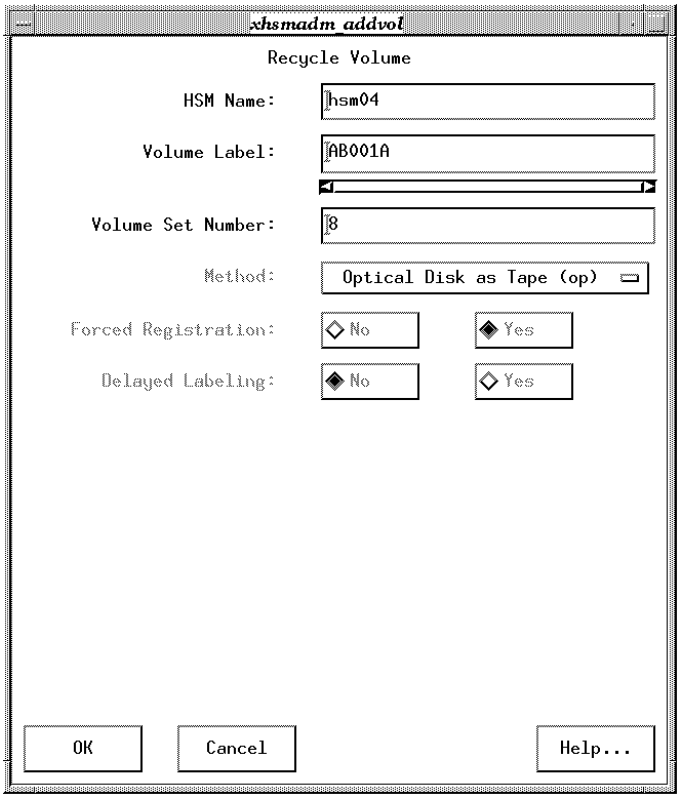
Each side of an optical disc is a separate volume, but VSM registers both sides with the same volume set number. If both sides of a rewritable disc (`op` method name) are empty, you can recycle the entire disc, change its attributes, and assign a new volume set number. If one side is empty and the other contains granules, you can recycle the empty volume but not change its attributes, and the volume set number remains the same for both sides. WORM discs (`ow` method name) cannot be recycled.

Recycling volumes is made easier by using either interface. This example shows `xhsmadm`.

1. From the main `xhsmadm` screen, select an *hsmname* that uses the database pathname for the volume you want to recycle, pull down the Volumes menu and select Volume Registration and Reports. This opens the `xhsmadm` volume registration screen used by that file system.
2. Highlight the volumes to be recycled.

3. Pull down the Edit menu and select Recycle. This opens the `xhsmadm` volume recycle screen. See Figure 119 (below).

Figure 119. xhsmadm_recycle



4. Change the volume recycle attributes to the desired values. The attributes in each entry are defined as follows:

HSM Name

The name of the selected file system.

Volume Label

The name of the volume to be recycled and removed from the VSM volume database VOLDB.

Volume Set Number

The integer number of the volume set of which this volume is a part. To recycle volumes as extras, specify volume set number 0. See “Keeping a Supply of Unused Volumes” on page 273.

Method Name

The method name for the volume to be recycled.

5. Click the OK pushbutton. VSM invokes the `migrecycle` command.

Removing Tape or Optical Volumes for Offline Storage

You can remove a volume from a library device for offline storage, making it less available. You configure availability with the *hint* field of METHOD1 through METHOD8 (see "Volume Set Availability (hint)" on page 87). Configure the *hint* field for the offline volume set as either `vault` or `operator`.

When removing a volume from an online library, use the NetBackup administrator interface (`jnbSA`) or the Media Manager's `xvmadm` or `vmadm` utilities to change the volume's physical location to indicate *Not Robotic*. To remove a single optical volume, remove the disc by using the *mailslot* capability of the optical-disc library.

Subsequent references to a *Not Robotic* volume generates an operator mount request. To satisfy the mount request, either manually mount the volume in a nonrobotic device or move the volume to the appropriate robot.

If you are going to continue writing to the volume set of the removed volume, replace the removed volume with another volume registered to the same volume set.

Use `xvmadm` or `vmadm` to move volumes to a robot. To move a single optical volume, simply insert the disc by using the *mailslot* capability of the optical-disc library. After moving the requested volume to a robot, resubmit the mount request through the operator interface.

Moving Files to a New Volume Set

There can be instances when you want to move migrated files to a different volume set (for example, to change media). There are two ways to do this:

- For `ct`, `mt`, `dt`, `op`, `ow`, or `ad` volumes, you can move active and obsolete files from one volume to another volume by using the consolidation procedure. See "Consolidating Volumes" on page 274.
- Another way to move files is with the `migmove` command. This makes it possible to move files from one volume set to another volume set providing the volume sets are on different migration levels. See "Move Files between Migration Levels" on page 293.

Removing Volume Database Entries

If a volume is lost, destroyed, becomes unreadable, or all its FHDB entries are obsolete, you can remove the entry from the VSM volume database (VOLDB). Use `migmdclean` with the `-O` option to force valid FHDB entries to obsolete and then to dead, and then use the `-R` option to remove the VOLDB entry. See man page `migmdclean(1M)` for caveats about using the `-O` option.

Cleaning volumes is made easier by using either interface. This example shows `xhsmadm`.

1. From the main `xhsmadm` screen, select an *hsmname* that uses the database pathname for the volumes you want to clean, pull down the Setup menu and select Volume Registration and Reports. This opens the `xhsmadm` volume registration screen used by that file system.
2. Highlight the volumes you wish to clean, pull down the Edit menu and select Delete via migmdclean. This cleans the volumes and databases by setting obsolete FHDB entries to dead and then removing the dead entries. When all FHDB entries for a volume are dead, the volume's entry is removed from the VOLDB. You can then recycle the volume. If you believe that the media is damaged, however, it is better not to recycle it.

Duplicating a Tape

If VSM can no longer read a tape volume containing migrated data, use this procedure to make a new copy of all the files on the tape. It is required that another copy of each file exist on another volume that is neither an `ft` nor an `nb` volume.

1. Identify the volume handle (VOL_HAND) of the damaged tape volume. The VOL_HAND is found in each FHDB entry for files that are on the volume. Print a report of all volumes by executing:

   `migdbrpt -a` *`hsmname`*

   In this example, the VOL_HAND for tape label `TP0004` is `00001008`.

   ```
   migdbrpt -a vdm2


   VOL_HAND       LABEL  POOL  METHOD
   00000000 <=>  DK0000        dk.0
   00001007 <=>  AD0001        ad.1
   00001008 <=>W TP0004        dt.3
   ```

   The method shows the method name and volume set number of each listed volume. This defines the volume set, and in this example the damaged tape `TP0004` is in volume set `dt.3`.

2. Mark all files dead in the FHDB that have all or part of their data on the damaged tape volume with the VOL_HAND identified in step 1 on page 288, and remove all of these dead FHDB entries. Remove the damaged tape volume from the VOLDB.

   `migmdclean -O -R` *hsmname label.method.volset*

   For the example in step 1 on page 288:

   `migmdclean -O -R vdm2 TP0004.dt.3`

3. Create another copy of all the files that were on the damaged tape volume.

   Do this either of two ways:

   - `migdbcheck` method

     1. Set the managed file system inactive.

     2. Run `migdbcheck -F -r` *hsmname* to produce a work list of all files that do not have enough copies.

     3. Run `migrc -R` *hsmname* to process the list produced by `migdbcheck`.

Note In some situations, the `migdbcheck` method may create three copies of the files on the damaged tape volume.

   - `migmove` method

     1. With either interface, set Move Badness, Move Minimum Age, and Move Minimum Size to 0 for the level that still has a good copy of the files that were on the damaged tape. You should also set Move Badness, Move Minimum Age, and Move Minimum Size to 0 for the Method Name that still has a good copy of the files that were on the damaged tape.

     2. Use `migmove -a -s` *source_level* `-d` *destination_level hsmname* to make another copy of the files that were on the damaged tape, where:

        source_level is the level that still has a good copy of the files

        destination_level is the level the damaged tape was at

     `migmove` will try to move all files, but if a file is currently at *destination_level*, **`migmove`** will *not* put another copy there. The `-a` option tells `migmove` to leave the files at *source_level* active.

     For example:

     `migmove -a -s 1 -d 2 vdm2`

VSM Migration Management

Migration management tasks that you perform as a VSM administrator include the following:

- Global Migration Control
- Calling Migration Commands
- Reconfiguring Storage Methods
- Performance Tuning

Global Migration Control

In addition to the local control files .migrate and .migstop described in the *Storage Migrator User's Guide*, there is also a set of global control files. These are part of the VSM configuration and apply to all VSM filesystems.

`/usr/var/openv/hsm/database/migrate`

The global migrate file, containing a list of the files or directories of files that VSM will migrate during automatic migration.

`/usr/var/openv/hsm/database/migstop`

The global stop file, containing a list of the files that VSM will not migrate.

The specifications only affect files that reside in a VSM-managed directory.

Rules for VSM Control Files

VSM applies the following rules to local and global control files.

General Rules

- All local migrate files must be named `.migrate`. All local stop files must be named `.migstop`. These names are not configurable.
- Local control files can reside anywhere, even outside VSM-managed file systems, but they only apply to VSM-managed files in the subtree of the directory in which the control file resides. Symbolic links are not traversed.
- Administrators can create one global migrate file and one global stop file as shown in "Global Migration Control" on page 290. These files apply to all VSM-managed file systems.
- The `.migrate` and `.migstop` files most local to the listed file override more remote control files in the directory tree. Local control files override global control files if the same file or directory is listed in both.

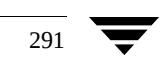
- If the same file or directory is listed in both a `.migrate` file and a `.migstop` file at the same level, the `.migrate` file overrides the `.migstop` file.
- Directories listed in a control file apply to all files and subdirectories residing in that directory unless overridden by another control file more local to the listed file or directory.

Syntax Rules

- Each line in a control file contains only one entry. Empty lines and lines starting with the pound sign (#) are ignored.
- In local control file entries, relative paths stem from the directory in which the local control file resides. Absolute paths only apply if they resolve to files or directories in the subtree in which the local control file resides.
- In global control file entries, relative paths stem from the mount point of each VSM file system. Absolute paths only apply if they resolve to files or directories in VSM-managed file systems.
- VSM does not traverse symbolic links if included in control file entries.
- VSM does not match parent directory (..) entries.
- VSM recognizes wildcards in control file entries. The wildcard metacharacters are as follows:
  - An asterisk (*) matches any character string including the null string.
  - A question mark (?) matches any single character.
  - A set of brackets ([]) matches any of the enclosed characters. A pair of enclosed characters separated by a hyphen (-) matches any character lexically between the pair, inclusively. If the first character following the first bracket ([) is an exclamation point (!), matches occur for any character not enclosed.
- VSM matches initial dot (.) file names and directories explicitly.
- VSM matches an ellipsis (...) used as a directory name matched any number of subdirectories.
- Special characters including a backslash (\) can be escaped with a backslash if they are to be matched explicitly in a file name or directory.

Files listed in these global control files are not subject to the quota limit. Local .migrate and .migstop files override the global control files if the same file or directory is listed in both.

If files are excluded from automatic migration by listing them in a `.migstop` file or global stop file, the super-user or the file's owner can still force their migration by using the `migrate -F` command. For more information, see man page `migrate(1)`.



Scheduling Migrations

VSM allows you to schedule automatic, unattended migration of files either by using the scheduling feature of VSM or by using `cron` to invoke the `migbatch` command (see “Backup and Migrate Script” on page 155). When setting up schedules, the main considerations are:

- “Best Times for Migrating Files” on page 108
- “How Often to Migrate Files” on page 108
- “How Often to Migrate Files” on page 108

Calling Migration Commands

Most file migration with VSM is handled by scheduled automatic and unattended migration operations. See “Scheduling Migrations” on page 107.

There are other situations, however, when you need to perform particular migration functions immediately. VSM enables you to call many of these migration commands directly from pull-down menus on either interface. These examples use `xhsmadm`.

Premigrate and Copy Files to Secondary Storage

From the main `xhsmadm` screen, select an *hsmname* that uses the database pathname for the files you wish to premigrate, pull down the Migration menu and select Premigrate and Copy (migbatch). This calls the `migbatch` command, which sweeps the file system and premigrates selected files. Later, VSM copies the data to secondary storage media as specified by the METHOD1 and METHOD2 parameters in the `migconf` configuration file. See Figure 10 on page 29, and the `migbatch(1M)` man page for more information.

Manage Free Space Threshold

You can examine and change the minimum percentage of available disk space you want to maintain in the managed server. From the main `xhsmadm` screen, select an *hsmname* that uses the database pathname you wish to manage.

To examine the current threshold on kernel-based implementations (Solaris *ufs* file systems), pull down the Configure menu and select Display Threshold. To change the current threshold, pull down the Configure menu and select Set Threshold per Configuration File. This invokes the `migthreshold -f` command for the selected *hsmname*. Additional `migthreshold` options are available through the command line interface; see man page `migthreshold(1M)` for more information. See also “High-Water Mark” on page 67 and “Assign File System Attributes” on page 133 for an explanation of the free space threshold.

The migthreshold command is not available on nonkernel-based implementations. You can change the free space threshold for a file system by editing the configuration file migconf and changing the free space attribute to the desired value. You must make this change using the xhsmadm interface, or the daemon will not recognize the change. See "Assign File System Attributes" on page 133.

Make Disk Space Available

After files are premigrated and copied to secondary storage, their disk space is still assigned on the managed server until the files are purged. From the main xhsmadm screen, select an *hsmname* that uses the database pathname for the files you wish to purge.

To purge files unconditionally from premigration and make disk space available, pull down the Migration menu and select Remove or Migrate (mignospace). This calls the mignospace command, which purges premigrated files from disk to increase free space. If there are no premigrated files to purge, mignospace initiates a migration operation to move additional files to secondary storage. See Figure 11 on page 31, and the mignospace(1M) man page for more information.

To purge files from premigration only if available disk space on the managed server is below the free space threshold, pull down the Migration menu and select Conditional mignospace (miglow). This launches the miglow command, which checks the free space threshold. If available disk space is less than the free space threshold, miglow first calls migbatch to premigrate and copy additional files to secondary storage, and then calls mignospace to purge premigrated files. See Figure 12 on page 33, and the miglow(1M) man page for more information.

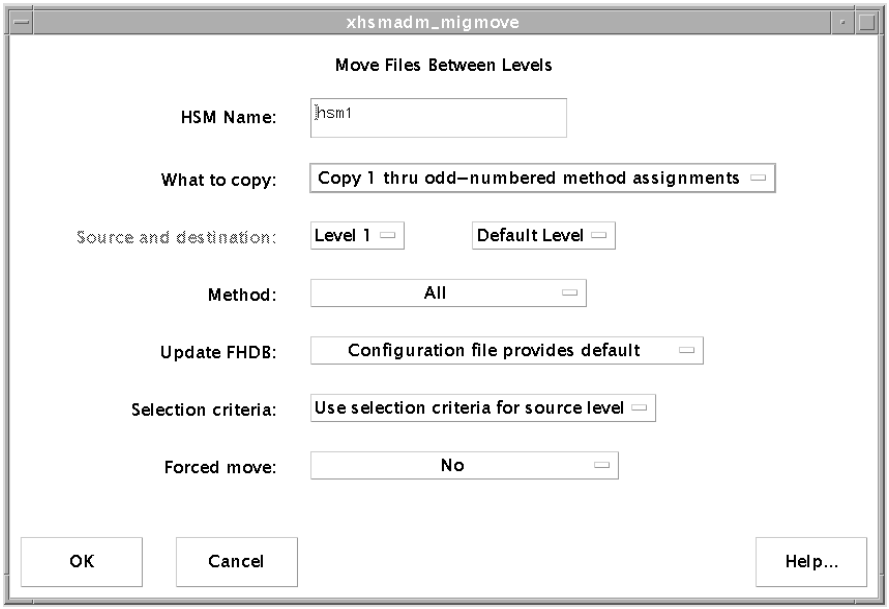
Move Files between Migration Levels

Initial migration to secondary storage sends the first copy to migration level 1 using storage method METHOD1. If two copies are configured, the second copy goes to migration level 2 using storage method METHOD2.

With VSM's multilevel migration feature, however, you can configure up to eight different migration levels. It is possible to move files between these migration levels at different times and in a variety of ways. See "Define Criteria for Moving Migrated Files" on page 80 for an explanation of multilevel migration.

1. From the main `xhsmadm` screen, select an *hsmname* that uses the database pathname for the files you wish to move to a different migration level. Pull down the Migration menu and select Move Between Levels. This opens the `xhsmadm` move files screen. See Figure 120 (below).

Figure 120. xhsmadm_migmove



2. Change the file move attributes to the desired values. The attributes in each entry are defined as follows:

   HSM Name

   The name of the selected file system.

   What to copy

   To use the default migration levels for copy 1, select Copy 1 thru Odd-numbered Method Assignments. To use the default migration levels for copy 2, select Copy 2 thru Even-numbered Method Assignments. To specify source and destination migration levels, select Specific Source and Destination.

   Source and destination

   Storage methods (METHOD1 through METHOD8) associated with a specified source migration level and destination migration level, where METHOD1 is associated with migration level 1, METHOD2 is associated with migration level 2, etc. The default destination migration level is source migration level plus 2.

Method

Method name used for selecting and moving files for all source migration levels processed. The default is all valid method names for all source migration levels processed.

Update FHDB

To leave the FHDB entries for the selected files *active* at the source migration level, select Leave FHDB Unchanged. To mark the FHDB entries for the selected files *obsolete* at the source migration level, select Mark Source Method File Handles Obsolete. To mark the FHDB entries for the selected files *dead* at the source migration level, select Mark Source Method File Handles Dead. (This option marks method name `ad` entries `obsolete`.) To mark the FHDB entries for the selected files according to what is specified in the `migconf` move flags (default), select Configuration File Provides Default.

3. Click the OK pushbutton.

Customize the VSM Policy and Method for Migrating Files

The administrator can customize the way VSM diverts specific files to suitable media. For example, smaller files can be migrated to optical disc and larger files to tape. Similarly, files from one owner can be migrated to one set of tapes, and those of a second owner sent to another. See “Choosing the Best Method” on page 90.

To do this, replace `/usr/openv/hsm/bin/admincmd/migpolicy.script` with an alternate script. A sample replacement is provided in `/usr/var/openv/hsm/example/database/sample.migpolicy.script`. This sample replacement script uses the size of a file to determine whether VSM will copy it to the first entry or to the second entry of a method definition.

For more information, see man page `migpolicy(1M)`.

Reconfiguring Storage Methods

Use this procedure switching for METHODx (METHOD1-8) to use a different method name. For example, you could change METHOD1 from using method name `ad` to using method name `dt`.

1. Using the interface, change the state of the managed file system to inactive.
2. Make sure all required copies have been made to the current method by executing:

   `migbatch -s` ***`hsmname`***

   Wait until this has finished before proceeding to the next step.
3. Execute

   `migdbcheck -F -r` ***`hsmname`***

If there are not enough copies and you are asked to run `migrc -R` also, do so now.

Note When the `migrc` command completes, other background processes may still be running. Wait for all recovery work to finish by checking to make sure `migrecover.sh` has completed.

4. Use multilevel migration to move any files you want to move from the current METHODx (optional).
5. Reconfigure the METHODx to be the way you want it.
6. Register volumes for the reconfigured METHODx.
7. Using the interface, change the state of the managed file system to active.

Files will now be migrated according to the reconfigured METHODx. Files previously migrated under the old METHODx can still be cached.

Performance Tuning

Tape Marks

The default behavior of VERITAS Storage Migrator when copying premigrated files to tape is to write a tape mark after every four G of file data (rounded up to include complete files) and after all files in the copy operation have been written to tape.

Note Writing tape marks more frequently degrades copy performance, and writing tape marks less frequently makes recovery from a system crash more difficult because the entire copy operation since the last tape mark must be repeated. In general, the default behavior is a good balance of these trade-offs.

To modify the default behavior and write tape marks differently, create the *dwpath*/database/*hsmname*.FLUSH file. This FLUSH file contains two values, separated by a blank.

value1 value2

where *value1* is the number of files to be copied before writing a tape mark, and *value2* (optional) is the number of kilobytes to be written before writing a tape mark. *Value1* = 0 is interpreted as no file limit, and *value2* = 0 is interpreted as unlimited kilobytes. Thus, if the FLUSH file contains a *value1 value2* of 0 0, Storage Migrator writes a tape mark only after all files are written.

Default tape marks are written in these situations:

- *hsmname*`.FLUSH` does not exist
- *hsmname*`.FLUSH` exists but is empty

- *hsmname* . `FLUSH` exists, *value1* = 0 and *value2* does not exist
- *hsmname* . `FLUSH` exists, *value1* = 0 and *value2* = 4104304 (4 Gbytes)

If both *value1* and *value2* are specified, a tape mark is written whenever either condition is met.

Constant Sweeps

You can tune VSM to perform constant sweeping of the managed file system instead of the normal sweeping process. To enable constant sweeping, execute this shell script:

`/usr/openv/hsm/bin/migconsweep [-s` *sleep_time*`]` *hsmname*

where `-s` *sleep_time* is the time in seconds that this command sleeps before resuming a sweep of the file system. Default is 60.

Constant sweeping uses system resources that may adversely affect overall VSM performance, particularly during periods of heavy system usage. Once initiated, constant sweeping continues to run until the process is terminated with the `kill` command. For more information, see

Partial File Caching

Implementations of VSM using the DMAPI interface support partial file caching. Partial file caching allows `read` access to a migrated file without caching the entire file. For more information, see “Partial File Caching” on page 17.

Partial file caching is preferable to total file caching if your applications read a small portion of the file data without reading the entire file. For more information on when to consider tuning your system with partial file caching, see “Total and Partial File Caching Trade-offs” on page 20.

Accelerated File Space Availability

The accelerated file space availability feature of VSM reduces the delay in freeing disk space when no premigrated and copied files exist. Rather than waiting for the entire process to run before making disk space available, VSM can optionally interrupt this process and purge files incrementally. For more information on when to consider tuning your system with partial file caching, see “Accelerated File Space Availability” on page 34.

VSM Export/Import Management

Note Storage Migrator Remote does not support export/import.

To move copies of migrated files from one VSM-managed file system to another VSM-managed file system, use the export/import feature of VERITAS Storage Migrator. See "File Export/Import" on page 40. VERITAS NetBackup is required for file export/import.

Make sure that all file systems exporting or importing files between them are distinct; each must be configured with a unique Machine ID. See "Assign Default Values" on page 127. This prevents the possibility of importing files that have the same file handles as files previously migrated on the importing file system.

Configure the appropriate migration level and corresponding method on the exporting system and copy files to that level before exporting them. Do the same on the importing system before accessing the imported files.

With export/import you move tape volumes or optical discs from one file system to another. Although VSM writes the data in the same format on different platforms, platform-specific device drivers can be incompatible with one another. For example, tapes written on HP-UX or Solaris platforms cannot be imported to IRIX platforms in all cases. If exporting optical media, the exporting and the importing file systems must run on like servers (the same platform type and operating system).

Planning File Exports

There are two ways to export files.

- Export all migrated and unmigrated files residing in the specified subdirectory (*dir*) in the managed file system.
- Export all migrated files residing at export level *ExpLevel*. In this case, do not specify the `-s` *dir* option. (Unmigrated files are not exported.)

If you want to export *only* migrated files from a subdirectory in the managed file system (*dir*), make sure that *all* files in that directory are migrated and copied. To do this, you can change to that directory and execute the following command:

```
find . -type f -print | xargs fls -l | grep -v "\[machid"
```

where *machid* is the machine ID displayed by the `fls` command for any migrated file in the same file system. See man page `fls(1)`. If all files have been migrated, there is no output from this command. To verify that all these migrated files have the required number of copies, run the `migdbcheck` command and look for the absence of messages such as this:

```
-- INFO: 2 files have less than 2 copies made.
```

Note See man page `migdbcheck(1M)`.

Files are exported from a migration level. Valid values are 1 to 8. Default is 8. You can also export files from a subdirectory in the managed file system. In this case, the `mignbexport` command moves the files to be exported to the export migration level first and then exports them. Make sure there are no extraneous active entries at the export migration level before executing the export command because all volumes at that level will be exported.

`mignbexport` does a user directed backup of the exported files and the FHDB entries and VOLDB entries for those exported files. The backup is done to a NetBackup class that must be defined in the NetBackup configuration prior to executing `mignbexport`. The NetBackup class must define a NetBackup volume pool that contains volumes that can be sent to the site that is doing the `mignbimport`. See "Registering Volumes for nb NetBackup Method" on page 168 for more information.

After running `mignbexport`, send the following items to the administrator of the importing file system:

- The VSM volumes listed in *dwpath*`/database/Volumes_to_export.`*pid*
- The NetBackup volumes belonging to the specified NetBackup class which were used during the `mignbexport` operation
- The NetBackup client name contained in *dwpath*`/database/Client_name.`*pid*

See man page `mignbexport(1M)` for more information and examples of this command.

Planning File Imports

Files are imported to a migration level. Valid values are 1 to 8. Default is 7.

The mount point (*fspath*) as defined in `migconfg` of the importing file system replaces the mount point as defined in `migconfg` of the exporting file system. The paths for the imported files below the managed directory as defined in `migconf` of the importing file system are identical to the paths for the exported files below the managed directory as defined in `migconf` of the exporting file system. See Figure 53 on page 133. These managed directories can be either at or below the mount point for the entire VSM-managed file system. See "File Systems to Manage" on page 58.

`mignbimport` does a user directed NetBackup restore of the exported files and the FHDB entries and VOLDB entries for those exported files. This restore also includes information about the original path of the exported VSM-managed filesystem.

The restore is done from a NetBackup class that must be defined in the NetBackup configuration prior to executing `mignbimport`. This class must have the same name as the class used with the `mignbexport`. The NetBackup class must reference a NetBackup

volume pool that contains only the NetBackup volumes that were written by `mignbexport`. See “Registering Volumes for nb NetBackup Method” on page 168 for more information.

The NetBackup volumes written by `mignbexport` must be imported into NetBackup at the site importing the files prior to running `mignbimport`. Use the NetBackup client name found in *dwpath*`/database/Client_name.`*pid*, sent from the exporting site. For more information on importing NetBackup images, see Chapter 6 of the *VERITAS NetBackup System Administrator's Guide*.

Place the VSM volumes sent from the exporting site in the appropriate storage devices and register them with Media Manager for the importing VSM-managed file system before running `mignbimport`. If this is not done, Media Manager will issue requests for the operator to assign this media. These volumes retain their original volume name so they must be unique. Do not register these volumes to VSM because `mignbimport` will take care of this automatically.

VSM Databases

The VSM database and workdir directories contain the VSM databases in addition to the configuration files. See Figure 17 on page 43.

You specify the path *dwpath* to these directories during configuration. See “Perform Global Configuration” on page 123 or “Advanced Wizard- Hierarchy Properties” on page 208.

Caution VSM databases in *dwpath* (except for the `.IHAND` file) consist of text files that you can view with any text file editor. Be very careful not to alter or otherwise damage any VSM database you view in this way. Doing so can make it difficult if not impossible to retrieve migrated files.

The topics in this section describe the contents of the database files on a managed server and also on remote volume servers.

Databases on a Managed Server

A complete list of databases on the managed server follows Figure 17 on page 43. Some of the principal databases are described below.

See “Database Problems” on page 311 for information on resolving problems with the VSM databases.

File-Handle Database (FHDB)

a VSM file handle is a unique sequence number that makes it possible to locate all copies of a migrated file, regardless of the storage methods used. The file-handle database (FHDB) contains one database entry for each copy of a file. If VSM divides the copy into smaller parts called granules, the database contains an entry for each granule of each copy.

The main processes that add entries to the file-handle database are those that premigrate files and copy them to secondary storage. During premigration, VSM extends the file-handle database as it moves each file into premigration. Then, it creates DVDB (destination-volume database) entries as it copies the files to secondary storage. After the copy phase completes, VSM merges the DVDB entries into the file-handle database.

Each file-handle database entry contains the following fields:

```
handle|machid|flags|volid|lock|size|offset|gransize|crc|uid|gid|
arch_date|copy_date|obsdate|method|path|hostname|user|group|hint|
seek_info|fh_seek_increment|comment|inode|rep_FHDB|mmlevel|
```

The *project_name* field in earlier VSM releases becomes the *fh_seek_increment* field. This field is used in conjunction with the *rep_FHDB* field to track migrated files with fewer FHDB entries than previously needed.

The *flags* field (in hexadecimal) is particularly important in determining the status of an entry. Flags can appear in combination.

`00000000` Entry is active

`00000002` Item failed; possibly corrupted

`00000004` Entry references obsolete data or volume

`00000008` Entry is dead

The *path* field is the full path to this file.

The *comment* field in the FHDB is used by the `migtie` command to indicate a group name.

The following are examples of FHDB entries for a single copy of a single file:

```
00001D93|000003E8|00000000|0000105A|00000000|007A1200|00000000|
00200000|0000D55B|00000000|00000000|335DFD17|335DFDE3|00000000|dt|
/xhsm1/acg/m8f1|hat|root|sys|library|2 0|41|Auto HSM run|
00000000|00000002|00000001|
```

```
00001D93|000003E8|00000000|0000105A|00000000|007A1200|00600000|
001A1200|000071BD|00000000|00000000|335DFD17|335DFDE4|00000000|dt|
/xhsm1/acg/m8f1|hat|root|sys|library|2 195||Auto HSM run|
00000000|00000000|00000001|
```


Volume Database (VOLDB)

The VSM volume database (VOLDB) contains one entry for each registered volume. The first entry is the disk entry, and it is required. VSM creates subsequent entries whenever you register a volume for one of the other methods.

VSM updates the database whenever it selects a volume to which it will migrate files and updates the database again when the copy is complete. If the copy operation fails and VSM does not update several of the fields in the database, this does not cause a problem.

Each volume database entry contains the following fields:

```
handle|machid|lock|flags|label|method|location|metric|date|size|
unused|inuse|files|volset|lt_file|blocks|gid|user|group|
project_name|pool_name|server_name|server_user|server_password|
```

The *migreg* command redefines three VOLDB fields when registering a NetBackup class as a volume for use by method name `nb`. The *location* field holds *class_name*, the *server_user* field holds *schedule*, and the *server_password* field holds *client_name*.

The *flags* field (in hexadecimal) is particularly important in determining the status of an entry.

`00000000` Volume can contain data; it is not assigned for writing

VSM clears the flags field either when the volume becomes full or when VSM encounters problems reading from a volume.

`00000010` Physical volume is dead - will not be used

(shows in `migdbrpt` output as `D`)

`00000020` Volume is empty and available for use

(shows in `migdbrpt` output as `E`)

`00000040` This volume assigned for writing copy

(shows in `migdbrpt` output as `W`)

`00000080` Do not write to this volume

(shows in `migdbrpt` output as `f`)

`00001020` This volume needs labeling when used

(shows in `migdbrpt` output as `L`)

`00002020` This volume needs FORCED label when used

(shows in `migdbrpt` output as `F`)

`00002000` This volume is corrupted

(shows in `migdbrpt` output as `C`)

Note VSM does not automatically set the corrupt flag. It must be set by `migsetdb`.

The following are example volume database entries:

```
00000000|000003E8|00000000|00000000|DK0000|dk||0|0000000A|
01048576|01048576|00000000|00000000|00000000|00000000|0|0|||||||


00001002|000003E8|00000000|00000040|EXB003|ct||0|2D0C6B51|
0047E000|0047D9C0|0000049E|0000000D|00000001|00000000|0|0|||||||
```

Work Lists (copydb files)

VSM uses work lists called copydb files to copy files to secondary storage and to move migrated files between migration levels. The names of `copydb` files have the form:

hsmname`.copydb.`*method_name*`.`*vol_set_number*`.`*hint*

For example:

```
alpha.copydb.ad.1.library
```

Entries in a `copydb` file represent files waiting either to be copied from premigration to secondary storage or to be moved from a source migration level to a destination migration level. Each `copydb` entry contains the following fields:

```
handle|machid|lock|flags|volset|copied|method|dst_seek|
age|size|badness|path|hint|comment|mmlevel|slevel|
```

The *copied* field remains clear until the copy (or move) operation is complete, at which time it is set to 1.

The *flags* field (in hexadecimal) is defined as follows:

`00000000` File has not been copied (moved) if *copied*=0

Copy (move) operation is completed if *copied*=1

`00000100` Copy (or move) operation has failed

The *path* field is the full path to this file.

Source and destination migration levels for multilevel migration are the *slevel* and *mmlevel* fields, respectively.

The following are example work list entries:

```
0000237F|000003E8|00000000|00000000|00000001|00000001|op||0.00|1|
0|/home3/abcd/reports/1Q|library|Auto HSM run|00000001|00000000|


0000237F|000003E8|00000000|00000000|00000005|00000001|ct||0.00|1|
0|/home3/abcd/reports/1Q|library|Auto HSM run|00000003|00000001|
```

The former represents the original migration of a file to optical storage, and the second represents a subsequent move of that file from migration level 1 to tape on migration level 3.

Destination-Volume Database (DVDB)

As VSM completes copying migrated files to secondary storage, it creates temporary file-handle database entries in a DVDB file. When all files are copied, VSM merges the DVDB entries into the file-handle database and deletes the DVDB file.

You seldom see DVDB files. However, if a problem prevents VSM from merging all temporary entries into the file-handle database, the DVDB file is left in the database directory. In this case, no data is lost. You simply execute `migrc -R`, which causes VSM to complete the unfinished copy and merge processes before deleting the DVDB file.

File-Handle-Database Lock File (FHDB.LK)

The file-handle-database Lock file (FHDB.LK) does not contain any data. It is used to provide a master lock on the file-handle database. Most processes use a shared lock on the file-handle database. The merge database process uses an exclusive lock. This ensures that the file-handle database is not being accessed while it is being sorted and merged.

File-Handle Sequence File (FHSEQF)

The file-handle sequence file (FHSEQF) contains the 6-digit hexadecimal value that VSM assigns to the next file handle.

For example:

```
00161e
```

Volume-Database Lock File (VOLDB.LK)

The volume-database Lock file (VOLDB.LK) does not contain any data. It is used to provide a master lock on the volume database. Most processes use a shared lock on the volume database. The merge database process uses an exclusive lock. This ensures that the volume database is not being accessed while it is being sorted and merged.

Volume-Sequence File (VOLSEQF)

The volume-sequence file (VOLSEQF) contains the 6-digit hexadecimal value that VSM assigns to the next volume ID (handle).

Next-Volume-Set Files (NEXTVOLM1...NEXTVOLM8)

The NEXTVOLM1 file contains the number of the volume set that VSM will select on the next migration to a volume for METHOD1. The NEXTVOLM2 file contains the number of the volume set that VSM selects on the next migration to a volume for METHOD2. The remaining files, NEXTVOLM3 through NEXTVOLM8, contain the number of the volume set to use for moving a migrated file to migration level 3 through 8, respectively.

migsweep.site

This program allows an intercept during `migsweep` processing if the site wants to do something other than the standard VSM badness or purge badness calculation. The same site-specified badness formula applies to both the selection of files to be migrated and the purging of files already migrated. See “Assign File System Attributes” on page 133 and Figure 54 on page 136 for information on how to select a site-specified badness formula. This intercept calls `migsweep.site` for each file that meets minimum age and size parameters. The input to `migsweep.site` is one parameter in the following format:

```
file-name|age-in-days|size-in-kilobytes|current-badness
```

Note For Solaris *ufs* file systems, the file name is the name of the file in the `migration/data` directory for calls to `migsweep.site`.

This routine must echo a true false migration flag to stdout. Migrate if output is 0. Do not migrate if anything else.

`migsweep.site` can be any type of program or script.

migsweepm.site

This program allows an intercept during `migsweepm` processing if the site wants to do something other than the standard VSM move badness calculation. See “Configure and Edit Storage Method Names” on page 139, and Figure 56 on page 143 or “Configure and Edit Migration Levels” on page 145 and Figure 57 on page 146 for information on how to select a site-specified move badness formula. This intercept calls `migsweepm.site` for each file that meets minimum move_age and move_size parameters. The input to `migsweepm.site` is one parameter in the following format:

```
file-name|age-in-days|size-in-kilobytes|current-badness|method|level
```

This routine must echo a true false migration flag to stdout. Move file if output is 0. Do not move file if anything else. `migsweepm.site` can be any type of program or script.

migconf

The `migconf` file is the configuration file containing migration criteria for the file systems using this database. See “Configure Migration Parameters” on page 126 or “Advanced Wizard- Filesystem Properties” on page 223 for detailed instructions on how to configure VSM.

Inode to Handle File (`.IHAND`)

The *hsmname* `.IHAND` file contains inode and handle information about migrated files (nonkernel-based implementations only). See “Administer Inode-to-Handle Files” on page 152 for a detailed description.

FLUSH

The *hsmname* `.FLUSH` file controls how often VERITAS Storage Migrator writes tape marks during file migration. See “Tape Marks” on page 296.

Databases on a Remote Volume Server

ID_LABEL File

ID_LABEL files exist on remote volume servers that have `ft` volumes. Each file system that is registered as an `ft` volume contains an ID_LABEL file that contains a single line of text identifying the label and the client name for the managed server. When you register the volume with the `migreg` command the file and label are created.

For example, if the file system is named `/hsmftvol1`, then `/hsmftvol1/ID_LABEL` is the path to the `ID_LABEL` file that the `migreg` command creates. If you register the volume to the server named `kiran` and use the label `kiranft1`, then the file entry is `kiran:kiranft1`.

When the managed server attempts to access the file system for migration and cache operations, it uses the contents of the ID_LABEL file to verify that the file system is registered to that managed server and only that system. Multiple registrations are not permitted.

Problem Solving

The following topics provide information on resolving problems you can encounter when using VSM.

If VSM manages more than one file system or directory, you can perform maintenance on only one while the others remain active. To do this, go to the main `xhsmadm` screen, select the *hsmname* for the file system or directory you want to work on, and toggle the State to

Inactive. When maintenance is complete, toggle the State back to Active. If using the VSM-Java interface, deactivate and activate the managed file system from the Actions menu.

Checking and Managing the Logs

The first step in resolving VSM problems is to check the log files. The messages in the logs can contain information that points you to a solution. In particular, look for any instances of ERROR or WARNING.

Note Using the `migchecklog` command is a convenient way to check VSM log files.

VSM provides two log files that you can use during troubleshooting:

- Global log (`/tmp/hsm.log`) which contains messages pertaining to all VSM configurations. Link the global-log file to a file not under `/tmp`. This ensures that the system maintains the log between system boots.
- Individual *hsmname* logs, which contain messages for specific managed file systems. You configure the path and name for the individual logs (see "Perform Global Configuration" on page 123).

If the logs accumulate too much data, you can use the `mignewlog` command to either:

- Truncate the desired log file

  or

- Copy the desired log to another file before truncating it

With either option, VSM recreates the log file for the first new log message that occurs. The copy option is useful if you want to start with a fresh log file, but need a record of previous log messages. See the man page for more information on the `mignewlog` command.

To change the level of log messages, go to the main `xhsmadm` screen, pull down the File menu and select Set Logging Level. If using the VSM-Java interface, set the logging level from the Actions menu.

Caution Avoid setting logging level to 9. This will cause all messages to include the message ID at the start of each line which will break some software operations in VSM.

Valid values are 1 through 9, with higher numbers logging more information. Default is 3.

Media and Database Information and Reports

There are several commands that you can use to obtain reports on VSM media and databases.

Volume Scan Reports

The `migtscan` command provide information on tape volumes (`ct`, `mt` or `dt` method). `migopscan` is for optical disc volumes (`op` or `ow` method), `migadscan` is for alternate magnetic disk volumes (`ad` method), `migftscan` is for remote volumes (`ft` method), and `mignbscan` is for NetBackup volumes (`nb` method).

The information includes data about the volume as a whole (for example, total capacity) and about individual granules on each volume (for example, granule size). You can use the information to resolve inconsistencies in the file-handle and volume databases.

See man pages `migtscan(1M)`, `migopscan(1M)`, `migadscan(1M)`, `migftscan(1M)`, and `mignbscan(1M)` for specific information about these commands.

Migration Database Report

The `migdbrpt` command provides information on both files and volumes. For example, you can produce a report that lists all the files on all volumes in a specific volume set. Or, you can request a report on the basis of a specific file handle or file path. See man page `migdbrpt(1M)` for instructions and examples on using this command.

Volume Usage Report

The `miggetvol` command generates a list of volumes sorted by method name in ascending order of percentage used. This command can help you to determine when to consolidate volumes. See manpage miggetvol(1M) for instructions and examples on using this command.

Global Configuration Information

The `migdbdir` command allows you to display information about a specific hsm on standard output. For example, you can display the database path and file system. See man page `migdbdir(1M)` for instructions on using this command.

Recovering from a System Crash

To restore a VSM-managed file system after a system crash, perform the following steps. In this example, the path for managed file system `alpha` is configured as `/hsm1` in the global configuration file.

Stop the daemons and run `startmigd`.

1. Run `fsck` to fix each VSM-managed file system.

a. If `fsck` does not find any errors, do the following:

- On kernel-based implementations (Solaris *ufs* file systems), mount the file system and then execute `startmigd` to start the VSM migration and volume daemons.
- On nonkernel-based implementations, execute `startmigd` to start the VSM migration and volume daemons and then mount the file system.
- Run *migrc -L alpha* to clear locks.
- If there is incomplete migration work, use `migrc -R alpha` to finish that work or wait until off-peak hours to do so.

b. On kernel-based implementations (Solaris *ufs* file systems), if `fsck` makes a trivial fix to the file system, do the following:

- Unmount the file system. Mount the file system as an *ufs* file system.
- Run `pfcheck /hsm1` and use the new `.PAIN` file.
- Unmount the file system and remount it as an *hsm* file system.
- Run `migrc -L alpha` to clear locks.
- If there is incomplete migration work, use `migrc -R alpha` to finish that work or wait until off-peak hours to do so.
- Run `startmigd` and the system will be ready for use.

c. On kernel-based implementations (Solaris *ufs* file systems), if `fsck` fails to fix the file system (disk head crash etc.), then do the following:

Caution The following procedure restores the system to a previous level, but some data can be lost.

- Make a new file system (using `mkfs` or `newfs`) and create a `.PAIN` file. See “Creating .PAIN Files” on page 149.
- Restore the database directory to the same level backup as the file system. See “Recovering File-Handle and Volume Databases” on page 313.

- Restore the file system from a backup tape, but do not restore the `.PAIN` file. See "Restoring VSM-managed File Systems" on page 310.
- Run `startmigd` and the system will be ready for use.

Restoring VSM-managed File Systems

Use the following procedures to backup a damaged or incomplete VSM-managed file system and then restore the entire file system to a previous backup level which is neither damaged or incomplete:

To Backup the Current File System

1. Make sure the VSM daemons `migd` and `migvold` are running.
2. Purge all premigrated files. To do so, run `migbatch` to write all premigrated files to volumes, and then run `mignospace -i` to purge premigrated files.
3. To be safe, back up the current incomplete VSM-managed file system to tape, making sure to exclude the `.PAIN` file on kernel-based implementations (Solaris *ufs* file systems), and the .IHAND file on nonkernel-based implementations. Also backup the VSM database directory. See "Backing Up VSM Databases and Managed File Systems" on page 260.

To Restore a Previously Backed Up File System

Caution The following procedure restores the system to a previous level, but some data can be lost.

1. Toggle the State for this *hsmname* to Inactive, or stop the VSM daemons by executing `stopmigd`.
2. Create the new file system (using `mkfs` or `newfs`). The entry in the global configuration file must not be changed.
3. Create the `migration/data` directory in the new file system.
4. On kernel-based implementations (Solaris *ufs* file systems), create the `.PAIN` file for the new file system. See "Creating .PAIN Files" on page 149.
5. Remount as an *hsm* file system.
6. Toggle the State for this *hsmname* to Active, and start the VSM daemons by executing `startmigd`.

Caution If you are restoring a file system that contains a backed up `.PAIN` or `.IHAND` file, you must exclude it.

7. Restore the most recent, previously backed up VSM database directory.
8. Restore the most recent, previously backed up file system, *not* the current damaged or incomplete VSM-managed file system.
9. Run the startup script `migrc -Lo` *hsmname* to clear locks and remove obsolete and dead database entries in VSM databases.

Extending VSM-managed File Systems

You can expand the size of an existing file system by increasing the number of available inodes.

Caution Stop all activity on the managed file system before extending it.

If you do add inodes to a managed file system in a nonkernel-based implementation, the `.IHAND` file will grow as needed to accommodate the larger file system. In kernel-based implementations (Solaris *ufs* file systems) you must also extend the `.PAIN` file for that file system. See "Extending a .PAIN File after Adding Inodes to the File System" on page 149.

Database Problems

Just like any other file, the FHDB and VOLDB are susceptible to errors due to system crashes or premature termination of programs that lock and unlock database entries. The topics in this section explain how to fix database problems and also how to recover a lost database.

Caution The FHDB and VOLDB are text files that you can view and modify with a text editor. However, before modifying any VSM database, be sure to stop the VSM daemons and any VSM processes that are executing.

Fixing the File-Handle Database

Symptoms that indicate problems with the file-handle database are:

- VSM processes wait indefinitely for FHDB locks. This is reported in the VSM log files.
- Migrated files (with file handle) are not in the FHDB.
- Removed files still have active entries in the FHDB.

If any of these conditions exist for file system *hsmname*, perform the following steps:

1. Toggle the State for this *hsmname* to Inactive, or stop the VSM daemons by executing `stopmigd`.
2. Execute `migrc -LR` *hsmname* to clear all locks and complete any interrupted migrations.

Note Use migdbcheck only if you suspect corruption of the FHDB due to a system crash or other malfunction, or if you have restored either the file system or the FHDB and you suspect that there may be a mismatch. You do not have to run this command under normal conditions.

3. Verify consistency between the file-handle database and the managed file systems by executing the `migdbcheck` command. This command checks that:
   - The managed file systems do not contain any migrated files that do not have a file-handle database entry.
   - The file-handle database does not contain entries for any files not present in the managed file systems.

   See man page `migdbcheck(1M)` for instructions on using this command and correcting problems.
4. Verify consistency between the media and the file-handle database by executing the `mediacheck` command. This command verifies that files recorded on the media also have entries in the file-handle database. See man page `mediacheck(1M)` for instructions on using this command and correcting common problems.
5. Toggle the State for this *hsmname* to Active, or start the VSM daemons by executing `startmigd`.

