Solstice DiskSuite 4.2.1 Reference Guide

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900
U.S.A.

Part Number 806–3204–10
February 2000

Please Recycle

Contents

Tables

Figures

Preface

Solstice™ DiskSuite™ 4.2.1 is a software product that manages data and disk drives.

DiskSuite runs on all SPARC™ systems running Solaris™ 8 and on all x86 systems running Solaris 8.

DiskSuite's diskset feature is supported only on the SPARC platform edition of Solaris. This feature is not supported on x86 systems.

Caution - If you do not use DiskSuite correctly, you can destroy data. As a minimum safety precaution, make sure you have a current backup of your data before using DiskSuite.

Who Should Use This Book

This book targets system administrators and others who manage disk storage.

How This Book Is Organized

This manual is organized as follows:

Chapter 1 gives an overview of DiskSuite and various DiskSuite objects, such as metadevices.

Chapter 2 gives an overview of DiskSuite metadevices.

Chapter 3 describes DiskSuite hot spares and hot spare pools.

Chapter 4 describes the DiskSuite graphical user interface.

Chapter 5 describes shared disksets.

Chapter 6 describes how to use various DiskSuite files to perform specific functions.

Chapter 7 provides configuration and planning information for using DiskSuite.

Appendix A describes DiskSuite Tool's error, status, and log messages, and the command line error and log messages.

Appendix B describes how to upgrade to later versions of Solaris while using DiskSuite metadevices.

Glossary provides definitions of DiskSuite terminology.

Ordering Sun Documents

The Sun Software Shop stocks select manuals from Sun Microsystems, Inc. You can purchase individual printed manuals and AnswerBook2™ CDs.

For a list of documents and how to order them, visit the Software Shop at `http://www.sun.com/software/shop/`.

Accessing Sun Documentation Online

The docs.sun.com℠ Web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is `http://docs.sun.com`.

Related Books

Sun documentation related to DiskSuite and disk maintenance and configuration includes:

- *Solstice DiskSuite 4.2.1 User's Guide*
- *Solstice DiskSuite 4.2.1 Installation and Product Notes*
- *System Administration Guide, Volume I*

What Typographic Changes Mean

The following table describes the typographic changes used in this book.

TABLE P–1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. machine_name% You have mail.
`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name% su` `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename*.
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide*. These are called *class* options. You *must* be root to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	Prompt
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	`#`

CHAPTER 1

Introduction to DiskSuite

This chapter explains the overall structure of DiskSuite. Use the following table to proceed directly to the section that provides the information you need.

What Does DiskSuite Do?

DiskSuite is a software product that enables you to manage large numbers of disks and the data on those disks. Although there are many ways to use DiskSuite, most tasks include:

- Increasing storage capacity
- Increasing data availability

In some instances, DiskSuite can also improve I/O performance.

How Does DiskSuite Manage Disks?

DiskSuite uses virtual disks to manage physical disks and their associated data. In DiskSuite, a virtual disk is called a *metadevice*.

A metadevice is functionally identical to a physical disk in the view of an application. DiskSuite converts I/O requests directed at a metadevice into I/O requests to the underlying member disks.

DiskSuite's metadevices are built from slices (disk partitions). An easy way to build metadevices is to use the graphical user interface, *DiskSuite Tool*, that comes with DiskSuite. DiskSuite Tool presents you with a view of all the slices available to you. By dragging slices onto metadevice objects, you can quickly assign slices to metadevices. You can also build and modify metadevices using DiskSuite's command line utilities.

If, for example, you want to create more storage capacity, you could use DiskSuite to make the system treat a collection of many small slices as one larger slice or device. After you have created a large metadevice from these slices, you can immediately begin using it just as any "real" slice or device.

For a more detailed discussion of metadevices, see "Metadevices" on page 21.

DiskSuite can increase the reliability and availability of data by using mirrors (copied data) and RAID5 metadevices. DiskSuite's hot spares can provide another level of data availability for mirrors and RAID5 metadevices.