Clearing File-Handle Database Locks

VSM processes maintain locks in the file-handle and volume database. If a process fails, it can leave a lock set and delay processes that are waiting for the locks to clear. If this occurs:

1. Use `HSMKiller` *hsmname* to kill any processes hung waiting on a database lock.
2. Clear the locks:
   - Toggle the State for this *hsmname* to Inactive, or stop the VSM daemons by executing `stopmigd`.
   - Execute `migrc -L` *hsmname*
3. Toggle the State for this *hsmname* to Active, or start the VSM daemons by executing `startmigd`.

Fixing the Volume Database

VSM records all its volumes in the VSM volume database (VOLDB) and updates VOLDB entries as it uses volumes for migration. Each VOLDB entry shows volume status, used space, and free space available in the volume.

VSM keeps the file-handle database (FHDB) and volume database synchronized. However, a system crash or other malfunction can cause empty volume database entries or file-handle database entries that have no corresponding entry in the volume database. In either case, VSM cannot cache the affected files or granules from the volume and VSM must use the second copy for caching (if one exists).

Perform the following steps if you suspect problems with the volume database:

1. Toggle the State for this *hsmname* to Inactive, or stop the VSM daemons by executing `stopmigd`.
2. Execute `migrc -L` *hsmname* to ensure that there are no database locks set.
3. Execute `mignospace -i` to ensure that there are no premigrated files.
4. Execute the `migdbcheck` command to check for consistency between the volume-database and file-handle database.
   - Use the `migdbrpt` command to obtain more information.
   - See the man page for instructions on using `migdbcheck` and correcting common problems.
5. Toggle the State for this *hsmname* to Active, or start the VSM daemons by executing `startmigd`.

Recovering File-Handle and Volume Databases

Perform the following steps to recover a lost file-handle or volume database.

1. Restore backup copies of all files in the *dwpath*`/database` directory.

Caution Restoring a previous VOLDB can cause the following problems:
- Loss of any volumes that were registered after the backup.
- If a tape or optical volume is written after the backup, the restored VOLDB will not be able to migrate to that volume.
- If an ft volume is written after the backup, the restored VOLDB will not have the correct available space for the volume.

Caution If you do not restore the FHSEQF, you may overwrite existing file handles. See "Cannot Find Next File Handle" on page 314.

2. Check your migration schedules to determine the migrations and volumes used since the last backup.

3. Depending on your system configuration, execute the appropriate scanning commands to scan the relevant volumes and use the information you obtain from those commands to reconstruct the FHDB and VOLDB. See the following man pages for more information: `migadscan(1M)`, `migtscan(1M)`, `migopscan(1M)`, `migftscan(1M)`, and `mignbscan(1M)`.

If necessary, you can use information from these scan commands to totally reconstruct the databases. However, it means scanning all volumes rather than just those used since the last backup.

Cannot Find Next File Handle

If the FHSEQF file is lost, VSM is unable to assign the next file handle and starts logging errors in the `/tmp/hsm.log` file. To correct the problem, check the end of the FHDB file to determine the next file handle. Write that handle to the FHSEQF file. Note that the file handles (hexadecimal format) in the FHDB use uppercase letters while those in the FHSEQF file use lowercase letters.

File Problems

Reloading Deleted Files

If a migrated file is deleted, it is possible to reload it from the VSM volume, providing the volume has not been consolidated since the file was removed.

Caution Be extremely careful not to corrupt the file-handle database when searching for file-handles.

1. Check the file-handle database (FHDB) to find the file handle that VSM assigned to the file. See "File-Handle Database (FHDB)" on page 301 for a description of the file-handle database.

   For example, assume that you are trying to reload the `/home2/jdoe/tprog/proga` file from `bunny` at the example site. This file is under the management of hsm2 and the machine ID is 3E8 (1000 decimal). You find the following entry in the file-handle database for `proga`:

```
000012D2|000003E8|00000000|00001002|00000000|00000005|00000000|
00000005|00001C18|00000000|00000001|2D0C5211|2D0C62F2|00000000|ct|
/home2/jdoe/tprog|proga|bunny|root|daemon|library|1 0||

Auto HSM run|00000000|00000000|00000001|
```

   The first field is the file handle (`000012D2`) and the second is the machine ID (`000003E8`).

2. Use the `migreconstruct` command to reconstruct deleted migrated files. See "Reconstructing Migrated Files" on page 316 and man page `migreconstruct(1M)` for more information.

   After *migreconstruct* completes, try to cache the file. If this fails, try the manual method outlined in step 3.

3. Restore files by executing:

   `migin` *hsmname filesystem machine-ID file-handle*

   For example, using the file handle and machine ID values from step 1:

   `migin hsm2 /home2 000003e8 000012d2`

   For kernel-based implementations (Solaris *ufs* file systems), VSM restores the file to premigration The name is of the form:

   machine-IDMfile-handle

   In our example, the file appears in the `/home2/migration/data` directory as:

   `3e8M12d2`

   Move this file to the appropriate location.

   `mv /home2/migration/data/3e8M12d2 /home2/jdoe/tprog/proga`

   For nonkernel-based implementations, VSM restores the file to its original location in the file system.

Users Cannot Open or Delete Migrated Files

If users are unable to open or delete migrated files, execute `startmigd -m` to start the migration daemon `migd`. Also check that the `state` parameter in the related *hsmname* entry is set to `1` (Active). When the files are on an `ft` remote volume, ensure that the managed server is able to access the file system on the remote volume server.

Reconstructing Migrated Files

The `migreconstruct` command lets a system administrator reconstruct migrated files either when they have been accidentally deleted or when the file system is damaged beyond repair. The preferred way to do this is to restore migrated files from the their backup copies, created previously by NetBackup. If no backup copies exist, however, use `migreconstruct` to recover the migrated files.

Before running *migreconstruct* in a file system that is damaged beyond repair, the administrator must first reinitialize the file system. Use `fsck()`, `newfs()`, or *mkfs()* as necessary. In addition for kernel-based implementations (Solaris *ufs* file systems), the `.PAIN` (parallel inode) file must be created, the path to the managed directory must exist, and the `migration/data` directory must be created in the managed directory.

For more information on this command, see man page `migreconstruct(1M)`.

Migration Problems

Restarting Migrations

If the system fails during migration or the migration does not finish (for example, if there were not enough tapes) execute `migrc -LR` to resume the migration operation after correcting the problem.

Automatic Removal or Migration Does not Occur

If `mignospace` does not automatically start when free space percentage is below the high-water mark percentage, check that the `migd` (migration daemon) and `ltid` (device manager daemon) are both running. If either daemon is stopped, the automatic removal or migration cannot occur. Use `startmigd` and `ltid` to start them. Also check that the `state` parameter in the related *hsmname* entry is set to `1` (Active). When files are being migrated to an `ft` remote volume, ensure that the managed server is able to access the file system on the remote volume server.

Stop the daemons and run `startmigd`.

Media Problems

Releasing VSM Tape Requests

If a tape process aborts with a tape left mounted, issue a `tpunmount` on the file or use the `HSMKiller` command to unmount the tape. Check the `/usr/var/openv/hsm/workdir` directory for the name of the requested tape file.

Not Enough Volumes Available

If there are no available volumes at the time of a `migbatch` run, the `lgpath` file indicates that a volume is not available. This stops the `migbatch` process. Use `migreg` to register additional media, and then execute `migbatch` again to resume migration.

Capacity Licensing

VSM licenses are capacity-based. Licensing is enforced when `migcopy` migrates data using the following storage methods: optical (`op` and `ow`), tape (`ct`, `dt`, and `mt`), and NetBackup (`nb`). Capacity includes active and obsolete granules, but not any granules marked dead. See man page `miglicense(1M)` for information about how to display both the current license capacity of VSM and the actual current capacity used.

VSM issues a warning message when current capacity exceeds 90% of licensed capacity, and an error message when current capacity exceeds licensed capacity. When current capacity exceeds licensed capacity, VSM suppresses further migrations with `migcopy` until additional license keys are added.

The VSM-Java interface displays information dialogs when the following conditions occur:

You are nearing the licensed capacity limit.

You are over the licensed capacity, migrations supressed.

You have not yet installed a capacity license key.

After acknowledging these messages, select the appropriate item from the Actions pull-down menu of VSM-Java.

Add License Key...

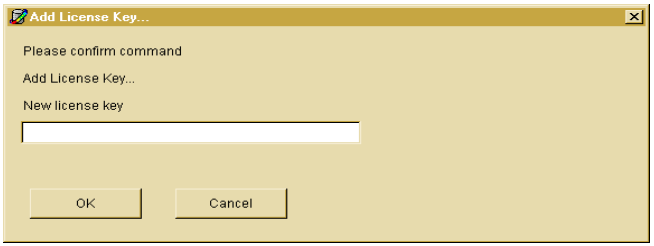
Show License Key

Show Capacity Usage

The last two menu choices both execute a job that appears in the bottom panel of VSM-Java. When the job completes, click the activity line to display the output text in the right panel.

To install a capacity license or increase your licensed capacity, select Add License Key... from the Actions pull-down menu. This opens a dialog for adding the text of the new license key.

Figure 121. Add License Key



Obtain license keys from your VERITAS representative.

Frequently Used VSM Commands

The man pages in appendix A of this manual provide detailed information about frequently used VSM commands.

Managing File Migration

`migbatch(1M)`

Premigrate files and copy to secondary storage

`mignospace(1M)`

Purge or migrate files to make disk space available

`migthreshold(1M)` -- (Kernel-based implementations only)

Set high-water mark to a new value until next boot

`miglow(1M)`

Start mignospace if file system is above high-water mark

`migpolicy(1M)`

Specify policy and method to write files to secondary storage

`miglicense(1M)`

Display the current license capacity of VSM

Managing Multilevel Migration

`migmove(1M)`

Move migrated files from one migration level to another level

`migsetdb(1M)`

Alter the flags field in the FHDB or the VOLDB

Managing File Caching

`migin(1M)`

Cache a file from secondary storage to disk

`migstage(1)`

Pre-cache files to avoid caching delays

`migtie(1)`

Designate the key caching file(s) for a caching group

Managing Media

`migreg(1M)`

Register and label volumes for VSM

`migcons(1M)`

Consolidate VSM tape and optical volumes

`migselect(1M)`

Select volumes for consolidation based on specified criteria

`migrecycle(1M)`

Reregister an empty volume

`migmdclean(1M)`

Remove obsolete entries from media

`migftscan(1M)`

Scan an `ft` remote volume and reconstruct database entries

Managing VSM Processes

`startmigd(1M)`

Start migration and volume daemons

`stopmigd(1M)`

Stop migration and volume daemons

`HSMKiller(1M)`

Kill active VSM processes and release tape requests

`migrc(1M)`

Clear locks, remove defunct lock files, restart migrations

`migconsweep(1M)`

Enable constant file system sweeping

User Controls

`fls(1)`

List and show migration status of files

`gethsm(1)`

Display the *hsmname* for files and directories

`migcat(1)`

Concatenate and display migrated files without caching them

`miggroup(1)`

Group files for migration and caching -- (DMAPI implementations only)

`migloc(1)`

Show location of migrated file

`migmode(1)` -- (Kernel-based implementations only)

Set mode for accessing migrated files

`migrate(1)`

Premigrate a specific file or files

`migpurge(1)`

Purge a specific file or files

`migstage(1)`

Pre-cache files to avoid caching delays

`migtie(1)`

Designate the key caching file(s) for a caching group

Exporting and Importing Migrated Files

`mignbexport(1M)`

Export VSM files

`mignbimport(1M)`

Import VSM files

Managing Databases

`migdbcheck(1M)`

Check the FHDB and/or the VOLDB for consistency with the file system

`mediacheck(1M)`

Check the media and FHDB for consistency

`migsetdb(1M)`

Alter the flags field in the FHDB or the VOLDB

Obtaining Reports

`migadscan(1M)`

Provide information on contents of archive disk volumes

`migtscan(1M), migopscan(1M)`

Provide information on contents of tape or optical volumes

`miggetvol(1M)`

List volumes in ascending order of percentage utilization

`migdbdir(1M)`

List global configuration values from `migconfg`

`migdbrpt(1M)`

Provide FHDB and VOLDB information on files and volumes

`migchecklog(1M)`

List the most recent messages from an error log file

`migftscan(1M)`

Scan an ft remote volume and reconstruct database entries

`mignbscan(1M)`

Scan a NetBackup volume and reconstruct database entries

Managing Logs

`mignewlog(1M)`

Copy or delete global or individual VSM log files

Tuning Migrations

`migconsweep(1M)`

Enable constant file system sweeping

`migtestbadness(1M)`

Evaluate configuring different file system attributes

Managing a Kernel-based Implementation

`pfcheck(1M)`

Check and fix the `.PAIN` file

`pfinit(1M)`

Create or extend the `.PAIN` file

`pfprint(1M)`

Print or alter the `.PAIN` file

Managing a Nonkernel-based Implementation

`ihprint(1M)`

Print or alter the `.IHAND` file

`rebuild_ihand(1M)`

Rebuild the `.IHAND` file from the FHDB

`migalter(1M)` -- (DMAPI implementations only)

Display or alter regions, events, or attributes

`migcleanup(1M)` -- (DMAPI implementations only)

Display or clean up DMAPI sessions

Recovering Data

`migreconstruct(1M)`

Reconstruct damaged or deleted migrated files

`migin(1M)`

Restore a file from secondary storage to disk

Troubleshooting

`migfind(1M)`

Determine the full pathname of a file

Graphical User Interface

`xhsmadm(1M)`

Administrative graphical user interface (Motif-based)

Man Pages A

This appendix contains man pages for the commands for VERITAS Storage Migrator and VERITAS Storage Migrator Remote. These commands are applicable to both products unless otherwise noted. You can also access any of these command descriptions online by using the `man` command, if you have the man pages installed.

The man pages are shown in alphabetical order.

fls(1)

NAME

fls – list contents of a directory and indicate whether files are migrated

SYNOPSIS

```
/usr/openv/hsm/bin/fls [-l]
```

DESCRIPTION

`fls` is a version of the standard `ls` command, modified to work with VSM. It supports most of the options supported by `ls`.

`fls` allows you to list the contents of your directories and when used with the `-l` option indicates which of those files have been migrated and which have been purged from premigration.

For example, assume you have a migrated file named `tproga` in your current working directory and enter:

```
/usr/openv/hsm/bin/fls -l  tproga
```

The response is:

```
mrwxr-xr-- 1 root   23 Nov 29 14:30 tproga [000003e8 0000100e]
```

The `m` in the mode bit field indicates that the file has been migrated. The numbers to the right of the file name indicate the machine ID and file handle, respectively.

If `tproga` has also been purged from premigration, the response is:

```
mrwxr-xr-t 1 root   23 Nov 29 14:40 tproga [000003e8 0000100e]
```

The `t` in the mode bit field indicates that the file has been purged and no longer resides on disk.

If `tproga` has been cached but has not been changed, the response is:

```
-rwxr-xr-t 1 root   23 Nov 29 15:30 tproga [000003e8 0000100e]
```

The `m` in the mode bit field is removed to indicate that the file has been cached back to disk. The `t` in the left column as well as the machine ID and file handle indicate that the migrated copy in secondary storage is still valid.

If `tproga` has been cached and also changed, the response is:

```
-rwxr-xr-- 1 root   23 Nov 29 15:40 tproga
```

The `t` in the mode bit field as well as the machine ID and file handle are removed to indicate that the cached file has been changed, which renders the migrated copy in secondary storage invalid.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

This command is intended for both administrators and users.

CAVEATS

- When `fls` is run with the `-F` option, it causes directories to be marked with a trailing `/`, sockets with a trailing `=`, symbolic links with a trailing `@`, executable files with a trailing *, and migrated files with a trailing %. The `fls` command does not mark fifos.
- `fls` has other options corresponding to those of the standard `ls` command, but those options have not been tested.
- If a migrated file is copied to a new file, the `t` bit in the mode bit field is also copied. This bit should be removed manually to indicate the correct status of the new file.

SEE ALSO

`ls(1), migloc(1)`

gethsm(1)

NAME

gethsm – display the hsmname for all managed filesystems or for a specified file or directory

SYNOPSIS

`/usr/openv/hsm/bin/gethsm [`*filename* | *dirname*`]`

DESCRIPTION

`gethsm` identifies and displays the hsmname assigned to the HSMDEV entry containing the file or directory specified in the command. If no file or directory is specified, `gethsm` identifies and displays the hsmnames for all VSM managed file systems.

Output is displayed to standard output. Error messages go to standard error.

This command may be run without regard to whether the migration daemon `migd` is running, but the hsm state must be active if an option is specified.

This command is intended for both administrators and users.

OPTIONS

filename The full pathname of the file for which to display the hsmname.

dirname The full pathname of the directory for which to display the hsmname.

EXAMPLES

To display all hsmnames, type the command without options:

```
gethsm
hsm1 hsm2 hsm3 hsm4
```

To display the particular hsmname for a particular file or directory, include the *filename* or *dirname* option in the command:

```
gethsm /hsmrel/dir/subdir/filename
hsm1
```

Error messages are returned if the *filename* or *dirname* are not valid:

```
gethsm /usr
ERROR: Path not configured for HSM.
```

In this case, `/usr` is not an hsm partition.

```
gethsm /hsm2/tom/file1
```

```
Error: HSMNAME is inactive.
```

In this case, the `hsm` file system is set to inactive.

FILES

dwpath`/database/migconf`

VSM configuration file for managed directories

`/usr/var/openv/hsm/database/migconfg`

Global-configuration file for VSM

SEE ALSO

`HSM(1M)`

HSM(1M)

NAME

HSM – Hierarchical Storage Management

DESCRIPTION

The hierarchical storage management software that runs on this platform increases the amount of file space available to users by migrating files from a local online managed file system to secondary storage (such as magnetic tape or another disk) as space is needed in the online file system.

Administrators can schedule migration operations to occur automatically. When properly configured, HSM selects files for migration based on configurable criteria such as file size and file age.

When a user accesses a migrated file, it is automatically retrieved from secondary storage and cached in the online file system. Except for the delay to perform the retrieval, users and programs are unaware that file migration and retrieval are taking place.

HSM implementations fall into two groups: kernel-based implementations or nonkernel-based implementations. Kernel-based implementations use a parallel inode (`.PAIN`) file. Nonkernel-based implementations (DMAPI) use an inode-to-handle (`.IHAND`) file. These implementations are platform-dependent but are functionally equivalent. Refer to your Release Notes for a detailed breakdown of which implementation applies to your system.

Two variations of VSM are available: VERITAS Storage Migrator and VERITAS Storage Migrator Remote. These are very similar, both in design and function. They use the same techniques for managing the local file system and for most administrative procedures. The main difference between them is the methods they use for migrating data.

VERITAS Storage Migrator

VERITAS Storage Migrator uses directly connected tape, optical disk, or magnetic disk devices as well as magnetic disk file systems on remote volume servers for secondary storage. Media Manager provides the interface to the tape and optical storage devices. Support for large-capacity library devices with robotic access mechanisms eliminates the need for operator action to either migrate or cache files. The net result is apparently unlimited online storage but at a lower cost per megabyte because the extra storage is on lower-cost media such as cartridge tape or optical disk.

There are nine methods that VERITAS Storage Migrator uses for secondary storage. They are disk file for premigration (`dk`), alternate magnetic disk (`ad`), three tape methods (`ct`, `dt`, and `mt`), two optical disk methods (`op` and `ow`), remote using ftp (`ft`), and NetBackup (`nb`). The `nb` method migrates files using VERITAS NetBackup.

VERITAS Storage Migrator is able to share storage volumes and devices with other applications like VERITAS NetBackup through the use of a common Media Manager.

VERITAS Storage Migrator supports multilevel migration. Administrators may configure up to eight migration levels, and can schedule migrated files to move from one level to another based on site-specified criteria. With multilevel migration, you can configure and manage cost-effective storage hierarchies that make best use of your storage equipment investment.

VERITAS Storage Migrator enables administrators to export migrated files from one managed file system and import them into another managed file system.

VERITAS Storage Migrator Remote

VERITAS Storage Migrator Remote is a subset of VERITAS Storage Migrator. It uses only four methods: disk file for premigration (`dk`), alternate magnetic disk (`ad`), remote using ftp (`ft`), and NetBackup (`nb`). The `ad` method is used to migrate files to a remote file system by using NFS. Using the `ft` method you can combine a local UNIX file system with one or more remote file systems to create the impression of one large *virtual* file system. The `nb` method migrates files using VERITAS NetBackup.

The remote volume servers can be from a variety of different vendors and have different capacities. Files are migrated and retrieved by using standard ftp to access a remote file system.

Neither the tape nor optical disk methods are available to VERITAS Storage Migrator Remote, and Media Manager is not used.

VERITAS Storage Migrator Remote does not support multilevel migration and file export/import.

SEE ALSO

Configuring VSM

`migconf(1M)`

VSM file system configuration file

`migconfg(1M)`

VSM global configuration file

Managing File Migration

`migbatch(1M)`

Premigrate files and copy to secondary storage

`mignospace(1M)`

Purge or migrate files to make disk space available

`migthreshold(1M)` -- (Kernel-based implementations only)

Set high-water mark to a new value until next boot

`miglow(1M)`

Start `mignospace` if file system is above high-water mark

`migpolicy(1M)`

Specify policy and method to write files to secondary storage

Managing Multilevel Migration

`migmove(1M)`

Move migrated files from one migration level to another level

`migsetdb(1M)`

Alter the flags field in the FHDB or the VOLDB

Managing File Caching

`migin(1M)`

Cache a file from secondary storage to disk

`migstage(1)`

Pre-cache files to avoid caching delays

`migtie(1)`

Designate the key caching file(s) for a caching group

Managing Media

`migreg(1M)`

Register and label volumes for VSM

`migcons(1M)`

Consolidate VSM tape and optical volumes

`migselect(1M)`

Select volumes for consolidation based on specified criteria

`migrecycle(1M)`

Reregister an empty volume

`migmdclean(1M)`

Remove obsolete entries from media

`migftscan(1M)`

Scan an ft remote volume and reconstruct database entries

Managing VSM Processes

`startmigd(1M)`

Start migration and volume daemons

`stopmigd(1M)`

Stop migration and volume daemons

`HSMKiller(1M)`

Kill active VSM processes and release tape requests

`migrc(1M)`

Clear locks, remove defunct lock files, restart migrations

`migconsweep(1M)`

Enable constant file system sweeping

User Controls

`fls(1)`

List and show migration status of files

`gethsm(1)`

Display the hsmname for files and directories

`migcat(1)`

Concatenate and display migrated files without caching them

`miggroup(1)`

Group files for migration and caching -- (DMAPI implementations only)

`migloc(1)`

Show location of migrated file

`migmode(1)` -- (Kernel-based implementations only)

Set mode for accessing migrated files

`migrate(1)`

Premigrate a specific file or files

`migpurge(1)`

Purge a specific file or files

`migstage(1)`

Pre-cache files to avoid caching delays

`migtie(1)`

Designate the key caching file(s) for a caching group

Exporting and Importing Migrated Files

`mignbexport(1M)`

Export VSM files

`mignbimport(1M)`

Import VSM files

Managing Databases

`migdbcheck(1M)`

Check the FHDB and/or the VOLDB for consistency with the file system

`mediacheck(1M)`

Check the media and FHDB for consistency

`migsetdb(1M)`

Alter the flags field in the FHDB or the VOLDB

Obtaining Reports

`migadscan(1M)`

Provide information on contents of archive disk volumes

`migtscan(1M), migopscan(1M)`

Provide information on contents of tape or optical volumes

`miggetvol(1M)`

List volumes in ascending order of percentage utilization

`migdbdir(1M)`

List global configuration values from `migconfg`

`migdbrpt(1M)`

Provide FHDB and VOLDB information on files and volumes

`migchecklog(1M)`

List the most recent messages from an error log file

`migftscan(1M)`

Scan an ft remote volume and reconstruct database entries

`mignbscan(1M)`

Scan a NetBackup volume and reconstruct database entries

Managing Logs

`mignewlog(1M)`

Copy or delete global or individual VSM log files

Tuning Migrations

`migconsweep(1M)`

Enable constant file system sweeping

`migtestbadness(1M)`

Evaluate configuring different file system attributes

Managing a Kernel-based Implementation

`pfcheck(1M)`

Check and fix the `.PAIN` file

`pfinit(1M)`

Create or extend the `.PAIN` file

`pfprint(1M)`

Print or alter the `.PAIN` file

Managing a Nonkernel-based Implementation

`ihprint(1M)`

Print or alter the `.IHAND` file

`rebuild_ihand(1M)`

Rebuild the `.IHAND` file from the FHDB

`migalter(1M)` -- (DMAPI implementations only)

Display or alter regions, events, or attributes

`migcleanup(1M)` -- (DMAPI implementations only)

Display or clean up DMAPI sessions

Recovering Data

`migreconstruct(1M)`

Reconstruct damaged or deleted migrated files

`migin(1M)`

Restore a file from secondary storage to disk

Troubleshooting

`migfind(1M)`

Determine the full pathname of a file

Graphical User Interface

`xhsmadm(1M)`

Administrative graphical user interface (Motif-based)

HSMKiller(1M)

NAME

HSMKiller – kill active VSM processes

SYNOPSIS

```
/usr/openv/hsm/bin/HSMKiller [hsmname]...
```

DESCRIPTION

`HSMKiller` kills VSM processes and unmounts secondary storage volumes that are left mounted by any killed processes.

If one or more *hsmnames* are specified, `HSMKiller` kills all active processes listed in `/usr/openv/hsm/bin/admincmd/HSMKiller.kill_list` for the specified *hsmnames*, but not the daemons `migd` and `migvold`. It also unmounts secondary storage volumes left mounted by any killed process, and signals `migvold` to clean up the mount table.

If no *hsmname* is specified, `HSMKiller` kills all active VSM processes listed in `/usr/openv/hsm/bin/admincmd/HSMKiller.kill_list` plus those listed in `/usr/openv/hsm/bin/admincmd/HSMKiller.global.kill_list`. It also kills the daemons `migd` and `migvold`, unmounts all secondary storage volumes, and removes the VSM mount table.

Use `stopmigd` if you want to terminate just the VSM daemons `migd` and `migvold`.

After running `HSMKiller`, run `migrc` to clean up any locks left set by the killed processes. Use `startmigd` to restart the VSM daemons `migd` and `migvold` after running `HSMKiller`.

Note Before using `HSMKiller`, make sure `migcopy` is not writing to any volume. Killing `migcopy` when writing to a volume will make it impossible to append any more data to that volume. Data previously written to the volume, however, will remain accessible.

Note Any process that cannot be terminated by a `kill-9` command after three attempts remains active, and a failure message is placed in the global-log file.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

OPTIONS

hsmname The HSMDEV entry for which you want to kill processes. More than one hsmname may be included on the command line. The default is all hsmnames if none are specified. See `migconfg(1M)` for more information on HSMDEV entries.

CAVEATS

- `HSMKiller` searches for processes to kill which are executing using the full pathname for VSM binaries, `/usr/openv/hsm/bin`. All VSM processes which call other VSM processes use this convention. For `HSMKiller` to work properly, all VSM processes called by customer defined scripts or `cron` schedules should follow this same convention instead of depending on PATH variables to allow shorthand command names.
- Use care either when deleting unused processes from `/usr/openv/hsm/bin/admincmd/HSMKiller.kill_list` to speed `HSMKiller` processing or when otherwise editing the two kill lists. This may result in a less comprehensive termination of active processes.

FILES

`/usr/openv/hsm/bin/admincmd/HSMKiller.kill_list`

`HSMKiller` kill list

`/usr/openv/hsm/bin/admincmd/HSMKiller.global.kill_list`

`HSMKiller` global kill list

SEE ALSO

`migconfg(1M), migrc(1M), startmigd(1M), stopmigd(1M)`

ihprint(1M)

NAME

ihprint– print or alter an .IHAND (inode-to-handle) file entry

SYNOPSIS

```
/usr/openv/hsm/bin/ihprint [-m machid] [-h handle]
          [-f flags] [-c slice] [-d dmhandle] [-H highest]
          [-I ihandfile] hsm inode | pathname
```

Caution ihprint is intended only for customer support engineers who are trained in its use--this especially applies to changing the .IHAND file. All other users should rely on the rebuild_ihand command to verify and change .IHAND entries.

AVAILABILITY

This command is only available on nonkernel-based implementations.

DESCRIPTION

ihprint prints the .IHAND entry for an inode in a managed file system. If any options are specified, ihprint sets the indicated field of the .IHAND entry, prints the new entry, and updates the .IHAND file.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon migd is running. but the file system must be mounted.

OPTIONS

-m *machid*

Set the machine id field of the appropriate .IHAND entry to *machid*. The value is assumed to be hexadecimal.

-h *handle*

Set the file handle field of the appropriate .IHAND entry to *handle*. The value is assumed to be hexadecimal.

-f *flags* Set the flags field of the appropriate .IHAND entry to *flags*. The value is assumed to be hexadecimal.

```
01  reloading
02  removing
04  parital_cache
08  cached, unmodified
10  being cached by migstage)
```

-c *slice* Set the slice field of the appropriate .IHAND entry to *slice*.

`-I` *ihandfile*
: Specifies an alternate path to an `.IHAND` file in which an entry is to be modified and/or printed. If *ihandfile* is specified, *hsm* and *inode* must also be specified and *pathname* cannot be specified.

hsm
: Specifies the *hsmname* for which an `.IHAND` entry is to be modified and/or printed. If *hsm* is specified, *inode* must also be specified and *pathname* cannot be specified.

inode
: Specifies the `.IHAND` entry to be modified and/or printed. If *hsm* is specified, *inode* must also be specified and *pathname* cannot be specified.

pathname
: Specifies the *pathname* of a file for which an `.IHAND` entry is to be modified and/or printed. If *pathname* is specified, do not specify the *hsm* and *inode* parameters.

`-d` *dmhandle*
: Set the DMAPI handle field of the appropriate `.IHAND` entry to *dmhandle*. The value is assumed to be hexadecimal.

`-H` *highest*
: Set the highest byte to read field of the appropriate `.IHAND` entry to *highest*.

EXAMPLE

This command prints the `.IHAND` entry for the inode at *pathname* `/managed_fs/file94`:

```
prompt> ihprint /managed_fs/file94
```

Typical output is shown below:

```
Current IHAND entry:
HANDLE   MACHID   FLAGS  SLICE   HIGHEST
1D18     1000     10     0       0
DMHANDLE-LENGTH DMHANDLE
10               008000140000271f0000000100000100
```

This command changes the machine id field of this same `.IHAND` entry to `3E8`:

```
prompt> ihprint -m 3E8 /managed_fs/file94
```

Typical output is shown below:

```
Current IHAND entry:
HANDLE   MACHID   FLAGS  SLICE   HIGHEST
1D18     1000     10      0      0
DMHANDLE-LENGTH DMHANDLE
```

```
10               008000140000271f0000000100000100

Modified IHAND entry:
HANDLE   MACHID   FLAGS  SLICE   HIGHEST
1D18     3E8      10      0      0
DMHANDLE-LENGTH DMHANDLE
10               008000140000271f0000000100000100

Do you want the IHAND entry reset? [yn] y

IHAND entry modified.
```

This command clears the flags field of the .IHAND entry for inode 10014 in managed file system vdm3:

prompt> ihprint -f 0 vdm3 10014

Typical output is shown below:

```
Current IHAND entry:
HANDLE   MACHID   FLAGS  SLICE   HIGHEST
100F     1000     10     0       0
DMHANDLE-LENGTH DMHANDLE
10               008000140000271e0000000100000100

Modified IHAND entry:
HANDLE   MACHID   FLAGS  SLICE   HIGHEST
100F     1000     0      0       0
DMHANDLE-LENGTH DMHANDLE
10               008000140000271e0000000100000100

Do you want the IHAND entry reset? [yn] y

IHAND entry modified.
```


FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath`/database/hsmname.IHAND`

Inode-to-handle file for VSM

dwpath`/database/FHDB`

File handle database for VSM

SEE ALSO

`rebuild_ihand(1M)`

mediacheck(1M)

NAME

mediacheck – check the media and file-handle database (FHDB) for consistency

SYNOPSIS

`/usr/openv/hsm/bin/mediacheck` *hsmname label method*

DESCRIPTION

The `mediacheck` command verifies that all files recorded on the secondary storage media are also recorded in the VSM file-handle database (FHDB). The *label* is scanned (using `migadscan`, `migtscan`, `migopscan`, `migftscan`, or `mignbscan`) and compared with the file-handle database entries.

You can use the `migdbrpt` command to display a summary of information from the FHDB. You can use the `migdbcheck` command to verify consistency between the file system and FHDB and also to verify the consistency between the volume database (VOLDB) and FHDB.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

OPTIONS

hsmname HSMDEV entry that specifies the path to the database files you want to check. See `migconfg(1M)` for more information on HSMDEV entries. This parameter is required.

label Label of the volume that is to be checked.

method Method the *label* is registered for. The allowed methods are `ad`, `ct`, `dt`, `mt`, `op`, `ow`, `ft`, and `nb`.

EXAMPLE

The following example verifies the media on optical disk OP001A, which is registered in the database specified by the *alpha* HSMDEV entry.

```
mediacheck alpha OP001A op
```

ERRORS

`-- FHDB entry 00001098 is locked by 0007700`

An entry is locked by the indicated process. Ensure that VSM is not being actively used and the VSM migration daemon (`migd`) is not running (see `stopmigd(1M)`). Run `migrc` to clear any leftover locks. Then, rerun `mediacheck`.

`-- FHDB entry 00001097 is out of order`

The FHDB is maintained in sorted order. If an entry is reported out of order, sort the FHDB according to the first two fields of each line.

`-- Missing FHDB gran 00001098, offset 1000, file /home/gls/wiggins`

A portion of the file recorded on the media is no longer in the FHDB. If the file is still in the active file system, you may have to reconstruct the FHDB entries as described in `migdbcheck`.

`-- Missing FHDB entry for file /home/gls/wiggins`

The media contains a file that does not have an entry in the FHDB. If the given file is still in the active file system, you may have to reconstruct the FHDB entries as described in `migdbcheck`.

`-- Missing media granule 00001098, offset 1000, file /home/gls/bird`

A portion of the file recorded in the FHDB is no longer on the media. If the file still exists in the active file system, the file data may be lost if you cannot recover it from a backup.

`-- No media file for orphan FHDB entry`

A file recorded in the FHDB is no longer on the media. If the given file still exists in the active file system, the file data may be lost if you cannot recover it from a backup.

SEE ALSO

`HSM(1M), migdbcheck(1M), migadscan(1M), migconfg(1M), migdbrpt(1M), migftscan(1M), mignbscan(1M), migtscan(1M), migopscan(1M), migrc(1M), stopmigd(1M)`

migadscan(1M)

NAME

migadscan – scan alternate disk volumes under VSM for file granules

SYNOPSIS

`/usr/openv/hsm/bin/migadscan -F [-n] [-s]` *hsmname label mount_point*

`/usr/openv/hsm/bin/migadscan [-n] [-s]` *hsmname label*

DESCRIPTION

`migadscan` scans an alternate magnetic disk (`ad`) volume and displays information about the volume as a whole in addition to information about each granule on the volume.

If you specify the `-F` option, the media can be scanned without a VOLDB entry. You must supply an alternate magnetic disk *mount_point* with the `-F` option.

The `migadscan` command creates two output files, FHDB.*label* and VOLDB.*label*, in the *dwpath*`/database` directory. The structure of these files is the same as the FHDB and VOLDB database files. These files may be used to rebuild the FHDB and VOLDB if they are corrupted or damaged (see `migdbcheck(1M)`).

You can sort and merge FHDB.*label* files for different magnetic disks in order to recreate the FHDB database. Similarly, you can merge and sort the VOLDB.*label* files for different magnetic disks to recreate the VOLDB database.

Note When recreating the VOLDB be sure to merge the VOLDB file in the `/usr/var/openv/hsm/example/database` directory to include the entry for the `dk` method.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

Only the system administrator can use this command.

OPTIONS

`-F` Force a scan of the volume for VSM granules. This is useful when the volume identity is not in the volume database. This parameter is optional. If omitted, the volume must be registered in the volume database. If specified, *mount_point* must also be specified.

`-n` Do not convert FHDB entries for granules written prior to R2.0. You cannot reconstruct a usable database with this option, but it is useful if you want to examine what is actually written on the media.

`-s` Silently scan the volume and create FHDB.*label* and VOLDB.*label* files. Do not display information on stdout.

hsmname HSMDEV entry specifying the path to the database files that contain entries for the volume you want to scan. See `migconfg(1M)` for more information on HSMDEV entries.

label Label of the volume you are scanning. This parameter is required. For a force scan, you can specify any name for the *label*.

mount_point
Indicates the mount point for the alternate magnetic disk. Specify *mount_point* if and only if using the `-F` option. If using NFS, make sure the file system has been mounted.

CAVEATS

Scanning an alternate magnetic disk volume that contains files that are not VSM granules will produce informative messages such as this:

```
<filename> ***Not a Valid Granule*** 10
```

EXAMPLE

The following example scans alternate magnetic disk volume `pjrad1` registered under `ad` method and displays a list of granule information for files migrated to the volume. Files FHDB.PJRAD1 and VOLDB.PJRAD1 are created in the VSM database directory.

migadscan alpha pjrad1

```
   Sample output
   -------------
eeyore [6]# migadscan alpha pjrad1
VOLUME PJRAD1 registered to HSM
   Volume Particulars
   ------------------
000003E8V000010AB PJRAD1     ad 14000000 09990800 #7
Volume Label Found ====> PJRAD1
-------------------------------
3E8M13BB.1.0  <=> 00005AFC 00000000 Thu Aug 22 11:01:15 1991 /home/gls/drat
3E8M13BC.1.0  <=> 0000009C 00000000 Thu Aug 22 11:01:16 1991 /home/gls/bmount
3E8M13BD.1.0  <=> 00001211 00000000 Thu Aug 22 11:01:17 1991 /home/gls/camel.log
3E8M13BF.1.0  <=> 00000047 00000000 Thu Aug 22 11:01:19 1991 /home/gls/x
3E8M13C0.1.0  <=> 0000B5F8 00000000 Thu Aug 22 11:01:20 1991 /home/gls/tiger
3E8M13C1.1.0  <=> 000110F4 00000000 Thu Aug 22 11:01:21 1991 /home/gls/bull *
```

Volume particulars displayed include (in order): volume handle, volume label, method, total capacity of the volume, total space in use on the volume and number of granules on the volume.

Particulars of granules displayed include (in order): granule file name, migrated file size, offset of the granule in the migrated file, date of archiving the granule and migrated file path name.

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath`/database/FHDB`

File-handle database for VSM

dwpath`/database/VOLDB`

Volume database for VSM

dwpath`/database/migconf`

VSM configuration file for managed file systems.

dwpath/database/FHDB.*label*

File-handle database for current volume

dwpath/database/VOLDB.*label*

Volume database for current volume

/usr/var/openv/hsm/database/migconfg

Global-configuration file for VSM

SEE ALSO

migdbcheck(1M), migconf(1M), migconfg(1M), migdbrpt(1M), migftscan(1M), migtscan(1M), migopscan(1M)

migalter(1M)

NAME

migalter – display or alter regions, events, or attributes for a file or managed file system

SYNOPSIS

```
/usr/openv/hsm/bin/migalter [-F] [-e event]...
          [-d event]... pathname

/usr/openv/hsm/bin/migalter [-r region] [-h handle]
          [-m machid] [-p slice] [-u token -s sid] pathname

/usr/openv/hsm/bin/migalter -I pathname

/usr/openv/hsm/bin/migalter -R pathname
```

Caution `migalter` is intended only for customer support engineers who are trained in its use.

AVAILABILITY

This command is only available on the DMAPI implementation of VSM.

DESCRIPTION

For files, `migalter` displays or alters managed regions, displays or enables events, displays or alters DM attributes, or punches a hole.

For managed file systems, `migalter` displays or alters events or DM attributes.

If no options are specified, `migalter` displays the enabled events for the file system, and the managed regions, DM attributes, file attributes, allocation information and DMAPI handle information for the file. If the file is partially cached, this is noted with the allocation information.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

OPTIONS

`-F` Display or alter enabled events for the managed file system of *pathname*. Default is display or alter enabled events, managed regions, and DM attributes for the file *pathname*.

The `-F` option can not be used with `-r`, `-h`, `-m`, `-p`, or `-u`.

`-e` *event* Add the specified *event* to the *pathname*. Valid values for *event* are as follows:

event = CREATE, POSTCREATE, REMOVE, POSTREMOVE, RENAME, POSTRENAME, LINK, POSTLINK, SYMLINK, POSTSYMLINK, DESTROY, NOSPACE, ATTRIBUTE, PREUNMOUNT, UNMOUNT, or NOEVENT.

NOEVENT clears all events.

Multiple instances are allowed. If both `-e` and `-d` are specified, events are added before they are deleted. See Examples.

`-d` *event* Delete the specified *event* from the *pathname*. Valid values for *event* are as follows:

event = CREATE, POSTCREATE, REMOVE, POSTREMOVE, RENAME, POSTRENAME, LINK, POSTLINK, SYMLINK, POSTSYMLINK, DESTROY, NOSPACE, ATTRIBUTE, PREUNMOUNT, or UNMOUNT.

Multiple instances are allowed. If both `-e` and `-d` are specified, events are added before they are deleted. See Examples.

`-r` *region* Add the managed *region* to the file *pathname*. Valid values for *region* are as follows:

region = READ, WRITE, TRUNCATE, NOREGION, MIGRATED, PURGED, or CACHED.

NOREGION clears all regions.

MIGRATED sets READ, WRITE, TRUNCATE.

PURGED sets READ, WRITE, TRUNCATE.

CACHED sets WRITE, TRUNCATE.

NOREGION, MIGRATED, PURGED, and CACHED all change the DM attributes for the *pathname*.

The *region* offset and size are always set to 0, which means the *region* applies to the whole file.

`-h` *handle* Set the file handle field of the DM attributes to *handle* for the file *pathname*. The value is assumed to be hexadecimal.

`-m` *machid*

Set the machid field of the DM attributes to *machid* for the file *pathname*. The value is assumed to be hexadecimal.

`-p` *slice* Punch a hole in the file *pathname* from the offset *slice* to the end of the file, where *slice* is specified in bytes. The value is assumed to be decimal.

`-u` *token* Unlock this *token* for the file *pathname*. The value is assumed to be decimal. Specifying `-u` requires specifying `-s`.

`-s` *sid* The session ID to which `-u` *token* belongs. The value is assumed to be decimal. Specifying `-u` requires specifying `-s`.

`-I` Initialize DM attributes on the managed file system or file *pathname*. You must first mount the file system before running `migalter -I`. Once `migalter -I` is run, you must have the VSM daemon `migd` running to mount the manged file system.

The `-I` option can not be specified with any other option. It is only available on VERITAS VxFS implementations.

`-R` Remove DM attributes from the managed file system or file *pathname*. You must first mount the file system before running `migalter -R`. Once `migalter -R` is run, you no longer need the VSM daemon `migd` running to mount the manged file system.

The `-R` option can not be specified with any other option. It is only available on VERITAS VxFS implementations.

pathname Specifies the pathname of a file or managed file system to which the options are applied.

EXAMPLE

A single command statement can both add and delete events. This example first adds the REMOVE event and then deletes the DESTROY event for the managed file system `/xhsm1`:

```
migalter -F -e REMOVE -d DESTROY /xhsm1
```

This example clears the regions for the file /xhsm2/file6:

```
migalter -r NOREGION /xhsm2/file6
```

This example initializes DM attributes on managed file system `/sys12` and prevents it from being mounted unless the VSM daemon `migd` is running:

```
migalter -I /sys12
```

This example removes DM attributes from managed file system `/sys12` and allows it to be mounted whether or not the VSM daemon `migd` is running:

```
migalter -R /sys12
```

This example displays the enabled events for file system `/sys14`, and the managed regions, DM attributes, file attributes, allocation information and DMAPI handle information for the file `/sys14/projects/file1`:

```
migalter /sys14/projects/file1
```

SEE ALSO

`migcleanup(1M)`

migbatch(1M)

NAME

migbatch – control migration of files from file system to secondary storage

SYNOPSIS

`/usr/openv/hsm/bin/migbatch [`*hsmname*`] | [`*file_system*`]`

DESCRIPTION

`migbatch` controls the migration of files to secondary storage. It sweeps either all file systems or the file system indicated by *hsmname* or the specified *file_system*, looking for files that meet the age and size criteria specified. A file in a managed file system is considered a candidate for migration if all of the following are true:

- It is a regular UNIX file without any new line (carriage return) characters, vertical bars (|), or null characters in its name.
- The file's computed badness value is greater than or equal to the badness value specified in `migconf`.
- It is not already migrated.
- The pathname of the file does not appear either in the global stop file or in a local stop file, `.migstop`, unless the stop file is overridden. See CAVEATS.
- The file does not reside in premigration.
- The file does not reside in another file system, even if that file system is mounted in a VSM managed directory.
- The file is not a symbolic link.
- The full pathname length is less than or equal to 1023 characters.

If the `migconf` file specifies a low-water mark, `migbatch` quits selecting files when this is reached. Otherwise, all files that meet the selection criteria are selected. This could result in migrating almost every regular file from the managed file system.

VSM then premigrates the data for each selected file. The site must set the METHOD1 and METHOD2 parameters in the `migconf` configuration file to specify the type of media, volume set, hint, and volume pool to which the premigrated files are written. VSM then writes the specified number of copies of the file to the specified secondary storage media.

After files are copied to secondary storage, files may be purged by `mignospace` to make disk space available. `mignospace` starts under any of the following conditions:

- For kernel-based implementations, the kernel detects that the system has run low on space and informs `migd`, which in turn starts `mignospace`.

- For nonkernel-based implementations, `migd` periodically checks the high-water mark threshold and starts `mignospace` if the threshold is exceeded.
- The administrator executes `mignospace` from the system prompt or from the `xhsmadm` GUI.
- The administrator executes `miglow` from the system prompt or from the `xhsmadm` GUI when file system free space is low.
- The administrator restarts migration and move processes from the VSM-Java GUI.

This command may be run without regard to whether the migration daemon `migd` is running, but the hsm state must be active.

OPTIONS

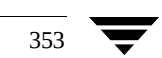
hsmname HSMDEV entry that specifies the file system you want `migbatch` to process. See `migconfg(1M)` for more information on HSMDEV entries.

file_system
Full path name of the file system on which to initiate migration.

If you do not specify either *file_system* or *hsmname*, then `migbatch` reads all configuration files and sweeps all file systems.

CAVEATS

- The migration process only migrates regular file data. It is not a substitute for a backup process which copies *all* directories and files. You must backup your system including VSM databases so you will be able to restore the system to that level following a catastrophic failure or loss of files.
- Before executing `migbatch`, use `migreg` or the GUI to register a sufficient number of volumes.
- If cartridge tapes or optical platters are used, the `ltid` daemon must be running. If the `ft` method name is used, the remote file system must be available.
- If VSM transfers files greater than 2 gigabytes with the `ft` method, the remote volume server must be configured to accept and process files of this size.
- Migration waits for caching operations to complete when migrating and caching from the same tape or optical volume. Migration and caching operations can occur in parallel for the ft and ad method names.
- The `.migrate` and `.migstop` files most local to the listed file override more remote control files in the directory tree. Local control files override global control files if the same file or directory is listed in both. If the same file is listed in both a `.migrate` file and a `.migstop` file at the same level, the `.migrate` control file overrides the `.migstop` file.



- Some processes spawned by `migbatch` create large temporary files and there may not be enough space in the `/tmp` directory to store these files. You can avoid this problem by defining the environment variable `TMPDIR` to contain the pathname of a directory in a file system that has sufficient space available. Then, if the process checks `TMPDIR`, it creates any temporary files in the specified directory instead of in the default `/tmp`.

EXAMPLES

- This example starts the migration for *hsmname* `alpha`.

  `migbatch alpha`

- This example starts the migration for all file systems:

  `migbatch`

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath`/database/migconf`

VSM configuration file for managed file systems

`/usr/var/openv/hsm/database/migconfg`

Global-configuration file for VSM

`/usr/var/openv/hsm/database/migrate`

Global migrate file for VSM

`/usr/var/openv/hsm/database/migstop`

Global stop file for VSM

SEE ALSO

`HSM(1M), migconf(1M), migconfg(1M), miglow(1M), mignospace(1M), migreg(1M), migrc(1M), startmigd(1M), stopmigd(1M)`

migcat(1)

NAME

migcat – concatenate and display migrated files without caching them to disk

SYNOPSIS

`/usr/openv/hsm/bin/migcat` *file* [*file*]...

DESCRIPTION

`migcat` is a version of the standard UNIX `cat` command, modified to work with VSM.

The `migcat` command sends the migrated file or files to `stdout` one granule at a time, and does not cache the file to disk. The user does not have to wait until the entire file is cached before reading it.

The standard UNIX `cat` command caches the entire file data to disk before the user can read it. This delay is reduced by using `migcat` instead of `cat`, particularly for large files.

This command may be run without regard to whether the migration daemon `migd` is running, but the hsm state must be active.

This command is intended for both administrators and users.

OPTIONS

file Specifies the file or list of files to concatenate. Wildcards are recognized. This parameter is required.

CAVEATS

- Files migrated with either method name `ft` or `nb` will be cached.
- Data from files that are premigrated but not purged will be read from disk, not from secondary storage. This will cause those files to be cached.

EXAMPLE

The following command reads the migrated files report21a, report21b, and report33 to `stdout`:

```
migcat rep*21? report33
```

SEE ALSO

`cat(1)`

migchecklog(1M)

NAME

migchecklog – list the most recent error messages from an error log file

SYNOPSIS

```
/usr/openv/hsm/bin/migchecklog [-d time_in_minutes]
          [-h] [hsmname | GLOBAL]
```

DESCRIPTION

`migchecklog` checks the specified VSM log file and displays the 20 most recent error messages logged.

A command option runs `migchecklog` periodically following a variable delay. If subsequent checks find newer error messages, older messages are dropped from the input. The display indicates that new errors exist and appends them to the list of up to twenty error messages.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

OPTIONS

`-d` *time_in_minutes*
: Check logs, and then delay for the sleep time specified by the variable *time_in_minutes* before checking logs again. Default is 0, which checks logs only once.

`-h`
: Print help information.

hsmname
: The HSMDEV entry that specifies the path to the log file you want `migchecklog` to check. If *hsmname*=`GLOBAL`, then `migchecklog` checks the VSM global-log file, `/tmp/hsm.log`. This is the default. See `migconfg(1M)` for more information on the global-configuration file and HSMDEV entries.

CAVEATS

- Because `migchecklog` lists only the latest 20 error messages, it is possible some error messages could be missed by this command. If the listing shows 20 new errors, the administrator is advised to check the actual log in question for any errors that may not have been reported.

EXAMPLES

- This example checks the global-log file once:

  migchecklog

- This example checks the log file for HSMDEV entry hsm1 at 8-minute intervals:

  migchecklog -d 8 hsm1

  A typical response looks like this (up to 20 messages):

```
-------  Wed Jul 12 15:03:58 CDT 1995  --------
07/12 14:36:21 migcopy[1084]: ERROR ** copy_for_method() ret=1
07/12 14:36:25 migmkspace[1113]: ERROR No entries in FHDB for filehandle 1000
07/12 14:36:25 migmkspace[1113]: ERROR No entries in FHDB for filehandle 107c
07/12 14:40:13 migcopy[1413]: ERROR choose_src_gran no gran for as_filep=x0
07/12 14:40:13 migcopy[1413]: ERROR read_data: no good source granules found
07/12 14:40:13 migcopy[1413]: ERROR copy_file()/copy_gran fail
07/12 14:40:13 migcopy[1413]: ERROR ** FAILED TO COPY: x107c
07/12 14:40:13 migcopy[1413]: ERROR ** copy_for_method() ret=1
```

  After 8 minutes, `migchecklog` prints the list again, appending any new messages and dropping older messages as necessary to maintain a maximum list of 20 messages:

```
-------  Wed Jul 12 15:11:58 CDT 1995  --------
New errors
07/12 14:36:25 migmkspace[1113]: ERROR No entries in FHDB for filehandle 1000
07/12 14:36:25 migmkspace[1113]: ERROR No entries in FHDB for filehandle 107c
07/12 14:40:13 migcopy[1413]: ERROR choose_src_gran no gran for as_filep=x0
07/12 14:40:13 migcopy[1413]: ERROR read_data: no good source granules found
07/12 14:40:13 migcopy[1413]: ERROR copy_file()/copy_gran fail
07/12 14:40:13 migcopy[1413]: ERROR ** FAILED TO COPY: x107c
07/12 14:40:13 migcopy[1413]: ERROR ** copy_for_method() ret=1
07/12 14:41:43 migcopy[1479]: ERROR choose_src_gran no gran for as_filep=x0
07/12 14:41:43 migcopy[1479]: ERROR read_data: no good source granules found
07/12 14:41:43 migcopy[1479]: ERROR copy_file()/copy_gran fail
07/12 14:41:43 migcopy[1479]: ERROR ** FAILED TO COPY: x107c
07/12 14:41:43 migcopy[1479]: ERROR ** copy_for_method() ret=1
07/12 14:42:42 migmkspace[1696]: ERROR No entries in FHDB for filehandle 1000
```


```
07/12 14:46:12 migmkspace[1929]: ERROR No entries in FHDB for filehandle 1000
07/12 14:46:17 migmkspace[2062]: ERROR No entries in FHDB for filehandle 1000
07/12 15:08:21 mignospace[2309]: ERROR: No threshold specified.
07/12 15:09:03 migmkspace[2419]: ERROR No entries in FHDB for filehandle 1000
07/12 15:09:07 migmkspace[2559]: ERROR No entries in FHDB for filehandle 1000
07/12 15:10:13 migcopy[2612]: ERROR choose_src_gran no gran for as_filep=x0
07/12 15:10:13 migcopy[2612]: ERROR read_data: no good source granules found
```

FILES

`/usr/var/openv/hsm/database/migconfg`

VSM global-configuration file

`/tmp/hsm.log`

VSM global-log file

SEE ALSO

`migconfg(1M), mignewlog(1M), migrc(1M), startmigd(1M)`

migcleanup(1M)

NAME

migcleanup – displays or cleans up DMAPI sessions

SYNOPSIS

```
/usr/openv/hsm/bin/migcleanup [[-e] | [-l] | [-k] | [-g]
          | [-r token]] [-R reply] [-s sid]
```

Caution `migcleanup` is intended only for customer support engineers who are trained in its use.

AVAILABILITY

This command is only available on the DMAPI implementation of VSM.

DESCRIPTION

`migcleanup` is used to clean up orphaned DMAPI sessions. `migcleanup` prints all generated token and session lists to standard output. The lists exclude the current session.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

OPTIONS

`-e` List all outstanding event tokens for all sessions only if *sid* is not specified. Otherwise, list all outstanding event tokens for the *sid*. Tokens are listed in decimal. If there are no tokens, the output list is identical to the one generated with the `-l` option.

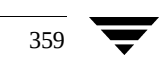
`-l` List all sessions only if *sid* is not specified. Otherwise, list *sid*.

`-k` Kill all sessions initiated by VSM for which the creating process no longer exists only if *sid* is not specified. Otherwise, kill *sid* unconditionally. Responds to all oustanding tokens.

`-g` Get all undelivered events for all sessions only if *sid* is not specified. Otherwise, get all undelivered events for *sid*.

Note The `-g` option does not respond to events. You must run `migcleanup` a second time with `-k` or `-r` to respond to events.

`-r` *token* Respond to event *token*. The value is assumed to be decimal. Requires that `-s` *sid* be specified.



`-R` *reply* Reply to event tokens with *reply*, defined as follows:

c = continue
a = abort (default)

`-R` is optional with `-k`, and `-r`.

`-s` *sid* Specifies the session to process. The value is assumed to be decimal. Required to be used with `-r` , and is optional with `-e`, `-l`, `-k`, and `-g`.

CAVEATS

- A session can not be killed if there are undelivered events. Use the `-g` option to get all undelivered events.
- A session can not be killed if it has outstanding tokens with exclusive rights. Use the `-r` option to respond to these tokens.

EXAMPLE

This example lists all outstanding event tokens for all sessions:

```
migcleanup
Session (37100) name: OVT.25719.tar-create
Session (37098) name: OVT.25714.tar-create
Session (37097) name: OVT.25713.tar-create
Session (37096) name: OVT.25712.tar-create
Session (36959) name: OVT.20137.migd
```

This example kills session 37100 unconditionally:

```
migcleanup -k -s 37100
Responding to events for sid 37100, tokens:
227
Destroying session 37100: OVT.25719.tar-create
```

To list the remaining sessions, execute migcleanup again:

```
migcleanup
Session (37098) name: OVT.25714.tar-create
Session (37097) name: OVT.25713.tar-create
Session (37096) name: OVT.25712.tar-create
Session (36959) name: OVT.20137.migd
```

This example kills all sessions initiated by VSM for which the creating process no longer exists:

```
migcleanup -k
Destroying session 37098: OVT.25714.tar-create
Destroying session 37097: OVT.25713.tar-create
Destroying session 37096: OVT.25712.tar-create
```

This example replies “continue” to token 6 for session 36959:

```
migcleanup -r 6 -s 36959 -R c
```

SEE ALSO

migalter(1M)

migconf(1M)

NAME

migconf – VSM file system configuration file

SYNOPSIS

dwpath/database/migconf

DESCRIPTION

Note The term *dwpath* refers to the path name of the directory containing the database and workdir directories. VSM uses the database directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the migconfg global-configuration file.

You define the migration parameters for each VSM managed file system in a *dwpath*/database/migconf file. Sites should change this file to match their specific needs. Use the xhsmadm GUI to make these changes.

The configuration file is divided into six general parameter blocks:

DEFAULTS

Describes general parameters for controlling the operation of VSM. For example, in this section you define whether to make one or two copies of each migrated file.

METHOD1 and METHOD2

Indicates the storage method (including types of media, volume sets, hints, and volume pools) to use for migrating the first and second file copies. For example, a site may choose to write the first copy to optical disk and the second copy to magnetic tape.

METHOD3 through METHOD8

Indicates the storage method (including types of media, volume sets, hints, and volume pools) to use for storing copies of migrated files at different migration levels. For example, a site may choose to take a file originally migrated to optical disk using METHOD1 and move it to magnetic tape at migration level 3 by using METHOD3.

METHOD Describes characteristics of all supported methods. Generally, sites do not need to change the defaults in this section. Some parameters, however, may be configured, including the criteria associated with each method for moving migrated files to another migration level.

`LEVEL` Defines the criteria associated with each migration level for moving migrated files to another migration level.

Note Move criteria for migration levels, if specified, take precedence over move criteria for method names, if specified.

`FILESYS` Defines a managed file system directory and its associated migration parameters such as the percentage of free space that should be remaining when VSM starts migrating files. Sites must change this section to specify the file systems and policies VSM is to use for selecting files for migration. There is a separate FILESYS entry for each managed file system that uses the `migconf` file.

Comments within a parameter block are currently not allowed. Parameters are separated by new lines or commas.

On startup, the migration daemon reads the configuration files. If you manually change the configuration file while the daemon is up, you can stop and restart the daemon so that it picks up the changes or you can signal the daemon as follows:

```
kill -INT `cat /usr/var/openv/hsm/workdir/migd.pid*`
```

If you use the `xhsmadm` GUI to make the changes, it signals the daemon for you.

Some parameter values for size or time in `migconf` can be specified as a base number followed by an optional multiplier. Possible multipliers are:

`B` 1 Block (512 bytes)

`K` 1 kilobyte (1,024 bytes)

`M` megabyte (1,048,576 bytes)

`G` gigabyte (1,073,741,824 bytes)

`d` 1 day (86400 seconds)

`h` 1 hour (3600 seconds)

`m` 1 minute (60 seconds)

`s` 1 second (1 second)

DEFAULTS SECTION

The parameters you can specify in the DEFAULTS section are as follows:

`mach_id` Integer (nonzero) identifier for each VSM managed file system. All file systems exporting or importing files between them must distinct; each must have a unique mach_id. Valid values range from 1 to FFFFFF. Default is 1000.

`quota` Maximum number of bytes a user may exclude from migration. The default is 10,000,000 bytes.

`copies` Number of migration copies of the disk file to be made. The maximum value is 2. The default is 2.

`unmount_delay`
Time in seconds a volume that is mounted in read mode remains mounted if no read request is received during that time. The default is 180.

`checksum`
If you set checksum to `1`, VSM calculates a check sum for each granule that is written. If `checksum` is set to `0`, VSM does not calculate a check sum. The default is `1`. During a read, VSM checks the check sum only if the granule was written with a check sum.

The following is an example DEFAULTS entry:

```
DEFAULTS quota=400000, copies=1, unmount_delay=300, checksum=1
```

METHOD1 AND METHOD2 SECTION

METHOD1 and METHOD2 parameters specify the types of media, volume sets, hints, and volume pools to which VSM writes migrated files. The METHOD1 parameter defines where the first copy is to be written. If copies=2, the METHOD2 parameter defines where the second copy is to be written. These values are used by `migpolicy`.

If copies=1, the optional METHOD2 parameter specifies the types of media, volume sets, hints, and volume pools for migration level 2.

The format is:

method_name.volume_set_number.hint.volume_pool

Names used for *method_name* must match the names listed in the METHOD section of the `migconf` file. Valid method names are:

`ad` Alternate magnetic disk

`ct` Tape

`dt` Tape

`mt` Tape

`op` Optical disk as tape with random seek - rewritable

`ow` Optical disk as tape with random seek - write once, read many

`ft` Remote method using ftp

`nb` NetBackup

Default attributes for tape method names are platform-dependent; refer to your *System Administrator's Guide* for details. Tape method names may be used interchangeably and optical disk method names may be used interchangeably if the default method attributes in `migconf` are modified to match the site configuration.

Note Method names `ct`, `dt`, `mt`, `op`, and `ow` do not apply to Storage Migrator Remote.

For *volume_set_number*, specify the number for the set of media IDs from which to choose the volume. Be sure to use different method names or volume set numbers for METHOD1 and METHOD2.

You can specify any of the following three values for the *hint* parameter:

`library` Access is immediate.

`operator`
Access requires operator action. 15 minutes is added to access time.

`vault` Access is off-site and requires 24 hours.

Use the optional *volume_pool* parameter to designate volume pools other than `HSM`. If not specified, the default volume pool is `HSM`.

Always configure the same volume pool for a given volume set, where a volume set is defined as a method name and its volume set number. The same volume set cannot exist in more than one volume pool.

If just one copy of each migrated file is to be made, configure the METHOD1 parameter and leave the METHOD2 parameter blank. For example, if the DEFAULTS parameter `copies`=1, then

```
METHOD1="ct.1.library"
METHOD2=""
```

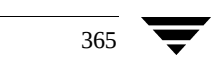
indicates that the file should be migrated using method name `ct`, volume set number 1, and that only one copy of the file is to be made.

If the DEFAULTS `copies` parameter indicates that two copies are to be made, use both METHOD1 and METHOD2, choosing different method names or volume set numbers for METHOD1 and METHOD2. For example, if `copies`=2, then

```
METHOD1="ct.1.library"
METHOD2="ct.2.vault"
```

causes one copy to be written to volume set 1 for cartridge tape, and one copy to volume set 2. On a cache, VSM first picks volumes on cartridge tape volume set 1 because the `hint` field indicates library.

Multiple volume sets may be specified in the same METHOD1 or METHOD2 parameter to allow VSM to use more than one drive at the same time. This speeds up the migration process although only one copy of the file is being made.



The following example causes migrated files to be distributed between two volume sets and uses two devices during migration. If `copies`=1, VSM makes only one copy of each file, but writes every other file to an alternate device to allow the writes to occur concurrently to two drives.

```
METHOD1="ct.1.library/ct.2.library"
```

METHOD3 THROUGH METHOD8 SECTION

Note This section of `migconf` does not apply to Storage Migrator Remote.

METHOD3 through METHOD8 parameters specify the types of media, volume sets, hints, and volume pools for migration levels 3 through 8, respectively.

The format is identical to that of METHOD1 and METHOD2:

method_name.volume_set_number.hint.volume_pool

Valid method names are:

`ad`	Alternate magnetic disk
`ct`	Tape
`dt`	Tape
`mt`	Tape
`op`	Optical disk as tape with random seek - rewritable
`ow`	Optical disk as tape with random seek - write once, read many

Default attributes for tape method names are platform-dependent; refer to your System Administrator's Guide for details. Tape method names may be used interchangeably and optical disk method names may be used interchangeably if the default method attributes in `migconf` are modified to match the site configuration.

The following are example METHOD3 and METHOD4 entries:

```
METHOD3="ct.10.library"
METHOD4="ct.20.vault"
```

This specifies an online tape library at migration level 3, and off-site tape storage at migration level 4.

METHOD SECTION

Each block of parameters in the METHOD section specifies the characteristics of all supported device method names. Generally, sites do not need to change the defaults in this section, but may want to modify the criteria associated with each method name for moving files from one migration level to another migration level.

The following METHOD parameters are available:

`name` The name of the device method. Valid values are `ct`, `dt`, `mt`, `op`, `ow`, `ad`, `ft`, and `dk`. Method name `dk`, used for premigration, is mandatory but not configurable.

Note Method names `ct`, `dt`, `mt`, `op`, and `ow` do not apply to Storage Migrator Remote.

`flags` Flags set for this method.

`MFLAG_OBSOLETE`

Media supports granule obsoletion (see `migmdclean(1M)`). Applies only to the `ad`, `ft`, `nb`, and `dk` methods.

Note Setting `MFLAG_OBSOLETE` causes dk entries to stay in the FHDB over cleanings initiated by `migrc`. This allows a cached file to be returned to premigration and processed by normal purge operations. To remove these dk entries and reduce the size of the FHDB, run `migmdclean`.

`MFLAG_APPEND`

Applies only to the `ct`, `dt`, `mt`, `op`, and `ow` methods. This flag allows VSM to place multiple migrations on the same volume by appending them to that volume until it is full to its configured limit. When writing to tape or optical media and `MFLAG_APPEND` is present, VSM continues to append to a volume until it is full and then writes on the next empty volume. This allows smaller migrations without wasting space. When `MFLAG_APPEND` is not present, each migration always starts on an empty volume.

When `MFLAG_APPEND` is present, VSM performs the following two steps to select a volume for a new migration:

1. Checks the *dwpath*`/database/VOLDB` file for a volume that belongs to the correct volume set that has the WRITING flag set in the VOLDB entry. (When VSM selects an empty volume for migration, it sets the WRITING flag in the VOLDB entry for that volume, indicating that VSM is using the volume for writing.)

2. If VSM finds a volume that is in the correct volume set and is also being written, it extends that volume with the new migration. Otherwise, it selects an empty volume (if possible) for the migration.

`MFLAG_EOT`

Applies only to `ct`, `dt`, and `mt` methods. This flag allows VSM to write to the media until end of tape (EOT) is encountered instead of using the capacity value. It is still necessary to specify capacity because VSM uses that value for calculating volume requirements during media consolidation.

`access` Time to access the media in seconds. VSM combines the `access` value with `hint`, `speed`, and file size to determine the relative time required to cache a copy of a file. The formula is as follows:

Relative cache time = access + hint + file size / speed

Where:

- Relative cache time is a value that VSM uses to determine which device method to use to cache first.
- Access is the value of the `access` parameter.
- Hint is the time delay indicated by the `hint` parameter.
- File size is the total size of the file in bytes.
- Speed is the value of the `speed` parameter.

If there is more than one copy of a file, VSM uses the relative cache time to determine which volume to select for a cache. VSM attempts to select the volume with the copy that it can cache in the least time. If that volume is not available, VSM then chooses another volume. VSM attempts to cache remote copies first (if they exist) before attempting to cache copies from local media.

`capacity` The capacity of the tape method (`ct`, `dt`, or `mt`) in bytes. You specify capacity for the `ft` and `nb` methods when registering the volumes (see `migreg(1M)`).

`speed` Relative rate at which the device can transfer data in bytes per second. As mentioned in the description for the `access` parameter, VSM uses speed when determining which copy of a file to cache.

`gran_size`
VSM divides files into granules. Each granule must fit on one volume. The `gran_size` parameter specifies the number of bytes in each granule that VSM writes to the device. The allowable granule sizes are: 128K, 256K, 512K, 1M, 2M, 4M, 8M, 16M, 32M, and 64M. Granule size is a power of 2 and an integral multiple of block size.

`block_size`
Block size in bytes to use when writing to the device. The allowable block sizes are: 512, 1K, 2K, 4K, 8K, 16K, 32K, 64K, 128K, and 256K.. The value must be a power of 2. Do not change this parameter after the initial configuration. It is not necessary to configure block size for method names `op` or `ow` because VSM determines the actual physical block size of optical volumes each time they are mounted or opened.

`density` Density of the tape or optical disk medium.

`dead_man_timeout`
: The maximum period of time in seconds that VSM should wait for an `ftp` or NetBackup request to complete or for a tape or optical mount to complete. The default is 3600 (one hour).

`port_number`
: `ftp` port number and used only for the `ft` method.

`demand_delay`
: The time a mount request waits before VSM unmounts a similar unused volume. If VSM identifies a similar mounted but unused volume whose unmount delay has not yet expired, it unmounts that volume as soon as the demand delay occurs. Otherwise, the mount request remains active until a drive becomes available. This is used only for the `ct`, `dt`, `mt`, `op`, and `ow` method names. Default values are 1 minute for method name `ct`, 3 minutes for method names `dt` and `mt`, and 20 seconds for method names `op` and `ow`.

Note Move criteria for method names do not apply to Storage Migrator Remote.

Note Move criteria for migration levels, if specified, take precedence over move criteria for method names, if specified.

`move_badness`
: The criterion that VSM uses to move files from one migration level to another after skipping those that are less than the minimum move age or size. VSM computes each migrated file's move badness and selects those with a move badness factor that equals or exceeds the configured value. The VSM move badness formula is as follows:

  move_badness=*move_age_weight* x (age of file in days)
  + or x (as specified by *move_weight_operator*)
  move_size_weight x (size of file in Kbytes)

  The default value is 30.

`move_age`
: Age of file (in days). VSM does not move files that have been migrated, moved, or accessed within this time. The default is 7. (Express value as a base number without a multiplier.)

`move_size`
: Size of file (in kilobytes). VSM does not move files smaller than this. The default is 0. (Express value as a base number without a multiplier.)

`move_age_weight`
: Weighting to be used for age in the move `badness` formula. The default is 1.

`move_size_weight`
: Weighting to be used for size in the move `badness` formula. The default is 1.

`move_weight_operator`
: The operator to be used (add or multiply) in calculating the `badness` factor. The default is multiply.

  Configuring `move_weight_operator = Site` specifies a site selected `badness` algorithm.

`move_flags`
: Mark FHDB entries for file copies at the source migration level either Dead, or Active, or Obsolete. Default is Dead for method names `ct`, `dt`, `mt`, `op`, and `ow`. Default is Obsolete for method name `ad`.

Be sure to delete or comment out method names not used at your site because extra unused methods cause additional processing. `dk` is always needed.

Method names `ct`, `dt`, and `mt` may be used interchangeably if the default method attributes in `migconf` are modified to match the site configuration. Method names `op` and `ow` may be used interchangeably if the default method attributes in `migconf` are modified to match the site configuration.
`/usr/var/openv/hsm/example/database/migconf`, the example `migconf` file, supports the following METHODs:

Disk File (dk)

This method is required for premigration. The access time for the staging area is always 0 to ensure that it is preferred over all other methods.

```
METHOD   name=dk,
         access=0,
         capacity=50M,
         speed=1M,
         gran_size=1M,
         block_size=64K
```

Tape (ct)

```
METHOD   name=ct,
         flags=MFLAG_APPEND | MFLAG_EOT
         block_size=32K,
         access=15s,
         gran_size=2M,
         capacity=20G,
         speed=10M,
```

```
            density=hcart,
            dead_man_timeout=3600,
            demand_delay=3m
```

Tape (dt)

```
   METHOD   name=dt,
            flags=MFLAG_APPEND | MFLAG_EOT
            block_size=32K,
            access=2m,
            gran_size=2M,
            capacity=35G,
            speed=5M,
            density=dlt,
            dead_man_timeout=3600,
            demand_delay=3m
```

Tape (mt)

```
   METHOD   name=mt,
            flags=MFLAG_APPEND | MFLAG_EOT,
            block_size=32K,
            access=30s,
            gran_size=2M,
            capacity=50G,
            speed=6M,
            density=8mm,
            dead_man_timeout=3600,
            demand_delay=3m
```

Alternate Magnetic Disk (ad)

```
   METHOD   name=ad,
            flags=MFLAG_OBSOLETE,
            block_size=1K,
            access=10s,
            gran_size=2M,
            capacity=320M,
            speed=800K
```

Optical Disk as Tape, rewritable (op)

```
   METHOD   name=op,
            flags=MFLAG_APPEND,
            block_size=512,
            access=1m,
            gran_size=2M,
```


```
            capacity=280M,
            speed=200K,
            density=odiskwm,
            dead_man_timeout=3600,
            demand_delay=20s
```

Optical Disk as Tape, write once, read many (ow)

```
   METHOD   name=ow,
            flags=MFLAG_APPEND,
            block_size=512,
            access=1m,
            gran_size=2M,
            capacity=280M,
            speed=200K,
            density=odiskwo,
            dead_man_timeout=3600,
            demand_delay=20s
```

Remote method using ftp (ft)

```
    METHOD  name=ft,
            flags=MFLAG_OBSOLETE,
            block_size=32K,
            access=20s,
            gran_size=2M,
            capacity=0,
            speed=300K,
            dead_man_timeout=3600,
            port_number=21
```

NetBackup (nb)

```
    METHOD  name=nb,
            flags=MFLAG_OBSOLETE,
            block_size=32K,
            access=2m,
            capacity=0,
            speed=300K,
            dead_man_timeout=3600
```

LEVEL SECTION

Note This section of `migconf` does not apply to Storage Migrator Remote.

Each block of parameters in the LEVEL section specifies the criteria associated with each migration level for moving files from one migration level to another migration level.

Note Move criteria for migration levels, if specified, take precedence over move criteria for method names, if specified.

`move_badness`
: The criterion that VSM uses to move files from one migration level to another after skipping those that are less than the minimum move age or size. VSM computes each migrated file's move badness and selects those with a move badness factor that equals or exceeds the configured value. The VSM move badness formula is as follows:

  move_badness=*move_age_weight* x (age of file in days)
  + or x (as specified by *move_weight_operator*)
  move_size_weight x (size of file in Kbytes)

  The default value is 30.

`move_age`
: Age of file (in days). VSM does not move files that have been migrated, moved, or accessed within this time. The default is 7. (Express value as a base number without a multiplier.)

`move_size`
: Size of file (in kilobytes). VSM does not move files smaller than this. The default is 0. (Express value as a base number without a multiplier.)

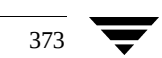
`move_age_weight`
: Weighting to be used for age in the move `badness` formula. The default is 1.

`move_size_weight`
: Weighting to be used for size in the move `badness` formula. The default is 1.

`move_weight_operator`
: The operator to be used (add or multiply) in calculating the `badness` factor. The default is multiply.

  Configuring `move_weight_operator = Site` specifies a site selected `badness` algorithm.

`move_flags`
: Mark FHDB entries for file copies at the source migration level either Dead, or Active, or Obsolete. Default is Dead for method names `ct`, `dt`, `mt`, `op`, and `ow`. Default is Obsolete for method name `ad`.



FILESYS SECTION

The FILESYS section specifies a managed file system directory and its associated migration parameters. VSM uses these parameters to manage space in the managed file system. There is a separate FILESYS entry for each managed file system that uses the `migconf` file. FILESYS parameters are as follows:

`name` The path to the directory in the file system that VSM manages. This may be less than a full file system. It must be the real name and not a link name, and must not contain a trailing slash. Do not specify more than one directory in the same managed file system. The default is `/hsm1`.

If you configure more than one name, each name must be in a unique managed file system.

`freespace`
Specifies the percentage of space to be kept free on the file system. If file system free space falls below this amount, migration should be restarted. The default is 10.

`lowwater`
The percentage of free space that stops the selection of files to be migrated. If specified, `lowwater` must be greater than or equal to the value specified for `freespace`. The default is 0 (the parameter is ignored).

Note Files listed in a user's `.migrate` file are selected after `lowwater` is reached, so the percentage of free space may be larger than expected.

Space in premigration is not considered free.

`migbatch` makes only one pass through the filesystem with the current badness as it tries to achieve `lowwater`. Each time `mignospace` executes and finds no files in premigration to purge, the current badness is cut in half, causing more files to be selected. `migrc` and `migbatch` reset the current badness to the value you initially configured in the `migconf` file.

`purgemark`
The percentage of free space that stops the purging of premigrated files. If specified, `purgemark` must be greater than or equal to the value specified for `freespace` and less than `lowwater` unless `lowwater=0`. The default is 0 (the parameter is ignored).

`badness` The criterion that VSM uses to select files for migration after skipping those that are less than the minimum age or size. Files with a `badness` factor that equals or exceeds this value are selected for migration.

badness=*age_weight* x (age of file in days)
+ or x (as specified by *weight_operator*)
size_weight x (size of file in Kbytes)

where file `age` is the number of days since the last access or since last modification, whichever is most recent.

The default value is 0.

`min_age` Age of file (in days). VSM does not select files for migration that have been either accessed or modified within this time. The default is 7. Generally, age should be greater than 0 to prevent VSM from migrating files on the same day they are created. (Express value as a base number without a multiplier.)

`min_size`
Size of file (in kilobytes). VSM does not select files for migration that are smaller than this. The default is 8. (Express value as a base number without a multiplier.)

`age_weight`
Weighting to be used for age in the `badness` formula. The default is 1.

`size_weight`
Weighting to be used for size in the `badness` formula. The default is 1.

`weight_operator`
The operator to be used (add or multiply) in calculating the `badness` factor. The default is multiply.

Configuring `weight_operator = Site` specifies a site selected `badness` algorithm.

`slice` Number of bytes from the head of a file to keep on disk when the file is migrated. These bytes are also migrated but VSM keeps a copy of them on disk even when it purges the file from premigration, thus allowing this number of bytes to be read without caching the file. Valid values range from 0 to 65536 (64 kilobytes). On DMAPI implementations the upper limit to the slice is 2147483648 (2 gigabytes). A value of 0 implies no bytes will be kept in the file system. The default is 0.

`readahead`
Minimum number of kilobytes beyond the current `read` request that VSM partially caches to disk. Applies only to DMAPI implementations of VSM. A `readahead` of 0 means partial file caching with no minimum caching beyond the read request. A `readahead` of -1 disables partial file caching (default).

The FILESYS section also specifies three parameters used in making additional file space available quickly.

`give_up_time` (Time Increment)
: Maximum time in minutes `migcopy` runs before stopping an accelerated file space availability operation. The default is 60. A value of 0 signifies no limit.

`give_up_files` (File Increment)
: Maximum number of files processed by `migcopy` and `migsweep` before stopping an accelerated file space availability operation. A value of 0 signifies no limit (default).

`give_up_kbytes` (Space Increment)
: Minimum amount of disk space (in kilobytes) freed by `migcopy` and `migsweep` before stopping an accelerated file space availability operation. The default is 1,048,576. A value of 0 signifies no limit.

The FILESYS parameters also specify the purge parameters for the file systems to which this specific *dwpath*`/database/migconf` file applies. The `migconf` file contains a separate set of purge parameters for each file system. Those parameters are as follows:

`purge_badness`
: The criterion that VSM uses to select premigrated files for purging after skipping those that are less than the minimum purge age or size. Files with a `purge_badness` factor that equals or exceeds this value are selected for purging.

  purge_badness=*purge_age_weight* x (days since last access)
  + or x (as specified by *purge_weight_operator*)
  purge_size_weight x (size of file in Kbytes)

  The default value is 0.

`purge_min_age`
: Age of file (in days since migrated). VSM does not purge premigrated files migrated within this time. The default is 0. (Express value as a base number without a multiplier.)

`purge_min_size`
: Size of file (in kilobytes). VSM does not purge files smaller than this. The default is 0. (Express value as a base number without a multiplier.)

`purge_age_weight`
: Weighting to be used for age in the `purge_badness` formula. The default is 1.

`purge_size_weight`
: Weighting to be used for size in the `purge_badness` formula. The default is 0.

`purge_weight_operator`
: The operator to be used (`add` or `multiply`) in calculating the `purge_badness` factor. The default is `add`.

Configuring `purge_weight_operator = Site` specifies a site selected `purge_badness` algorithm.

An example FILESYS configuration is as follows:

```
FILESYS  name=/hsm6,
         freespace=10, badness=2000,
         min_age=1, min_size=1,
         age_weight=1, size_weight=1,
         weight_operator=multiply,
         slice=8192, purgemark=0
```

In the above example, VSM manages the file system directory `/hsm6`. With the above parameter values, VSM does not migrate files on the same day they were created (because min_age=1), nor files less than a kilobyte (because min_size=1). If VSM encounters a 102400 byte file that has not been accessed for 30 days, it computes the badness to be 3000 (30 days x 100 Kbytes). Because the badness value is set to 2000 in this example, this file would be a candidate for migration by VSM.

SEE ALSO

`HSM(1M)`, `migconfg(1M)`, `miglow(1M)`, `migmdclean(1M)`, `migreg(1M)`, `migtestbadness(1M)`, `migthreshold(1M)`, `startmigd(1M)`

migconfg(1M)

NAME

`migconfg` – global configuration file for VSM

SYNOPSIS

`/usr/var/openv/hsm/database/migconfg`

DESCRIPTION

This file has one or more entries, each starting with the string HSMDEV followed by one or more parameters. Not all parameters are required. The parameters specified in this file let you partition VSM into easily manageable partitions by name (*hsmname*) and define location of files for each hsmname. Use the `xhsmadm` GUI to make changes to `migconfg`.

The following parameters are available:

hsmname The logical name for this HSMDEV entry. *hsmname* must be a unique alphanumeric value, such as `project_x` or `px45`. Although numeric values such as *1234* are accepted, their use is not recommended as the *hsmname* is embedded in path names and some utilities may not work correctly. The maximum length is 32 characters.

fspath The path name of the managed file system for this HSMDEV entry. It is used for computing the device number of the file system. *fspath* is the mount point for the file system. It cannot be the full path to the filesys name if the filesys name is a subdirectory in *fspath*. This parameter is required.

Caution Always create the database directory in a local file system that VSM does not manage. This eliminates the possibility of migrating files from the database or workdir directories.

dwpath The path name of the directory containing the database and workdir directories for this HSMDEV entry. The default is `/usr/var/openv/hsm/`*hsmname*, where *hsmname* is the value you supply for *hsmname*.

lgpath The path name of the log file for this HSMDEV entry. The default is `/tmp/hsm.`*hsmname.log*, where *hsmname* is the value you supply for *hsmname*.

`state` Specifies whether VSM processes this HSMDEV entry when it automatically migrates or caches files from the file system indicated by *fspath*. The `state` parameter can have a value of either `1` or `0`, where `1` is on and `0` is off. Default is `1` (on). If you set `state` to `0` (off), users cannot

access migrated files and an error is returned to the user's application program. Setting `state` to `0` also prevents automatic migrations from occurring when the kernel detects a low-space condition.

The following list shows VSM files in the *dwpath*`/database` directory (for a configured VSM):

dwpath`/database/FHDB - File-handle database file`

dwpath`/database/FHDB.LK* - FHDB lock file`

dwpath`/database/FHSEQF - Next-file-handle sequence number`

dwpath`/database/NEXTVOLM1 - Next volume set for copy 1`

dwpath`/database/NEXTVOLM2 - Next volume set for copy 2`

dwpath`/database/NEXTVOLM3 - Next volume set for moving to level 3`

dwpath`/database/NEXTVOLM4 - Next volume set for moving to level 4`

dwpath`/database/NEXTVOLM5 - Next volume set for moving to level 5`

dwpath`/database/NEXTVOLM6 - Next volume set for moving to level 6`

dwpath`/database/NEXTVOLM7 - Next volume set for moving to level 7`

dwpath`/database/NEXTVOLM8 - Next volume set for moving to level 8`

dwpath`/database/VOLDB - Volume-database file`

dwpath`/database/VOLSEQF - Next volume sequence number`

dwpath`/database/migconf - HSM configuration file`

dwpath`/database/migsweep.site - Site migration and purge criteria`

dwpath`/database/migsweepm.site - Site move criteria`

dwpath`/database/`*hsmname*`.FLUSH - Tape mark frequency`

In addition, the following file is applicable to DMAPI implementations:

dwpath`/database/`*hsmname*`.IHAND - Inode to handle conversion`

The following lists show the files in the VSM directory `workdir` (for a configured VSM):

- Files for each volume set (method name.volume set number), for example:

  dwpath`/workdir/`*hsmname*`/copydb.`*xx*`.1.library`

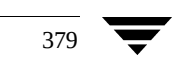
  dwpath`/workdir/`*hsmname*`/copydb.`*xx*`.2.library`

  where *xx* is the method name(`ct`, `dt`, `mt`, `op`, `ow`, `ad`, `ft` or `nb`).

- Temporary files, the absence of which does not necessarily indicate a problem, for example:

  dwpath`/workdir/`*hsmname*`/.migbatch.running - migbatch PID`

  dwpath`/workdir/`*hsmname*`/.migconsweep.running - migconsweep PID`



dwpath/workdir/*hsmname*/.miglow.running - miglow PID

dwpath/workdir/*hsmname*/.migmdclean.running - migmdclean PID

dwpath/workdir/*hsmname*/.migmove.running - migmove PID

dwpath/workdir/*hsmname*/.mignospace.running - mignospace PID

dwpath/workdir/*hsmname*/.migrc.running - migrc PID

- Other files, for example:

  dwpath/workdir/mig.lck - Managed file system lock file

  dwpath/workdir/*hsmname*/.p_badness - Current purge badness value

  dwpath/workdir/*hsmname*/.s_badness - Current badness value

To find the actual filenames used in your system, look at the *dwpath* and *hsmname* parameters in your global-configuration file.

VSM uses a global-log file to log messages from the migration daemon and volume daemon, and a log file for each HSMDEV entry to log messages from migration processes running on behalf of file system the HSMDEV entry defines. The path name of the global-log file is:

```
/tmp/hsm.log
```

You can specify a path name for the HSMDEV log files with the *lgpath* parameter in the /usr/var/openv/hsm/database/migconfg file.

You can obtain the values of migconfg parameters by using the migdbdir command. For example, the following prints the value of the *lgpath* parameter for the HSMDEV entry named *alpha*:

```
migdbdir alpha 3
```

On startup, the migration daemon reads the configuration files. If you change the global-configuration file while the daemon is up, you can stop and restart the daemon so that it picks up the changes or you can signal the daemon as follows:

```
kill -INT `cat /usr/var/openv/hsm/workdir/migd.pid*`
```

If you use the xhsmadm GUI to make the changes, it signals the daemon for you.

EXAMPLE

The following are example /usr/openv/hsm/database/migconfg entries.

```
HSMDEV   hsmname=HSMNAME,
         fspath=/sd7,
         dwpath=/usr/var/openv/hsm/HSMNAME,
         lgpath=/tmp/hsm.HSMNAME.log,
         state=1
```

```
HSMDEV   hsmname=home,
         fspath=/usr/home,
         dwpath=/usr/var/openv/hsm/home,
         lgpath=/tmp/hsm.home.log,
         state=1
```

SEE ALSO

HSM(1M), migconf(1M), migdbdir(1M)

migcons(1M)

NAME

migcons – consolidate VSM tape and optical disk volumes

SYNOPSIS

```
/usr/openv/hsm/bin/migcons [-F] [-N] hsmname
        one | two out_method out_volume_set
        label.method.volset [label.method.volset] . . .
```

DESCRIPTION

Periodic volume consolidation is necessary because VSM does not release unusable volume space automatically. Therefore, as files are migrated, cached, and modified, the amount of unusable space on a volume grows. Run `miggetvol` or `migdbrpt` to help determine which VSM volumes have low space utilization or otherwise are candidates for consolidation.

Unusable space is occupied by file data that has been marked obsolete or dead. Obsolete data represents an earlier version of a modified file. It is possible to retrieve obsolete data up until the time these entries are marked dead. To mark obsolete entries dead, run `migmdclean` before consolidating volumes.

`migcons` consolidates one or more input volumes onto a set of output volumes. Input volumes can belong to different methods while output volumes must all belong to a single method.

The `migcons` command:

- Copies active data from the input volume(s) onto the output volume(s). If you did not run `migmdclean` first, `migcons` also copies obsolete data onto the output volumes. `migcons` does not copy data marked dead in the FHDB.
- Removes all references to the file granules on input volumes from the file-handle database
- Removes the volume entry for the input volume from the volume database (VOLDB)

Note When one side of an optical disk is consolidated, the volume entry for that input volume is marked dead and not removed from the VOLDB unless the other side of that optical disk is also marked dead.

After consolidation, all data other than dead entries is available on the output volumes. The input volumes must be reregistered before VSM will use them again (see the `migreg(1M)` man page). Although it is possible to consolidate ow volumes (write once, read many), it is not possible to recycle them.

migcons can use two drives simultaneously for reading and writing volumes. If sufficient drives are not available, migcons supports a two-step consolidation process, which is described in the options section.

You can use migselect to generate a list of volumes that need to be consolidated.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon migd is running.

OPTIONS

`-F` Perform consolidation without feasibility check to assess the capacity of the output volumes. See CAVEATS.

`-N` Selects a new (empty) volume for output, disregarding the MFLAG_APPEND flag for the method. If you do not specify `-N`, migcons selects the output volume based on the MFLAG_APPEND flag:

- If MFLAG_APPEND is set, migcons selects and appends the data to the volume currently being written.
- If MFLAG_APPEND is not set, migcons selects a new volume.

hsmname HSMDEV entry specifying the path to the database files for the volumes you want to consolidate. See migconfg(1M) for more information on HSMDEV entries.

`one | two`

Specifies whether the consolidation occurs in one or two steps. This parameter is required.

- One-step consolidation copies files directly from the input media to empty volumes under the output method.
- Two-step consolidation copies from the input media to alternate magnetic disk media (method ad) and then to empty volumes under the specified output method.

One-step consolidation is always faster. However, sites that consolidate input volumes using the same method as the output method but have only one device for the output method must use a two-step process. For example, a site that uses cartridge tape but has only one cartridge-tape drive available must use a two-step process.

Prior to performing two-step consolidation, the site must register a volume in alternate magnetic disk (ad) volume set 0 (see migreg(1M)).

out_method

Output method name for consolidation. This parameter is required. Valid values are ct, dt, mt, op, or ow. See CAVEATS for method ad.

out_volume_set

Output volume set to use for consolidation. Volumes selected for writing belong to this volume set. You can use this parameter to ensure that multiple copies of the same file are consolidated on different volumes. This parameter is required.

label.method.volset

Label, method, and volume set of the input volume (for example, CWra01.mt.1). You must supply at least one volume. Do not specify output volumes here; VSM automatically selects output volumes. Valid method values are `ct`, `dt`, `mt`, `op`, or `ow`. See CAVEATS for method `ad`.

CAVEATS

- Do not use `migcons` on method ad volumes. Instead, use the `migmdclean` command to free up disk space.
- You cannot consolidate volumes that use the `ft` or `nb` method.
- `migcons` performs a feasibility check before attempting consolidation and allows consolidation only if the empty volumes for the output method have enough room for the files from the input volumes. No additional media is registered.
- If consolidation is forced with the `-F` option and output volume capacity is inadequate, available tape or optical disk media in the same volume pool as the first input volume being consolidated are registered automatically as needed to provide adequate output volume capacity for writing the data from the input volumes. If output volume capacity is still inadequate, the command will fail when output volumes become full. These output volumes are marked full in the VOLDB, but no FHDB changes occur for input volumes being consolidated.
- Locked VOLDB entries for the input volumes fail the feasibility check and prohibit consolidation. Run the `migrc` command prior to consolidation to clear the locks.
- Do not run `migcons` and `migmove` simultaneously if they both are taking source from the same volumes. The results of such an action are undefined.

EXAMPLE 1

The following example consolidates `reel01` and `reel02` cartridge tape volumes (method `ct`) to new cartridge tapes selected by VSM. A one-step process is used because the site has multiple devices available.

```
migcons  alpha one  ct 1  reel01.ct.1  reel02.ct.1
```

EXAMPLE 2

Assuming that a site has just one tape device under method `ct`, the following example consolidates volumes `reel01` and `reel02` under method `ct` to volume(s) under out_method `ct`. This occurs in two steps, with volumes using method `ad` being taken as staging volumes.

```
migcons  alpha two  ct 1  reel01.ct.1  reel02.ct.1
```

EXAMPLE 3

The following example selects input volumes from `ct` volume set 2 based on volume occupancy limits ranging from 0.00 through 5.50 percent full. The consolidation is a one-step process to output method `ct`.

```
migcons  alpha one  ct 2  `migselect  alpha  0.00-5.50 2 ct`
```

EXAMPLE 4

The following example consolidates volumes under multiple input methods `ct` and `dt` to volumes under output method `ct`.

```
migcons  alpha one  ct 1  `migselect  alpha  0.00-5.50 1 ct dt`
```

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath`/database/VOLDB`

Volume database.

dwpath`/database/FHDB`

File-handle database.

dwpath`/database/CONS_INVOL`

Consolidation input volume list file generated during consolidation.

dwpath`/database/CONS_OUTVOL`

Consolidation output volume list file generated during consolidation.