Once you have set up your configuration, you can use DiskSuite Tool to report on its operation. You can also use DiskSuite's SNMP trap generating daemon to work with a network monitoring console to automatically receive DiskSuite error messages.

DiskSuite Tool

DiskSuite Tool is a graphical user interface for setting up and administering a DiskSuite configuration. The command to start DiskSuite Tool is:

```
# metatool &
```

DiskSuite Tool provides a graphical view of DiskSuite objects—metadevices, hot spare pools, and the MetaDB object for the metadevice state database. DiskSuite Tool uses drag and drop manipulation of DiskSuite objects, enabling you to quickly configure your disks or change an existing configuration.

DiskSuite Tool provides graphical views of both physical devices and metadevices, helping simplify storage administration. You can also perform tasks specific to administering SPARCstorage™ Arrays using DiskSuite Tool.

However, DiskSuite Tool cannot perform all DiskSuite administration tasks. You must use the command line interface for some operations (for example, creating and administering disksets).

To learn more about using DiskSuite Tool, refer to Chapter 4.

Command Line Interface

Listed here are all the commands you can use to administer DiskSuite. For more detailed information, see the man pages.

TABLE 1–1 Command Line Interface Commands

DiskSuite Command	Description
`growfs(1M)`	Expands a UFS file system in a non-destructive fashion.
`mdlogd(1M)`	The `mdlogd` daemon and `mdlogd.cf` configuration file enable DiskSuite to send generic SNMP trap messages.
`metaclear(1M)`	Deletes active metadevices and hot spare pools.
`metadb(1M)`	Creates and deletes state database replicas.
`metadetach(1M)`	Detaches a metadevice from a mirror, or a logging device from a trans metadevice.
`metahs(1M)`	Manages hot spares and hot spare pools.
`metainit(1M)`	Configures metadevices.
`metaoffline(1M)`	Places submirrors offline.
`metaonline(1M)`	Places submirrors online.
`metaparam(1M)`	Modifies metadevice parameters.
`metarename(1M)`	Renames and switches metadevice names.

TABLE 1–1 Command Line Interface Commands *(continued)*

DiskSuite Command	Description
`metareplace(1M)`	Replaces slices of submirrors and RAID5 metadevices.
`metaroot(1M)`	Sets up system files for mirroring root (/).
`metaset(1M)`	Administers disksets.
`metastat(1M)`	Displays status for metadevices or hot spare pools.
`metasync(1M)`	Resyncs metadevices during reboot.
`metatool(1M)`	Runs the DiskSuite Tool graphical user interface.
`metattach(1M)`	Attaches a metadevice to a mirror, or a logging device to a trans metadevice.

Overview of DiskSuite Objects

The three basic types of objects that you create with DiskSuite are metadevices, state database replicas, and hot spare pools. Table 1–2 gives an overview of these DiskSuite objects.

TABLE 1–2 Summary of DiskSuite Objects

DiskSuite Object	What Is It?	Why Use It?	For More Information, Go To ...
Metadevice (simple, mirror, RAID5, trans)	A group of physical slices that appear to the system as a single, logical device	To increase storage capacity and increase data availability.	“Metadevices” on page 21
Metadevice state database (state database replicas)	A database that stores information on disk about the state of your DiskSuite configuration	DiskSuite cannot operate until you have created the metadevice state database replicas.	“Metadevice State Database and State Database Replicas” on page 24
Hot spare pool	A collection of slices (hot spares) reserved to be automatically substituted in case of slice failure in either a submirror or RAID5 metadevice	To increase data availability for mirrors and RAID5 metadevices.	“Hot Spare Pools” on page 27

Note - DiskSuite Tool, DiskSuite’s graphical user interface, also refers to the graphical representation of metadevices, the metadevice state database, and hot spare pools as “objects.”

Metadevices

A *metadevice* is a name for a group of physical slices that appear to the system as a single, logical device. Metadevices are actually pseudo, or virtual, devices in standard UNIX terms.

You create a metadevice by using concatenation, striping, mirroring, RAID level 5, or UFS logging. Thus, the types of metadevices you can create are concatenations, stripes, concatenated stripes, mirrors, RAID5 metadevices, and *trans metadevice*s.