`/usr/var/openv/hsm/database/migconfg`

Global-configuration file for VSM.

SEE ALSO

migconfg(1M), migdbrpt(1M), miggetvol(1M), migmdclean(1M), migreg(1M), migrc(1M), migselect(1M), migrecycle(1M)

migconsweep(1M)

NAME

migconsweep – enable constant file system sweeping

SYNOPSIS

`/usr/openv/hsm/bin/migconsweep [-s` *sleep_time*`]` *hsmname*

DESCRIPTION

The `migconsweep` command enables constant sweeping of the managed file system instead of the normal sweeping process performed by VSM. Sweeps occur periodically, following interruptions determined by the *sleep_time* variable. This makes it less likely that the file list of migration candidates will be empty when needed.

Constant sweeping attempts to keep the list of migration candidates from becoming empty by periodically checking the list and resuming sweeping if necessary.

If `mignospace` is running when the file system is swept by `migconsweep`, the accelerated file space availability feature of `mignospace` is implemented whereby the sweeping process is interrupted as soon as any one of three configurable file system attributes is satisfied: time increment, file increment, or space increment. See man page mignospace(1M) for more information.

Once initiated, constant sweeping continues to run until the process is terminated with the `kill` command.

This command may be run without regard to whether the migration daemon `migd` is running, but the hsm state must be active.

OPTIONS

`-s` *sleep_time*
: Pause for this interval between sweeping operations, where *sleep_time* is the time in seconds that this command pauses before resuming a sweep of the file system. Default is 60.

hsmname
: HSMDEV entry specifying the path to the database files for the volumes you want to consolidate. See `migconfg(1M)` for more information on HSMDEV entries.

CAVEATS

- Constant sweeping uses system resources that may adversely affect overall VSM performance, particularly during periods of heavy system usage.

EXAMPLE

The following command initiates constant sweeping with a sleep time of 10 minutes (600 seconds) between sweeps for hsmname alpha.

```
migconsweep -s 600 alpha &
```

To terminate constant sweeping, kill the process with this command.

```
kill pid
```

FILES

dwpath/workdir/*hsmname*.migconsweep.running

The process identifier (pid) for migconsweep.

dwpath/workdir/*hsmname*.migsweep

The list of files selected to be premigrated.

SEE ALSO

migbatch(1M), mignospace(1M)

migdbcheck(1M)

NAME

migdbcheck – check the FHDB and / or the VOLDB for consistency

SYNOPSIS

```
/usr/openv/hsm/bin/migdbcheck [-F | -P] [-V]
          [-v -r -s -h] hsmname
```

DESCRIPTION

`migdbcheck` checks the file-handle database (FHDB) and checks the volume database (VOLDB) for consistency with the filesystem. Each database may be checked independently or both databases may be checked together with a single command.

Note `migdbcheck` defaults to check only the FHDB. Checking both the FHDB and VOLDB with a single command is faster than checking the databases independently with two commands.

You should correct database inconsistencies immediately to guard against future problems.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running. See CAVEATS.

FHDB Checking

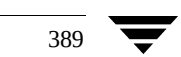
`migdbcheck` checks the file-handle database (FHDB) in two ways:

- It verifies that the filesystem does not contain any files that have a file handle but no entry in the FHDB.
- It verifies that the FHDB does not have any entries without files in the file system.

`migdbcheck` automatically checks all filesystems that use the FHDB specified by *hsmname*. `migdbcheck` can also check for consistency between the FHDB and premigrated files.

This command generates a list of all migrated files from the filesystem(s) specified, and compares this list with the contents of the FHDB. The following error conditions are detected by `migdbcheck`:

- Migrated files exist that do not have enough entries in the FHDB and do not have a valid `dk` method name entry. If there is a valid `dk` method name entry, VSM checks the premigration directories and the copydb databases.
- Active FHDB entries exist for which there are no corresponding migrated files in the filesystem. These are sometimes referred to as orphan entries.



- FHDB entries which have been flagged *obsolete* are present for files that still exist. Each migration level is checked.
- FHDB entries which have been flagged *dead* are present for files that still exist. Each migration level is checked.
- The FHDB is not in sorted order.
- Duplicate FHDB handles exist in the file system(s).
- The FHDB sequence number in the FHSEQF file is incorrect.
- FHDB entries exist that are flagged as *locked*.
- FHDB entries exist that are flagged as *failed*.
- There are `dk` entries in the FHDB but the data is purged.
- There are files in the premigration directory that do not correspond to entries in the FHDB. (Applicable only to kernel-based implementations).
- There are files in the file system whose full pathnames exceed 1024 characters. These files cannot be selected for migration, and may eventually fill the file system.

VOLDB Checking

`migdbcheck` checks the volume database (VOLDB) in two ways:

- It verifies that all files recorded in the VSM file-handle database (*dwpath*`/database/FHDB`) have an associated entry in the VSM volume database (*dwpath*`/database/VOLDB`).
- It verifies that all active VOLDB entries are associated with at least one FHDB entry.

`migdbcheck` automatically checks all filesystems that use the VOLDB specified by *hsmname*. The following error conditions are detected by `migdbcheck`:

- Active FHDB entries exist that reference nonexistent VOLDB entries.
- Active VOLDB entries exist for which there are no corresponding FHDB references.
- VOLDB entries exist in write mode for which there are no corresponding FHDB references.

OPTIONS

`-F`	Check all entries in the FHDB configured for *hsmname*. This is the default condition if neither `-P` nor `-V` are specified.
`-P`	Check consistency between `dk` entries and premigrated files. This overrides `-F`.
`-V`	Check all entries in the VOLDB configured for *hsmname*.

`-v`	Verbose mode; output is directed to standard output as well as to the output files.
`-r`	Repair FHDB entries. See CAVEATS and EXAMPLES. (Does not repair VOLDB entries.)
`-s`	Save the list of migrated files.
`-h`	Print help information about this command.
hsmname	HSMDEV entry that specifies the path to the database files you want to check. Check all file systems sharing this FHDB or VOLDB. See `migconfg(1M)` for more information on HSMDEV entries.

CAVEATS

- If `migdbcheck` is run while the migration daemon `migd` is running and the VSM state is active, a warning message tells you that the results of this command may be in error because of simultaneous migration activity. Nevertheless, VSM gives you an option to continue, but these results are for information only. If the VSM state is active, `migdbcheck` does not repair the FHDB.
- If `migbatch` or `miglow` or `migmove` run simultaneously with `migdbcheck`, the results of `migdbcheck` may be in error.
- Be sure to analyze very carefully any error messages generated by `migdbcheck` before taking corrective action. The examples that follow show only a few of the error situations that could arise.

EXAMPLES

Example 1: FHDB Checking

The following example checks all filesystems that use the FHDB specified by hsmname alpha. In this case, filesystem ov3 specified by hsmname beta shares the same FHDB, so it is also checked.

```
migdbcheck -F alpha
```

Typical messages in response to this command are shown below:

```
-- INFO: Also checking file system /ov3 for hsmname=beta
-- INFO: 3 files are in the premigration directory
-- INFO: There are 54 migrated files in the /alpha file system
-- INFO: There are 33 migrated files in the /beta file system
-- INFO: There are 87 migrated files.
-- INFO: There are 1302 entries in the FHDB.
```


```
-- ERROR: 1 FHDB entries with no file with a matching handle
   were found.
-- INFO: 1 files have the error flag set.
-- INFO: 2 files have fewer than 2 copies made.
-- INFO: 21 files have more than 2 copies made.
--       Extra copies found at level 1
--       Extra copies found at level 2
```

On kernel-based implementations, the following error message could appear:

```
-- ERROR: 3 FHDB 'dk' entries with no corresponding files in
   the premigration directories were found.
```

To repair FHDB entries, set VSM state to inactive and specify the `-r` option in the command:

```
migdbcheck -F -r alpha
```

A typical error message in response to this command is shown below:

```
-- ERROR: 1 FHDB entries with no file with a matching handle
   were found.
```

This would be followed by:

```
Do you wish to set them inactive?? [aynq](n):
```

An answer of A will correct all the entries with no further prompting.
An answer of Y will then lead to a one-by-one listing of the problem entries, as shown below.
An answer of N or Q will move on to the next problem analysis and leave the file `migdbcheck-orphan.pid` in the `/tmp` directory.

```
Set handle: 00001000  flags: 00000000 path:
/hsm1/test/test_foo  to inactive? [aynq](n):
```

A response of A will correct all remaining entries without prompt.
A response of Y will correct the entry.
A response of N will not correct the entry.
A response of Q will stop processing the list and not correct any more errors.

Another typical error message in response to this command is shown below:

```
-- ERROR: 3 FHDB 'dk' entries with no corresponding files in
   the premigration directories were found.
```

This would be followed by:

```
Do you wish to set them inactive? [aynq](n):
```

and processing would proceed as indicated above, using the file `migdbcheck-dk-orphan.pid` in the `/tmp` directory.

Another response could report extra copies found at different migration levels.

```
-- INFO: 21 files have more than 2 copies made.
--       Extra copies found at level 1
--       Extra copies found at level 2
```

It is possible to alter the flags field of the FHDB entry for these extra copies using the `-i` argument of the `migsetdb` command, but the administrator must first evaluate the files to decide which flags to alter for each level. The files `migdbcheck-level-X.`*hsmname*`.pid` in `/tmp` are used to indicate the files that exist at each migration level. The character *X* is replaced by the appropriate migration level, 1 through 8.

An error message alerts the administrator when files exist in the file system that can never be migrated because their full pathnames are too long.

```
-- WARNING: Files with path lengths exceeding 1024 were found
   in the filesystem.
```

The following information message indicates that VSM has cached some premigrated files before making any copies:

```
-- INFO: 6 cached files with no active entries in the FHDB
   were found.
```

Note If this message continually shows large numbers of files, VSM is selecting too many frequently accessed files for migration. Tune the file badness formula to be less aggressive.

VSM lists all such files in verbose mode. Run `migloc` on each file so identified. There is no problem if the output resembles this:

```
migloc /hsmr/raa/7/6/4/4/1/1/2/f0
Status: Cached      code    raa    /hsmr/usr1/7/6/4/4/1/1/2/f0
No matching entries found in the FHDB handle: A14C2.
Problems getting migration data on file.
```

The next time the file gets migrated it will get a new handle and be placed in the copydb worklists.

Example 2: VOLDB Checking

The following example checks all filesystems that use the VOLDB specified by hsmname delta.

```
migdbcheck -V delta
```


Typical messages in response to this command are shown below:

```
-- INFO: checking hsmname=delta fspath=/hsm1
-- INFO: There are 15 entries in the VOLDB.
```

If you have just installed VSM and have not registered any volumes using `migreg` or if no files have been migrated out, the following message appears:

```
-- WARNING: no volumes in VOLDB referenced by FHDB
```

If an entry is locked by the indicated process, the following message appears. This could occur if you run `migdbcheck` when VSM is actively being used and the migration daemon (`migd`) is running. See CAVEATS.

```
-- WARNING: VOLDB entry 00001000 is locked by 4660
```

VSM maintains the VOLDB in sorted order. If an entry is out of order, the following message error appears:

```
-- ERROR: /usr/var/openv/hsm1/database/VOLDB is not in the
correct order! handle 1000 preceeded 999.
You must sort the file and rerun migdbcheck. ABORTING
```

Use the `sort` command to put the VOLDB in sorted order.

Another typical error message in response to this command is shown below:

```
-- ERROR: Files on missing volid 00001000
```

To list all such FHDB entries, run `migdbcheck` in verbose mode.

```
migdbcheck -Vv delta
-- ERROR: Files on missing volid 00001000
00001000|00000001|00000000|00001000|00000000|000181CD|00000000
|000181CD|0000A15E|00000000|00000000|3155C9F9|3155CB02|00000000
|ad|/hsm1/machid1/1|yak|root|sys|library|1 0||Auto HSM run
|00000008|00000000|00000001|
```

A similar error message is produced by `migdbrpt`. The FHDB references a volume that is not in the VOLDB. In this example, volume ID 1000 was not in the VOLDB. Following the initial error is a display of the FHDB entries that reference the missing volume. Use this list to determine the specific files that the error affects and take the following actions:

1. If the missing entry exists on a recently backed up VOLDB, you may add that entry to the VOLDB.
2. Run `migtscan`, `migopscan`, `migadscan`, `migftscan`, or `mignbscan` on the volume. This creates a file *dwpath*`/database/VOLDB.label`. If this file looks reasonable, use an editor to insert this file into *dwpath*`/database/VOLDB`. Keep the

file *dwpath*/database/VOLDB sorted with respect to handle and machine ID, which are the first two fields of each line. Rerun migdbcheck to determine the effect of this change.

3. If only a few files are affected, you can restore the user files from the site's backup tapes and delete the entries from *dwpath*/database/FHDB.

Other typical messages indicate that no file currently references this volume:

```
-- 00000998 in VOLDB, but no FHDB reference to it
-- Writing Entry 00000999 in VOLDB, but no FHDB reference to it
```

This is normal if all of the files on the volumes have been cached or removed. Registered but unwritten volumes do not appear in this list. The first message indicates the writing flag is not set for the VOLDB entry; the second message indicates the writing flag is set for the VOLDB entry.

To list all such VOLDB entries for these volumes, run migdbcheck in verbose mode.

```
migdbcheck -Vv delta
-- 00000998 in VOLDB, but no FHDB reference to it
00000998|00000001|00000000|00000000|AD2|ad|/hsm1_6|0|3155C971
|00050000|000ECFAC|000009C0|0000000A|00000001|00000000|0|0||||||||
-- Writing Entry 00000999 in VOLDB, but no FHDB reference to it
00000999|00000001|00000000|00000040|AD3|ad|/hsm1_8|0|3155C971
|00050000|000ECFAC|000009C0|0000000A|00000001|00000000|0|0||||||||
```

These volumes are good candidates for consolidation and reuse (see migcons(1M)). Before consolidating the second volume, clear the write flag in its VOLDB flags field.

```
migsetdb -Va delta AD3
```

FILES

Note The term *dwpath* refers to the path name of the directory containing the database and workdir directories. VSM uses the database directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the migconfg global-configuration file.

dwpath/database/FHDB

File-handle database

dwpath/database/VOLDB

Volume database

dwpath/database/FHSEQF

File-handle sequence number file

dwpath/database/VOLSEQF

Volume ID sequence number file

dwpath/database/migconf

VSM configuration file for managed file systems

dwpath/workdir

VSM work directory

fspath/migration/data

VSM premigration directory (applicable only to some kernel-based implementations)

/usr/var/openv/hsm/database/migconfg

VSM global-configuration file

dwpath/database/*hsmname*.IHAND

VSM inode-to-handle file (nonkernel-based implementations only)

The following output files are created and used by migdbcheck to list FHDB inconsistencies discovered:

/tmp/migdbcheck-migratedfiles.*hsmname.pid*

A list of all migrated files in the file system(s)

/tmp/migdbcheck-premigfiles.*hsmname.pid*

A list of files that exist in the premigration directory (kernel-based implementations only)

/tmp/migdbcheck-dk-obsolete.*hsmname.pid*

A list of FHDB entries that are obsolete but the files exist in the premigration directory (kernel-based implementations only)

/tmp/migdbcheck-obsolete.*hsmname.pid*

A list of FHDB entries other than dk entries that are obsolete or dead but the files exist as migrated files in the file system

/tmp/migdbcheck-orphan.*hsmname.pid*

A list of migrated files that have an entry in the FHDB but the files do not exist

/tmp/migdbcheck-dk-orphan.*hsmname.pid*

A list of files that have an active dk entry in the FHDB but the files either do not exist or are not migrated (data not on disk)

`/tmp/migdbcheck-missing.`*hsmname.pid*

A list of migrated files that do not have enough entries in the FHDB

`/tmp/migdbcheck-dup-handles.`*hsmname.pid*

A list of migrated files with duplicate FHDB handles

`/tmp/migdbcheck-no-fhdb.`*hsmname.pid*

A list of migrated files that do not have any entries in the FHDB

`/tmp/migdbcheck-cached.`*hsmname.pid*

A list of cached files that do not have any entries in the FHDB because they were cached before any copies were made

`/tmp/migdbcheck-copies-needed.`*hsmname.pid*

A list of files that do not have the required number of migrated copies

`/tmp/migdbcheck-error-flag.`*hsmname.pid*

A list of files whose FHDB entry contains an error flag that is either *locked* or *failed*

`/tmp/migdbcheck-level-X.`*hsmname.pid* where 1<= X <= 8

A list of files that exist at migration level `X`, where `X` represents migration levels 1 through 8, and there are more valid copies found than defined in `migconf`

The following output files are created and used by `migdbcheck` to list VOLDB inconsistencies discovered:

`/tmp/migdbcheck-voldb-missing.`*hsmname.pid*

A list of volumes in the FHDB that are not in the VOLDB. Verbose mode lists FHDB entries residing on the missing volumes.

`/tmp/migdbcheck-voldb-consolidate.`*hsmname.pid*

A list of volumes in the VOLDB with no FHDB references. Verbose mode lists VOLDB entries for these volumes.

`/tmp/migdbcheck-voldb-writing.`*hsmname.pid*

A list of volumes in the VOLDB that are in write mode with no FHDB references. Verbose mode lists VOLDB entries for these volumes.

`migdbcheck` checks the environment variable *TMPDIR*, which allows the administrator to use a path other than the default `/tmp`. This can save files through system reboots or make use of a larger file system to avoid running out of space. The path defined by *TMPDIR*, if set, is used instead of `/tmp` as the directory in which to place any temporary files. The process id of the `migdbcheck` that is executing replaces pid.

Files which could indicate problems or which could be used to obsolete or activate FHDB or VOLDB entries are left in /tmp (or the path defined by *TMPDIR*). No output files should be left if `migdbcheck` executes successfully and finds no errors.

Note Some output files created by `migdbcheck` may be used as input files for `migsetdb`.

SEE ALSO

`migadscan(1M)`, `migbatch(1M)`, `migconfg(1M)`, `migcons(1M)`, `migdbrpt(1M)`, `migftscan(1M)`, `miglow(1M)`, `mignbscan(1M)`, `mignospace(1M)`, `migtscan(1M)`, `migopscan(1M)`, `migrc(1M)`, `migsetdb(1M)`

migdbdir(1M)

NAME

migdbdir – get VSM information from the global-configuration file

SYNOPSIS

`/usr/openv/hsm/bin/migdbdir` *hsmname id*

DESCRIPTION

`migdbdir` prints the desired value or device for the specified *hsmname* on standard output. The *id* parameter specifies the desired information.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

OPTIONS

hsmname HSMDEV entry about which you want information. See `migconfg(1M)` for more information on HSMDEV entries.

id Integer identifying the desired information. The accepted values are:

1 – Database path (*dwpath*)
2 – File system path (*fspath*)
3 – Log file path (*lgpath*)
4 – VSM state flag (state)
5 – File system device number
6 – Mounted status

EXAMPLES

The following example prints the device number of the VSM-managed alpha1 file system.

```
migdbdir alpha1 5
33554455
```

The following example prints whether or not the alpha3 file system is mounted correctly. In this case, it is not mounted correctly.

```
migdbdir alpha3 6
not mounted
```

FILES

`/usr/var/openv/hsm/database/migconfg`

VSM global-configuration file

SEE ALSO

migconfg(1M)

migdbrpt(1M)

NAME

migdbrpt – report on VSM migration database

SYNOPSIS

```
/usr/openv/hsm/bin/migdbrpt [-g] -f file_path
        hsmname

/usr/openv/hsm/bin/migdbrpt [-g] -h file_handle | -H
        hsmname

/usr/openv/hsm/bin/migdbrpt [-g] -l label | -a
        hsmname

/usr/openv/hsm/bin/migdbrpt [-g] -m method | -a
        hsmname

/usr/openv/hsm/bin/migdbrpt [-g] -s vol_set
        hsmname

/usr/openv/hsm/bin/migdbrpt [-g] -v vol_handle
        hsmname
```

DESCRIPTION

`migdbrpt` prints reports on database information maintained by VSM. Preserving periodic `migdbrpt` reports helps trace volumes for important files and make decisions on volume consolidation. The system administrator can issue `migdbrpt` at any time.

You can base enquiries about files on:

- The *file_path* (`-f` option)
- The *file_handle* (`-h` option) that VSM assigns to the file during the migration phase

The `-H` option displays information about all migrated files. The displayed information includes where the file is migrated, as well as the number of granules and the percentage of volume capacity for the file.

You can base inquiries about volumes on:

- The volume label (`-l` option)
- The method name (`-m` option)
- The volume set (`-s` option)
- The volume handle (`-v` option) that VSM assigns when the volume is registered with `migreg`

The –a option reports on all volumes. migdbrpt displays the percentage of the volume in use and the number of both live and obsolete granules on the volume. This information is useful in making decisions on consolidating volumes (see migcons(1M)). All numbers are displayed in decimal.

migdbrpt also displays database entries that have duplicate file handles or duplicate volume handles.

If a report indicates possible problems, use migdbcheck to verify the databases. Correct any database inconsistency immediately to guard against future catastrophe.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon migd is running.

OPTIONS

–g Prints granule-related information. You can use this parameter with any of the other options.

–f *file_path*
Reports on the file with the specified file path. The file path must be specified with the full path.

–h *file_handle*
Reports on the file with the specified file handle (specify the file handle in hexadecimal).

–H Reports on all migrated files.

–l *label* Reports on the volume with the specified label. You can use this option with the –m option to report on all volumes under a given method and label pair.

–m *method*
Reports on all volumes registered under the specified method.

–a Reports on all volumes registered.

–s *vol_set*
Reports on all volumes registered under the specified volume set. You can use this option with the –m option to report on all volumes under a given method and volume set pair.

–v *vol_handle*
Reports on the volume with the specified volume handle (specify the volume handle in hexadecimal).

hsmname HSMDEV entry that defines the path to the database files from which you want to derive the report. See migconfg(1M) for more information on HSMDEV entries.

EXAMPLE 1

The following example prints information about the `/home/pjt/f1` file. Information is displayed about what volumes the file resides on, as well as its granule count and what percentage of the volume is occupied by this file.

```
migdbrpt -g -f /home/pjt/f1 alpha
```

This results in a display similar to the following:

```
FILE_HAND  VOL_HAND METHOD FIL_SIZE OFFSET   GRN_SIZE F GRN# SEEKINFO FILE_PATH
0009C2EA   00000000     dk 0072C668 00000000 0072C668 L    0 00000000 /home/pjt/f1
0009C2EA * 00001004     dt 0072C668 00000000 00200000 L 4980 00000000 /home/pjt/f1
0009C2EA * 00001004     dt 0072C668 00200000 00200000 L 4980 00000041 /home/pjt/f1
0009C2EA * 00001004     dt 0072C668 00400000 00200000 L 4980 00000082 /home/pjt/f1
0009C2EA * 00001004     dt 0072C668 00600000 0012C668 L 4980 000000C3 /home/pjt/f1

VOL_HAND       LABEL   POOL  METHOD  CAPACITY   #LGNS  #OGNS   GRAN_USE  PERCENT
LOC
00000000 <=>   DK0000        dk.0 17483290624     1      0     7521896   0.04%
00001003 <=>C  VLS001  HSM   dt.1 19327352832     4      0     7521896   0.04%
00001004 <=>W  VLS001  HSM   dt.1 19327352832     4      0     7521896   0.04%
```

The items in this display have the following meanings:

`FILE_HAND`
: File handle for the file, followed by an asterisk (*) if locked.

`VOL_HAND`
: Handle assigned to the volume and referenced by entries in FHDB.

`<*>` : Indicates this volume is currently locked for use

`W` : Current volume to be written

`<=>` : Indicates the volume has been written (if not followed by a letter)

`C` : volume is corrupt- this volume will not be read or written

`D` : Indicates a dead volum

`E` : Indicates an empty volume

`f` : Do not write to this volume

`F` : Volume needs to be force labeled

`L` : Volume needs to be labeled

METHOD Method name. Valid values are ct, dt, mt, op, ow, ad, ft, nb, and dk. In some displays the volume set number is also shown.

FILE_SIZE
Size of the file (hexadecimal).

OFFSET Offset from the beginning of the file for each granule.

GRN_SIZE
Size of each granule in the file.

F File status: O for obsolete; L for active.

GRN# The number of the file on the media.

SEEKINFO
Seek information for the file.

FILE_PATH
File path name.

LABEL Label specified when this volume was registered.

POOL Name of the volume pool for this volume.

CAPACITY
Total space in bytes taken from the capacity parameter in the METHOD section of the migconf file for the file system.

#LGNS Total number of active granules on the volume. VSM derives this value from a scan of the FHDB.

#OGNS Total number of obsolete granules on the volume. VSM derives this value from a scan of the FHDB.

GRAN_USE
Space used on the volume, in bytes.

PERCENT Percentage of space used on the volume. Anything over 86% may be full. If drivers compress the data before writing it to a volume, however, PERCENT could exceed 100 and the volume may still not be full.

LOC Path to the registered directory. This applies to methods ad, and ft.

SERVER Server name.

USER User name (root).

EXAMPLE 2

The following example prints a report on all volumes registered under the configured methods.

```
migdbrpt -a alpha
```

This results in a display similar to the following:

```
VOL_HAND       LABEL   POOL  METHOD   CAPACITY    #LGNS  #OGNS   GRAN_USE  PERCENT LOC
00001000 <*>W  OPT01A   HSM   dt.1     375896384    572   13317   290711094  77.36%
00001009 <=>C  TST003   HSM   op.1     293601280   1140      11   256408241  87.33%
00000000 <=>   DK0000         dk.0   17483290624   6083       0   287560268   1.64%
00001001 <=>E  OPT01B   HSM   op.1     293601280      0       0           0   0.00%
```

EXAMPLE 3

The following example prints information about volume *rao001* under method `ct`. The display will include information about the migrated files, their granule count, and the volume occupancy.

```
migdbrpt -g -l rao001 -m ct  alpha
```

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath`/database/migconf`

VSM configuration file for managed file systems

dwpath`/database/FHDB`

File-handle database for migrated files

dwpath`/database/VOLDB`

Volume database

`/usr/var/openv/hsm/database/migconfg`

Global-configuration file for VSM

SEE ALSO

`migdbcheck(1M), migadscan(1M), migconf(1M), migconfg(1M), migcons(1M), migreg(1M), migtscan(1M), migopscan(1M)`

migfind(1M)

NAME

migfind – determine the full pathname of a file

SYNOPSIS

`/usr/openv/hsm/bin/migfind -h` *file_handle* `-m` *machid hsmname*

`/usr/openv/hsm/bin/migfind -d` *DMAPI_handle hsmname*

DESCRIPTION

`migfind` determines the full pathname of a file, given either its VSM file handle and machine ID or its DMAPI handle (nonkernal-based implementations only).

Output is displayed to standard output. Error messages go to standard error.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running. The managed filesystem must be mounted.

Note `migfind` command is not supported on the HP-UX 10.20 operating system.

OPTIONS

`-h` *file_handle*
: The VSM file handle of the file. File handles are stored in the file-handle database (FHDB) for the file system.

`-d` *DMAPI_handle*
: The DMAPI handle of the file (nonkernal-based implementations only).

`-m` *machid*
: The integer (nonzero) identifier for the VSM managed file system from which the file was migrated, as configured with the machid parameter in the DEFAULTS section of the *dwpath*`/migconf` file.

hsmname
: HSMDEV entry that specifies the file system in which the file resides. For more information on HSMDEV entries, see `migconfg(1M)`.

EXAMPLE

The following command determines the full path of a file in hsmname `hsm2` whose VSM file handle is `1f29` and machine id is `3e8`.

```
/usr/openv/hsm/bin/migfind -h 1f29 -m 3e8 hsm2
/hsm2/acg/acg1
```

Identical results are achieved by using the DMAPI handle of the file.

```
/usr/openv/hsm/bin/migfind -d \
00000000008000a900000006000000000000000000001a0000010070c00080 hsm2
    /hsm2/acg/acg1
```

The following output shows the full path of a file in hsmname `hsm2` whose VSM file handle is `25c1` and machine id is `3e8`, but reports that no FHDB entry exists for this file.

```
/usr/openv/hsm/bin/migfind -h 25c1 -m 3e8 hsm2
INFO: No FHDB entries exist
/hsm2/acg/acg33
```

The following output reports that neither the FHDB entry nor the file exist for VSM file handle `25c1` in hsmname `hsm2` and machine id `3e9`.

```
/usr/openv/hsm/bin/migfind -h 25c1 -m 3e9 hsm2
INFO: No FHDB entries exist
No such file
```

The following output reports that the file with the specified DMAPI handle does not exist in hsmname `hsm2`.

```
/usr/openv/hsm/bin/migfind -d \
00000000008000a900000003000000000000000000001a0000010070c00090 hsm2
No such file
```

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath`/database/FHDB`

File handle database for VSM.

SEE ALSO

`HSM(1M)`

migftscan(1M)

NAME

migftscan – scan ft volume and reconstruct database entries for the `ft` storage method.

SYNOPSIS

```
/usr/openv/hsm/bin/migftscan [-F] [-n] [-s] hsmname
        label [server_name server_path [user]]
```

DESCRIPTION

`migftscan` scans a remote file system and displays information about the volume as a whole as well as information about each file granule on the volume. It also reconstructs the FHDB and VOLDB entries for all scanned granules.

The remote disk volumes are registered by using the `migreg` command before they can be used as archive media.

A file is migrated to a remote file system as a single granule. The granule header is separately archived in the remote file system as a file. The granule header contains FHDB and VOLDB entry information.

The `migftscan` command creates two output files, FHDB.*label* and VOLDB.*label*, in the *dwpath*`/database` directory. The structure of these files is the same as the FHDB and VOLDB database files. These files may be used to rebuild the FHDB and VOLDB if they are corrupted or damaged (see `migdbcheck(1M)`).

Sorting and merging of `FHDB.`*label* files for different remote file systems can be used to recreate the FHDB. Similarly, merging multiple `VOLDB.`*label* files for different remote file systems can be used to recreate the `VOLDB` database.

Note When recreating the VOLDB be sure to merge the VOLDB file in the `/usr/var/openv/hsm/example/database` directory to include the entry for the `dk` method.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

OPTIONS

The following parameters are available under `migftscan`:

`-F` Force scan the volume for granules in case the volume identity is not in the volume database (VOLDB). This parameter is optional. If omitted, the volume must be registered in the volume database. If specified, the optional parameters *server_name server_path user* must also be supplied. A password prompt is issued.

`-n` Do not convert FHDB entries for granules written prior to R2.0. You cannot reconstruct a usable database with this option, but it is useful if you want to examine what is actually written on the media.

`-s` Silently scan the volume and create FHDB.*label* and VOLDB.*label* files, but do not display information on stdout.

hsmname HSMDEV entry that specifies the path to the database containing information on the desired volume. For more information on HSMDEV entries, see `migconfg(1M)`.

label Remote-file-system volume label used when registering the volume. This parameter is required. For a force scan, you may specify *label* as anything.

server_name
Name of the remote volume server. This can be the internet id or number. VSM uses this name on the `ftp open` command as the host parameter. It must be a valid `ftp` host. This parameter is required when you specify `-F`.

server_path
Path of the file system on the remote volume server. This parameter is required when you specify `-F`.

user User name used for the `ftp login` request when VSM migrates or caches files from the remote volume server. If you do not specify user, then root is assumed. This parameter is required when you specify `-F`.

CAVEATS

- Only the system administrator may use this command.
- The remote volume server file system must be available via `ftp` for this command.
- `migftscan` performance depends on the number of migrated files and the performance of `ftp` to the server.

EXAMPLE

The following example scans the archive disk volume FT005 that is registered under the `ad` method. The display shows a list of granule information for files archived onto the volume. The FHDB.FT005 and VOLDB.FT005 files are created in the *dwpath*`/database` directory.

```
migftscan openv2 FT0005
VOLUME FT0005 registered to HSM
   Volume Particulars
   ------------------
000003E8V000010AF FT0005     ft 00000B71 00000883 #31
```


```
   Volume label Found ====> hare:FT0005
---------------- /usr/home/agilbert/ft_storage_1/3E8M122D.1.0.GLABEL <=>
0000000000100000000 Thu Apr 21 13:09:40 1994 /sd7/ov2fs/b1f2
/usr/home/agilbert/ft_storage_1/3E8M1231.1.0.GLABEL <=> 00001036
00000000 Thu Apr 21 13:09:41 1994 /sd7/ov2fs/f1.c

/usr/home/agilbert/ft_storage_1/3E8M1235.1.0.GLABEL <=> 00076000
00000000 Thu Apr 21 13:09:42 1994 /sd7/ov2fs/f1.x

/usr/home/agilbert/ft_storage_1/3E8M1239.1.0.GLABEL <=> 00000F3B
00000000 Thu Apr 21 13:09:44 1994 /sd7/ov2fs/f2.sh

/usr/home/agilbert/ft_storage_1/3E8M123D.1.0.GLABEL <=> 0000188E
00000000 Thu Apr 21 13:09:45 1994 /sd7/ov2fs/f3.rcs

/usr/home/agilbert/ft_storage_1/3E8M1241.1.0.GLABEL <=> 0000565C
00000000 Thu Apr 21 13:09:47 1994 /sd7/ov2fs/f4.o

/usr/home/agilbert/ft_storage_1/3E8M1245.1.0.GLABEL <=> 00001036
00000000 Thu Apr 21 13:09:48 1994 /sd7/ov2fs/f5.c
.
.
.
/usr/home/agilbert/ft_storage_1/3E8M126D.1.0.GLABEL <=> 00001FFF
00000000 Thu Apr 21 13:10:03 1994 /sd7/ov2fs/k8-1f5

/usr/home/agilbert/ft_storage_1/3E8M1276.1.0.GLABEL <=> 00002000
00000000 Thu Apr 21 13:10:07 1994 /sd7/ov2fs/k8f8

/usr/home/agilbert/ft_storage_1/3E8M127A.1.0.GLABEL <=> 00001000
00000000 Thu Apr 21 13:10:08 1994 /sd7/ov2fs/kef4

/usr/home/agilbert/ft_storage_1/3E8M1283.1.0.GLABEL <=> 00004592
00000000 Thu Apr 21 13:10:12 1994 /sd7/ov2fs/nrof.3

/usr/home/agilbert/ft_storage_1/3E8M1287.1.0.GLABEL <=> 00002000
00000000 Thu Apr 21 13:12:52 1994 /sd7/ov2fs/k8f2
```

Volume particulars displayed include (in order): volume handle, volume label, method, total capacity of the volume, total space in use on the volume and number of granules on the volume.

Particulars of granules displayed include (in order): granule file name, migrated file size, offset of the granule in the migrated file, date of archiving the granule and migrated file pathname.

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath/database/FHDB

File handle database for VSM.

dwpath/database/VOLDB

Volume database for VSM.

dwpath/database/migconf

Configuration file for VSM.

dwpath/database/FHDB.*label*

File handle database for current volume.

dwpath/database/VOLDB.*label*

Volume database for current volume.

server_name:/*server_path*

File system on the server.

SEE ALSO

migdbcheck(1M), migconf(1M), migconfg(1M), migdbrpt(1M), migreg(1M), migadscan(1M), mignbscan(1M), migtscan(1M), migopscan(1M)

miggetvol(1M)

NAME

miggetvol – print a list of VSM volumes on stdout by method name in ascending order of percentage utilization

SYNOPSIS

`/usr/openv/hsm/bin/miggetvol` *hsmname* [*method*]

DESCRIPTION

`miggetvol` prints a list of volumes registered under the specified method. This list is sorted by method name in ascending order according to the percentage of space used on the volumes. The total size of files currently migrated to the volume determines space used on the volume.

The entries in the list include:

`label`

`method`

`empty`

`gran_count`

`obsolete_gran`

`percent_use%`

`volume_set`

`date_last_written`

You can use `miggetvol` to monitor volume usage, such as required in determining when to consolidate volumes. The `migselect` command uses `miggetvol` for selection of volumes to consolidate.

You must have superuser privileges to use `miggetvol`.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

OPTIONS

hsmname HSMDEV entry that specifies the path to the database files from which you want to derive the report. For more information on HSMDEV entries, see `migconfg(1M)`.

method Method name under which `miggetvol` is to return information. By default, volumes under all methods are returned.

CAVEATS

- Inconsistency between VOLDB and FHDB entries cause a wrong *percent_use* for the volume.
- `miggetvol` does not consider granule space overhead on the volume in arriving at volume occupancy.

EXAMPLE

The following command prints a list of tape volumes in ascending order of percentage utilization under the `ct` method.

```
miggetvol  alpha ct
```

Sample output:

```
C00000      ct E      0   0   0.00 %  1 Thu May 16 14:35:11 1991
C00003      ct -     80   5   7.81 %  1 Thu May 16 17:59:28 1991
C00002      ct -      2   0 25.47 %  2 Thu May 16 17:00:35 1991
C00001      ct -      7   2 79.86 %  2 Thu May 16 17:11:31 1991
```

where `E` indicates empty, `W` indicates writing, `D` indicates dead, and `-` is a place-holder indicating none of the above. See DESCRIPTION for a listing of the fields shown in the usage report.

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath`/database/VOLDB`

Volume database for VSM

dwpath`/database/FHDB`

File-handle database for VSM

`/usr/var/openv/hsm/database/migconfg`

Global-configuration file for VSM

SEE ALSO

`migconfg(1M), migcons(1M), migdbrpt(1M), migselect(1M)`

miggroup(1), migungroup(1)

NAME

miggroup - group files for migration and caching

SYNOPSIS

`/usr/openv/hsm/bin/miggroup [-d] [-M] [-P]` *dir*

`/usr/openv/hsm/bin/miggroup -l [-v]` *dir*

`/usr/openv/hsm/bin/miggroup -u` *dir*

`/usr/openv/hsm/bin/migungroup` *dir*

AVAILABILITY

This command is only available on the DMAPI implementation of VSM.

DESCRIPTION

`miggroup` allows users to premigrate and group all of the files in a directory and all of its subdirectories in VSM-managed file systems. Users with root privilege can group directories owned by any user. Other users can group their own regular directories if allowed to use this command.

For nonroot users, the `miggroup` command performs only the premigration phase of migration. Optionally, users can also purge copied files. The files in a grouped directory do not actually migrate to secondary storage until the next time `mignospace`, `migbatch`, `migrc -R`, or `miglow` execute. Users with root privilege can also optionally copy and purge premigrated files in a grouped directory.

Grouping causes files to be added to a tie group, `MigGroup`, as key caching files so they will be cached together when any one file in the group is cached. Files listed in global and local stop files are not included in the group.

Use the `-l` option to determine if a directory has been grouped, and to see the status of the files in that directory. Adding the `-v` option gives the status of each file in the directory. If many files in the directory are already migrated, they may reside on multiple volumes. This means the directory is fragmented. The `-d` option will defragment the directory by caching migrated files before grouping the directory.

After a grouped directory is migrated to secondary storage, accessing any file in the group causes VSM to transparently cache all of the files in the group back to disk. Defragmenting a directory will cache and remigrate previously migrated data to a minimal number of tapes, which reduces the mount time whenever the directory is cached.

Directory grouping is *not* dynamic. If files are created in a grouped directory, removed from the directory, or moved in or out of the directory, the grouping remains the same. It is good practice, therefore, to group active directories periodically.

Normal file selection criteria are not applicable to `miggroup`, but normal VSM rules apply to files that have not been grouped in grouped directories, and these files continue to be selected, migrated, and cached individually.

Users can ungroup a grouped directory using either the `migungroup` command or `miggroup` with the `-u` option.

OPTIONS

`-d`	Defragment the directory. If specified, all migrated files in the directory are cached and their FHDB entries are marked obsolete before the directory is grouped. If not specified (default), all migrated files in the directory remain migrated, and the directory is grouped. In either case, files not previously grouped are included in the grouped directory.
`-M`	Run `migbatch` to make copies of the grouped premigrated files. Available only to users with root privilege; generates an error message if used by nonroot users. If the directory is not yet grouped, using `miggroup -M` or `migungroup -M` groups the directory.
`-P`	Purge copied files in the directory. If the directory is not yet grouped, using `miggroup -P` or `migungroup -P` groups the directory.
`-l`	List the status of a directory without grouping the directory.
`-v`	Generate a verbose list showing the status of a directory. Must be used with the `-l` option.
`-u`	Ungroup a previously grouped directory. Previously grouped migrated files are cached individually if accessed.
dir	Path of the directory to group (relative or absolute).

CAVEATS

- By their very nature, grouped directories can increase migration and caching activity unnecessarily if used in situations where files are used individually. Group and migrate directories only where applicable.
- Files created in or moved to a grouped directory are not added to the group until the grouped directory is grouped again. Until they are added to the group, the grouped behavior of these files when cached is undefined.
- Files moved out of a grouped directory remain in the directory group. The behavior of these files when cached is undefined.

- Unless specifying the `-l` option, the directory is grouped again each time `miggroup` is executed.
- The tie group name `MigGroup` is reserved for `miggroup`. Using this name for other uses may cause unpredictable behavior. Also using `migtie` and `miggroup` on the same directory and files will cause unpredictable results.
- Making changes to a directory while `miggroup` is processing the directory may produce unpredictable behavior. Do not access migrated files, add files, or remove files in the directory being processed by `miggroup`.
- Grouping and defragmenting a directory does not guarantee that all grouped files will migrate together. Other migration activity in the same file system may cause other files to get intermingled with the grouped files on the same media.
- Running two `miggroup` commands on the same directory at the same time may produce undefined results.
- Moving grouped directories or files will cause unpredictable behavior. It is best not to group directories that will be subject to moving.
- Defragmenting a directory requires disk space to cache migrated files from secondary storage. If this space is not available the defragmenting operation will fail. If the operation fails, create space on disk and try again. A partially completed defragmentation operation may leave unnecessary data from migrated files on disk.
- If a migrated and purged file in a grouped directory is truncated to size zero, the migration daemon `migd` does not cache the file, nor does it cache any of the other files in the grouped directory.
- Files whose pathname length exceeds 1023 characters will not be migrated.
- The VSM state must be active and the migration daemon `migd` must be running when you issue a `miggroup` command.

EXAMPLES

In the following example, the user requests that VSM group and premigrate all of the files in directory `project123` without first defragmenting that directory.

```
miggroup /home/abc/project123
```

In the following example, the user defragments directory `project123` before grouping and premigrating all of the files in that directory.

```
miggroup -d /home/abc/project123
```

If the user has root privilege, the following command defragments directory `project123` before grouping, premigrating, copying, and purging all of the files in that directory. (Option `-M` does not apply to nonroot users.)

```
miggroup -d -MP /home/abc/project123
```

The following command gives the status of directory `project123` (specifying the absolute pathname) by listing the number of volumes holding data for the grouped files in that directory.

```
miggroup -l /home/abc/project123
Regular                    3 files       0.046080 Mbyte
Grouped                    9 files       0.101412 Mbyte
Not Grouped                3 files       0.046080 Mbyte

 Migrated/Grouped files are on  1 volume.
```

The following command gives the status of directory `project123` (specifying the relative pathname) by listing the number of volumes holding data for the grouped files in that directory.

```
miggroup -l project123
Regular                    3 files       0.046080 Mbyte
Grouped                    9 files       0.101412 Mbyte
Not Grouped                3 files       0.046080 Mbyte

 Migrated/Grouped files are on  1 volume.
```

The following command gives a more detailed status of directory `project123` by listing the number of volumes holding data for the grouped files in that directory on a file-by-file basis. Grouped files in subdirectories are also shown.

```
miggroup -lv /home/abc/project123
Regular                    3 files       0.046080 Mbyte
Grouped                    9 files       0.101412 Mbyte
Not Grouped                3 files       0.046080 Mbyte

 Migrated/Grouped files are on  1 volume.

 Not Yet Copied         3 files       0.046080 Mbyte
 State          Size  File
  m--          15360  dir1/f4
  m--          15360  d1/d1/d3/d4/d5/d6/f4
  m--          15360  f4
```


```
Volume  CT0001         9 files       0.101412 Mbyte
State          Size  File
 mg-          11268  f1
 mgt          11268  f2
 mg-          11268  f3
 mg-          11268  dir1/f1
 mgt          11268  dir1/f2
 mg-          11268  dir1/f3
 mg-          11268  d1/d1/d3/d4/d5/d6/f1
 mgt          11268  d1/d1/d3/d4/d5/d6/f2
 mg-          11268  d1/d1/d3/d4/d5/d6/f3
```

Either of the following two commands will ungroup directory `project123`.

`miggroup -u /home/abc/project123`

`migungroup /home/abc/project123`

FILES

`/usr/var/openv/hsm/database/migstop`

Global stop file for VSM.

SEE ALSO

`HSM(1M), migbatch(1M), migpurge(1), migrate(1), migsetdb(1M), migstage(1), migtie(1)`

migin(1M)

NAME

migin - caches a file from secondary storage to disk

SYNOPSIS

`/usr/openv/hsm/bin/migin` *hsmname file_system machid file_handle*

DESCRIPTION

The `migin` command allows the administrator to cache a file in cases where the file has been removed or lost and therefore cannot be cached simply by accessing it. This is useful in recovering files that were migrated or cached and then perhaps accidentally deleted. In kernel-based implementations the file is cached to premigration. In nonkernel-based implementations the file is cached to the original location as a regular file.

It is possible to recover the file only if the FHDB (File Handle Database) has not been cleaned up. VSM automatically removes obsolete entries from the FHDB after 7 days.

Note This process does not restore original file ownership or mode bits.

This command may be run without regard to whether the migration daemon `migd` is running, but the hsm state must be active.

OPTIONS

hsmname Name of the hsm that is controlling migrations for the file system in which the file you are recovering resided, as configured with the *hsmname* parameter in the `/usr/var/openv/hsm/database/migconfg` file.

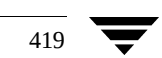
file_system Path to the file system that contained the file, as configured with the *fspath* parameter in the `/usr/var/openv/hsm/database/migconfg` file.

machid The integer (nonzero) identifier for the VSM managed file system from which the file was migrated, as configured with the machid parameter in the DEFAULTS section of the *dwpath*`/migconf` file.

file_handle File handle of the file you are recovering, as recorded in the FHDB.

CAVEATS

- If a file has been designated as a key caching file by the `migtie` command, caching that file manually with `migin` will *not* cause all migrated files in the associated group to be cached as well.



- If a file has been included in a grouped directory by the `miggroup` command, caching that file manually with `migin` will *not* cause all migrated files in the grouped directory to be cached as well.
- If both a file and its directory have been removed, `migin` will not cache the file until you have made a new directory with its full path to replace the original directory.
- Manual `migin` is not intended to be used on existing files. If `migin` is used on an existing purged file, the command restores the data and leaves the file in a purged state.

EXAMPLE

Assume you run `migdbcheck` and find that a file handle exists for a file that is not in the file system. Upon further checking you discover that you need this file. The file resided in the `/home1` file system, was migrated by `hsm1`, the machine ID for the host is `3e8`, and the file-handle is `12d2`. You execute the following:

```
migin hsm1 /home1 3e8 12d2
```

For kernel-based implementations, `migin` restores the file to the premigration directory for `/home1` in this example. The complete file path is:

```
/home1/migration/data/3e8M12d2
```

Then move the file manually to the correct location, and set the desired file permission and ownership.

For nonkernel-based implementations, `migin` restores the file to its original pathname as a regular, unmigrated file. Set the desired file permission and ownership.

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

`/usr/var/openv/hsm/database/migconfg`

Global-configuration file for VSM

dwpath`/database/migconf`

VSM configuration file for managed file systems

SEE ALSO

`migdbcheck(1M), migconf(1M), migconfg(1M), migreconstruct(1M), migpurge(1), migtie(1)`

miglicense(1M)

NAME

miglicense - displays the current license capacity of VSM

SYNOPSIS

```
/usr/openv/hsm/bin/miglicense

/usr/openv/hsm/bin/miglicense -l [-v]

/usr/openv/hsm/bin/miglicense -u [-f]

/usr/openv/hsm/bin/miglicense -a | -d textofkey
```

DESCRIPTION

The `miglicense` command lets the administrator determine the current license capacity of VSM. If used without options, `miglicense` displays the total valid capacity under license. The `-l` option displays an itemized list of each VSM license key including both valid and invalid keys.

You can compare the licensed capacity with the actual current capacity used on storage media by using the `-u` option. VSM issues a warning message when current capacity exceeds 90% of licensed capacity, and an error message when current capacity exceeds licensed capacity. When current capacity exceeds licensed capacity, VSM suppresses further migrations with `migcopy` until additional license keys are added.

Licensing is enforced when `migcopy` migrates data using the following storage methods: optical (`op` and `ow`), tape (`ct`, `dt`, and `mt`), and NetBackup (`nb`). Capacity includes active and obsolete granules, but not any granules marked dead.

VSM calculates current capacity and increments the total as new migrations occur, recording the sum in `/usr/var/openv/hsm/database/CAPACITY_USED`. When VSM periodically recalculates current capacity, only one copy of each migrated file is counted, regardless of how many copies exist and the levels on which they are located.

Note VERITAS recommends the practice of making two copies of each migrated file.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

OPTIONS

`-l [-v]`

Lists the text value and capacity of each VSM license key. The `-verbose` option is for debugging or customer support, and also displays NetBackup keys.

`-u`	Lists the previously calculated, current capacity used on storage media.
`-f`	Recalculate the current capacity used on storage media. Must be used with the `-u` option.
`-a`	Add a license key of format *textofkey*.
`-d`	Delete a license key of format *textofkey*.
textofkey	The text of the license key in the format *AAAA-BBBB-CCCC-DDDD-EEEE-FFFF*.

CAVEATS

- Whenever current capacity exceeds licensed capacity, VSM suppresses further migrations with `migcopy`. This continues until licensed capacity is increased. During this period, migrated files can continue to be cached if there is enough free space on disk to hold them.

FILES

`/usr/var/openv/hsm/database/CAPACITY_USED1`

`/usr/var/openv/hsm/database/CAPACITY_USED2`

The current capacity used

SEE ALSO

`HSM(1M)`

migloc(1)

NAME

migloc – display the location of a migrated file

SYNOPSIS

```
/usr/openv/hsm/bin/migloc [-R] pathname [pathname ...]
/usr/openv/hsm/bin/migloc -f filelist
```

DESCRIPTION

The `migloc` command shows where migrated files are located.

Note `fls` with the `–l` option displays `m` as the first character in the mode bit field of a file that has been migrated.

The resulting `migloc` display has the following format:

```
Status: Migrated  mach_name  owner  file_pathname
Handle: mach_idMhandle  file_size  gran_count  date_of_migration
Medium: label  method  level  gran_count  seek_info  vol_location
```

The meaning of each field is as follows:

- Status fields (if the file is not migrated, VSM generates only this field):

  Migrated

  Indicates whether the file is Regular (unmigrated), Migrated, Staging (currently being staged), Cached, Partial (partially cached), or a Directory.

  mach_name

  Host name of the machine on which the file resides.

  owner

  Owner of the file.

  file_pathname

  Pathname of the file.

- Handle fields:

  mach_idMhandle

  Machine ID and file handle in HEX separated by M.

  file_size

File size in bytes.

gran_count

Total number of active granules of this file on all media.

date_of_migration

Date when the file was last migrated.

- Medium fields:

label

Volume label.

method

Volume method name and volume set number.

level

Volume migration level.

gran_count

Number of granules (of the file) on this media.

seek_info

Information used by VSM to get to the file on the media. The information printed depends on the volume method. For `dk` and `ad` it is the name of the corresponding file. For `ct`, `dt` and `mt` it consists of two numbers separated by a slash(/). The first number is the file number of the file on the volume relative to the beginning. The second number is the granule number of the first granule on the volume.

vol_location

Path to file system in which the volume resides. For the ft remote method, it indicates the location of both the server and file system in the format:

The method name indicates the type of media as follows:

`ct`	Tape - as defined in `migconf`
`dt`	Tape - as defined in `migconf`
`mt`	Tape - as defined in `migconf`
`op`	Optical disk - as defined in `migconf`
`ow`	Optical disk - as defined in `migconf`
`ad`	Alternate magnetic disk
`dk`	Disk file (required for premigration)

`ft` Remote `ftp` method

`nb` NetBackup

After migration, the file data may exist both on disk and secondary storage until `mignospace` releases the space from disk. If the data still exists on disk, the display shows media type `dk`.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

OPTIONS

`-R` Recursively list subdirectories encountered.

pathname Files or directories that `migloc` should locate. If a directory is specified, all files in the directory and its subdirectories are located. May be given as a relative path or absolute path. You can specify multiple files or directories, or use standard regular expressions. Wildcards are recognized.

`-f` *filelist* Locate the files listed in *filelist*. The format of *filelist* is one file name per line, showing either the full pathname of each file or a pathname relative to the current directory in which `migloc` is executed, not the directory in which the file in the *filelist* resides. Wildcards are not recognized.

EXAMPLE 1

In this example, the `tproga` file resides on cartridge tape and the data has not yet been released from disk.

```
migloc tproga


Status: Migrated          aspen   root              tproga
Handle: 3E8M10E0            133   49                Fri Jan 28 12:48:36 1994
Medium: DK0000     dk.0       0   1    3e8M10e0     /sd7/ov2fs/migration/data
Medium: EXB007     ct.1       1   48   60/3055
```

EXAMPLE 2

In this example, the `acg1` file resides on a remote `ft` volume and has not yet been released from local disk.

```
migloc acg1


Status: Migrated          star   root              acg1
Handle: 3E8M140A             6   2                 Tue Apr 12 08:09:52 1994
```


```
Medium: DK0000     dk.0     0   1   3e8M140A      /sdisk3/migration/data
Medium: FT0001     ft.1     1   1   3e8M140A.1.0  star:/sdisk4/ft_storage
```

EXAMPLE 3

In this example, the `rpt1` file has been cached before all of the copies have been made.

```
migloc rpt1

Status: Cached            moon    root            rpt1
No matching entries found in FHDB: 3E8M160A
Problems getting migration data on file.
```

SEE ALSO

`fls(1)`

miglow(1M)

NAME

miglow – check the VSM-managed file systems for low space

SYNOPSIS

`/usr/openv/hsm/bin/miglow` [*hsmname*] | [*file_system*]

DESCRIPTION

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

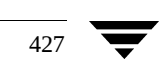
The `miglow` command allows a site to maintain and manage disk space on a VSM-managed file system by initiating migration when free space is below the `freespace` percentage specified in the *dwpath*`/database/migconf` file.

You can execute this command from the system prompt or call it periodically from a `crontab` entry. Assuming free space for the designated *hsmname* or file_system is below the `freespace` percentage, `miglow` premigrates files to the low-water mark (`lowwater` percentage), then copies them to secondary storage and purges the premigrated files from disk until free space increases to the purge mark (`purgemark` percentage). Messages are displayed about the current and new disk space use on the VSM-managed file systems.

`miglow` can be useful if you want to directly manage file system space rather than use the automatic disk-space-management capability that `migd` provides. For example, if you turn off `migd` or set the `state` parameter to `0` (off) for a specific HSMDEV entry in `migconfg`, automatic migration does not occur. This also disables caching so VSM does not cache a file when a user or program accesses it. In this case, you can use `miglow` to periodically migrate files when free space is below the `freespace` percentage specified in the *dwpath*`/database/migconf` file.

For kernel-based implementations of VSM, another approach is to disable automatic migration but not caching by using `migthreshold` to set the automatic migration threshold to 0 or some low percentage. This effectively disables automatic migration but caching still occurs. `miglow` continues to initiate migration when free space is below the `freespace` percentage specified in the *dwpath*`/database/migconf` file, not any other threshold values reset by `migthreshold`.

This command may be run without regard to whether the migration daemon `migd` is running, but the hsm state must be active.



OPTIONS

hsmname HSMDEV entry that specifies the file system you want to check. For more information on HSMDEV entries, see `migconfg(1M)`.

file_system Full path name of the file system you want to check.

If you do not specify an *hsmname* or *file_system*, `miglow` checks all VSM-managed file systems.

SEE ALSO

`HSM(1M), migconfg(1M), migbatch(1M), mignospace(1M), migthreshold(1M)`

migmdclean(1M)

NAME

migmdclean – clean VSM migration media

SYNOPSIS

```
/usr/openv/hsm/bin/migmdclean [-O] [-i] [-a days]
          [-R] hsmname label.method [.volset]
          [label.method [.volset] ] . . .
```

DESCRIPTION

When a user modifies or deletes migrated files, VSM does not remove the migrated files and granules from the medium, but simply marks them as obsolete in the file-handle database (FHDB). VSM is unable to reuse the space on the media until all FHDB entries for the volume are obsoleted and set to dead.

`migmdclean` allows you (the system administrator) to clean the media and databases by setting obsolete FHDB entries to dead and then removing all dead FHDB entries. When all FHDB entries for a volume are dead, the `-R` option can remove the volume's entry from the VSM volume database (VOLDB). You can then recycle the media by registering and relabeling it for use with VSM. By using the `migselect` command, you can select multiple media under different methods.

Normally, you use the `migrc` command at startup to remove obsolete and dead entries (see `migrc(1M)`). You can also use the `migcons` command to consolidate volumes and free them for reuse. However, `migmdclean` has additional options that may be useful under certain conditions. For example, if the medium is damaged you can use the `-O` and `-R` options to forcibly obsolete database entries and remove the empty volume from VSM.

For the `ad` and `ft` methods, `migmdclean` also attempts to remove files from the media when the `MFLAG_OBSOLETE` flag is specified in the *dwpath*`/database/migconf` configuration file for these methods.

For the `ct`, `dt`, `mt`, `op`, and `ow` methods, `migmdclean` never attempts to remove files from the media. If all granules on the volume are already dead, use `migrecycle` to reregister the media and remove files. If the volume contains obsolete granules, use `migcons` and then reregister the media with `migreg` to remove files. Although it is possible to clean `ow` media (write once, read many), it is not possible to recycle these optical disks.

Note When one side of an optical disk is consolidated, the volume entry for that input volume is not removed from the VOLDB unless and until both sides of that optical disk are empty.

For the dk method, migmdclean sets obsolete FHDB entries to dead and removes them only if the MFLAG_OBSOLETE flag is specified in the *dwpath*/database/migconf configuration file for this method.

Note Setting MFLAG_OBSOLETE causes dk entries to stay in the FHDB over cleanings initiated by migrc. This allows a cached file to be returned to premigration and processed by normal purge operations. To remove these dk entries and reduce the size of the FHDB, run migmdclean.

Only the system administrator may use migmdclean.

Note Although VSM sets FHDB entries to obsolete if a migrated file is modified or deleted, VSM must set these entries to dead before actually removing them from the FHDB. With the ad and ft methods, migmdclean may also unlink the file on the media so it is no longer possible to recover the data using VSM utilities. Prior to unlinking, you could recover the file by simply setting the corresponding FHDB entry back to active. In the following discussions, removal refers to unlinking and applies only to media used by the ad and ft methods.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon migd is running.

OPTIONS

By default, migmdclean marks all obsolete FHDB entries as dead regardless of method or volume. For ad and ft volumes, the command also attempts to remove obsolete files from the media if MFLAG_OBSOLETE is set. For nb volumes, the command will expire the NetBackup image when the last FHDB entry to reference it is deleted.

–O
: Forces valid FHDB entries to obsolete and then to dead.

Caution Use the –O parameter with extreme caution, as it might prevent user access to migrated files residing on the volume. See CAVEATS.

: For ad and ft volumes, this option also attempts to remove the files from the media. If this option is not specified, migmdclean removes only currently obsolete entries.

–i
: Marks the FHDB entries as dead but inhibits removal of files from the media. If this option is not specified, VSM attempts to remove files. This option applies to the ad and ft methods.

–a *days*
: Selects only files that have been obsoleted for the specified number of days. By default, VSM considers all obsoleted entries. An important use of the age parameter [–a *days*] is to specify that VSM delete only those files obsoleted prior to the last full backup.

`-R` Specifies that if all FHDB entries for the volume are dead, `migmdclean` removes the volume entry from the VOLDB. If this option is not specified, the volume remains in the VOLDB as a valid entry.

hsmname Specifies the HSMDEV entry that defines the path to the database files for the volumes that you want `migmdclean` to process. See `migconfg(1M)` for more information on HSMDEV entries.

label.method.volset
Selects the media id that has a matching label in the volume set for the method. *volset* is optional. You can supply a list for this parameter.

CAVEATS

- Locked database entries for the media fail the cleanup operation. Run `migrc -L` prior to running `migmdclean` to unlock the locked entries.
- Execute `migdbcheck` before running `migmdclean` to ensure database consistency.
- Side effects of running `migmdclean -O -R` after all copies have been made:

  If there is only one migrated copy and the file is purged from premigration, the file cannot be cached and the data is lost.

  If another active or obsolete copy exists and the file is purged from premigration, the file can still be cached.

  Unpurged files will not be purged and can still be cached.

  Copies of unpurged files will not be remade.

- If using NetBackup in conjunction with VSM, set the *age* variable for `migmdclean` at a value higher than the longest NetBackup retention period on the managed filesystem. Do this by adding the following line to `/usr/openv/hsm/bin/migmdclean`:

  `FLGS="-a nnn $FLGS"`

  before the line:

  `HSMNAME=$1`

  where *nnn* is the longest Netbackup retention period.

EXAMPLE

The following example removes obsolete FHDB entries at least 7 days old from media in cartridge disk `rao001` volume set 1.

```
migmdclean  -a 7  alpha  rao001.ct.1
```


The following example selects media in alternate magnetic disk (ad) volume set 1 that have no active files and removes the granules from them. When removal is successful, the corresponding FHDB entries are marked as dead.

```
migmdclean  alpha  `migselect alpha 0-0 1 ad`
```

FILES

Note The term *dwpath* refers to the path name of the directory containing the database and workdir directories. VSM uses the database directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the migconfg global-configuration file.

dwpath/database/FHDB

File-handle database

dwpath/database/VOLDB

Volume database

dwpath/database/migconf

VSM configuration file for managed file systems

/usr/var/openv/hsm/database/migconfg

Global-configuration file for VSM

SEE ALSO

migdbcheck(1M), migreg(1M), migconf(1M), migconfg(1M), migcons(1M), migdbrpt(1M), migrc(1M), migrecycle(1M), migselect(1M)

migmode(1)

NAME

migmode – control VSM operating modes

SYNOPSIS

```
/usr/openv/hsm/bin/migmode [-v] [-n 0 | 1]
```

AVAILABILITY

This command is only available on kernel-based implementations.

DESCRIPTION

`migmode` allows users to control whether caching of migrated files is automatic and whether they wait when writing to a file system that is out of space. They can execute the command directly or include it in their login scripts. On Solaris `ufs`, child processes do not inherit the values of the mode from the parent.

Note Option `-n` does not apply to Solaris releases before 2.6.

OPTIONS

`-v` Specifies verbose mode. In this mode, the parameters are displayed before and after setting. The default is to not display the parameters.

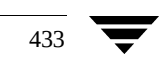
`-n 0 | 1`

(This option is available only for Solaris platforms.) Allows the superuser to control the behavior of the NFS server processes as seen from NFS clients. The two options are as follows (default is `-n 0`):

`-n 0`: NFS client write operation does not wait for space when the file system is full. Instead, the write operation fails with the error ENOSPC (no space left on device).

`-n 1`: NFS client write operation waits until additional space is available. If the migration daemon is active and the state parameter is set to 1 (Active) in the HSMDEV entry for the file system, `mignospace` makes space available and the write operation continues (see `mignospace(1M)`).

For this to work transparently, configure the client mount options so the client request does not time out before space becomes available. The client will see the usual messages during this period,"NFS server not responding, still trying."



SEE ALSO

HSM(1M)

migmove(1M)

NAME

migmove – move migrated files from one migration level to another level

SYNOPSIS

```
/usr/openv/hsm/bin/migmove [-A] [-m methname]
          [-f a | o | d] [hsmname | file_system]

/usr/openv/hsm/bin/migmove -c 1 | 2 [-A]
          [-m methname] [-f a | o | d]
          [hsmname | file_system]

/usr/openv/hsm/bin/migmove -s miglevel [-d miglevel]
          [-A] [-m methname] [-f a | o | d]
          [hsmname | file_system]
```

AVAILABILITY

This command is not available with Storage Migrator Remote.

DESCRIPTION

The `migmove` command controls the movement of active migrated files from one migration level to another migration level. It scans the configured migration levels associated with the specified *hsmname* or *file_system*, selecting and moving files that meet the configured age and size criteria. A file is selected for movement if all of the following are true:

- The file is active (not marked `error`, `obsolete` or `dead` in the FHDB).
- The file's calculated move badness value is greater than or equal to the configured move badness value for the source migration level. File move badness is calculated using either the standard move badness formula (which factors in file age and size) or a site-specified move badness formula.
- The destination migration level and corresponding storage method are configured in `migconf`.
- No copy of the file already exists at the destination migration level, even if that file has the error flag set or is marked `obsolete` or `dead` in the FHDB. (Files marked `obsolete` or `dead` can be made active again by running `migsetdb`.)
- The source migration level method name is `ad`, `ct`, `dt`, `mt`, `op`, or `ow`.

VSM supports a total of up to eight migration levels. By default the `migmove` command assumes there will be two migrated copies, and divides the eight migration levels into a symmetrical hierarchy where the first copy is associated with odd-numbered levels and the second copy is associated with even-numbered levels. Defaults are set accordingly.

Sites can create asymmetrical hierarchies by migrating only one file copy, or by specifying both the source and destination migration levels in `migmove` commands.

For each configured migration level, the site must set the corresponding storage method parameter (METHOD1, METHOD2, ...METHOD8) in `migconf` to specify the method name, volume set number, hint, and volume pool with which the moved files are written.

Using the `xhsmadm` GUI you may configure move criteria for migration levels or for method names or for both. Move criteria for migration levels, if specified, take precedence over move criteria for method names, if specified.

VSM first attempts to retrieve the data from the volume at the source migration level. If that fails, VSM attempts to retrieve the data from another volume regardless of its migration level, starting with the storage device with the shortest file transfer time.

After a file moves from a source migration level to a destination migration level, the file at the source level can remain active or it can be marked obsolete or dead. Volumes can later be consolidated and recycled to reclaim the file space occupied by obsolete and dead files.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

The `migmove` command is normally run on a `cron` schedule or by the administrator.

OPTIONS

`-c 1 | 2`

For `-c 1`, move copy 1 of the migrated files. In the default configuration with all eight migration levels configured, the source migration levels 5, 3, and 1 are processed in that order. Files at level 5 move to level 7, then files at level 3 move to level 5, and then files at level 1 move to level 3. Configured source migration levels are scanned only if their respective destination migration levels are also configured.

For `-c 2`, move copy 2 of the migrated files. In the default configuration with all eight migration levels configured, the source migration levels 6, 4, and 2 are processed in that order. Files at level 6 move to level 8, then files at level 4 move to level 6, and then files at level 2 move to level 4. Configured source migration levels are scanned only if their respective destination migration levels are also configured.

The default is move both copy 1 and copy 2 files when neither `-c` nor `-s` are specified. If `-c` is specified, `-s` cannot be specified.

`-s` *miglevel*

Move files from the source migration level indicated by *miglevel*, where *miglevel* is an integer in the range 1 to 8. If the destination migration level `-d` is not specified, *miglevel* cannot equal 7 or 8. The default destination migration level is the source migration level +2.

If `-s` is specified, `-c` cannot be specified.

`-d` *miglevel*

Move files to the destination migration level indicated by *miglevel*, where *miglevel* is an integer in the range 1 to 8. The destination migration level cannot equal the source migration level. The default is the source migration level +2.

If `-d` is specified, `-s` must be specified.

`-A` Select all files at source level ignoring selection criteria.

`–m` *methname*

Scan for this method name when selecting and moving files for all source migration levels processed. Allowed values for *methname* are `ad`, `ct`, `dt`, `mt`, `op`, and `ow`. Default is to scan for all allowed and configured method names for all source migration levels processed.

If `-m` is specified, either `-c` or `-s` may also be specified.

`–f a | o | d`

For `-f a`, leave the FHDB entries for the selected files `active` at the source migration level.

For `-f o`, mark the FHDB entries for the selected files `obsolete` at the source migration level.

For `-f d`, mark the FHDB entries for the selected files `dead` at the source migration level. If the method name is ad, mark the FHDB entries for the selected files `obsolete` at the source migration level.

When `–f a|o|d` is not specified, the default is what is specified in the `migconf` move flags.

`–h` Print help information.

hsmname The HSMDEV entry that specifies the file system to be processed for moving files from one migration level to another migration level. See `migconf(1M)` for more information on HSMDEV entries. *hsmname* may be in either the active or inactive state.

file_system The full path name of the file system to be processed for moving files from one migration level to another migration level. The *hsmname* for this file system must be in the active state.

If neither *hsmname* nor *file_system* is specified, all *hsmname*s are processed. whether active or inactive

CAVEATS

- Although the default for configuring VSM is to make one copy of each migrated file, the defaults for `migmove` assume two migrated copies and a symmetrical hierarchy where the first copy is associated with odd-numbered migration levels and the second copy is associated with even-numbered migration levels. To implement any other multilevel migration hierarchy, configure the migration levels you need and then specify both the source and destination migration levels in `migmove` commands.
- You cannot run more than one `migmove` operation simultaneously on a shared VSM database.
- If `migmove` is interrupted by a system crash, the last incomplete move by `migmove` of a file to another migration level will complete when the VSM startup script `migrc -M` is run.
- `migmove` will deadlock and time out if the same storage unit drive is the only one available to process files at both the source migration level and the destination migration level.
- You must not run `migcons` and `migmove` simultaneously on a shared VSM database. Even if `migcons` and `migmove` are run simultaneously on different databases, you must make sure there are enough storage unit drives to avoid a deadlock condition.
- The `migmove` command can be used to restore missing copies of files at METHOD1 (or METHOD2 if two copies of files are made). Before using `migmove` in this manner, change the state of the *hsmname* to inactive. This avoids problems that may occur if either a `migbatch` or `mignospace` operation occurs simultaneously with `migmove`. You must specify *hsmname* in the migmove syntax.
- Do not run `migcons` and `migmove` simultaneously if they both are taking source from the same volumes. The results of such an action are undefined.

EXAMPLES

The *hsmname* for the file system to be processed in these examples is alpha.

The following example is the most general case.

```
migmove alpha
```

It assumes there are two migrated copies and all eight migration levels are configured into a symmetrical hierarchy where the first copy is associated with odd-numbered levels and the second copy is associated with even-numbered levels. The default conditions will move selected copy 1 and copy 2 files with any method name from source migration levels 5 and 6 respectively to their corresponding default destination migration levels, 7 and 8 respectively, and then process the other source migration levels in decreasing numerical sequence. After they are moved, selected files at all source migration levels are marked `dead` by default unless the method name is ad, in which case they are marked `obsolete`.

In the special case where only migration levels 1, 2, and 3 are configured, this same command

```
migmove alpha
```

moves selected files with any method name only from source migration level 1 to default destination migration level 3.

The following example moves only copy 2 of migrated files. Assuming all migration levels are configured, the command moves selected files with method name `ct` sequentially from source migration levels 6, 4, and 2 to their default destination migration levels, 8, 6, and 4 respectively. After they are moved, selected files at all source migration levels are marked `dead` by default.

```
migmove –c 2 -m ct alpha
```

The following example moves selected files with any method name from source migration level 4 to default destination migration level 6. After they are moved, selected files at source migration level 4 remain `active`.

```
migmove -s 4 -f a alpha
```

The following example is that of an asymmetrical hierarchy where the destination migration level is specified to be something other than the default value (source migration level plus 2). This command moves selected files with any method name from source migration level 3 to destination migration level 4. After they are moved, selected files at source migration level 3 are marked `obsolete`.

```
migmove -s 3 -d 4 -f o alpha
```

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath`/database/FHDB`

File-handle database

dwpath`/database/migconf`

VSM configuration file for managed file systems

`/usr/var/openv/hsm/database/migconfg`

VSM global-configuration file

dwpath`/database/migsweep.site`

Site migration and purge criteria

dwpath/database/migsweepm.site

Site move criteria

/usr/var/openv/hsm/example/database/sample.migpolicy.script

Sample migpolicy replacement script

SEE ALSO

migconf(1M), migconfg(1M), migpolicy(1M), migsetdb(1M)

mignbexport(1M)

NAME

mignbexport – export VSM files

SYNOPSIS

```
/usr/openv/hsm/bin/mignbexport [-s dir] [-e ExpLevel]
            [-S schedule] [-C class] [-D timeout] hsmname
```

AVAILABILITY

This command is not available with Storage Migrator Remote.

DESCRIPTION

`mignbexport` is used to export files from one VSM managed filesystem so they can be imported into a different VSM managed filesystem using the companion `mignbimport` command. Migrated files, if any, are not cached by either `mignbexport` or `mignbimport`.

Using this command requires that NetBackup be installed and running on the server executing `mignbexport`.

`mignbexport` does a user directed backup of the exported files and the FHDB entries and VOLDB entries for those exported files. These NetBackup images are exported to the site doing the `mignbimport`.

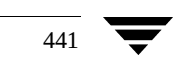
The backup is done using a NetBackup class that must be defined in the NetBackup configuration prior to executing `mignbexport`. The NetBackup class must define a NetBackup volume pool that contains volumes that can be sent to the site doing the `mignbimport`.

The VSM volumes that contain data for any migrated files being exported must also reside on VSM volumes that can be sent to the site doing the `mignbimport`.

There are two ways to export files.

- Export all migrated and unmigrated files residing in the specified subdirectory (*dir*) in the managed file system.
- Export all migrated files residing at export level *ExpLevel*. In this case, do not specify the `-s` *dir* option.

After running `mignbexport`, *dwpath*`/database/Volumes_to_export.`*pid* contains a list of all VSM volumes that are exported. Send these volumes to the site doing the `mignbimport`.



After running mignbexport, *dwpath*/database/Client_name.*pid* contains the NetBackup client name that initiated the user directed backup. This client name is needed when doing the NetBackup import at the import site.

After running mignbexport you must identify which NetBackup volumes belonging to the specified NetBackup class were used during the mignbexport operation. Send these volumes to the site doing the mignbimport.

This command may be run without regard to the VSM state (active or inactive), but the migration daemon migd must be running.

OPTIONS

-s *dir* Full path of the directory to export. This must be a subdirectory in the managed file system. mignbexport moves files in *dir* to *ExpLevel* and exports all migrated and unmigrated files in *dir*.

Prior to running mignbexport a METHOD*ExpLevel* must be defined in migconf and there must be VSM volumes registered for this METHOD*ExpLevel.* After mignbexport moves migrated data to these VSM volumes, the volumes are sent to the site doing the mignbimport.

All files residing in *dir* following a mignbexport operation remain there.

If -s *dir* is specified and there are any migrated files in *dir* that reside at level *ExpLevel* prior to running mignbexport, the command will abort with an error message.

If -s *dir* is not specified, mignbexport will only export files that currently reside at *ExpLevel*. No files will be moved to *ExpLevel* and unmigrated files will not be exported.

-e *ExpLevel*

Migration level of volumes to export. Valid values range from 1 to 8. Default is 8. See the -s option for an explanation of the relationship between *ExpLevel* and *dir*.

-S *schedule*

NetBackup schedule in class *class* to use for the user directed backup. The default is HSMEXPS. The schedule must be defined in the NetBackup configuration prior to running mignbexport.

-C *class* NetBackup class to use for the user directed backup. The default is HSMEXPC. The class must be defined in the NetBackup configuration prior to running mignbexport. The class must use NetBackup volumes that can be moved to the site doing the mignbimport.

-D *timeout*

A deadman timeout value in minutes. If NetBackup does not respond to the user directed backup request of mignbexport within the *timeout* period, mignbexport will abort. The default is 60 minutes.

hsmname The name of the file system from which files are exported. For more information, see `migconf(1M)`.

CAVEATS

- All systems exporting and importing files between them must have unique Machine IDs for managed file systems. Valid hexadecimal values range from 1 to FFFFFF. Default is 1000.
- All systems exporting and importing files between them must have unique volume labels for the VSM volumes containing the migrated files being exported. These volumes are sent to the site doing the `mignbimport`.
- All systems exporting and importing files between them must have unique volume labels for the NetBackup volumes sent to the site doing the `mignbimport`.
- All systems exporting or importing files between them must have unique NetBackup client names for the client doing the user directed backup.
- If `mignbexport` does not terminate successfully you may have to cleanup the exporting file system before trying again. If the `-s` option was specified, the VSM volumes containing the migrated files being exported may have to be recycled.
- Configure the export migration level (*ExpLevel*) and corresponding method before using this command.

EXAMPLE

In this example, the exporting site will move copies of all migrated and unmigrated files in directory `/hsm04/export/6/4` to default level 8 for export from file system `hsm04`. The Netbackup schedule is the default `HSMEXPS`, and the NetBackup class is the default `HSMEXPC`.

```
mignbexport -s /hsm04/export/6/4 hsm04
```

A typical response is shown below:

```
Setting HSM hsm04 state to "inactive"

Setting HSM hsm04 state to "active"

Waiting on NetBackup backup ...

Waiting on NetBackup to complete ...
```

Note The dots are printed in sequence while tapes are writing or rewinding.

```
The HSM volumes exported are listed in file
   /usr/var/openv/hsm04/database/Volumes_to_export.225

The NetBackup client name is listed in file
   /usr/var/openv/hsm04/database/Client_name.225
```


After running `mignbexport`, remove the VSM volumes listed in *dwpath*`/database/Volumes_to_export.`*pid* and send them to the importing site. Use the NetBackup and Media Manager GUIs to identify which volumes are in class `HSMEXPC`, and also send the ones that have been written to the importing site.

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath`/database/FHDB`

File handle database for VSM.

dwpath`/database/VOLDB`

Volume database for VSM.

dwpath`/database/migconf`

Configuration file for VSM.

dwpath`/database/Volumes_to_export.`*pid*

A list of all VSM volumes that are exported

dwpath`/database/Client_name.`*pid*

NetBackup client name

SEE ALSO

`migconf(1M), migconfg(1M), migdbcheck(1M), migdbrpt(1M), mignbimport(1M), migmove(1M)`

mignbimport(1M)

NAME

mignbimport – import VSM files

SYNOPSIS

```
/usr/openv/hsm/bin/mignbimport [-i ImpLevel]
          [-C class] -m client_name [-D timeout] hsmname
```

AVAILABILITY

This command is not available with Storage Migrator Remote.

DESCRIPTION

`mignbimport` takes files previously exported from one VSM managed filesystem using the companion `mignbexport` command and imports them into a different VSM managed filesystem.

Using this command requires that NetBackup be installed and running on the server executing `mignbimport`.

`mignbimport` does a user directed NetBackup restore of the exported files and the FHDB entries and VOLDB entries for the exported files. This restore also includes information about the original path of the exported VSM managed filesystem.

`mignbimport` merges the FHDB and VOLDB entries created by `mignbexport` into the FHDB and VOLDB of the importing managed file system, *hsmname*. When the files are restored, the portion of the path that was the original path of the exported VSM managed filesystem is replaced with the path of the VSM managed filesystem into which the files are being imported.

The VSM volumes that were exported must be placed in the desired library before running `mignbimport`. These volumes retain their original volume name so they must be unique. Do not register these volumes to VSM because `mignbimport` will take care of this automatically.

A NetBackup class must be defined prior to running `mignbimport` and this class must have the same name as the NetBackup class used with the `mignbexport`. This class must reference a NetBackup volume pool that contains only the NetBackup volumes that were written by `mignbexport`. The NetBackup volumes that were written by `mignbexport` must be imported into NetBackup at the site importing the files prior to running `mignbimport`. For more information on importing NetBackup images, see Chapter 6 of the *VERITAS NetBackup System Administrator's Guide*.

This command may be run without regard to the VSM state (active or inactive), but the migration daemon `migd` must be running.

OPTIONS

`-i` *ImpLevel*

Migration level of the imported volumes. Valid values range from 1 to 8. Default is 7.

`mignbimport` will change the FHDB entries for the imported files to indicate that the files reside at this level. `mignbimport` will also change the VOLDB entries for the imported volumes so they appear to be full and will never be written on again.

`-C` *class* NetBackup class to use for the user directed restore. It must be the same as the *class* used with `mignbexport`. The default is `HSMEXPC`. The class must be defined in the NetBackup configuration prior to running `mignbimport`. The class must use NetBackup volumes that were exported from *client_name* and sent from the site doing the `mignbexport`. These volumes must be imported into NetBackup prior to running `mignbimport`.

`-m` *client_name*

Name of the NetBackup client from which the VSM files were exported. See the *dwpath*`/database/Client_name.`*pid* file created when the `mignbexport` operation finished.

`-D` *timeout*

A deadman timeout value in minutes. If NetBackup has not responded within the *timeout* period, `mignbimport` will abort. The default is 60 minutes.

hsmname The name of the file system from which files are exported. For more information, see `migconf(1M)`.

CAVEATS

- All systems exporting and importing files between them must have unique Machine IDs for managed file systems. Valid numerical values range from 1 to FFFFFF. Default is 1000.
- All systems exporting and importing files between them must have unique volume labels for the VSM volumes containing the migrated files being exported. These volumes are sent from the site doing the `mignbexport`.
- All systems exporting and importing files between them must have unique volume labels for the NetBackup volumes sent from the site doing the `mignbexport`.
- Do not run `mignbimport` twice on a system to import the same files without first cleaning up any errors from the first run.

- Imported files lose any designations they may have as tied files, and must be redesignated. See man page `migtie(1)`.
- Configure the import migration level (*ImpLevel*) and corresponding method before using this command.
- The NetBackup class used for the user directed restore must match the NetBackup class used in the `mignbexport` operation, and must be defined in the NetBackup configuration prior to running `mignbimport`.
- Imported VSM volumes must be loaded into a library prior to running `mignbimport`.
- The NetBackup volumes that were exported must be in a NetBackup pool referenced by *class*, and must have been imported into NetBackup as *client_name* prior to running `mignbimport`.

EXAMPLE

Place the NetBackup volumes containing files to be imported in the appropriate storage devices and register them with Media Manager for the importing VSM managed file system, giving them a unique pool name. Define the NetBackup class `HSMEXPC` to use this unique pool name

Using NetBackup, import the client *client_name*. In this example, this is hat.

Place the VSM volumes containing file data to be imported in the appropriate storage devices and register them with Media Manager for the importing VSM managed file system, giving them a pool name of `HSM`. If this is not done, Media Manager will issue requests for the operator to assign this media.

In this example, the site will import copies of files to default level 7 on file system `hsm01c`. The name of the NetBackup client is `hat`.

```
mignbimport -m hat hsm01c
```

A typical response is shown below:

```
Setting HSM hsm01c state to "inactive"
Waiting on NetBackup restore in Phase 1...
Setting HSM hsm01c state to "active"
Waiting on NetBackup restore in Phase 2...
mignbimport complete.
```

Note The dots are printed in sequence while tapes are reading or rewinding.

The imported files now are ready for access by VSM on the file system at the new location.

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath`/database/FHDB`

File handle database for VSM.

dwpath`/database/VOLDB`

Volume database for VSM.

dwpath`/database/migconf`

Configuration file for VSM.

SEE ALSO

`migconf(1M), migconfg(1M), migdbcheck(1M), migdbrpt(1M), mignbexport(1M), migmove(1M), migtie(1)`

mignbscan(1M)

NAME

mignbscan – scan nb volume and reconstruct database entries for the nb storage method

SYNOPSIS

```
/usr/openv/hsm/bin/mignbscan [-s] [-h] hsmname label
        NB_master class_name NB_client level
```

DESCRIPTION

mignbscan scans a NetBackup class and displays information about the volume as a whole as well as information about each file granule on the volume. It also reconstructs the FHDB and VOLDB entries for all scanned granules.

A file is migrated to NetBackup as a single granule. The granule header, <*machid*>M<*handle*>.GLABEL.<*level*>, is separately backed up as a file. The granule header contains FHDB and VOLDB entry information.

The mignbscan command creates two output files, FHDB.*label* and VOLDB.*label*, in the *dwpath*/database directory. The structure of these files is the same as the FHDB and VOLDB database files. These files may be used to rebuild the FHDB and VOLDB if they are corrupted or damaged (see migdbcheck(1M)).

You can sort and merge FHDB.*label* files for different volumes to recreate the FHDB. Similarly, merging VOLDB.*label* files for different volumes can recreate the VOLDB.

Note When recreating the VOLDB be sure to merge the VOLDB file in the /usr/var/openv/hsm/example/database directory to include the entry for the dk method.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon migd is running.

OPTIONS

The following parameters are available under migftscan:

-s	Silently scan the volume and create FHDB.*label* and VOLDB.*label* files, but do not display information on stdout.
-h	Print help information.
hsmname	HSMDEV entry that specifies the path to the database containing information on the desired volume. For more information on HSMDEV entries, see migconfg(1M).

label NetBackup volume label used when registering the volume. This parameter is required.

NB_master
Name of the master NetBackup server for `nb` volumes. *NB_master* must be the first name the server is known by to NetBackup.

class_name
Name of the NetBackup class. The NetBackup class must be active.

NB_client Name of the NetBackup client. Check the NetBackup GUI, `xbpadm`, to identify the correct value to use for *client*.

level The migration level of this volume. Valid values are 1 or 2.

CAVEATS

- Only the system administrator may use this command.

EXAMPLE

The following example scans the NetBackup disk volume NB003 that is registered under the `nb` method. The NetBackup server is duo, the NetBackup class is hsmnb2, the NetBackup client is trio, and the migration level for this volume is 1. The display shows a list of granule information for files backed up onto the volume. The FHDB.NB003 and VOLDB.NB003 files are created in the *dwpath*`/database` directory.

```
mignbscan hsm04 NB003 duo hsmnb2 trio 1

Volume label Found ====> NB003 duo hsmnb2
------------------------------

Obtaining list of granules.

Restoring granule header files

..
/hsm04/migration/data/3e8M24c5.GLABEL.1 <=> 00014B6B 00000000
Mon Apr  8 12:54:35 1996 /hsm04/files/a

/hsm04/migration/data/3e8M24c6.GLABEL.1 <=> 00014B6B 00000000
Mon Apr  8 12:54:36 1996 /hsm04/files/b

/hsm04/migration/data/3e8M24c7.GLABEL.1 <=> 00014B6B 00000000
Mon Apr  8 12:54:36 1996 /hsm04/files/c

/hsm04/migration/data/3e8M24c8.GLABEL.1 <=> 00014B6B 00000000
Mon Apr  8 12:54:36 1996 /hsm04/files/d

/hsm04/migration/data/3e8M24c9.GLABEL.1 <=> 00014B6B 00000000
Mon Apr  8 12:54:37 1996 /hsm04/files/e

/hsm04/migration/data/3e8M24ca.GLABEL.1 <=> 00014B6B 00000000
Mon Apr  8 12:54:37 1996 /hsm04/files/f
```

```
/hsm04/migration/data/3e8M24cb.GLABEL.1 <=> 00014B6B 00000000
Mon Apr  8 12:54:37 1996 /hsm04/files/g

/hsm04/migration/data/3e8M24cc.GLABEL.1 <=> 00014B6B 00000000
Mon Apr  8 12:54:37 1996 /hsm04/files/h
```

Particulars of granules displayed include (in order): granule file name, migrated file size, offset of the granule in the migrated file, date of backing up the granule, and migrated file pathname.

Including an incorrect class name in the command gives this result:

mignbscan hsm04 NB003 duo badname trio 1

```
Warning NO Volume label found active for duo badname
```

There are no output files, and the hsmlog contains:

```
ERROR - the specified class does not exist in the configuration database
Class:migrate exit code:230
```

Including an incorrect migration level in the command gives this result:

mignbscan hsm04 NB003 duo hsmnb2 trio 0

```
Volume label Found ====> NB003 duo hsmnb2

------------------------------

Obtaining list of granules.

Restoring granule header files

No files found in /tmp/mignbscan2.2401Z
```

There are no output files, and the hsmlog contains:

```
ERROR 13:29:29 INF - Status = no files specified in the file list.
```

Including an incorrect NetBackup client in the command gives this result:

mignbscan hsm04 NB003 duo hsmnb2 badclient 1

```
% echo $status

4
```

The output files are deleted, and the hsmlog contains:

```
ERROR: bpflist failure ABORTING no entity was found
```

Including an incorrect NetBackup server in the command gives this result:

mignbscan hsm04 NB003 badserver hsmnb2 trio 1

```
% echo $status

3
```


There are no output files, and the hsmlog contains:

```
ERROR 13:33:38 INF - Status = no entity was found.
```

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath`/database/FHDB`

File handle database for VSM.

dwpath`/database/VOLDB`

Volume database for VSM.

dwpath`/database/migconf`

Configuration file for VSM.

dwpath`/database/FHDB.`*label*

File handle database for current volume.

dwpath`/database/VOLDB.`*label*

Volume database for current volume.

SEE ALSO

`migconfg(1M), migdbcheck(1M), migdbrpt(1M), migreg(1M), migadscan(1M), migftscan(1M), migtscan(1M), migopscan(1M)`

mignewlog(1M)

NAME

mignewlog – truncate or restart and optionally make a copy of the VSM log

SYNOPSIS

```
/usr/openv/hsm/bin/mignewlog
        hsmname [destination_file] | GLOBAL [destination_file]
```

DESCRIPTION

VSM maintains a global-log file and can also have a separate log file for each HSMDEV entry in the `/usr/var/openv/hsm/database/migconfg` file.

- The pathname to the global-log file is not configurable and is always `/tmp/hsm.log` (it may be a symbolic link).
- You define the log file paths for each VSM entry during configuration. The term *lgpath* refers to these file paths. See your `/usr/var/openv/hsm/database/migconfg` file for the actual log file names.

VSM uses each individual HSMDEV log file *lgpath* and the global-log file (`/tmp/hsm.log`) to store messages, status, and errors. To ensure that the global-log is maintained between boots and does not fill up the `/tmp` partition, you should link it to a file that resides on a file system other than `/tmp`.

If the *lgpath* or `/tmp/hsm.log` files become too large, you can use the `mignewlog` command to truncate or restart them and release the associated disk space. Note that using `rm` to remove the global-log file `/tmp/hsm.log` does not release the disk space associated with that file if the migration daemon (`migd`) is running.

Before truncating or restarting any log files, you should inspect them for abnormal errors.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

OPTIONS

hsmname The HSMDEV entry that specifies the path to the log file you want `mignewlog` to truncate or restart. If *hsmname*=`GLOBAL`, then `mignewlog` uses the VSM global-log file, `/tmp/hsm.log`. See `migconfg(1M)` for more information on the global-configuration file and HSMDEV entries.

destination_file
If you specify this parameter, `mignewlog` copies the log to the specified destination before truncating or restarting it. If you do not specify a *destination_file*, the log is removed.

CAVEATS

- If migration to secondary media is taking place, the disk space for *lgpath* is not actually released until the copy-out operation completes.

FILES

`/usr/var/openv/hsm/database/migconfg`

VSM global-configuration file

`/tmp/hsm.log`

VSM global-log file

SEE ALSO

`migconfg(1M), migchecklog(1M), migrc(1M), startmigd(1M)`

mignospace(1M)

NAME

mignospace – make disk space available

SYNOPSIS

```
/usr/openv/hsm/bin/mignospace [-N] [-i | -h]
        hsmname | file_system
```

DESCRIPTION

The `mignospace` command attempts to make space available in the indicated file system. This command starts in either of two ways:

- The administrator executes it directly from the system prompt, from the `xhsmadm` GUI, or with a `crontab` entry.
- A user attempts to write to the file system when the actual free space is less than the configured free space (see `migthreshold(1M)`). Here, the kernel informs `migd` of the low-space condition and `migd` starts `mignospace`.

`mignospace` purges premigrated file space in a sequence determined by the purge badness attributes configured for the file system. The default purge badness attributes will purge space for the oldest files first, regardless of size. See `migconf(1M)`.

`mignospace` starts by checking for premigrated files.

- If some premigrated files exist that exceed purge badness, `mignospace` either purges this premigrated file space to the `purgemark` percentage and stops or purges all this premigrated file space and stops, whichever comes first.
- If premigrated files exist but none exceed purge badness, `mignospace` cuts the current purge badness in half and exits.
- If there are no premigrated files, `mignospace` cuts the current badness in half and selects enough files to increase free space to the low-water mark percentage. `mignospace` then premigrates the selected files, copies them to secondary storage, and determines if any exceed purge badness. If some premigrated files exceed purge badness, `mignospace` either purges this premigrated file space to the `purgemark` percentage and stops or purges all this premigrated file space and stops, whichever comes first. If no premigrated files exceed purge badness, `mignospace` cuts the current purge badness in half and exits.

`migrc` and `migbatch` reset current badness and current purge badness to their original *dwpath*`/database/migconf` values.

If the *dwpath*/database/migconf file specifies a lowwater value, mignospace selects only enough files to satisfy lowwater. If lowwater is unspecified, mignospace selects all files meeting the selection criteria. Files in premigration are not considered free, and files in .migrate are not used in calculating lowwater.

If the *dwpath*/database/migconf file specifies a purgemark value, mignospace purges only enough premigrated file space to satisfy purgemark. (To ignore a specified purgemark value and purge all premigrated file space, use the -i parameter with the mignospace command.) If purgemark is unspecified, mignospace purges all premigrated file space.

The log for the HSMDEV entry (see *hsmname* below) indicates the action taken by mignospace.