DiskSuite uses a special driver, called the *metadisk driver*, to coordinate I/O to and from physical devices and metadevices, enabling applications to treat a metadevice like a physical device. This type of driver is also called a logical, or pseudo, driver.

You can use either the DiskSuite Tool graphical user interface or the command line utilities to create and administer metadevices.

Table 1–3 summarizes the types of metadevices:

TABLE 1–3 Types of Metadevices

Metadevice	Description
Simple	Can be used directly, or as the basic building blocks for mirrors and trans devices. There are three types of simple metadevices: stripes, concatenations, and concatenated stripes. Simple metadevices consist only of physical slices. By themselves, simple metadevices do not provide data redundancy.
Mirror	Replicates data by maintaining multiple copies. A mirror is composed of one or more simple metadevices called submirrors.
RAID5	Replicates data by using parity information. In the case of missing data, the missing data can be regenerated using available data and the parity information. A RAID5 metadevice is composed of slices. One slice's worth of space is allocated to parity information, but it is distributed across all slices in the RAID5 metadevice.
Trans	Used to log a UFS file system. A trans metadevice is composed of a master device and a logging device. Both of these devices can be a slice, simple metadevice, mirror, or RAID5 metadevice. The master device contains the UFS file system.

How Are Metadevices Used?

You use metadevices to increase storage capacity and data availability. In some instances, metadevices can also increase I/O performance. Functionally, metadevices behave the same way as slices. Because metadevices look like slices, they are transparent to end users, applications, and file systems. Like physical devices, metadevices are accessed through block or raw device names. The metadevice name changes, depending on whether the block or raw device is used. See "Metadevice Conventions" on page 23 for details about metadevice names.

You can use most file systems commands (`mount(1M)`, `umount(1M)`, `ufsdump(1M)`, `ufsrestore(1M)`,and so forth) on metadevices. You cannot use the `format(1M)` command, however. You can read, write, and copy files to and from a metadevice, as long as you have a file system mounted on the metadevice.

SPARC and x86 systems can create metadevices on the following disk drives:

- **SPARC –** IPI, SCSI devices, and SPARCStorage Array drives
- **x86 –** SCSI and IDE devices

Metadevice Conventions

- **How are metadevices named?**

  Metadevice names begin with the letter “d” followed by a number (for example, `d0` as shown in Table 1–4).

- **What are the default metadevice names?**

  DiskSuite has 128 default metadevice names from 0-127. Table 1–4 shows some example metadevice names.

TABLE 1–4 Example Metadevice Names

`/dev/md/dsk/d0`	Block metadevice d0
`/dev/md/dsk/d1`	Block metadevice d1
`/dev/md/rdsk/d126`	Raw metadevice d126
`/dev/md/rdsk/d127`	Raw metadevice d127

- **Can metadevice names be abbreviated?**

  Yes. Instead of specifying the full metadevice name, such as `/dev/md/dsk/d1`, you can use `d1`. You can use either the command line interface or DiskSuite Tool to name metadevices.

- **What is the maximum number of metadevices possible?**

  1024 (though the default number of metadevices is 128). You can increase the number of default metadevices by editing the `/kernel/drv/md.conf` file. See “System and Startup Files” on page 29 for more information on this file.

- **Where are metadevice names stored?**

  Like physical slices, metadevices have logical names which appear in the file system. Logical metadevice names have entries in `/dev/md/dsk` (for block devices) and `/dev/md/rdsk` (for raw devices).

- **Can metadevices be renamed?**

  Yes. DiskSuite enables you to rename a metadevice at any time, as long as the name being used is not in use by another metadevice, and as long as the metadevice itself is not in use. For a file system, make sure it is not mounted or being used as `swap`. Other applications using the raw device, such as a database, should have their own way of stopping access to the data.

  You can use either DiskSuite Tool (via a metadevice’s Information window) or the command line (the `metarename(1M)` command) to rename metadevices.