OPTIONS

-N	Make file space available in increments.
-i	Ignore the purgemark and purge all available premigrated file space.
-h	Print help information.
hsmname	HSMDEV entry for the file system in which you want to provide space. See migconfg(1M) for more information on HSMDEV entries.
file_system	Full path name of the file system in which you want to provide space.

CAVEATS

- Always maintain a sufficient number of VSM-registered tapes, optical platters, or alternate magnetic disks since mignospace may be started automatically by the system on low space conditions. See migreg(1M).
- The migration daemon migd and device-manager daemon (see ltid(1M)) must be running and the hsm state must be active for mignospace to start automatically on low space conditions.
- Some processes spawned by mignospace create large temporary files and there may not be enough space in the /tmp directory to store these files. You can avoid this problem by defining the environment variable TMPDIR to contain the pathname of a directory in a file system that has sufficient space available. Then, if the process checks TMPDIR, it creates any temporary files in the specified directory instead of in the default /tmp.

EXAMPLE

The following example, calls `mignospace` to increase the space available on the file system named `/mig2` and then on the file system specified by the `migconfg` HSMDEV entry named `alpha`.

```
mignospace /mig2
mignospace alpha
```

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath`/database/migconf`

VSM configuration file for managed file systems

`/usr/var/openv/hsm/database/migconfg`

Global-configuration file for VSM

SEE ALSO

`HSM(1M), ltid, migconf(1M), migconfg(1M), startmigd(1M),`

migopscan(1M)

NAME

migopscan – scan the optical volume for file granules

SYNOPSIS

```
/usr/openv/hsm/bin/migopscan [-F] [-n] [-s]
        hsmname volume_label method
```

DESCRIPTION

See `migtscan(1M)`, `migopscan(1M)` for a description of this command.

migpolicy(1M)

NAME

migpolicy – specify VSM policy and method to write files to secondary storage

SYNOPSIS

`/usr/openv/hsm/bin/migpolicy` *hsmname inputfile*

DESCRIPTION

`migpolicy` reads the input file and adds the number of copies and the destination method to `migcopy`'s input script. `migpolicy` is used whenever the controlling commands `migbatch` or `mignospace` are run. The `migpolicy` command calls `/usr/openv/hsm/bin/admincmd/migpolicy.script`, which is responsible for filling in fields 5 (volset), 7 (method), 13 (hint), 14 (comment), and 15 (mmlevel) of the input file, and for placing the altered information on worklists, the names of which are echoed as an exit condition.

The administrator normally should not alter `migpolicy` or use it directly. However, it is possible to enhance the migpolicy.script to divert specific files to suitable media. You can divert files by file size, by owner, or by any other distinguishable file characteristic.

This is accomplished by replacing `/usr/openv/hsm/bin/admincmd/migpolicy.script` with an alternate script. A sample replacement is provided in `/usr/var/openv/hsm/example/database/sample.migpolicy.script`. This sample replacement script uses the size of a file to determine whether VSM will copy it to the first entry or to the second entry of a method definition.

OPTIONS

hsmname HSMDEV entry that specifies the file system from which you are migrating files. See `migconfg(1M)` for more information on HSMDEV entries.

inputfile List of files to be migrated; its format is similar to the `copydb.xx.x` work file.

```
0000103A|000003E8|00000000|00000100|0|00000000|ad||0.92|512|472|
/home1/migtie/asd|library|Auto HSM run|00000001|00000000|
```

where:

```
1       2       3     4      5       6       7       8         9
handle|machid|lock|flags|volset|copied|method|dst_seek|age|

10    11       12    13    14       15       16
size|badness|path|hint|comment|mmlevel|slevel|
```


```
flags x200 is success code.
flags x100 is a failure.
```

This file is specified in the VSM controlling scripts.

CAVEATS

The method specified in `migpolicy` must be a valid method defined in the *dwpath*`/database/migconf` configuration file.

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath`/database/migconf`

VSM configuration file for managed file systems

`/usr/var/openv/hsm/database/migconfg`

VSM global-configuration file

`/usr/var/openv/hsm/example/database/sample.migpolicy.script`

Sample `migpolicy` replacement script

hsmname`.copydb.`*method_name*`.`*vol_set_number*`.`*hint*

VSM work lists

SEE ALSO

`HSM(1M), migbatch(1M), migconf(1M), migconfg(1M), mignospace(1M)`

migpurge(1)

NAME

migpurge -purge migrated file disk space

SYNOPSIS

`/usr/openv/hsm/bin/migpurge` *filename* [*filename* `...`]

`/usr/openv/hsm/bin/migpurge -f` *filelist*

DESCRIPTION

`migpurge` purges disk space for the specified migrated files. This is useful if you want to make more disk space available immediately. File space is not purged if all copies of the specified files have not been made. See Caveats.

Users cannot purge files owned by another user. Root users can purge files owned by anyone.

The system administrator must provide the proper permissions before users can use this command. See “Enable User Permissions” on page 156.

OPTIONS

filename	Files that `migpurge` should purge. May be given as a relative path or absolute path. You can specify multiple files or use standard regular expressions. Wildcards are recognized.
`-f` *filelist*	Purge the files listed in *filelist*. The format of *filelist* is one file name per line, showing the full pathname of each file or a pathname relative to the current directory in which `migpurge` is executed, not the directory in which the file in the *filelist* resides. Wildcards are not recognized.

CAVEATS

- The VSM state must be active and the migration daemon `migd` must be running when you issue a `migpurge` command.
- If desired, to make sure all migration copies have been made before using `migpurge`, the administrator can first run `migrc -R` *hsmname* to ensure that any unfinished migration work is completed.

EXAMPLES

The following command purges file space for f1 and f2.

```
migpurge -f purgelist
```

where the file purgelist reads as follows:

```
/usr/mydir/subdir/f1
/usr/anotherdir/subdir/f2
```

The following command purges file space for f4, f5, and f6 in the current working directory.

```
migpurge f4 f5 f6
```

The following command purges all file space in the current working directory whose file names begin with the string `July_15`.

```
migpurge July_15*
```

SEE ALSO

HSM(1M), fls(1), migloc(1), mignospace(1M), migrate(1)

migrate(1)

NAME

migrate – request premigration of a file

SYNOPSIS

`/usr/openv/hsm/bin/migrate [-F]` *filename* `[`*filename* `...]`

`/usr/openv/hsm/bin/migrate [-F] -f` *filelist*

DESCRIPTION

`migrate` allows users to request that VSM migrate a file to secondary storage. Users can premigrate regular files that they own in VSM-managed file systems if allowed to use this command. The system administrator must provide the proper permissions before users can use this command. See “Enable User Permissions” on page 156.

All files specified with a single `migrate` command must be in the same managed file system. The first file determines that file system.

Use the `fls` command to determine if a file has been migrated or premigrated.

The `migrate` command performs only the premigration phase of migration. The files do not actually migrate to secondary storage until the next time `mignospace`, `migbatch`, or `miglow` execute. After a file is migrated to secondary storage, accessing it causes VSM to transparently cache the file back to disk.

If the `migrate` command is used to premigrate an unmodified cached file for which enough valid copies already exist in secondary storage, VSM does not recopy the file. VSM purges the data immediately from disk if the `dk` method entry for the file does not exist in the FHDB, otherwise the data is purged later by `mignospace`. Use the `migloc` command to determine whether or not file data was purged; if the file remains in premigration, `migloc` will show a `dk` method entry for the file.

The `migrate` command creates a list of files in `/tmp/migrate_list.`*pid* that meet the migration file selection criteria. The *pid* is the process ID of the `migrate` command that is executing. This list is of the form:

age size 0 | 0 | *path* | *machid* | *handle* |

where the third field is a default badness value of 0, and the fourth field is 0 which indicates the migration level for premigration. The *machid* and *handle* are both 0 if the file has never migrated before.

OPTIONS

`-F` Force file migration even if the file is listed in the global stop file or a local stop file, `.migstop`. This argument is applicable only to Super Users or to owners of the file.

filename Files that `migrate` should migate. May be given as a relative path or absolute path. You can specify multiple files or use standard regular expressions. Wildcards are recognized.

`-f` *filelist* Migate the files listed in *filelist*. The format of *filelist* is one file name per line, showing either the full pathname of each file or a pathname relative to the current directory in which `migrate` is executed, not the directory in which the file in the *filelist* resides. Wildcards are not recognized.

CAVEATS

- Unless the `-F` argument is given, `migrate` will not premigrate any files listed in the global stop file or in a local stop file, `.migstop`, unless the stop file is overridden. A typical error response is as follows:

  ```
  /ov2/stop/nono found in user stopfile not allowed to migrate
  ```

  The `.migstop` file most local to the listed file overrides other stop files higher in the directory tree. Local control files override global control files if the same file or directory is listed in both. If the same file is listed in both a `.migrate` file and a `.migstop` file at the same level, the `.migrate` control file overrides the `.migstop` file.

- Files whose pathname length exceeds 1023 characters will not be migrated.

- For non-root users, the VSM state must be active and the migration daemon `migd` must be running when a `migrate` command is issued.

  For root users, this command may be run without regard to the VSM state (active or inactive) or whether the migration daemon is running.

EXAMPLES

In the following example, the user requests that VSM migrate a file named `tproga`.

```
migrate /home/gls/tproga
```

In the following example, the user has a file named migrate_list that contains a list of files to be migrated, one file per line.

```
migrate `cat migrate_list`
```

The following example uses the `find` command to select for migration all regular files under a directory, and then uses `fls` to check if they are migrated.

```
migrate `find /home/gls/hawks -type f -print`
```

```
fls -l /home/gls/hawks
total 94
mr--r--r--  1 root other  23368  May 9  17:20 osprey
mr--r--r--  1 root other  23368  May 9  17:21 peregrine
```

FILES

`/usr/var/openv/hsm/database/migrate`

Global migrate file for VSM

`/usr/var/openv/hsm/database/migstop`

Global stop file for VSM

The following output file is created and used by `migrate` to premigrate files:

`/tmp/migrate_list.`*pid*

A list of files in the file system that meet the migration file selection criteria

The *pid* is the process id of the `migrate` command that is executing. `migrate` checks the environment variable *TMPDIR*, which allows the administrator to use a path other than the default `/tmp`. This can save files through system reboots or make use of a larger file system to avoid running out of space. The path defined by *TMPDIR*, if set, is used instead of `/tmp` as the directory in which to place any temporary files.

SEE ALSO

`HSM(1M), fls(1), migloc(1), migmode(1), migpurge(1)`

migrc(1M)

NAME

migrc – VSM startup script

SYNOPSIS

```
/usr/openv/hsm/bin/migrc [-LRMh] [-o | -O db_type]
          [-a age] hsmname
```

DESCRIPTION

`migrc` should be run when the system reboots. Depending on the options you select, `migrc` clears locks in databases, removes obsolete and dead entries, clears database locks, and restarts any migrations or moves that may still be pending.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

OPTIONS

Note At least one of the options `L`, `R`, `M`, `h`, `o` or `O` must be specified.

`-L` Clear all locks and unmount volumes. For kernel-based implementations, this option clears all in progress flags in the `.PAIN` file. For nonkernel-based implementations, this option clears all in progress flags in the `.IHAND` file. Always execute `migrc -L` following a system crash.

Caution Only run `migrc -L` if the HSM daemon (`migd`) is not running and no HSM processes are active. If `migrc -L` is run while other HSM processes are running, the HSM locks may be left in an inconsistent state. Data loss could result.

`migrc -L` *hsmname* may be run to clear locks for one *hsmname*. This may be done while `migd` is running, if *hsmname* is inactive and no other HSM processes are running for *hsmname*.

`-R` Restart interrupted premigration and migration. Execute `migrc -RM` following a system interrupt to ensure that any unfinished migration and multilevel migration (`migmove`) work is completed.

`-M` Restart interrupted `migmove` work and FHDB flag updates. Execute `migrc -RM` following a system interrupt to ensure that any unfinished migration and multilevel migration (`migmove`) work is completed.

`-h` Print help information.

`-o` (Lower case o) Remove obsolete / dead entries from all three databases: FHDB, VOLDB and COPYDB.

`-O` *db_type*
(Upper case O) Remove obsolete / dead entries from the single database type specified by *db_type*: FHDB, VOLDB or COPYDB.

`-a` *age* The *age* parameter indicates the age (in days) of obsolete entries to remove from the FHDB. The default *age* is 7 days. This argument only has meaning when used in conjunction with either the `-O FHDB` option or with the `-o` option as it pertains to FHDB entries.

Caution You should use the *age* parameter to specify removing only files that were obsoleted prior to the last full backup.

hsmname The HSMDEV entry that you want VSM to process. See `migconfg(1M)` for more information on HSMDEV entries.

Note If you do not specify *hsmname*, `migrc` performs the selected operation(s) on all VSM managed file systems.

CAVEATS

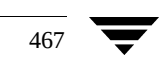
- Ensure that a sufficient number of volumes are available for migrations that may occur when you run `migrc`. Use `migreg` to register volumes.
- Running `migrc` resets the current badness and current purge badness to their respective values as specified in `migconf`.
- If using NetBackup in conjunction with VSM, set the *age* variable for `migrc` at a value higher than the longest NetBackup retention period on the managed filesystem. Do this by changing the line:

  `AGE=7`

  to:

  `AGE=`*nnn*

  where *nnn* is the longest NetBackup retention period.
- If MFLAG_OBSOLETE is set, `dk` entries will stay in the FHDB when `migrc` is run. This allows a cached file to be returned to premigration and processed by normal purge operations. To remove these `dk` entries and reduce the size of the FHDB, run `migmdclean`.
- When `migrc` completes there may still be VSM processes running that `migrc` has started.



SEE ALSO

HSM(1M), migconfg(1M), migreg(1M), startmigd(1M), stopmigd(1M)

migrd(1M)

NAME

migrd – start the VSM migrd daemon

SYNOPSIS

```
/usr/openv/hsm/bin/migrd
```

DESCRIPTION

`migrd` is available to the system administrator to start the VSM migrd daemon (`migrd`). You must have super-user privileges to execute this command. You should start this daemon as part of system startup. If `migrd` is not running when you use VSM-Java, the GUI will not connect to the server. Consequently, `migrd` is critical to the functionality of the VSM-Java GUI.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

SEE ALSO

`HSM(1M)`

migreconstruct(1M)

NAME

migreconstruct – Reconstruct lost or deleted migrated files

SYNOPSIS

```
/usr/openv/hsm/bin/migreconstruct [-v -l out_file
        -f -w in_file -p permissions] hsmname
```

DESCRIPTION

`migreconstruct` lets a system administrator reconstruct migrated files either when they have been accidentally deleted or when the file system is damaged beyond repair. (The command cannot reconstruct files that have not been migrated.) The preferred way to do this is to restore migrated files from their backup copies, if created previously by NetBackup. If no backup copies exist, however, use `migreconstruct` to recover the migrated files. `migreconstruct` uses information in the file-handle database (FHDB) to do the reconstruction.

If reconstructing a damaged file system, the administrator must first reinitialize the file system. Use `fsck`, `newfs`, or `mkfs` as necessary. In addition, the `.PAIN` (parallel inode) file must be created on kernel-based implementations, the *hsmname*`.IHAND` (inode-to-handle) file must be removed before starting `migd` on nonkernel-based implementations, the path to the managed directory must exist, and the `migration/data` directories must be created in the managed directory.

`migreconstruct` does not overwrite existing files. It only reconstructs lost or deleted migrated files.

OPTIONS

The following options and parameters are available with `migreconstruct`:

`-v` Verbose; lists the full path of all files reconstructed. Default is none are listed.

`-l` *out_file*

Lists on *out_file* all files that have an entry in the VSM file handle database (FHDB). Does not reconstruct any of these files. This parameter may not be used with `-f` or `-w`. Each line in *out_file* contains the following information:

handle | *machid* | *size* | *uid* | *gid* | *permissions* | *file_path*

The *handle* and *machid* fields are in hexadecimal; *size*, *uid* and *gid* are decimal; and *permissions* is octal. Default is no *out_file*.

`-f` Reconstructs all files that have an entry in the FHDB. This parameter may not be used with `-l` or `-w`. Default is no files are reconstructed.

`-w` *in_file* Reconstructs all files listed in *in_file*. This parameter may not be used with `-l` or `-f`.

Each line in *in_file* should contain the following information:

handle | *machid* | *size* | *uid* | *gid* | *permissions* | *file_path*

The *handle* and *machid* fields are in hexadecimal; *size*, *uid* and *gid* are decimal, and *permissions* is octal. *file_path* may be set to any full path in the managed file system. Default is no *in_file*.

`-p` *permissions*

Permissions to be used when creating files. *permissions* is assumed to be an octal value. Default is 1755. All files created as a result of the `-f` option will be given these permissions. All entries in the *out_file* will be given these permissions. The permission field in the *in_file* overrides *permissions*.

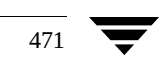
hsmname Name of the file system that VSM is managing.

Note One of the options `-l`, `-f`, or `-w` is required to run `migreconstruct`.

To restore accidentally deleted files, create proper *in_file* entries and use `migreconstruct` with the parameter.

CAVEATS

- All directories created by `migreconstruct` belong to the user of this utility, and take on whatever permissions are set in the user's umask. This would normally be root.
- `migreconstruct` will not reconstruct files from method `dk`.
- `migreconstruct` will reconstruct obsolete migrated files if VSM has not yet removed their corresponding FHDB entry. Obsolete files are no longer obsolete after they have been reconstructed.
- Moved or renamed migrated files will be reconstructed in their original locations according to the full path recorded in the FHDB when the files were first migrated.
- The hsm state must be active and the migration daemon `migd` must be running when you issue a `migreconstruct` command.
- All reconstructed files will have a 0 slice value.



EXAMPLE

In the following example, all files that have a full path in the FHDB of VSM-managed file system alpha will be reconstructed and granted default permission 1755.

```
migreconstruct -v -f alpha
```

In the following example, all files in VSM-managed file system openv2 with the path name `/tmp/out_list` will be reconstructed.

```
migreconstruct -w /tmp/out_list openv2
```

where `/tmp/out_list` contains:

```
1861|3e8|105|0|1|1755|/home1/home1/file-23_#
1862|3e8|108|0|1|1755|/home1/home1/file-24_$
1863|3e8|111|0|1|1755|/home1/home1/file-25_%
1864|3e8|114|0|1|1755|/home1/home1/file-26_&
1865|3e8|117|0|1|1755|/home1/home1/file-27_'
```

This shows the *handle machid size uid gid permissions* and *file_path* for the five files to be reconstructed.

FILES

dwpath`/database/FHDB`

File-handle database

SEE ALSO

`HSM(1M), migin(1M), newfs, fsck, mkfs, pfinit(1M)`

migrecycle(1M)

NAME

migrecycle – Reregister an empty volume

SYNOPSIS

```
/usr/openv/hsm/bin/migrecycle [-v volset [-P poolname]
          -d device -c capacity -s server -u user
          -l new_label | -h] hsmname label method
```

DESCRIPTION

`migrecycle` lets a system administrator reregister an empty registered VSM volume. If the volume is not empty, an error is produced without any change in registration. A volume is considered to be empty either if it has no FHDB entries for granules or if all FHDB entries for granules on the volume are marked dead. The volume is reregistered with the parameters in the VOLDB unless they are specified as `migrecycle` options. The volume set, volume pool, device, capacity, server, and the password may be changed by supplying them on the command line.

Note When one side of an optical disk is consolidated, the volume entry for that input volume is not removed from the VOLDB unless and until both sides of that optical disk are empty. See `migcons(1M)`.

`migrecycle` first calls `migmdclean -R` with the supplied label and method in order to verify that the volume is empty. This is followed by a call to `migreg` if the volume is empty.

After a volume is reregistered, any data previously stored on it can no longer be recovered.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

OPTIONS

The following options and parameters are available with `migrecycle`:

`-v` *volset* Volume set number used to reregister this volume. If not specified, the volume set number is not changed.

`-P` *poolname*
The name of the volume pool if other than the default, `HSM`.

`-d` *device* For recycling `ad` method volumes, this is the path to the file system. For recycling `ft` method volumes, this is the file system path on the remote volume server.

`-c` *capacity*
: Capacity of the remote file system for the `ft` method.

`-s` *server*
: Name of the remote volume server system for `ft` method.

`-u` *user*
: For the `ft` method, this is the valid `ftp` user name on the remote volume server system.

`-l` *new_label*
: Label used to reregister the volume.

`-h`
: Print help information.

hsmname
: HSMDEV entry containing the path to the database that will contain an entry for the volume you are recycling. For more information on HSMDEV entries, see `migconfg(1M)`.

label
: This is the name of the label that VSM assigns to the remote file system for identification. The label is a single line of text in a file named ID_LABEL that is stored in the remote file system.

method
: Volume method name. It can be `ct`, `dt`, `mt`, `ad`, `op`, or `ft`. Volumes for the `ow` and `nb` methods cannot be recycled.

Note If the method is `ft`, a prompt is issued for the password when the volume is reregistered.

CAVEATS

- When recycling an `ft` volume, the remote file system must be available so the ID_LABEL file, if present, can be read.
- If one side of a rewritable optical disk (method name op) is empty and the other contains granules, you can recycle the empty volume but not change its attributes, and the volume set number remains the same for both sides.

EXAMPLE

In this example, an `mt` volume EXB001 is reregistered in volume set 1.

```
migrecycle -v 1 alpha EXB001 mt
```

FILES

dwpath`/database/VOLDB`

Volume database.

dwpath`/database/migconf`

Configuration file.

SEE ALSO

HSM(1M), migreg(1M), migmdclean(1M)

migreg(1M)

NAME

migreg – Register and label volumes for VSM

SYNOPSIS

```
/usr/openv/hsm/bin/migreg [-F] [-D] [-P poolname] hsmname
        methname volume_set_number volume_name [volume_name]...

/usr/openv/hsm/bin/migreg [-F] hsmname ad volume_set_number
        mount_point volume_name [mount_point volume_name]...

/usr/openv/hsm/bin/migreg [-F] [-u user [-p password]]
        hsmname ft volume_set_number
        server_name server_directory capacity volume_name
        [server_name server_directory capacity volume_name]...

/usr/openv/hsm/bin/migreg [-F] hsmname nb
        volume_set_number class_name NB_master NB_client
        schedule capacity volume_name
        [schedule capacity volume_name]...
```

DESCRIPTION

The `migreg` command allows you to register and label volumes for VSM.

For local secondary storage, you can register tapes for use by method names `ct`, `dt`, and `mt`, or optical disk surfaces for use by method name `op` or `ow`. You can also register disk partitions as alternate magnetic disk volumes for use with method name `ad`.

Note When using `migreg` to register and label new, unused tape volumes, it is normal behavior to receive some routine error messages.

For remote secondary storage, you can register remote file system directory names for use by method name `ft`. You can also register disk partitions as alternate magnetic disk volumes for use with method name `ad` on remote systems. You can register a NetBackup class as a volume for use by method name `nb`.

Note The `migreg` command does not register volumes for method name `dk`, which is used only for premigration.

`migreg` registers all volumes in the VSM volume database, *dwpath*`/database/VOLDB` for the *hsmname*. After registration, VSM can use these volumes on subsequent migrations requiring the corresponding method name associated with each volume.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

OPTIONS

The following options and parameters are available with `migreg`:

Caution Use the `-F` option with extreme caution, as it may prevent access to files already migrated to the volume.

`-F` Forces the relabeling of a previously labeled volume. The volume must not be registered currently in any VSM volume database. If necessary, you can use `migmdclean -R` to remove an entry from a VOLDB. You may also use `migsetdb` to set an empty volume to `dead` in a VOLDB so it can be removed by `migrc` and relabled by `migreg`. If `-F` is not specified, VSM returns an error if you attempt to register a volume that has already been labeled.

`-D` Delay labeling of tape or optical disk until needed. This argument affects only volumes registered for the `ct`, `dt`, `mt`, `op`, and `ow` method names. If specified for other method names, VSM posts a message and then proceeds with the normal registration process. If `-D` is not specified, the volume is labled at the time `migreg` is run.

`-P` *poolname*
The name of the volume pool if other than the default, `HSM`.

`-u` *user* User name used for the `ftp` login request when VSM migrates files or caches files from the server with method name `ft`. If not supplied on the command line, it is read from stdin. If stdin is a tty, a prompt is issued.

`-p` *password*
Password used for the `ftp` login request with method name `ft`. It must be a valid password for *user* on the remote volume server. If not supplied on the command line, it is read from stdin. If stdin is a tty, a prompt is issued.

hsmname HSMDEV entry that specifies the path to the database in which you want to record the volume information. See `migconfg(1M)` for more information on HSMDEV entries.

methname
The method name under which this volume will be recorded. Valid values are `ct`, `dt`, `mt`, `op`, and `ow`. Any method name used with `migreg` (including `ad`, `ft`, and `nb`) must be defined in the *dwpath*`/database/migconf` configuration file for *hsmname*.

volume_set_number

The integer number of the volume set of which this volume is a part. Setting the *volume_set_number* parameter to `0` causes the volume to be assigned to any volume set with the same method when it is needed for writing.

Note A unique volume set is denoted by *both* the method name and volume set number. (Examples are ct.1, ct.2, dt.1, and op.2.) Volume sets provide a means within VSM of specifying the volumes on which a particular copy of a migrated file is placed. This is useful when a library device is being used to manage your migrations.

volume_name

The name of the volume to be recorded on the volume and in the volume database VOLDB. For NetBackup volumes, the *volume_name* is only recorded in the VOLDB, not on the volume. VSM restricts volume names to an alphabetic character followed by up to five alphanumeric characters, and converts all lower case input to upper case.

mount_point

The filesystem mount point required when registering a volume for use with method name `ad`. Make sure the filesystem is mounted before registering the volume.

server_name

Name of the remote volume server for `ft` volumes. This can be the internet id or number of the server. VSM uses this name on the `ftp open` command as the host parameter. It must be a valid `ftp` host.

server_directory

Absolute pathname of a directory on the host identified by *server_name* for use by method name `ft`. The *user* must have read and write permissions to this directory. This can be any directory on the remote volume server identified by *server_name* that is not already registered for VSM.

capacity Capacity, in bytes, of the remote file system or NetBackup volume that is available for VSM to use for storing migrated files. A value of 0 is interpreted as unlimited storage capacity.

class_name

The name of the NetBackup class.

NB_master

Name of the master NetBackup server for `nb` volumes. *NB_master* must be the first name the server is known by to NetBackup.

NB_client Name used for `nb` volumes to register the client under NetBackup. It must be the name used to register the client in the NetBackup class.

schedule The name of the NetBackup schedule.

CAVEATS

- Using the `-F` option to force write a new label destroys all information that may be on the volume.
- Always use the `–F` option if the media is already labeled and you wish to reuse the volume.
- The device-manager daemon `ltid` must be running if you are registering `ct`, `dt`, `mt`, `op` or `ow` method names.
- When you register one side of an optical disk, VSM also automatically registers the other side with the same volume set number. This avoids deadlocks during volume consolidation and when moving files between migration levels.
- If migrating two copies of a file, it is good practice to migrate copy 1 and copy 2 to different volume sets, such as to ct.1 and ct.2 or to ct.1 and dt.1. This avoids the chance of losing both copies when a volume is damaged or lost.
- To specify a password on the command line with the `-p` parameter is less secure that to let the command line prompt for input.

EXAMPLES

The following example for hsmname alpha registers a dlt tape (method name `dt`) with the name ARC001, and assigns it to volume set number 2.

```
migreg alpha dt 2 ARC001
```

The following example for hsmname alpha registers three cartridge tapes (method name `ct`) with the names CT001, CT002 and CT003, and assigns them to volume set number 0. Later, when a different `ct` volume set needs another volume, VSM can assign this tape to that volume set and re-register it with the new volume set number. Tapes are labeled when they are needed.

```
migreg -D alpha ct 0 CT001 CT002 CT003
```

The following example for hsmname alpha registers two disk partitions on device names `/dev/dsk01` and `/dev/dsk02` as alternate magnetic disk volumes (method name `ad`) with the names ARC900 and ARC901, and assigns them to volume set number 4.

```
migreg alpha ad 4 /dev/dsk01 ARC900 /dev/dsk02 ARC901
```


The following example for hsmname alpha attempts to register two sides of optical disk AB001, but cannot do so because the volume on side B of the platter is already registered. (VSM had registered side B automatically the first time it had registered side A.)

```
migreg alpha op 0 AB001A AB001B
Tape label: AB001B already registered to HSM
```

The following example for hsmname alpha registers a remote file system directory (method name `ft`) at pathname `/sdisk3` on server daneel with the name FTD001, and assigns it to volume set number 1 with a capacity of 600000000 bytes. The user name is fred and the password is HsMk2$.

```
migreg -u fred -p HsMk2$ alpha ft 1 daneel /sdisk3 600000000 FTD001
```

The following example for hsmname alpha registers NetBackup class hsmnb on NetBackup server duo as a NetBackup volume (method name `nb`), and assigns it to volume set number 1 with NetBackup schedule sk01 and infinite capacity (0).

```
migreg alpha nb 1 hsmnb duo duo sk01 0 NB001
Capacity set to unlimited.
Registered vh_label=NB001 [at hsmnb] as vh_handle=104A and
method= nb
```

FILES

dwpath`/database/VOLDB`

Volume database.

dwpath`/database/migconf`

Configuration file.

SEE ALSO

`HSM(1M), migbatch(1M), migconf(1M), migconfg(1M), migdbrpt(1M), migmdclean(1M), mignospace(1M), migsetdb(1M)`

migselect(1M)

NAME

migselect – select volumes for consolidation

SYNOPSIS

`/usr/openv/hsm/bin/migselect [-o]` *hsmname*
low_high volume_set method

DESCRIPTION

`migselect` selects a set of volumes for consolidation according to the percentage of volume usage. The percentage occupancy can have low-range and high-range values. This utility provides only a model in selecting the volumes and a site can modify it to meet specific requirements.

The output is of the form *label.method.volume_set*. `migcons` accepts this format in a list of input volumes to consolidate.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

Only the system administrator may use this utility.

OPTIONS

`-o` Causes `migselect` to base its selection on the percentage of a full volume that is obsolete. If this parameter is not specified, the selection is based upon the percentage of a full volume that is both active and obsolete.

hsmname HSMDEV entry that specifies the path to the database containing information on these volumes. See `migconfg(1M)` for more information on HSMDEV entries. This parameter is required.

low_high Volume use expressed as limits of low and high percent (for example, 20.05-30.75). This parameter is required.

volume_set
The name of the volume set to use. This parameter is required.

method Method name under which `migselect` selects volumes. Allowed values for *method* are `ad`, `ct`, `dt`, `mt`, `op`, or `ow`. This parameter is required.

CAVEATS

- This command does not select volumes currently selected for writing.
- This command ignores dead volumes.
- Write overhead on the volume is not considered in choosing the low and high range values for percent occupancy.

EXAMPLE

The following example selects volumes in `ct` volume set 1 and `ad` volume set 1, that have percentage occupancy between 1 and 10 percent.

```
migselect  alpha 01.00-10.00 1 ct ad
RAO001.ct.1
GLS002.ad.1
```

The following example selects cartridge tape volumes less than 20 percent full that are on volume set 2. The output of `migselect` is used as input to consolidation (see `migcons(1M)`).

```
migcons  alpha one  ct 2 'migselect  alpha 0.00-20.00 2 ct'
```

The following example selects volumes in `op` volume set 1, that are at least 50 percent obsolete and consolidates them to `op` volume set 1.

```
migcons alpha one op 1 'migselect alpha  -o  50.0-100 1 op'
```

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath`/database/VOLDB`

Volume database for VSM.

dwpath`/database/FHDB`

File-handle database for VSM.

`/usr/var/openv/hsm/database/migconfg`

Global-configuration file for VSM

SEE ALSO

`migconf(1M), migcons(1M), miggetvol(1M), migmdclean(1M), migrecycle(1M)`

migsetdb(1M)

NAME

migsetdb - allows you to alter the flags field in the FHDB or the VOLDB

SYNOPSIS

```
/usr/openv/hsm/bin/migsetdb -F [-a | d | O]
          [-m methname] -i worklist_file [-s setlevel] hsmname

/usr/openv/hsm/bin/migsetdb -F [-a | d | O]
          [-m methname] [-M machid] [-s setlevel]
          hsmname handle [handle]...

/usr/openv/hsm/bin/migsetdb -V
          [-a | c | d | e | f | l | w | R] hsmname label [label]...

/usr/openv/hsm/bin/migsetdb -V [-u new_user]
          [-p] hsmname label [label]...

/usr/openv/hsm/bin/migsetdb -V [-u new_user]
          [-P password] hsmname label [label]...

/usr/openv/hsm/bin/migsetdb -V -U user
          [-P password] hsmname
```

DESCRIPTION

This command is normally used by VSM to select and change the state of file handle database (FHDB) and volume database (VOLDB) entries automatically. Administrators can also use `migsetdb` to fix inconsistencies not corrected by `migdbcheck`.

A `migsetdb` command can perform one of the following functions:

- It may be run by the administrator to mark FHDB entries in a supplied worklist or file handle at a specific migration level as `active`, `obsolete` or `dead`. Finer selectivity is achieved by also specifying method name or machine ID.
- It may be called by `migmove` to mark all FHDB entries in the supplied worklist file as `active`, `obsolete` or `dead`.
- It may be used by the administrator to alter the flags field in the VOLDB to any valid value for the label(s) specified, or to change the password, user name, or both for the volumes that match the label(s) specified.
- It may be used by the administrator to change all passwords for individual users in the VOLDB.

`migsetdb` creates the file `/tmp/migsetdb.`*`pid`* when changing flags on FHDB entries. For changing users or passwords in the VOLDB, `migsetdb` creates the file `/tmp/migsetdb-tmp-voldb.`*`pid`*. However, `migsetdb` checks the environment variable TMPDIR, which allows the administrator to use a path other than the default `/tmp`. This can save files through system reboots or make use of a larger file system to avoid running out of space. The path defined by TMPDIR, if set, is used instead of `/tmp` as the directory in which to place any temporary files. The process ID of the `migsetdb` that is executing replaces *pid*. No files should be left after successful execution.

Note The FHDB and VOLDB in general and the flags field of FHDB and VOLDB entries in particular are discussed in the System Administrator's Guide for VSM.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

OPTIONS

Caution Inappropriate use of `migsetdb` may make migrated files inaccessible or make the FHDB inconsistent.

`-F` Make changes to the filehandle database, FHDB.

`-V` Make changes to the volume database, VOLDB.

`-a` Set the flags field of the specified FHDB or VOLDB entry to `active` (or 0). This is the default for the VOLDB when neither `-a` nor `-d` are specified. Default for the FHDB when neither `-a` nor `-d` nor `-O` are specified is the condition specified in `migconf`.

`-c` Set the corrupt flag field of the specified VOLDB entry.

Note VSM does not automatically set the corrupt flag. It must be set by `migsetdb`.

`-d` Set the flags field of the specified FHDB or VOLDB entry to `dead`.

`-O` Set the flags field of the specified FHDB entry to `obsolete`.

`-e` Set the flags field of the specified VOLDB entry to `empty`.

`-f` Set the flags field of the specified VOLDB entry to `full`.

`-l` Set the flags field of the specified VOLDB entry to NEEDLBL, indicating that this VOLDB entry needs to be labeled.

`-w` Set the flags field of the specified VOLDB entry to `write`.

`-R` Set the flags field of the specified VOLDB entry to NDLBLFRC, indicating that this VOLDB entry needs to be force-labeled.

`-m` *methname*

Set the flags field for FHDB entries with migration level *setlevel* only if they match method name *methname* and either *handle* or the files specified in *worklist_file*. These method names must be defined in the *dwpath*`/database/migconf` configuration file for *hsmname*. When `-m` is not specified, entries at *setlevel* will be modified without regard to *methname*. Specifying `-M` *machid* further restricts which FHDB entries are modified.

`-i` *worklist_file*

Set the flags field for FHDB entries with migration level *setlevel* only for the files specified in *worklist_file*. The format of each line of *worklist_file* is as follows:

handle | *machid* | *lock* | *flags* | *volset* | *copied* | *method* | *dest_seek* | *badness* | *size* | *age* | *path* | *hint* | *comment* | *mmlevel* | *slevel* |

Do not set the flags field for FHDB entries with migration level *mmlevel*, where *mmlevel* is an integer in the range 1 to 8. FHDB flags at *mmlevel* will not be obsolete.

`-s` *setlevel*

The migration level for which to set FHDB flags, where *setlevel* is an integer in the range 0 to 8. VSM logs and outputs a warning message if *setlevel* is greater than 8. Specifying `-m` *methname* or `-M` *machid* further restricts which FHDB entries are modified. If `-s` *setlevel* is not specified, all levels except 0(dk) are set.

`-M` *machid*

Set the flags field for FHDB entries with migration level *setlevel* only if they match machine ID *machid* and *handle*. When `-M` is not specified, entries at *setlevel* are modified without regard to *machid* only if they match *handle* and all entries for a particular *handle* have the same machine ID. VSM skips entries at *setlevel* that match a particular *handle* but have more than one machine ID, and logs an error message advising you to execute `migsetdb` with the `-M` option in order to set the flags field for the *handle* in question.. Specifying `-m` *methname* further restricts which FHDB entries are modified.

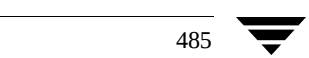
hsmname The HSMDEV entry that specifies the path to the database files you want to alter. See `migconf(1M)` for more information on HSMDEV entries.

handle Set the flags field for FHDB entries with migration level *setlevel* only for the files with file handle *handle*.

label The volume label of the entry to change in the VOLDB.

`-u` *new_user*

(Lower case ‘u’) Change the user name to *new_user* for each entry with the volume label specified by *label*.



`-U` *user* (Upper case 'U') Change the password at the prompt for all VOLDB entries that match the user name specified by *user*.

`-p` (Lower case 'p') Change the password at the prompt(s) for each entry with the volume label specified by *label*.

`-P` *password*
(Upper case 'P') Change the password as specified by the *password* variable on the command line, rather than at the prompt.

CAVEATS

- Using the lower case 'p' option to change the password at a prompt is generally more secure than using the upper case 'P' option and specifying the password as part of the command line.
- It is possible to obsolete all entries in the FHDB for any file. This could cause them to be deleted when `migrc` is run to consolidate the FHDB, making the files inaccessible.
- The `migrc` command removes all FHDB and VOLDB entries marked dead.
- Do not mark entries for method name `ad` or `ft` `dead`; mark them `obsolete` instead. This prevents them from being incorrectly removed by `migrc`. These entries are correctly removed from the FHDB by `migmdclean` as it cleans the disk.

EXAMPLES

The following example for hsmname alpha sets the FHDB flags field to the condition specified in the configuration file `migconf` for all entries with a migration level of 6.

```
migsetdb -F -s 6 alpha
```

The following example for hsmname alpha sets the FHDB flags field to `dead` for all entries with method name `ct` and a migration level of 1.

```
migsetdb -Fd -m ct -s 1 alpha
```

The following example for hsmname alpha sets the FHDB flags field to `obsolete` for all files specified in the worklist file wlst0001 with a migration level of 3 and a mmlevel other than 3.

```
migsetdb -FO -i wlst0001 -s 3 alpha
```

The following example for hsmname alpha sets the VOLDB flags field to `dead` for all entries with label FT001.

```
migsetdb -Vd alpha FT001
```

In this case, the hsm log would show a record resembling this:

```
03/31 13:56:15 migsetdb[8506]: VOLDB label: FT001 flags=0010
```

indicating the flags are set to `dead`.

Similarly, the following example for hsmname alpha clears the VOLDB flags field for all entries with label FT001.

```
migsetdb -Va alpha FT001
```

In the following example for hsmname alpha, `migsetdb` is called by `migmove`.

```
migmove -s 5 alpha
```

```
In this case, the hsm log would show records resembling these:
   03/11 14:35:49 migsetdb[28010]: fh=3e8Me1035 level=5: flags=08
   03/11 14:35:50 migsetdb[28010]: fh=3e8Me1036 level=5: flags=08
   03/11 14:35:50 migsetdb[28010]: fh=3e8Me1037 level=5: flags=08
```

indicating the flags are set to `dead`.

The following example for hsmname alpha sets the new user name to name01 and prompts for a new password for all entries with label FT001 or FT002.

```
migsetdb -Vp -u name01 alpha FT001 FT002
```

A password is requested and confirmed for each specified volume.

```
Enter the password for label: FT001 user:name01
Reenter the password for label: FT001 user:name01
Enter the password for label: FT002 user:name01
Reenter the password for label: FT002 user:name01
```

These volumes now list name01 as the new user name in the VOLDB.

The following example for hsmname alpha prompts once for a new password for user name fred.

```
migsetdb -V -U fred alpha
```

To change many volumes to a new owner with a single password, you can use the previous two forms of this command in combination. In the following example for hsmname alpha, the first instance sets the new user name to jane for all entries with the specified labels, and the second instance prompts once for jane's new password.

```
migsetdb -V -u jane alpha VOL1 VOL2 VOL3 VOL4 VOL5 VOL6 VOL7
migsetdb -V -U jane alpha
```

This avoids repetitive password prompts for each reassigned volume.

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath`/database/VOLDB`

Volume database

dwpath`/database/FHDB`

File-handle database

dwpath`/database/migconf`

VSM configuration file for managed file systems

`/usr/var/openv/hsm/database/migconfg`

VSM global-configuration file

`/tmp/hsm.log`

VSM global-log file

Note The term *lgpath* refers to the path name of the log file for an HSMDEV entry. The default is `/tmp/hsm.`*hsmname*`.log`, where *hsmname* is the value you supply for `hsmname`. The path name of this log file is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

lgpath`/hsm.`*hsmname*`.log`

VSM log file

SEE ALSO

`HSM(1M), migconf(1M), migconfg(1M), migmove(1M), migdbcheck(1M)`

migstage(1)

NAME

migstage - cache one or more files in advance

SYNOPSIS

`/usr/openv/hsm/bin/migstage [-w]` *pathname* [*pathname* `...]`

`/usr/openv/hsm/bin/migstage [-w] -f` *filelist*

DESCRIPTION

`migstage` examines a set of files and caches them back to disk. Optionally, `migstage` will wait until all of the files have been cached before exiting.

This command lets you cache one or more migrated files before you need to access them in order to avoid caching delays at the time of access.

The specified files are cached in their entirety, even when partial file caching is enabled.

On nonkernel-based implementations of VSM, `migstage` attempts to optimize the reload operation by grouping all of the files for a given *volume ID* and *hsmname* together. There is no optimization on kernel-based implementations, and `migstage` simply causes each file to be cached back by reading the last byte of the file.

OPTIONS

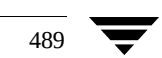
`-w` Specifies that `migstage` is to wait for all of the files to be cached back to disk before exiting. The default is to initiate caching and then exit while caching operations continue in the background.

pathname Files or directories that `migstage` should pre-cache. If a directory is specified, all files in the directory and its subdirectories are cached. May be given as a relative path or absolute path. You can specify multiple files or directories, or use standard regular expressions. Wildcards are recognized.

`-f` *filelist* Pre-cache the files listed in *filelist*. The format of *filelist* is one file name per line, showing either the full pathname of each file or a pathname relative to the current directory in which `migstage` is executed, not the directory in which the file in the *filelist* resides. Wildcards are not recognized.

CAVEATS

- The VSM state must be active and the migration daemon `migd` must be running when you issue a `migstage` command.



EXAMPLES

The following command pre-caches all files in the current directory and its subdirectories. The directory does not need to be a grouped directory.

```
migstage .
```

The command exits quickly while caching continues in the background.

The following command waits for all caching to complete before exiting.

```
migstage -w .
```

The following command pre-caches files `f1` and `f2`.

```
migstage -f stagelist
```

where the file stagelist reads as follows:

```
/usr/mydir/subdir/f1
/usr/yourdir/subdir/f2
```

The following command pre-caches files `f4`, `f5`, and `f6`.

```
migstage f4 f5 f6
```

The following command pre-caches all files whose file names begin with the string `June_23`.

```
migstage June_23*
```

SEE ALSO

`HSM(1M), fls(1), miggroup(1), migungroup(1), migloc(1), migtie(1)`

migtestbadness(1M)

NAME

migtestbadness - allows you to check what effect changing the badness computation parameters has on migration operations

SYNOPSIS

```
/usr/openv/hsm/bin/migtestbadness hsmname pathname
          [badness] [min_age] [min_size] [age_weight]
          [size_weight] [weight_operator 0=X, 1=+, 2=site]
          [usage 0=ffs, 2=hsm] [lowwater 0=none]
          [kbytes_to_reduce 0=find all]
          [test_mode 1=print headers 2=no headers]
          [number_of_files 0=no limit]
```

DESCRIPTION

`migtestbadness` gives administrators a way to assess the results of configuring different values for badness, minimum file age and file size, weighting factors and weight operators to determine the amount of free space that could be achieved during a typical migration operation. Each test of a file system lists the files whose calculated badness exceeds the specified badness, and computes the total filespace that could be freed up by this operation. This total filespace is recomputed each time `migtestbadness` is run with different values for the file system attributes. By testing different file systems in this way, the administrator is able to choose the optimum values to use in configuring each file system.

Note The values for each `migtestbadness` option can be customized. You can view these settings in the `migconfg` global-configuration file.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

OPTIONS

hsmname The hsm name under which the swept file system is defined.

pathname The directory to sweep. It should be specified as a full absolute path.

badness The badness criteria to be used in this test. Files whose `badness` factor is greater than this value are identified. This is a user-configurable value. If this value is not configured, the default is 30. Badness is computed using either the following VSM formula or one specified by the site. (See *weight_operator*).

badness=*age_weight* x (age of file in days)
+ or x (as specified by *weight_operator*)
size_weight x (size of file in Kbytes)

Where file `age` is the number of days since the last access or since last modification, whichever is greater.

min_age Age of file (in days). Only include files that have neither been accessed nor modified within this time. This is a user-configurable value. If this value is not configured, the default is 1.

min_size Size of file (in kilobytes). Only include files that are at least this large. This is a user-configurable value. If this value is not configured, the default is 1.

age_weight
Weighting to be used for age in the badness factor. This is a user-configurable value. If this value is not configured, the default is 1.

size_weight
Weighting to be used for size in the badness factor. This is a user-configurable value. If this value is not configured, the default is 1.

weight_operator
The operator to be used in calculating the `badness` factor. This is a user-configurable value. If this value is not configured, the default is 0:

0 = multiply
1 = add
2 = site selected

If 2 (site selected) is used, `migsweep.site.sh` is called.

usage File system type. This is a user-configurable value. If this value is not configured, the default is 0:

0 = ffs
2 = hsm

lowwater To identify all available files, set *lowwater*=0. If *lowwater* does not equal 0, you must specify *kilobytes_to_reduce* to stop this test. This is a user-configurable value. If this value is not configured, the default is 0.

Note If *lowwater* is larger than 0, you must also set the *kbytes_to_reduce* value.

kbytes_to_reduce
The maximum amount of free space to be found before stopping this test. This value is valid and must be specified if *lowwater* does not equal 0. If *lowwater*=0, this value, if specified, is ignored.

Note You must set the *kbytes_to_reduce* value if *lowwater* is larger than 0.

test_mode List files to output. This is a user-configurable value. If this value is not configured, the default is 1:

1 = print header messages
2 = don't print headers

number_of_files

The maximum number of files to be found before stopping this test. This is a user-configurable value. If this value is not configured, the default is 0:

0 = no limit

EXAMPLES

1. In the first test, *badness*=1, and *kbytes_to_reduce*=4000:

```
migtestbadness hsm1_6 /hsm1_6 1 1 1 1 1 1 0 9 4000
   age + size = badness  pathname
   ---   ----   -------  --------
   1.10   1025      1026 /hsm1_6/test1/retrieve_file.1.6713
   1.11   1025      1026 /hsm1_6/test1/retrieve_file.1.7813
   9.04     10        19 /hsm1_6/test1/1
   9.04     10        19 /hsm1_6/test1/2
   2.10    100       102 /hsm1_6/test1/filler.93.7813
   2.10    100       102 /hsm1_6/test1/filler.94.7813
   2.10    100       102 /hsm1_6/test1/filler.95.7813
   2.10    100       102 /hsm1_6/test1/filler.96.7813
   2.10    100       102 /hsm1_6/test1/filler.97.7813
   1.11   1025      1026 /hsm1_6/test1/retrieve_file.1.9805
   2.10    100       102 /hsm1_6/test1/filler.15.9805
   2.10    100       102 /hsm1_6/test1/filler.16.9805
   2.10    100       102 /hsm1_6/test1/filler.17.9805
   2.10    100       102 /hsm1_6/test1/filler.18.9805
   2.10    100       102 /hsm1_6/test1/filler.19.9805
   2.10     10        12 /hsm1_6/test1/filler.20.9805
   2.10     10        12 /hsm1_6/test1/filler.21.9805
   2.10     10        12 /hsm1_6/test1/filler.22.9805
   2.10     10        12 /hsm1_6/test1/filler.23.9805
```


```
   2.10     10        12 /hsm1_6/test1/filler.24.9805
   2.10    100       102 /hsm1_6/test1/filler.9.10960
migsweep finished: 21 files, 4077 Kbytes 3 migrated files 3051 Kbytes
```

2. In the second test, badness=30, min_age=2, and kbytes_to_reduce=400:

```
migtestbadness hsm1_6 /hsm1_6 30 2 1 1 1 1 0 9 400
   age + size = badness  pathname
   ---   ----   -------  --------
   2.10    100       102 /hsm1_6/test1/filler.93.7813
   2.10    100       102 /hsm1_6/test1/filler.94.7813
   2.10    100       102 /hsm1_6/test1/filler.95.7813
   2.10    100       102 /hsm1_6/test1/filler.96.7813
   2.10    100       102 /hsm1_6/test1/filler.97.7813
migsweep finished: 5 files, 460 Kbytes 0 migrated files 0 Kbytes
```

Note Kbytes=460 not 500 (100x5). The reason for this is that hsm1_6 is set up with a slice value of 8K. This means actual disk savings are 92x5=460 Kbytes.

3. The third test is the same as the second test, but no headers are printed:

```
migtestbadness hsm1_6 /hsm1_6 30 2 1 1 1 1 0 9 400 2
   2.10    100       102 /hsm1_6/test1/filler.93.7813
   2.10    100       102 /hsm1_6/test1/filler.94.7813
   2.10    100       102 /hsm1_6/test1/filler.95.7813
   2.10    100       102 /hsm1_6/test1/filler.96.7813
   2.10    100       102 /hsm1_6/test1/filler.97.7813
```

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath`/database/migconf`

VSM configuration file for managed file systems

`/usr/var/openv/hsm/database/migconfg`

VSM global-configuration file.

SEE ALSO

HSM(1M), migconf(1M)

migthreshold(1M)

NAME

migthreshold – set or display the file system threshold value

SYNOPSIS

```
/usr/openv/hsm/bin/migthreshold [-d | -f |
        -p percentage | -v number | -h] hsmname
```

AVAILABILITY

This command is only available on kernel-based implementations.

DESCRIPTION

The `migthreshold` command may be used by the system administrator to set or display the file system threshold value of a configured file system.

Each time migd is started, it sets the threshold value for all configured file systems based on the freespace parameter setting in the configuration file. A freespace value of zero does not change the current file system threshold.

The administrator can subsequently use `migthreshold` to:

- Display the current threshold setting.
- Set the threshold to a specific value in kilobytes.
- Set the threshold specific percentage.
- Reset threshold to the *percentage* specified in the `migconf` file.

OPTIONS

You can specify one (and only one) of the following options when you execute this command. The default is `-d` (display threshold).

`-d` Display the current threshold value in kilobytes for the *hsmname* file system. This is the default.

`-f` Force the threshold value for *hsmname* file system to the *percentage* specified by the freespace parameter in the *dwpath*`/datapath/migconf` file.

`-p` *percentage*
Set the threshold value for *hsmname* file system to the specified *percentage*. For example:

```
migthreshold -p 20 alpha
```

Do not enter a percent sign (%).

`-v` *number*
: Set the threshold value for *hsmname* file system to the specified *number* in kilobytes.

`-h`
: Display help.

hsmname
: HSMDEV entry that specifies the file system for which you want to set the threshold. See `migconfg(1M)` for more information on HSMDEV entries. This parameter is required.

CAVEATS

- Increasing the threshold may cause VSM to migrate files the next time space is needed in *hsmname* file system.
- Resetting the threshold does not affect the operation of `miglow`, which continues to initiate migration when free space is below the `freespace` percentage specified in the *dwpath*`/database/migconf` file.
- The hsm state must be active and the migration daemon `migd` must be running when you issue a `migthreshold` command.
- Saving the global-configuration file from the `xhsmadm` GUI causes the threshold to be set to the freespace parameter value in the configuration file except when freespace is 0, in which case the threshold remains unchanged.
- Some systems reserve a certain percentage of the file system for superusers only. The default reserve value on such systems is 10%. The migthreshold command takes this into account. The file system threshold is computed based on total capacity less the reserve space.

EXAMPLE

This file system includes 10% of file space reserved for superusers:

```
Filesystem          kbytes     used   avail capacity  Mounted on
/dev/md/dsk/d1    4278217  3850395       0   100%    /hsm01c
```

Execute `migthreshold -f` to reset the threshold to equal the freespace parameter in `migconf`, which is set at 2% in this example:

```
(delux):root 81% migthreshold -f hsm01c
Current threshold: 77007 Kbytes for hsm01c
/hsm01c/hsm total 3850395 Kbytes, freespace wanted 2%
New threshold: 77007 Kbytes for hsm01c
```

where 77007 is 2% of 3850395 Kbytes.

To set the threshold to 50%, use the `-p` option:

```
(delux):root 82% migthreshold -p 50 hsm01c
Current threshold: 77007 Kbytes for hsm01c
New threshold: 1925197 Kbytes for hsm01c
```

where 1925197 is 50% of 3850395 Kbytes. In this case, the threshold setting no longer equals the configured freespace value.

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

`/usr/var/openv/hsm/database/migconfg`

VSM global-configuration file.

dwpath`/database/migconf`

VSM configuration file for managed file systems

SEE ALSO

`HSM(1M), migconf(1M), migconfg(1M), miglow(1M)`

migtie(1)

NAME

migtie – designate the key caching file(s) for a caching group

SYNOPSIS

```
/usr/openv/hsm/bin/migtie -a | -d groupname
          file_path [file_path] . . .

/usr/openv/hsm/bin/migtie -a | -d -h hsmname
          groupname file_handle [file_handle] . . .

/usr/openv/hsm/bin/migtie [-l] hsmname groupname
```

DESCRIPTION

The `migtie` command allows you to add, delete, or list *key caching files* for a caching group. A *caching group* is a list of files to be cached together. A *caching group file* defines this list.

1. Create a caching group file before using `migtie`.

If the caching group file is a zero length file, every file in the directory containing the caching group file will be cached back along with every file in all of its subdirectories.

If the caching group file contains pathnames relative to the current directory, it will cache all files and directories specified in the caching group file. If a directory is cached, every file within that directory and and all of its subdirectories will be cached. Wildcard characters are recognized.

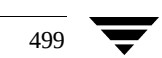
If the caching group file contains full file pathnames, every file in the caching group will be cached back. Wildcard characters are not recognized. Directory names are ignored.

Key caching files may but need not be listed in the caching group file.

The name of the caching group file is *.groupname*. Locate the *.groupname* file at a level in the managed file system higher than or equal to all of the key caching files to which it is tied, and include it in the user's `.migstop` file to prevent its migration.

2. Choose which files you will designate as key caching files. Use the `migtie` command to tie one or more of these key caching files to a caching group file.

Users can use `migtie` to add, delete, or list only those key caching files which they own. The command exits if it encounters a key caching file not owned by the command user.



Whenever a key caching file is cached, all files listed in the caching group file tied to it are also cached. The key caching file and all files listed in the caching group file are cached in their entirety, even when partial file caching is enabled.

The system administrator must provide the proper permissions before users can use this command. See "Enable User Permissions" on page 156.

This command may be run without regard to whether the migration daemon `migd` is running, but the hsm state must be active.

OPTIONS

`-a` Add key caching files for a caching group.

`-d` Delete key caching files for a caching group.

`-l` List the key caching files for a caching group (default).

`-h` Identify key caching files by file handle rather than by file path.

groupname
Identifies the caching group file listing the files to be cached together. The *groupname* cannot exceed eleven characters.

hsmname HSMDEV entry that specifies the file system for `migtie`. See `migconfg(1M)` for more information on HSMDEV entries.

file_path Absolute or relative pathname of the key caching file(s).

file_handle File handle of the key caching file(s).

CAVEATS

- Creating a caching group file of zero length or one containing relative pathnames can increase caching activity unnecessarily if used in situations where it is not necessary to cache entire directories and their subdirectories at one time.
- When `migtie` designates a file to be a key caching file it modifies the comments field of the file's FHDB entry. Only files with a FHDB comments field containing "Auto VSM run" or "User selected" or a term with a dot "." prefix can be designated to be key caching files.
- If you choose a file to designate as a key caching file while it is still in premigration, and then cache that file before it has been fully migrated to secondary storage, the designation will be lost when the file is later remigrated. Although this is a rare occurrence, you can ensure that the designation will not be lost by using the `migloc` command to verify the migrated status of the file before either initially designating the file or caching it for the first time.
- If you cache the files in a caching group tied to a key caching file before all the copies of those files have been made, remigrating the files at a later time will create new file handles. Execute `migtie` again for each of these files.

- If you change the name or full path of any file listed in a caching group file, caching a key caching file tied to that caching group file will no longer cache the changed file unless you modify the caching group file accordingly.
- A reference only to the slice portion of a key caching file will not cause the files listed in the caching group file to be cached.
- On nonkernel-based implementations, if a key caching file has not been purged, caching it will not cause the files listed in the caching group file to be cached.
- If the key caching files are renamed after they are included in a caching group, they no longer work as key caching files.
- The `miggroup` command is implemented using a tie group named `MigGroup`. Using this same name in your own `migtie` command will interfere with any current or future use of `miggroup` in this file system.

EXAMPLE 1

Create a caching group file in `/home/user1/.month_tie` for the following caching group:

```
/home/usr1/monthly_project/run_project.1
/home/usr1/monthly_project/run_project.2
/home/usr1/monthly_project/data.1
/home/usr1/monthly_project/data.2
/home/usr1/quarterly_project/data
/home/boss/reports/monthly_project/user1/her_input
```

Designate key caching files for this caching group with the following command:

```
migtie -a month_tie /home/usr1/monthly_project/run_project.1 \
    /home/usr1/monthly_project/run_project.2
```

When the user executes either program `run_project.1` or `run_project.2`, VSM will automatically cache all of the files listed in `/home/usr1/.month_tie`. Normal read, write, and execute permissions pertain to these cached files.

If the user accesses only `/home/usr1/monthly_project/data.1`, which is not a key caching file, no other files would be cached automatically.

EXAMPLE 2

Create a caching group file in `/home/user1/.month_tie` for the following caching group:

```
monthly_project
quarterly_project/d*
```


Designate key caching files for this caching group with the following command:

```
migtie -a month_tie /home/usr1/monthly_project/run_project.?
```

When the user executes either program `run_project.1` or `run_project.2`, VSM will automatically cache all of the files listed in `/home/usr1/.month_tie`. This includes all files in the `/home/user1/monthly_project` directory and its subdirectories as well as any files or directories beginning with the character `d` in the `/home/user1/quarterly_project` directory.

EXAMPLE 3

Create a caching group file in `/home/user1/.quarter_tie` for the following caching group:

```
/home/usr1/monthly_project/data/output
/home/usr1/quarterly_project/data
/home/boss/reports/monthly_project/user1/her_input
```

Designate a key caching file for this caching group with the following command:

```
migtie -a quarter_tie /home/usr1/quarterly_project/run_project
```

When the user executes the program `run_project`, VSM will automatically cache all of the files listed in `/home/usr1/.quarter_tie`. Note that the key caching file does not need to be listed in `/home/usr1/.quarter_tie` for this to occur.

Note Files can appear in more than one caching group file, as shown in Examples 1 and 2. A key caching file can only be tied to one caching group file. A caching group file may list a key caching file for another caching group.

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath`/database/FHDB`

File-handle database for VSM

SEE ALSO

`HSM(1M), gethsm(1), migin(1M), miggroup(1), migungroup(1), migstage(1),`

migtscan(1M), migopscan(1M)

NAME

migtscan, migopscan – scan the tape or optical volume for file granules

SYNOPSIS

```
/usr/openv/hsm/bin/migtscan [-F] [-n] [-s] [-t]
        hsmname  volume_label  method

/usr/openv/hsm/bin/migopscan [-F] [-n] [-s]
        hsmname  volume_label  method
```

AVAILABILITY

These commands are not available with Storage Migrator Remote.

DESCRIPTION

`migtscan` and `migopscan` scan the specified tape (`ct`, `dt`, or `mt`) or optical platter (`op` or `ow`) *volume_label*. The resulting display contains information about the volume as a whole and also about each granule on the volume.

When VSM migrates a file to secondary storage, it writes the file as granules, with an entry in the VSM file-handle database (FHDB) for each granule. The granule size is configured for the method name (`ct`, `dt`, `mt`, `op`, or `ow`) in the VSM configuration file. Each granule on tape also contains FHDB and VOLDB entry information.

The `migtscan` command and `migopscan` command create two output files, FHDB.*label* and VOLDB.*label*, in the *dwpath*`/database` directory. The structure of these files is the same as the FHDB and VOLDB database files. These files may be used to rebuild the FHDB and VOLDB if they are corrupted or damaged (see `migdbcheck(1M)`).

You can sort and merge FHDB.*label* files for different volumes to recreate the FHDB. Similarly, merging VOLDB.*label* files for different volumes can recreate the VOLDB.

Note When recreating the VOLDB be sure to merge the VOLDB file in the `/usr/var/openv/hsm/example/database` directory to include the entry for the `dk` method.

FHDB entries for granules written prior to R2.0 might not contain the full pathname for the file and will not contain a valid migration level number. By default, `migtscan` and `migopscan` convert these FHDB entries to include full pathnames and correct level numbers.

Note These commands convert FHDB entries for granules written prior to R2.0 by using the copies parameter and storage methods in `migconf` to determine whether the level field in a new FHDB entry should be 1 or 2.

These commands may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

Only the system administrator may use these commands.

OPTIONS

`-F` Force scan the volume for VSM granules in case the volume identity is not in the VOLDB. This parameter is optional. If omitted, the volume must be registered in the VOLDB.

`-n` Do not convert FHDB entries for granules written prior to R2.0. You cannot reconstruct a usable database with this option, but it is useful if you want to examine what is actually written on the media.

`-s` Silently scan the volume and create FHDB.*label* and VOLDB.*label* files, but do not display information on `stdout`.

`-t` Use values of file and record numbers that were recorded on tape. When not specified, use actual values. The actual values may differ from the recorded values in some cases; the actual value is needed for VSM to cache files correctly .

hsmname HSMDEV entry that specifies the path to the database containing information on the desired volume. See `migconfg(1M)` for more information on HSMDEV entries.

volume_label
Tape volume label used when registering the volume. This parameter is required.

method Method name under which the volume is registered (`ct`, `dt`, or `mt` for `migtscan`, and `op` or `ow` for `migopscan`).

CAVEATS

- Results of scanning a tape volume that does not conform to VSM standards are unknown.

EXAMPLE

The following example uses method `mt` to scan a tape volume called *lostid*. The resulting display shows a list of granule information for files archived on the volume.

The `FHDB.lostid` and `VOLDB.lostid` files are created in the VSM database directory.

```
% migtscan alpha lostid mt

VOLUME LOSTID registered to HSM

VOLUME particulars ======>>

000003E8V00001098 LOSTID    mt 09600000 0072DB69 #23
--------------------------

000003E8M00001342 V00001098 mt 00005B8C 00000000 00005B8C
#1 1 #1 1 /home/ckb/ed

000003E8M00001343 V00001098 mt 00005B8C 00000000 00005B8C
#2 1 #2 1 /home/ckb/an
```

Volume information on the display includes (in order):

```
Volume handle

Volume label

Method

Total capacity of the volume

Total space in use on the volume

Number of files on the volume.
```

File granules information on the display includes (in order):

```
File handle

Volume ID

Method

File size

Offset of the granule in the file

Granule size

Recorded granule number

Recorded file number

Computed granule number

Computed file number

File name.
```


The recorded granule number is the granule number recorded on the tape, and should equal the computed granule number. The computed granule number is based on the scan of the tape. Similarly, the recorded file number is the file number recorded on the tape, and should equal the computed file number.

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath`/database/FHDB`

File-handle database for VSM

dwpath`/database/VOLDB`

Volume database for VSM

dwpath`/database/migconf`

VSM configuration file for managed file systems

dwpath`/database/FHDB.`*label*

File-handle database for current volume

dwpath`/database/VOLDB.`*label*

Volume database for current volume

`/usr/var/openv/hsm/database/migconfg`

VSM global-configuration file

SEE ALSO

`migdbcheck(1M), mediacheck(1M), migconf(1M), migconfg(1M), migdbrpt(1M), migreg(1M), migadscan(1M), migftscan(1M), mignbscan(1M)`

pfcheck(1M)

NAME

pfcheck - checks consistency of the VSM `.PAIN` (parallel inode) file.

SYNOPSIS

`/usr/openv/hsm/bin/pfcheck [-p] [-s]` *file_system*

AVAILABILITY

This command is only available on kernel-based implementations.

DESCRIPTION

`pfcheck` sweeps through the VSM managed file system and checks all files against the `.PAIN` file. A new file named `.PAIN.NEW` is created that the administrator can use to replace the old `.PAIN` file if there were problems.

The following is done when `pfcheck` sweeps the file system:

- If the inode number of the file is not in the `.PAIN` file, a warning is issued. If this happens, use `pfinit -e` to extend the `.PAIN.NEW` file.
- Each regular file is checked against the `.PAIN` file to see if:
  - Inode value in `.PAIN` matches the file inode number.
  - Version in `.PAIN` entry is correct.
  - Flags in `.PAIN` are correct. Only the migrated or cached bits may be set. If migrated and vmiglock bit is set, vmiglock is cleared.
  - Size of file equals size in `.PAIN` entry. This check occurs only if the file is migrated or has been cached without subsequent modification.
  - Slice value is less than or equal to the file size indicated in the `.PAIN` entry. The file must be migrated or be cached without subsequent modification to have a slice value (zero slice value is legal).
  - Modification date in `.PAIN` matches the file's modification date. The file must be migrated or cached and unmodified for the date value to be correct.
- If a file entry in the `.PAIN` file is not correct the `.PAIN.NEW` file entry is initialized.
- For files that are NOT regular files, the `.PAIN` file is checked for initialized entries. If the `.PAIN` file entry is not initialized, the `.PAIN.NEW` file entry is initialized.

- For `.PAIN` file entries that have no file, the contents of the `.PAIN.NEW` file entry are initialized.

Note An initialized `.PAIN.NEW` entry has the version and inode number filled in and flags set to unmigrated.

OPTIONS

`-p` Turns off prompting. In this mode, `pfcheck` provides messages pertaining to incorrect entries that it finds but does not prompt you before changing the `.PAIN.NEW` file entry.

`-s` Turns on silent mode. In this mode, `pfcheck` checks the file and makes necessary changes to the `.PAIN.NEW` file without showing you any prompts or messages.

file_system Specifies the absolute path to the managed file system to be checked. This path must match the one defined by the *fspath* parameter for the file system in the `/usr/var/hsm/database/migconfg` file. `pfcheck` checks the file system against the *fspath*`/.PAIN` file.

CAVEATS

- The file system must not be mounted as an `hsm` file system when you issue a `pfcheck` command.

EXAMPLES

This example checks the `.PAIN` file in `/home/.PAIN` and creates the `.PAIN.NEW` file in `/home/.PAIN.NEW`.

```
pfcheck /home
```

FILES

fspath`/.PAIN`

VSM parallel inode file

fspath`/.PAIN.NEW`

New VSM parallel inode file

ALSO SEE

`pfinit(1M), pfprint(1M), HSM(1M)`

pfinit(1M)

NAME

pfinit - initialize the VSM .PAIN (parallel inode) file.

SYNOPSIS

```
/usr/openv/hsm/bin/pfinit [-c | -e] file_system
```

AVAILABILITY

This command is only available on kernel-based implementations.

DESCRIPTION

`pfinit` creates or extends the `.PAIN` file for a file system which is to be managed by VSM. `pfinit` will either create or extend the `.PAIN` file to match the total number of file inodes in the file system.

- You must use `pfinit` to create the `.PAIN` file before your initial attempt to mount a file system under VSM control.
- If you expand the file system with the `growfs` command, you must use `pfinit` to extend to `.PAIN` before remounting the file system under VSM control.

OPTIONS

`-c` This option directs `pfinit` to create the `.PAIN` file if it does not exist. If the file already exists, the `-c` option does nothing.

`-e` This option directs `pfinit` to extend the `.PAIN` file if it exists. `pfinit` does nothing if the file does not exist and the `-e` option is specified.

file_system Specifies the absolute path to a file system as defined by the *fspath* parameter in the VSM global-configuration file, `/usr/var/openv/hsm/database/migconfg`. `pfinit` will create or extend the `.PAIN` file as *fspath*`/.PAIN`.

CAVEATS

- When using `pfinit`, *file_system* must be mounted as a normal file system, not as an *hsm* file system.
- `pfinit` is available only to root users.

EXAMPLES

The following command will create the /home/.PAIN file by using the number of inodes in the /home file system:

```
pfinit -c /home
```

If /home is mounted as an hsm file system, the following error message is returned:

```
pfinit: Filesystem must not be mounted as an hsm filesystem.
# echo $status
101
```

If /home is not mounted at all, the following error message is returned:

```
pfinit: error locating /home in mount table.
# echo $status
22
```

FILES

fspath/.PAIN

VSM parallel inode file

dwpath/database/migconf

VSM configuration file for managed file system

/usr/var/openv/hsm/database/migconfg

Global-configuration file for VSM

SEE ALSO

growfs(1M), pfcheck(1M), pfprint(1M), HSM(1M)

pfprint(1M)

NAME

pfprint - prints or alters a .PAIN (parallel inode) file entry

SYNOPSIS

```
/usr/openv/hsm/bin/pfprint [-v version] [-f flags]
          [-c slice] [-s size] [-t mtime] [-m mchid]
          [-h handle] file_system inode
```

Caution pfprint is intended only for customer support engineers who are trained in its use--this especially applies to changing the .PAIN file. All other users should rely on the pfcheck command to verify and change .PAIN entries.

AVAILABILITY

This command is only available on kernel-based implementations.

DESCRIPTION

pfprint prints the *fspath*/.PAIN entry for *inode* to standard output. If you specify any options, pfprint sets the indicated field in the .PAIN entry for the inode, prints the entry, and updates the .PAIN file. Values not specified remain unchanged. pfprint does not check changed values for correctness.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon migd is running.

OPTIONS

-v *version*
: Sets the version field to this value.

-f *flags*
: Sets the flags field of .PAIN entry *inode* to this value. The value is assumed to be hexadecimal.

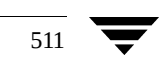
-c *slice*
: Sets the slice field of .PAIN entry *inode* to this value.

-s *size*
: Sets the size field of .PAIN entry *inode* to this value.

-t *mtime*
: Sets the mtime field of .PAIN entry *inode* to this value.

-m *mchid*
: Sets the *handle*.machid field of .PAIN entry *inode* to this value. The value is assumed to be hexadecimal.

-h *handle*
: Sets the *handle*.handle field of .PAIN entry *inode* to this value. The value is assumed to be hexadecimal.



file_system Specifies the absolute path to a file system as defined by the *fspath* parameter in the VSM `/usr/openv/hsm/database/migconfg` global-configuration file. `pfprint` will use the *fspath*`/.PAIN` file.

inode Specifies that the *fspath*`/.PAIN` entry be printed and or altered. If the *inode* is past the end of the `.PAIN` file, an error will be reported.

EXAMPLES

This command will change `/home/.PAIN` file entry 1234 to be set as a normal file, not a migrated and not an unchanged cached file.

```
pfprint -v OV01 -f 0 -c 0 -s 0 -t 0 -m 0 -h 0 /home 1234
```

FILES

fspath`/.PAIN`

VSM parallel inode file

dwpath`/database/migconf`

VSM configuration file for managed file system

`/usr/var/openv/hsm/database/migconfg`

Global-configuration file for VSM

SEE ALSO

`pfinit(1M), pfcheck(1M), HSM(1M)`

rebuild_ihand(1M)

NAME

rebuild_ihand – rebuild .IHAND file based on FHDB and file system

SYNOPSIS

/usr/openv/hsm/bin/rebuild_ihand *hsmname*

AVAILABILITY

This command is only available on nonkernel-based implementations.

DESCRIPTION

rebuild_ihand reconstructs a lost or corrupted inode-to-handle (.IHAND) file if the managed file system and file-handle database (FHDB) are intact.

This command will not detect all discrepancies between the file system and the FHDB; run migdbcheck to do this.

The output of rebuild_ihand is a reconstructed .IHAND file in *dwpath*/database/*hsmname*.IHAND.*pid*. To use this reconstructed .IHAND file, copy it to the real .IHAND file in *dwpath*/database/*hsmname*.IHAND.

The exit status of rebuild_ihand is a count of the number of errors encountered. A zero status indicates there were no errors. Error messages are written to stderr and to the VSM logfile.

This command may be run without regard to the VSM state (active or inactive), but the VSM daemons must not be running.

OPTIONS

hsmname HSMDEV entry of the file system for which you want to rebuild the .IHAND file.

EXAMPLE

This command will reconstruct the *dwpath*/database/alpha.IHAND file based on *dwpath*/database/FHDB.

```
prompt> rebuild_ihand alpha
```

If the command detects an active FHDB entry for a nonexistent file, the following error message is issued:

ERROR: Cannot stat *pathname*, handle *handle*

and no IHAND entry is created.

If the command detects that the file size recorded in the FHDB differs from the actual file size, the following error message is issued:

ERROR: *pathname* FHDB size *size*, != stat() size *size*

and an IHAND entry is created using the actual file size.

FILES

Note The term *dwpath* refers to the path name of the directory containing the `database` and `workdir` directories. VSM uses the `database` directory to store all its database files. The path name of this directory is site configurable and can be different for each HSMDEV entry that you define in the `migconfg` global-configuration file.

dwpath/`database`/*hsmname*.`IHAND`

Inode-to-handle file for VSM

dwpath/`database`/*hsmname*.`IHAND`.*pid*

Reconstructed inode-to-handle file for VSM

dwpath/`database`/`FHDB`

File handle database for VSM

SEE ALSO

`ihprint(1M)`

startmigd(1M)

NAME

startmigd – start the VSM migration and volume daemons

SYNOPSIS

```
/usr/openv/hsm/bin/startmigd [-m | -v] [-t time]
```

DESCRIPTION

`startmigd` is available to the system administrator to start the VSM migration daemon (`migd`) and the VSM volume daemon (`migvold`). You must have superuser privileges to execute this command.

You should start `migvold` and `migd` as part of system startup. The startup scripts that you copied during installation do this for you automatically. The `startmigd` command verifies that the daemon tasks are not already running and then starts them.

On startup, the migration daemon reads the configuration files. If you manually change the configuration file while the daemon is up, you can stop and restart the daemon so that it picks up the changes or you can signal the daemon as follows:

```
kill -INT `cat /usr/var/openv/hsm/workdir/migd.pid*`
```

If you use the GUI `xhsmadm` to make the changes, the interrupt is automatically sent to `migd`.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

OPTIONS

`-m`	Start only the migration daemon(s) (`migd`).
`-v`	Start only the volume daemon (`migvold`).
`-t` *time*	Sets the *time* interval in seconds that determines the frequency with which the migration daemon (`migd`) checks the high-water mark threshold. Default is 60. Applicable only to nonkernal-based implementations.

The default is to start all VSM daemons.

CAVEATS

- Mount all managed file systems before starting the migration-daemon task and the volume-daemon task.

- If the migration daemon is not running or the corresponding HSMDEV entry `state` parameter is set to `0` (Inactive) in the `migconfg` file, only root users can use migrated files. The kernel returns an error response (EARDR NOT AVAIL) to all file-system requests. In addition, the kernel cannot automatically start a migration when file-system space is low. If the daemon task dies, use the VSM log files to determine the cause of failure, correct the problem, and restart the daemon.
- If the volume daemon is not running, users cannot read migrated files from tape or optical volumes.

EXAMPLE

The following command starts both the migration and volume daemons, and returns messages similar to these:

```
startmigd
HSM volume daemon migvold started
migration daemon migd started
```

FILES

`/usr/var/openv/hsm/workdir/migd.pid`

The process ID (pid) of the VSM migration daemon (if applicable)

`/usr/var/openv/hsm/workdir/migd1.pid`

The process ID (pid) of VSM migration daemon on channel 1 (if applicable)

`/usr/var/openv/hsm/workdir/migd2.pid`

The process ID (pid) of VSM migration daemon on channel 2 (if applicable)

`/usr/var/openv/hsm/workdir/migvold.pid`

The process ID (pid) of the VSM volume daemon.

SEE ALSO

`HSM(1M), HSMKiller(1M), migconf(1M), migconfg(1M), stopmigd(1M)`

stopmigd(1M)

NAME

stopmigd – stop the VSM migration and volume daemons

SYNOPSIS

```
/usr/openv/hsm/bin/stopmigd [-m | -v]
```

DESCRIPTION

`stopmigd` allows the system administrator to stop the VSM migration daemon (`migd`) and the VSM volume daemon (`migvold`). The daemons should only be stopped in the event of a problem, or perhaps when operator coverage is not available to mount tapes. When the migration software is installed on a machine, the migration daemon becomes an integral part of the operating system software on that machine.

Note On some platforms `migd` forks into two processes.

The command `stopmigd` only terminates the VSM daemons. The command `HSMKiller` may be used to halt all running VSM processes. See caveats.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

OPTIONS

`-m` Stop only the migration daemon (`migd`).

`-v` Stop only the volume daemon (`migvold`).

The default is to stop all VSM daemons.

The exit value (n) is the number of daemons stopped.

CAVEATS

- If the migration daemon is not running, only root users can use migrated files. The kernel returns error responses to all file system requests.
- If the migration daemon is not running, automatic migrations do not occur when free space is above the high-water threshold and a user attempts to open a migrated file.
- If the volume daemon is not running, users cannot read migrated files from tape or optical volumes.
- Current VSM migration and caching operations run to completion but may leave incorrect results after `stopmigd` stops the VSM daemons.

EXAMPLE

The following command stops both the migration and volume daemons, and returns messages similar to these:

```
stopmigd
migration daemon migd stopped
HSM volume daemon stopped
echo $status
2
```

The exit value shows this command stopped a total of two daemons.

FILES

`/usr/var/openv/hsm/workdir/migd.pid`

The process ID (pid) of the VSM migration daemon (if applicable)

`/usr/var/openv/hsm/workdir/migd1.pid`

The process ID (pid) of VSM migration daemon on channel 1 (if applicable)

`/usr/var/openv/hsm/workdir/migd2.pid`

The process ID (pid) of VSM migration daemon on channel 2 (if applicable)

`/usr/var/openv/hsm/workdir/migvold.pid`

The process ID (pid) of the VSM volume daemon.

SEE ALSO

`HSM(1M), HSMKiller(1M), startmigd(1M)`

xhsmadm(1M)

NAME

xhsmadm - administrative graphical user interface

SYNOPSIS

`/usr/openv/hsm/bin/xhsmadm [X` *options*`]`

DESCRIPTION

The `xhsmadm` command initiates the X-based VSM administrative user interface. You must use an X terminal or X server software on a workstation, compatible with MIT's release 11.4 (or later) of the X Window System.

The administrator can use `xhsmadm` to configure VSM and monitor its operations. You must have root user privileges.

The System Administrator's Guide for VSM and `xhsmadm`'s popup help windows provide detailed operating instructions.

`xhsmadm` uses OSF / Motif for standardized look and feel. Users who are unfamiliar with Motif are advised to refer to the *OSF/Motif User's Guide*, written by the Open Software Foundation and published by Prentice-Hall, Inc., ISBN 0-130640509-6.

This command may be run without regard to the VSM state (active or inactive) or whether the migration daemon `migd` is running.

OPTIONS

`xhsmadm` supports the standard command-line options for X programs. Most important of these is the `-d` option which can be used to force the name of the X terminal or server. Most users will already have their DISPLAY environment variable defined and can routinely ignore the `-d` option.

Other useful X options are:

`-font` Allows you to enlarge text for visibility. It is best to use fixed-pitch fonts because `xhsmadm` formats some text into columns. These columns can appear uneven with proportional fonts.

`-geometry` Allows you to control the initial size and position of the `xhsmadm` window.

`-title` Controls the window manager title bar and can be useful if you run several instances of `xhsmadm` at once.

SEE ALSO

HSM(1M)

Command Requirements B

This appendix shows what is required for VSM commands to run.

Table 9. VSM Command Requirements

Command Name	VSM State Inactive and `migd` not running	VSM State Active and `migd` not running	VSM State Inactive and `migd` running	VSM State Active and `migd` running
`fls`	Runs	Runs	Runs	Runs
`gethsm`	No	Runs	No	Runs
`HSMKiller`	Runs	Runs	Runs	Runs
`ihprint`	Runs	Runs	Runs	Runs
`mediacheck`	Runs	Runs	Runs	Runs
`migadscan`	Runs	Runs	Runs	Runs
`migalter`	Runs	Runs	Runs	Runs
`migbatch`	No	Runs	No	Runs
`migcat`	No	Runs	No	Runs
`migchecklog`	Runs	Runs	Runs	Runs
`migcleanup`	Runs	Runs	Runs	Runs
`migcons`	Runs	Runs	Runs	Runs
`migconsweep`	No	Runs	No	Runs
`migdbcheck`	Runs	Runs	Runs	Runs
`migdbdir`	Runs	Runs	Runs	Runs
`migdbrpt`	Runs	Runs	Runs	Runs
`migfind`	Runs	Runs	Runs	Runs
`migftscan`	Runs	Runs	Runs	Runs
`miggetvol`	Runs	Runs	Runs	Runs

Table 9. VSM Command Requirements

Command Name	VSM State Inactive and migd not running	VSM State Active and migd not running	VSM State Inactive and migd running	VSM State Active and migd running
miggroup	No	No	No	Runs
migin	No	Runs	No	Runs
miglicense	Runs	Runs	Runs	Runs
migloc	Runs	Runs	Runs	Runs
miglow	No	Runs	No	Runs
migmdclean	Runs	Runs	Runs	Runs
migmode	No	No	No	Runs
migmove	Runs	Runs	Runs	Runs
mignbexport	No	No	Runs	Runs
mignbimport	No	No	Runs	Runs
mignbscan	Runs	Runs	Runs	Runs
mignewlog	Runs	Runs	Runs	Runs
mignospace	No	No	No	Runs
migopscan	Runs	Runs	Runs	Runs
migpurge	No	No	No	Runs
migrate (root)	Runs	Runs	Runs	Runs
migrate (non-root)	No	No	No	Runs
migrc	Runs	Runs	Runs	Runs
migreconstruct	Runs	No	No	No
migrecycle	Runs	Runs	Runs	Runs
migreg	Runs	Runs	Runs	Runs
migselect	Runs	Runs	Runs	Runs
migsetdb	Runs	Runs	Runs	Runs
migstage	No	No	No	Runs
migtestbadness	Runs	Runs	Runs	Runs
migthreshold	No	No	No	Runs
migtie	No	Runs	No	Runs
migtscan	Runs	Runs	Runs	Runs

Table 9. VSM Command Requirements

Command Name	VSM State Inactive and `migd` not running	VSM State Active and `migd` not running	VSM State Inactive and `migd` running	VSM State Active and `migd` running
`pfprint`	Runs	Runs	Runs	Runs
`rebuild_ihand`	Runs	Runs	No	No
`startmigd`	Runs	Runs	Runs	Runs
`stopmigd`	Runs	Runs	Runs	Runs
`xhsmadm`	Runs	Runs	Runs	Runs

NOTE: `pfinit` and `pfcheck` must be run with the file system *not* mounted as an *hsm* file system.

Planning Worksheets C

This appendix contains reproducible copies of the planning worksheets found in chapter 2 of this manual.

- Global Configuration Worksheet
- Managed-File-System Planning Worksheet
- Managed-File-System Configuration Planning Worksheet (1 of 3)
  - File System Migration Tresholds
  - Migration Stop Files
- Managed-File-System Configuration Planning Worksheet (2 of 3)
  - Storage Methods, 1 through 8
  - Method Names
- Managed-File-System Configuration Planning Worksheet (3 of 3)
  - Move Attributes by Method Name
  - Move Attributes by Migration Level

Global Configuration Worksheet

VSM Name: _____

Mount Point: _____

Database Pathname _____

Logfile Pathname: _____

State: _____

VSM Name: _____

Mount Point: _____

Database Pathname _____

Logfile Pathname: _____

State: _____

VSM Name: _____

Mount Point: _____

Database Pathname _____

Logfile Pathname: _____

State: _____

VSM Name: _____

Mount Point: _____

Database Pathname _____

Logfile Pathname: _____

State: _____

Managed-File-System Planning Worksheet

VSM Name _____ File System Name (mount point)_____

File System Thresholds for Migration

High-water Mark for this file system: _____%

Purge Mark for this file system: _____%

Low-water Mark for this file system: _____%

File Selection Criteria for Migration	File selection Criteria for Purging
Migrate files with Badness ≥ _____	Purge files with Badness ≥ _____
Age weight: _____	Age weight: _____
Size weight: _____	Size weight: _____
Weight operator: _____	Weight operator: _____
Min. age to migrate (days): _____	Min. age to purge (days): _____
Min. size to migrate (kbytes): _____	Min. size to purge (kbytes): _____

Best Storage Methods for File System

Copies: _____ (1 or 2) METHODn: `op.1.library/ct.3.vault/...`

METHOD1: _____

METHOD2: _____

METHOD3: _____

METHOD4: _____

METHOD5: _____

METHOD6: _____

METHOD7: _____

METHOD8: _____

Managed-File-System Configuration Planning Worksheet (1 of 3)

Database Pathname: _____

File System Migration Thresholds

VSM Name: _____ _____ _____ _____

File System Name: _____ _____ _____ _____

High-Water Mark (free space): _____ _____ _____ _____

Purge Mark: _____ _____ _____ _____

Low-Water Mark: _____ _____ _____ _____

Badness: _____ _____ _____ _____

Minimum Age (days): _____ _____ _____ _____

Minimum Size (kbytes): _____ _____ _____ _____

Age Weight: _____ _____ _____ _____

Size Weight: _____ _____ _____ _____

Weight Operator: _____ _____ _____ _____

Purge Badness: _____ _____ _____ _____

Purge Minimum Age (days): _____ _____ _____ _____

Purge Minimum Size (kbytes): _____ _____ _____ _____

Purge Age Weight: _____ _____ _____ _____

Purge Size Weight: _____ _____ _____ _____

Purge Weight Operator: _____ _____ _____ _____

Migration Parameters for all File Systems Using this Database

Machine ID: _____ User-Space-Restriction Quota (bytes): _____

Copies: _____ Unmount Delay (seconds): _____ Checksum (y/n): _____

Managed-File-System Configuration Planning Worksheet (2 of 3)

Database Pathname: _____

Storage Methods for All File Systems Using this Database

Copies: _____ (1 or 2)

METHOD1: _____

METHOD1: _____

METHOD1: _____

METHOD1: _____

METHOD1: _____

METHOD1: _____

METHOD1: _____

METHOD1: _____

Method Names for All File Systems Using this Database

Method Name:	ct	dt	mt	op	ow	ad	ft	nb
Flags:								
Access Time:							na	na
Capacity:				na	na		na	na
Speed:							na	na
Granule Size:							na	na
Block Size:				na	na	na	na	na
Density:						na	na	na
DeadmanTimeout:						na		
Demand Delay:						na	na	na

Managed-File-System Configuration Planning Worksheet (3 of 3)

Database Pathname: _____

Move Attributes for all File Systems Using this Database by Method Name

Method Name:	ct	dt	mt	op	ow	ad
Move Badness:						
Move Minimum Age:						
Move Minimum Size:						
Move Age Weight:						
Move Size Weight:						
Move Weight Operator:						

Move Attributes for all File Systems Using this Database by Migration Level

Storage Method (level):	METH1	METH2	METH3	METH4	METH5	METH6	METH7	METH8
Move Badness:								
Move Minimum Age:								
Move Minimum Size:								
Move Age Weight:								
Move Size Weight:								
Move Weight Operator:								

Setting Up a Schedule D

This chapter describes how to set up a schedule for a task.

The following topics are included in this chapter:

Table 10. Topics in This Chapter

Section	Description
"What Is a Schedule?" on page 539	Defines a time window and run schedule.
"Overview of the Scheduling Service Dialog Box" on page 542	Provides an overview of the Scheduling Service dialog box and explains how to navigate the dialog box and configure schedule settings.
"Viewing a Schedule Summary" on page 547	Describes how to view a summary of your schedule.
"General Options" on page 549	Explains how to specify when a schedule becomes effective, the time when a task starts, and whether the task restarts during the run day.
"Run Day Options" on page 557	Describes different ways to specify the days or dates on which a task runs.

Overview of Schedule Options

The following table summarizes scheduling options:

Table 11. Summary of Schedule Options

Option	Description
Effective Date	Determines when your schedule goes into effect. A task can not run prior to its effective date. By default, the effective date is the current date when you create the schedule.
Time Window	Period of time during which a task can begin on any scheduled run day.
Restart Time Interval	Interval at which a task repeats within a time window. A task can run multiple times within the time window defined for any scheduled run day. The task is scheduled to begin at the starting time of your time window and repeat at a regular interval that you specify in hours, minutes, and seconds.
Week Days	Schedules a task for certain days of each week, weeks of each month, or days on particular weeks of the month. A check mark entered for a day indicates that the task is scheduled to run on that day of its respective week. All days are selected by default.
Day Interval	Schedules a task to run every certain number of days calculated from a particular date. By default, the date from which the interval is calculated is the date on which you create the schedule.
Days of the Month	Schedules a task for certain days of the month. For example, you might want a task to run on the first and 15th day of each month. You can also schedule a task to run on the last day of each month, regardless of the actual date.
Specific Dates	Schedules a task for specific dates as an alternative to a recurring schedule or in addition to a recurring schedule. Use the controls at the top of the calendar to change the month or year.
Exclude Dates	Exclude specific dates, for example holidays, from a run schedule. Use the controls at the top of the calendar to change the month or year.
Summary	Allows you to verify your schedule settings. Run days are shown in bold font on calendars covering a fixed period. Use the calendar controls to view the next period.

What Is a Schedule?

By setting up a schedule for a task, you can define what time and on which days the task can run. A schedule consists of the following:

- Time window—Determines the time period during which a task can start on any given run day. A task can restart within the time window.
- Run days—Consist of specific dates, a recurring pattern of days, or both.

The Scheduling Service dialog box provides several options for configuring schedules. These options can be applied exclusively or in combination with each other.

General options allow you to specify the time when a task can start, whether it repeats during the day, and on which date you want the schedule to take effect. The general options that you specify apply every day on which a task is scheduled to run.

Run day options allow you to schedule the days or dates on which the task can run. Run days consist of specific dates, a recurring pattern of days, or both. By default, a task is scheduled to run every day indefinitely.

Related topics

- “Understanding the Time Window” on page 540
- “Understanding Run Days” on page 541
- “General Options” on page 549
- “Run Day Options” on page 557

Understanding the Time Window

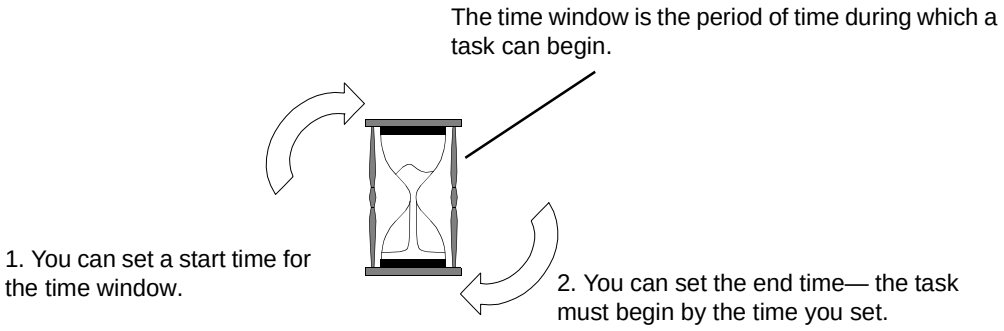
By default, a task can run anytime between 12:00:00 AM and 11:59:59 PM. This means that, for example, if you schedule a task to run every Monday using the default time window, the task can start at or anytime after midnight early Monday morning. It cannot, however, start after 11:59:59 PM Monday night.

Time window

A time window is the period of time during which a task can begin.

You can control when a task runs by changing the start and end times of its time window. The following illustration summarizes the process required to customize a time window.

Figure 122. Steps Required To Define the Time Window for a Task



You can set up a time window that extends past midnight. For example, if you don't want a task to interfere with daytime operations, you can schedule the task to start after 5:00 PM and before 3:00 AM the next day.

You can also configure the task to restart within the time window. This enables you to run the task several times each day.

The time window you specify applies for every day on which the task is scheduled to run. If you schedule a task to start today, the task can run today provided the specified time window has not elapsed.

Understanding Run Days

By default, a task is scheduled to run every day indefinitely. You can, however, set up any type of schedule you want for a task. The Scheduling Service dialog box provides several options for defining run days.

Run days

Run days make up a defined set of days or dates on which a task can run. They can be specific dates, a recurring pattern of days, or both.

You can set up a schedule that enables a task to run on certain days each week, certain weeks each month, or certain days on particular weeks of the month. For example, you might want a task to run on the first Monday of every month, the last week of every month, the last Friday of the last week every month, or all of the preceding.

You can also base the run cycle on particular days of the month, for example, by scheduling a task for the first and 15th days. This option allows you to schedule tasks for the last day of each month regardless of the date.

A task can run at regular intervals. This feature is useful, for example, to back up weekly, quarterly, or annual records that are modified every certain number of days.

You can specify explicit dates for the task using a calendar. The task can run on these dates rather than follow a recurring schedule, or the dates can be in addition to a recurring schedule. Once the calendar is displayed, you can select dates in any month of any year.

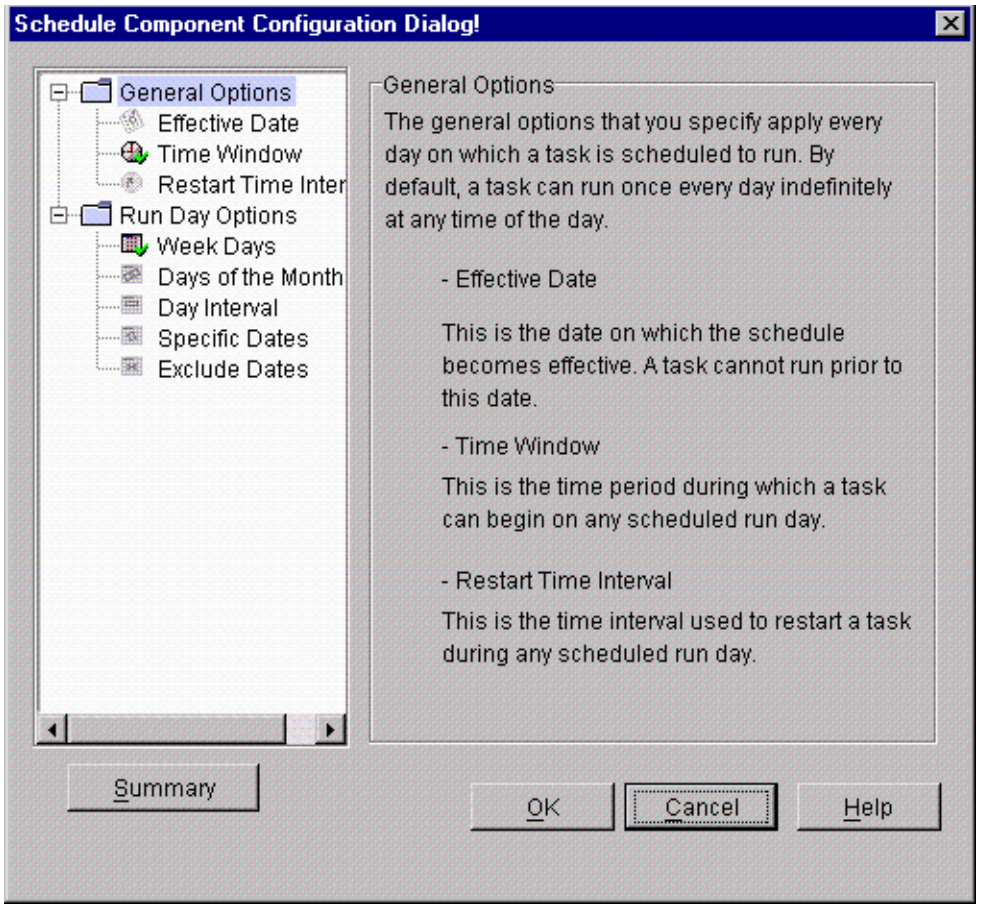
After configuring a schedule, if you don't want the task to run on certain days falling within the schedule, you can exclude those dates. This option is useful for excluding holidays.

Finally, the Scheduling Service dialog box allows you to combine all available options. For example, you could schedule a task to run on the first and third Thursday of the month, the last day of each month, every 90 days, and on specific dates, such as December 25 and July 4.

Overview of the Scheduling Service Dialog Box

In the Scheduling Service dialog box, shown belowyou define what time and on which days a task can run.

Figure 123. Scheduling Service Dialog Box



The Scheduling Service dialog box contains the following elements:

Table 12. Scheduling Service Dialog Box Elements

Element	Description
Options tree list	The list in the pane on the left contains the various options you can use when scheduling a task. The Summary button directly beneath the list enables you to view all settings for a schedule.

Table 12. Scheduling Service Dialog Box Elements (continued)

Element	Description
Configuration pane	The pane on the right lets you configure settings for the option that is currently highlighted in the options list.
Button bar	The button bar at the bottom enables you to accept settings and close the dialog box, cancel settings, or get help about the pane currently displayed. The Summary button displays a summary of your current task settings.

The following sections provide procedures for:

- "Navigating the Schedule Options Tree" on page 544
- "Configuring Schedule Settings" on page 546

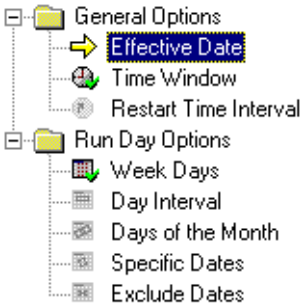
Navigating the Schedule Options Tree

The tree in the left pane of the Scheduling Service dialog box lists the following options for setting up a schedule.

- General options allow you to control the time when a task begins, whether and how often it restarts during the run day, and on which date the schedule goes into effect. The general options that you specify apply every day on which a task is scheduled to run.
- Run day options enable you to specify the days on which a task can run. Run days consist of specific dates, a recurring pattern of days, or both. You can also exclude specific dates from a schedule.

The schedule options tree is shown below:

Figure 124. Schedule Options Tree



You can expand and collapse the schedule options tree, view the available options, and enter or change the settings for an option. The following steps describe how to manipulate the tree and choose options:

- A plus sign [+] next to a folder indicates that the folder contains suboptions. Select [+] to expand the section and display all its suboptions.
- Select the folder itself (or the folder's name) to expand the section as well as display the associated option in the Configuration pane.
- A minus sign [-] next to a folder indicates that the folder's contents are displayed. Select [-] to collapse the section and hide its suboptions.
- An icon next to an option indicates there are no additional suboptions. Select the icon to display the option in the Configuration pane.

Each schedule option has its own icon. The icon provides the status of a configuration option in the following ways:

- A darkened icon with a check mark indicates that settings have been entered for the option.

- A grayed icon indicates that no settings have been entered for the option.

Note By default, every schedule is configured with time window and week day settings.

As an example, the following table provides status information for the Effective Date icon:

Table 13. Effective Date Icons

Icon	Option Status
	Indicates that no effective date has been configured for the task.
	Indicates that an effective date has been configured for the task.
	Indicates that the Effective Date pane is currently displayed and can be configured.

Configuring Schedule Settings

By default, a task is scheduled to run indefinitely once a day at any time of the day. You can, however, set up a schedule that defines what time and on which days the task can run.

From the Scheduling Service dialog box, you can view and edit settings that correspond to the tree options in the left pane. First, you must display the appropriate pane in the Scheduling Service dialog box.

▼ **To view or edit schedule settings:**

1. Expand the options tree if necessary to display the schedule options.
2. Select the option you want to view or edit.

   The associated configuration pane is displayed.
3. Make the changes you want in the pane.
4. When you have finished making changes, select OK.

   The Scheduling Service dialog box accepts the changes you made and closes.

Tip After making changes, you can view a summary of the modified schedule by selecting the Summary button.

Related topics

- "Navigating the Schedule Options Tree" on page 544
- "Viewing a Schedule Summary" on page 547

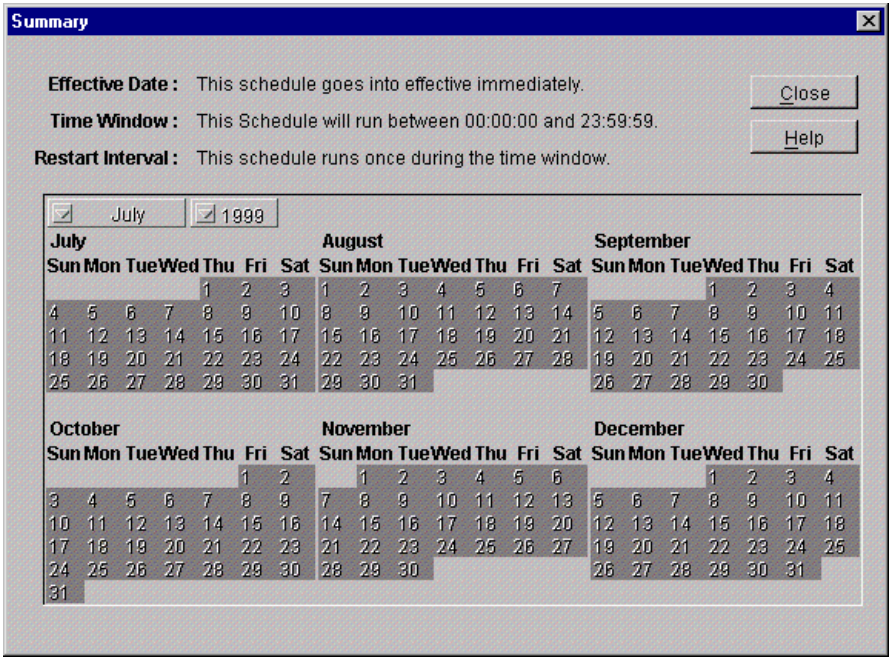
Viewing a Schedule Summary

You can view current schedule settings for your task. This is especially helpful when you've made changes to your schedule and want to verify that the new schedule is correct.

▼ To view a schedule summary:

- ❖ Select the Summary button beneath the schedule options tree. The Summary dialog box is displayed. This is a read-only dialog box that cannot be edited.

Figure 125. Summary Dialog Box



The Summary dialog box displays the run days and time window defined for a task during the current summary period. If you scheduled a task to repeat during the day, the restart interval is shown as well. Run days are shown in boldface on the calendars.

After viewing the current summary, you can advance the calendars and view the next set of months.

▼ **To advance the calendars:**

1. Select the month button (labeled the starting month for the current six-month summary).

   A drop-down listbox displays starting months of the two six-month summaries for this task.

2. Select the starting month for the next six-month summary.

   The calendar displays the next six months of the current year.

3. To advance the year, select the year button.

   A drop-down list of years is displayed.

4. Select the year you want.

Related topics

- "Configuring Schedule Settings" on page 546
- "Navigating the Schedule Options Tree" on page 544
- "Overview of the Scheduling Service Dialog Box" on page 542

General Options

The General Options category of the options tree list allows you to configure settings that apply every day on which a task is scheduled to run. You set general options to specify:

- Effective date—This is the date on which the schedule you define becomes effective. A task can not run prior to this date.
- Time window—This is the time period during which the task can begin on any scheduled run day.
- Restart time interval—This is the time interval used to restart a task during any scheduled run day.

After setting general options, you can specify the days on which the task runs by setting run day options.

The following sections describe how to configure general options:

- “Setting an Effective Date” on page 550
- “Specifying a Time Window” on page 552
- “Restarting a Task Within Its Time Window” on page 555
- “Working with a Time Window That Extends Past Midnight” on page 554

Setting an Effective Date

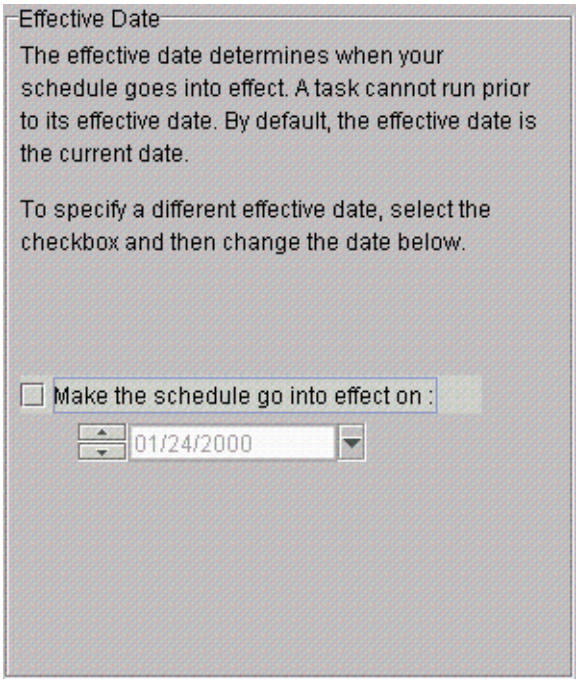
The effective date determines when your schedule goes into effect. A task can not run prior to its effective date. By default, the effective date is the current date.

▼ **To specify an effective date:**

1. Open the Scheduling Service dialog box.
2. From the schedule options list in the left pane, select the Effective Date option.

   The Effective Date pane is displayed.

Figure 126. Effective Date Pane



3. Select the Make the schedule go into effect checkbox.

   The field directly beneath the checkbox becomes enabled.
4. In the field, enter the date on which you want the schedule to go into effect. Do either of the following:
   - Click the arrow to the right of the field and select the date you want from the drop-down calendar.

- Use the increment buttons to the left of the field to change the month, day, or year (mm:dd:yyyy format). Click the portion of the date you want to change and then select the up or down increment button.

5. When you have finished setting up a schedule, select OK.

   The Scheduling Service dialog box accepts the changes you made and closes.

Specifying a Time Window

You can specify the time when a task can start by changing its time window. For example, if you want a task to run sometime between 09:00:00 and 11:00:00, you would change the start time and end time to those values. By default, a task can start any time between 00:00:00 and 23:59:59 on a run day.

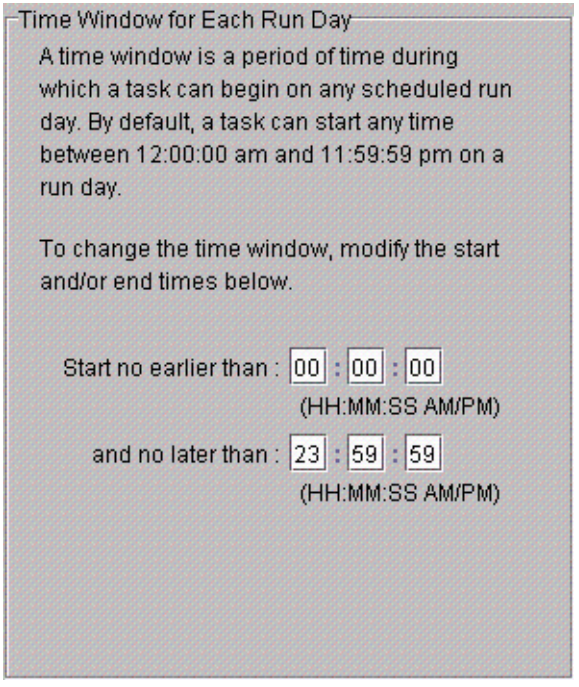
Note The time window does not extend beyond 24 hours, or more specifically, beyond 23 hours, 59 minutes, and 59 seconds. For example, you can not set a time window starting at 03:00:00 and ending at 05:00:00 the following day.

▼ **To specify a time window for a task:**

1. Open the Scheduling Service dialog box.
2. From the schedule options list in the left pane, select the Time Window option.

   The Time Window for Each Run Day pane is displayed. This pane contains two fields that correspond to the start and end times of your time window.

Figure 127. Time Window for Each Run Day pane



3. In the top field, type or select the time after which the task can start. Perform any of the following actions:

- To enter hours, select the hour value and enter the hour you want for the start time.
- To enter minutes, select the minutes value and enter the minutes you want for the start time.
- To enter seconds, select the seconds value and enter the seconds you want for the start time.

4. In the bottom field, select or type the time by which the task must start. Perform any of the following actions:
   - To enter hours, select the hour value and enter the hour you want for the end time.
   - To enter minutes, select the minutes value and enter the minutes you want for the end time.
   - To enter seconds, select the seconds value and enter the seconds you want for the end time.

Caution Use caution when scheduling a time window that extends past midnight, as this can affect the days on which the task runs. For more information, see "Working with a Time Window That Extends Past Midnight" on page 554.

5. When you have finished setting up a schedule, select OK.

   The Scheduling Service dialog box accepts the changes you made and closes. The task is scheduled to start during the time window you specified.

Working with a Time Window That Extends Past Midnight

When setting up the time during which a task runs, you can enter a time window that extends past midnight and into the next day. You do this by providing an end time that occurs the next morning.

Bear in mind, however, that this type of time window changes the days on which the task can run. For example, if you schedule a task to run between 8:00 PM Friday night and 4:00 AM the next morning, it's possible for the task to run on Saturday anytime before or at 4:00 AM. If you don't want the task to run on Saturday, you must alter the time window, for example, by changing the ending value from 04:00:00 AM to 11:59:59 PM. Then, the task is confined to one day.

Tip When a time window crosses midnight, the start time is later in the day than the end time.

For details about specifying a time window, see "Specifying a Time Window" on page 552.

Restarting a Task Within Its Time Window

You can set up a task to restart at regular intervals within the specified time window. This allows a task to run multiple times each scheduled day based on an interval that you specify in hours, minutes, and seconds. The interval you specify is measured with respect to the first value of your defined time window. For example, if you set up a task to run between 2:00 AM and 9:00 AM and repeat every three hours, the run schedule is 2:00 AM, 5:00 AM, and 8:00 AM.

If you schedule a task to start today, the task can run at the next scheduled interval. If the last interval has passed, the task can not run today.

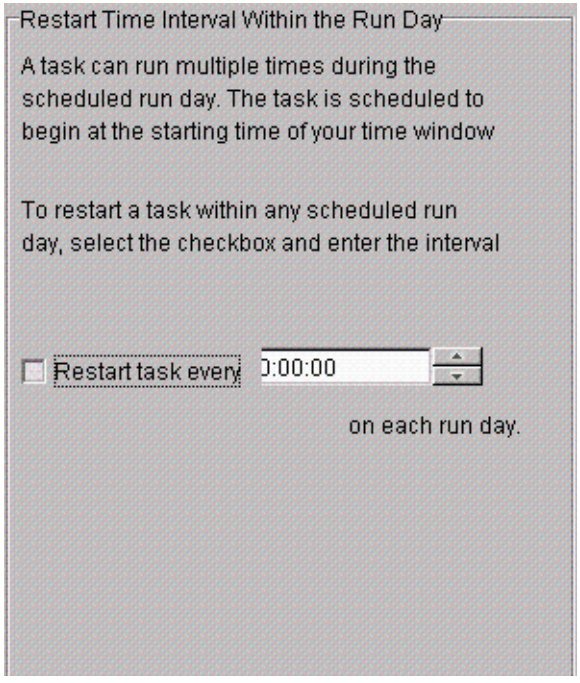
▼ **To restart a task during the scheduled time window:**

1. Open the Scheduling Service dialog box.
2. If necessary, specify the time window during which you want the task to run.

   See “Specifying a Time Window” on page 552 for procedures.
3. From the schedule options list in the left pane, select the Restart Time Interval option.

   The Restart Time Interval Within the Run Day pane is displayed.

Figure 128. Restart Time Interval Within the Run Day Pane



4. Select the Restart task every checkbox.

5. In the listbox to the right of the checkbox, select or type the restart interval. Perform any of the following actions:

   - To enter hours, select the hour value and enter the number of hours you want for the restart interval.

Tip If using the increment buttons, you need only click in the portion of the field you want to change (hours / minutes / seconds) and then select the up or down increment button.

   - To enter minutes, select the minutes value and enter the number of minutes you want for the restart interval.
   - To enter seconds, select the seconds value and enter the number of seconds you want for the restart interval.

Note The maximum values you can choose are 23 hours, 59 minutes, and 59 seconds.

6. When you have finished setting up a schedule, select OK.

   The Scheduling Service dialog box accepts the changes you made and closes. The task is scheduled to restart at the specified interval.

Run Day Options

By default, a task is scheduled to run every day indefinitely. You can, however, specify particular run days for a task. Run days consist of specific dates, a recurring pattern of days, or both. You can set up a schedule based on:

- Week days (for example, every Saturday)
- Days of the month (for example, the first, 15th, and / or last day of each month)
- Regular intervals (for example, every 45 days)
- Specific dates (for example, December 25 and July 4)

These options can be applied exclusively or in combination with each other. The task is eligible to run on any of the selected run days.

After configuring a schedule, if you don't want the task to run on certain days falling within the schedule, you can exclude those dates. This option is useful for excluding holidays.

Regardless of the option you use, your task is scheduled to start during the time window you defined for the task. If you schedule a task to start today, the task can run today provided the specified time window has not elapsed.

The following sections describe how to set up run days:

- "Scheduling a Task for Certain Days of the Week or Weeks of the Month" on page 559
- "Scheduling a Task for Certain Days of Each Month" on page 567
- "Scheduling a Task To Run at Regular Intervals" on page 565
- "Specifying Explicit Dates for a Task" on page 569
- "Excluding Specific Dates from a Schedule" on page 571

Combining Schedule Options

You have considerable flexibility when setting up run days for your schedule. You can overlap schedule options and use them in combination with each other. For example, you can schedule a task to run the first and third Thursday of the month as well as the last day of the month. Or, you might base a run cycle on a 90-day interval and also run the same task the last day of each month. Another combination might be to schedule a task to run on certain holidays—for example, December 25, January 1, and July 4—as well as every Sunday of each month.

Caution It's important to remember that your task runs on each day that is selected in the Scheduling Service dialog box. If you schedule a task for certain days of each week and later decide to base the schedule on a specific interval instead, you must deselect the week days if you don't want the task to run on those days as well.

To verify that your schedule is correct, it’s a good idea to view the schedule summary after making changes. See “Viewing a Schedule Summary” on page 547 for details.

Scheduling a Task for Certain Days of the Week or Weeks of the Month

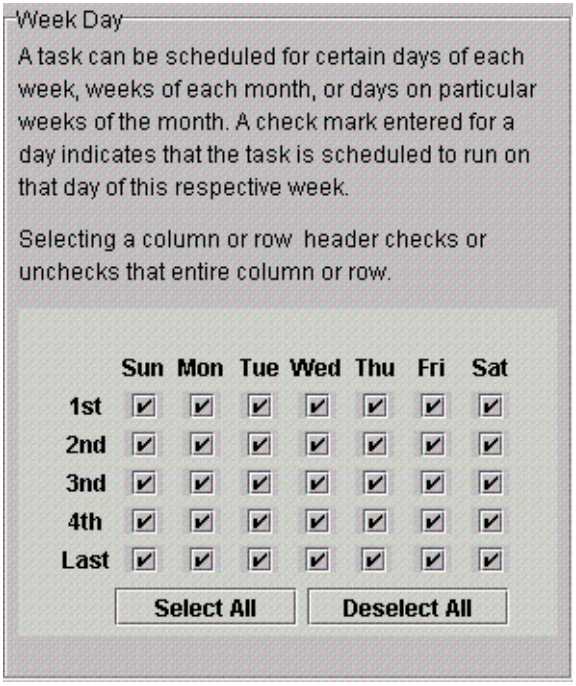
You can schedule a task to run on certain days each week, certain weeks each month, or certain days on particular weeks of the month. For example, you might want a task to run on the first and third Thursday of every month. In another example, you could schedule a task to run the last week of every month. Finally, you could schedule a task that includes both of the previous examples.

▼ To schedule week days for a task:

1. Open the Scheduling Service dialog box.
2. From the schedule options list in the left pane, select the Week Days of the Month option.

   The Week Days of the Month pane is displayed.

Figure 129. Week Days of the Month



This pane contains a weekday matrix. A check mark entered for a day indicates that the task is scheduled to run on that day of its respective week. All days are selected by default.

Note The weekday matrix is not a calendar. It is simply a matrix used to select days and weeks in a month.

3. You can modify the matrix by doing either of the following:

   - "Selecting Days in the Week Days of the Month Matrix" on page 561—Use this option if you want the task to run only on a few days each month. You may need to deselect all days first and then select the days you want.
   - "Deselecting Days from the Week Days of the Month Matrix" on page 563—Use this option if you want the task to run on every day with the exception of a few particular days.

4. When you have finished setting up a schedule, select OK.

   The Scheduling Service dialog box accepts the changes you made and closes.

Selecting Days in the Week Days of the Month Matrix

The Week Days of the Month pane contains a weekday matrix with all days and weeks in any given month. Each check mark in the matrix indicates that the task is scheduled to run on that day of the month.

The matrix has all days selected by default. If you want a task to run only once or twice a month, deselect all days in the calendar matrix and then select the days on which you want the task to run.

▼ **To select days in the matrix:**

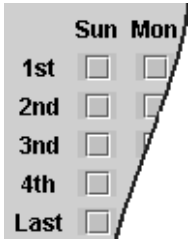
1. Open the Scheduling Service dialog box.
2. From the schedule options list in the left pane, select the Week Days of the Month option.

   The Week Days of the Month pane is displayed.
3. If necessary, select Deselect All to remove any existing check marks.
4. To select individual days, select the checkbox for the days you want included in the schedule. For example, if you select the checkbox beneath Sun in the first week, the task runs on the first Sunday of each month.

Tip If you make a mistake, select a day again to exclude it from the schedule.

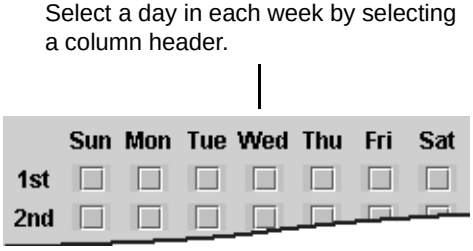
5. To select an entire week, select the appropriate row number to the left of the matrix. For example, to select the second week of each month, select 2nd.

Select a week by selecting its row number.



Note If you select the Last row, the task runs the last week of each month regardless of how many weeks there are in any given month.

6. To select a particular day in each week, select the appropriate column header. For example, to schedule a task for every Wednesday, select Wed.



7. When you have finished setting up a schedule, select OK.

The Scheduling Service dialog box accepts the changes you made and closes. The task is scheduled to run on the days you selected.

Deselecting Days from the Week Days of the Month Matrix

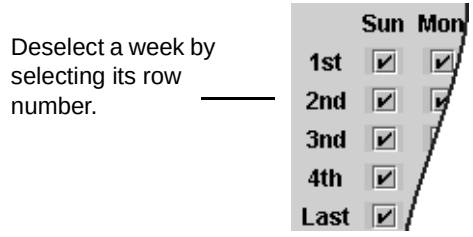
The Week Days of the Month pane contains a weekday matrix with all days and weeks in any given month. Each check mark in the matrix indicates that the task is scheduled to run on that day of the month. If you deselect an individual day by removing its check mark, the task does not run on that day.

▼ **To deselect days from the matrix:**

1. Open the Scheduling Service dialog box.
2. From the schedule options list in the left pane, select the Week Days of the Month option.

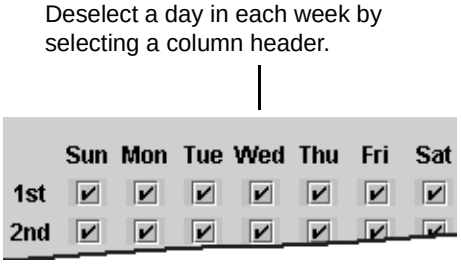
   The Week Days of the Month pane is displayed.
3. To remove a check mark for a particular day, select the corresponding checkbox. You uncheck the checkbox for the days you want to exclude from the schedule.

   If you make a mistake, select a checkbox again to return its check mark.
4. To deselect an entire week, select the appropriate row number to the left of the matrix. For example, to deselect the second week of each month, select 2nd.



Note This feature works as described only if the entire week is currently checked. If the row is only partially checked, you need to select the row number twice.

5. To deselect a particular day in each week of the month, select the appropriate column header. For example, selecting Wed would remove the check mark from each Wednesday of the month and prevent the task from running on Wednesdays.



Note This feature works as described only if the entire column is currently checked. If the column is only partially checked, you need to select the column header twice.

6. When you have finished setting up a schedule, select OK.

The Scheduling Service dialog box accepts the changes you made and closes. The task does not run on the days you unchecked.

Scheduling a Task To Run at Regular Intervals

You can schedule a task to run every certain number of days calculated from a particular date. For example, if you set up a task to repeat every three days starting on March 30, 2002, the task can run on March 30, April 2, April 5, and so on beginning in the year 2002. By default, the date from which the interval is calculated is the current date.

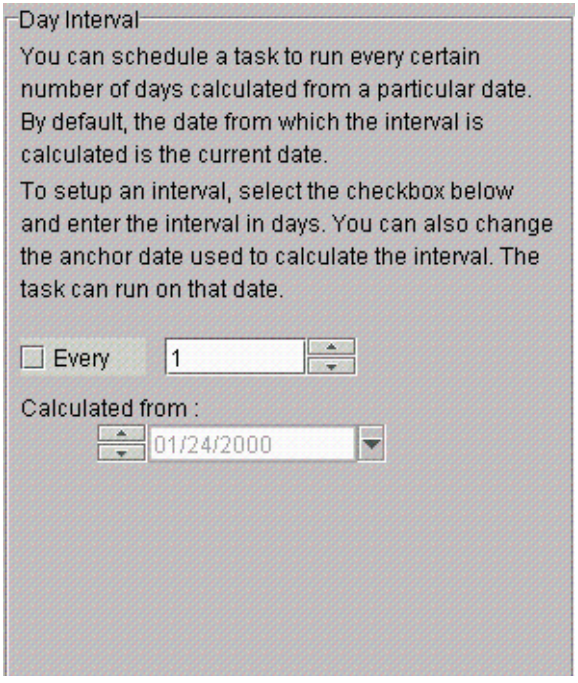
Recurring tasks run during the specified time window.

▼ To schedule an interval for a task:

1. Open the Scheduling Service dialog box.
2. From the schedule options list in the left pane, select the Day Interval option.

   The Day Interval pane is displayed.

Figure 130. Day Interval Pane



3. In the Day Interval panel, select the Every checkbox.
4. In the drop-down listbox to the right of the checkbox, select or enter the restart interval in days.

   The maximum value you can enter is 9,999 days.

5. To change the date used to calculate the interval, type or select the new date in the Calculated from field using the format mm:dd:yyyy.

You can do either of the following:

- Click the arrow to the right of the field and select the date you want from the drop-down calendar.
- Use the increment arrows to the left of the field to change the month, day, or year. Click the portion of the date you want to change and then select the up or down increment arrow.

Note The date entered here does not override the effective date specified in the Effective Date pane. For example, if your effective date is next month and you set up an interval of 30 days calculated from today's date, the task can not run prior to its effective date next month.

---

6. When you have finished setting up a schedule, select OK. The Scheduling Service dialog box accepts the changes you made and closes.

Scheduling a Task for Certain Days of Each Month

Some tasks require a run cycle that is based on the day of the month. For example, you might want to back up certain financial transactions that occur on the first or the 15th day of each month. You can also schedule a task to run on the last day of each month, whether it falls on the 31st, 30th, 29th, or the 28th.

The schedule begins with the current month and runs each month on the days you select during the specified time window.

▼ **To select days each month on which a task runs:**

1. Open the Scheduling Service dialog box.
2. From the schedule options list in the left pane, select the Days of the Month option.

   The Days of the Month pane is displayed.

Figure 131. Days of the Month Pane

3. Select the button that corresponds to the day you want to include in your schedule. The button remains pushed in after it is selected.

   Continue selecting a button for each day you want to schedule. If you make a mistake, select the button again to remove the day from your schedule.

4. To choose all days of the month, select the Set All button. The Clear All button clears all days that have been selected.

5. To schedule your task for the last day of the month regardless of the actual date, select the Last Day checkbox.

Note If you select buttons for the 29th, 30th, or 31st days, the task does not run on those days during months that do not contain the specified number. If you want a task always to run on the last day of each month, select the Last Day checkbox.

6. When you have finished setting up a schedule, select OK.

   The Scheduling Service dialog box accepts the changes you made and closes.

Specifying Explicit Dates for a Task

A task can run on specific dates rather than follow a recurring schedule, and you can add specific dates to a recurring schedule. You can schedule dates in any month of any year.

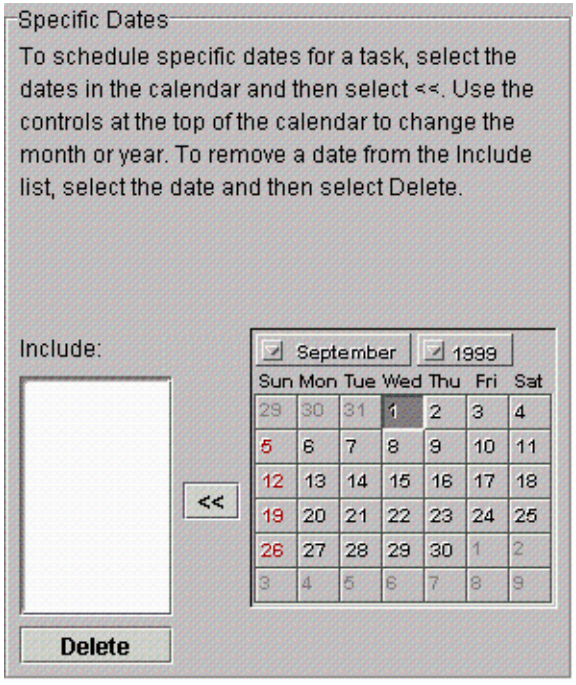
Scheduled tasks run during the time window specified for the task.

▼ To schedule specific dates:

1. Open the Scheduling Service dialog box.
2. From the schedule options list in the left pane, select the Specific Dates option.

   The Specific Dates pane is displayed.

Figure 132. Specific Dates Pane



   This pane contains a calendar used to select dates.

3. If applicable, change the calendar month or year in which you want the task to run. See “Navigating the Specific Dates Calendar” on page 574 for details.
4. Select the dates you want and then select <<. For help selecting dates, see “Selecting Specific Dates in the Calendar” on page 573.

   The dates are listed in the Include listbox.

Note If you make a mistake, you can remove a date from the Include listbox. Select the date and then select Delete.

5. When finished, select OK.

The Scheduling Service dialog box accepts your changes and closes.

Excluding Specific Dates from a Schedule

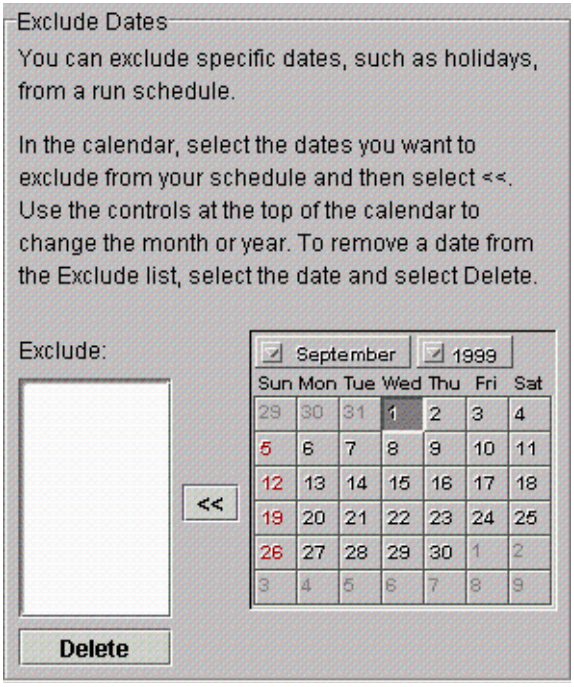
You can remove specific dates, for example holidays, from a run schedule if you don't want the task to run on those dates.

▼ **To exclude schedule specific dates from a schedule:**

1. Open the Scheduling Service dialog box.
2. From the schedule options tree in the left pane, select the Exclude Specific Dates option.

   The Exclude Dates pane is displayed.

Figure 133. Exclude Dates Pane



3. If applicable, change the calendar month or year in which you want the task to run. See "Navigating the Specific Dates Calendar" on page 574 for details.
4. Select the dates you want to exclude and then select <<. For help selecting dates, see "Selecting Specific Dates in the Calendar" on page 573.

   The dates are listed in the Exclude listbox.

Note If you made a mistake, you can remove a date from the Exclude listbox. Select the date and then select Delete.

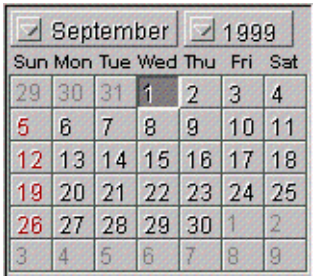
5. When finished, select OK.

   The Scheduling Service dialog box accepts your changes and closes.

Selecting Specific Dates in the Calendar

The Specific Dates pane contains a calendar used to select specific dates for a schedule. Similarly, the Exclude Specific Dates pane contains a calendar used to exclude specific dates from a schedule.

Figure 134. Specific Dates Calendar



The following steps describe how to select one or more dates in the calendar.

▼ **To select dates in the calendar:**

- To choose an individual date, select the date and select <<. You can continue selecting individual dates and clicking << for each.
- To choose multiple dates in a month, select the first date, press Ctrl, and select each additional date. To deselect a date, hold Ctrl and click the date again.

  When the highlighted dates are correct, select <<.
- To choose dates in multiple months, do the following:

  a. Select each date you want to schedule for the current month and select <<.

  b. Change to the next month or year you want to schedule for this task. See “Navigating the Specific Dates Calendar” on page 574 for details.

  c. Select the dates you want and select **<<**.

  d. Repeat the process for each month or year during which you want the task to run.

Navigating the Specific Dates Calendar

The calendar in the Specific Dates and Exclude Dates panes allows you to select specific days. Using the calendar, you can schedule exclude dates in any month or year.

Figure 135. Specific Dates Calendar

The procedures that follow describe ways to navigate within the calendar.

▼ **To change the month:**

1. Select the month name.

   A pop-up menu is displayed.

2. Choose the month you want.

▼ **To change the year:**

1. Select the year.

   A pop-up menu is displayed.

2. Choose the year you want.

Glossary

ad method

designates the method name for migrating to alternate magnetic disk.

alternate level

pertains to the second copy migrated. See primary level.

API

application programming interface.

archive

the process of backing up files and directories to a storage unit and then deleting the original files and directories.

backup

the process of copying and saving files and directories to storage media.

badness

the calculated value for a file that determines whether or not it is selected to be migrated.

badness, move

the calculated value for a file that determines whether or not it is selected to be moved from one migration level to another.

badness, purge

the calculated value for a file that determines whether or not it is selected to be purged from premigration.

cache

the process of copying migrated files back to the managed file system for access.

caching delay

the time an application is blocked after accessing migrated data and before the data is cached.

class, NetBackup

defines the backup policy for a group of one or more clients that have similar backup requirements.

concurrent recording

the process of copying data to more than one storage device at the same time.

configuration file (migconf)

a file that contains the migration parameters for a single managed file system or managed directory.

configuration file, global (migconfg)

a file that defines a collection of managed file systems (mount points) along with their attributes, specifying them as separate HSMDEV entries.

consolidation, volume

the process of moving active and obsolete files from volumes to be recycled to other volumes, and updating the FHDB and VOLDB.

ct method

designates a method name for migrating to tape, defaulted as 8mm double density technology.

daemon

a program that runs in the background and performs some task (for example, starting other programs when they are needed).

daemon, migd

the migration daemon.

daemon, migrd

the VSM-Java daemon.

daemon, migvold

the volume daemon.

defragment, directory

the process of caching previously migrated data in a directory before grouping that directory and migrating the files together to a minimal number of tapes, which reduces the mount and search time whenever the grouped directory is cached.

demand delay parameter

the time in seconds a mount request waits before VSM unmounts a similar unused volume.

device manager

the part of media manager that allocates or deallocates drives based on availability.

directory group

a directory that has been grouped so its files and those in its subdirectories will be premigrated (and cached) as a group.

dk method

designates the method name for migrating to disk file (used only for premigration).

DMAPI

Data Management Application Programming Interface.

dt method

designates a method name for migrating to tape, defaulted as DLT 2000 native technology.

empty volume

a volume that contains no active or obsolete files. See recycling.

ENOSPC

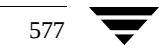
no space left on device.

export

the process of removing migrated files from one VSM managed file system that will be imported to another VSM managed file system. See import.

file handle

a unique sequence number that VSM uses to identify migrated files and unmodified cached files. File handles are stored in the file-handle database (FHDB) for the file system.



FHDB

the file-handle database. Each file system can use a separate file-handle database or several file systems can share a database.

free space

the space in the managed file system that is unused.

freespace parameter

sometimes referred to as high-water mark.

ft method

designates the method name for migrating to a remote volume using FTP.

FTP

file transfer protocol.

global configuration file - see configuration file, global (migconfg)

global-stop file - see stop file, global

granule

a configurable size attribute VERITAS Storage Migrator uses for dividing files before migrating them to secondary storage. Each granule has a unique file handle.

grouped directory- see directory group

GUI

graphical user interface. This release of VSM uses two types of graphical user interfaces: Java-based and Motif-based. The Motif-based GUI (`xhsmadm`) will not be supported in future releases of VSM.

handle - see file handle

hierarchical storage management

the process of automatically migrating selected files from a managed file system to specified migration levels on secondary storage while maintaining transparent user access to those files.

hierarchy

a collection of managed file systems (mount points) along with their attributes. See HSMDEV entry and configuration file, global (`migconfg`).

high-water mark

disk utilization where VSM begins migration operations: when the percentage of free space falls below the high-water mark.

hint parameter

the volume set availability: library, operator, or vault. See storage method.

HSM

an abbreviation that represents both VERITAS Storage Migrator and VERITAS Storage Migrator Remote; may also be referred to as *VSM*. See also *Hierarchical Storage Management*.

HSMDEV entry

an entry in the global configuration file specifying the attributes of a managed file system or managed directory.

hsmname parameter

the name assigned to an HSMDEV entry.

.IHAND file

the inode to handle file, containing inode and handle information about migrated files. (nonkernel-based implementations only)

I/O

input/output.

import

the process of adding migrated files to one VSM managed file system that had been exported from another VSM managed file system. See export.

inode

a data structure that defines the existence of a single file.

kernel

the nucleus of an operating system.

level, migration

the numbered level associated with each storage method. Multi-level migration with VERITAS Storage Migrator has up to eight migration levels.

low-water mark

disk utilization where VSM stops selecting files for migration: when the percentage of free space reaches the low-water mark.

managed directory

a directory managed by VSM.

managed file system

a file system managed by VSM.

managed server

the server upon which the managed file systems reside, and where VSM software executes.

media

the physical magnetic tapes, optical discs, or magnetic disks upon which data is stored.

Media Manager

software that provides device management and removable media management of tapes and optical discs.

method name

the name assigned to a set of parameters referring to storage device and media. See storage method.

migrate

the process of copying files to secondary storage while retaining the file names in the managed file system.

migrate file, global

a list of files or directories of files that the administrator eants VSM to select for automatic migration.

migrate file, local

a list of files or directories of files that a user wants VSM to select for automatic migration.

mount point

the point where a file system on a disk logically connects to a system's directory structure so the file system is available to users and applications.

mt method

designates a method name for migrating to tape, defaulted as 4mm DAT DDS-1 technology.

multilevel migration

the process of moving migrated files to and from up to eight migration levels with VERITAS Storage Migrator.

MUM

minimized unmounting of storage media following read operations.

name parameter - see hsmname parameter.

nb method

designates the method name for using NetBackup to make copies of files for migration.

NFS

network file system.

offline storage

storage media not physically loaded in a storage unit.

op method

designates a method name for migrating to optical disc as tape with random seek, defaulted as rewritable optical disc.

ow method

designates a method name for migrating to optical disc as tape with random seek, defaulted as WORM optical disc.

.PAIN file

the parallel inode file, entries of which are indexed by inode number and show the migration status of each file migrated from the file system. (kernel-based implementations only)

partial file caching

a process which allows `read` access to a migrated file without caching the entire file.

premigration

the first step in the migration process where selected files are moved to the premigration directory within the file system.

primary level

pertains to the first copy migrated. See alternate level.

pseudodevice

a UNIX device with a `/dev` entry and a device driver, but requiring no real device for I/O.

purge

the process of deleting files from the premigration directory after all migrated copies have been made.

purge mark

disk utilization where VSM stops purging files from the premigration directory: when the percentage of free space reaches the purge mark.

quota parameter

the maximum number of bytes that each user can restrict from migration.

recycling

the process of recovering wasted space on storage media by reregistering empty volumes.

remote volume server

the server upon which the remote volume resides.

remote storage

storage of migrated files on a remote volume server. See ft method and ad method.

restore

the process of recovering selected files and directories from a previous backup and returning them to their original directory locations (or to an alternate directory).

secondary storage

storage of migrated files on storage units connected locally to the the managed server.

slice, configured

a portion of the front of a file retained on disk even when the file is completely migrated.

slice, effective

that variable portion of a migrated file which is partially cached to disk.

stop file, global

a list of files or directories of files that the administrator does not want to migrate.

stop file, local

a list of files or directories of files that a user does not want to migrate.

storage method

the manner in which migrated files are copied to different migration levels in secondary storage. A storage method consists of one or more stripes.

storage unit

refers to the type of storage device on which VSM or NetBackup stores files. It may be a robotic device or a single tape drive.

stripe

the set of parameters used to define a storage method: method name, volume set number, volume set availability (hint), and pool name. A storage method consists of one or more stripes.

thresholds

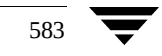
the parameters used in the VSM-Java GUI for selecting files to be migrated, moved, or purged. See badness, badness (move), and badness (purge).

threshold parameter

disk utilization in kernel-based implementations where automatic migration operations begin for the kernel: when the percentage of free space falls below the threshold. See high-water mark

unmount delay parameter

the time in seconds a volume that is mounted in read mode remains mounted pending another read request.



VOLDB

the VSM volume database, which contains attributes of each volume registered with VSM.

volume

a tape, optical disc surface, or magnetic disk partition that has been registered and labeled.

volume manager - see Media Manager

volume pool

a group of volumes to be used by a single application and protected from access by other applications and users.

volume set

one or more volumes sharing the same method name and volume set number, e.g. op.1.

volume set number

the number assigned to a volume when media is registered (labeled). See storage method.

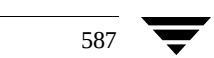
VSM

an abbreviation that represents both VERITAS Storage Migrator and VERITAS Storage Migrator Remote; may also be referred to as *HSM*. See also *Hierarchical Storage Management*.

WORM

write once, read many (optical discs).

Index



G