The `metarename(1M)` command with the –x option can “switch” metadevices that have a parent-child relationship. Refer to *Solstice DiskSuite 4.2.1 User’s Guide* for procedures to rename and switch metadevices.

Example — Metadevice Consisting of Two Slices

Figure 1–1 shows a metadevice “containing” two slices, one each from Disk A and Disk B. An application or UFS will treat the metadevice as if it were one physical disk. Adding more slices to the metadevice will increase its capacity.

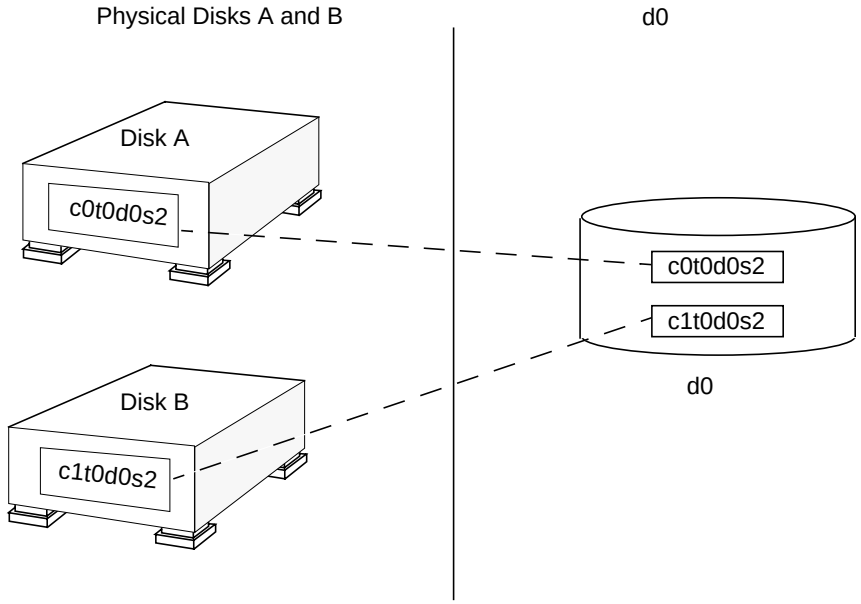

Figure 1–1 Relationship Among a Metadevice, Physical Disks, and Slices

Metadevice State Database and State Database Replicas

A *metadevice state database* (often simply called the state database) is a database that stores information on disk about the state of your DiskSuite configuration. The metadevice state database records and tracks changes made to your configuration. DiskSuite automatically updates the metadevice state database when a configuration or state change occurs. Creating a new metadevice is an example of a configuration change. A submirror failure is an example of a state change.

The metadevice state database is actually a collection of multiple, replicated database copies. Each copy, referred to as a *state database replica*, ensures that the data in the database is always valid. Having copies of the metadevice state database protects against data loss from single points-of-failure. The metadevice state database tracks the location and status of all known state database replicas.

DiskSuite cannot operate until you have created the metadevice state database and its state database replicas. It is necessary that a DiskSuite configuration have an operating metadevice state database.

When you set up your configuration, you have two choices for the location of state database replicas. You can place the state database replicas on dedicated slices. Or you can place the state database replicas on slices that will later become part of metadevices. DiskSuite recognizes when a slice contains a state database replica, and automatically skips over the portion of the slice reserved for the replica if the slice is used in a metadevice. The part of a slice reserved for the state database replica should not be used for any other purpose.

You can keep more than one copy of a metadevice state database on one slice, though you may make the system more vulnerable to a single point-of-failure by doing so.

How Does DiskSuite Use State Database Replicas?

The state database replicas ensure that the data in the metadevice state database is always valid. When the metadevice state database is updated, each state database replica is also updated. The updates take place one at a time (to protect against corrupting all updates if the system crashes).

If your system loses a state database replica, DiskSuite must figure out which state database replicas still contain non-corrupted data. DiskSuite determines this information by a *majority consensus algorithm*. This algorithm requires that a majority (half + 1) of the state database replicas be available before any of them are considered non-corrupt. It is because of this majority consensus algorithm that you must create at least three state database replicas when you set up your disk configuration. A consensus can be reached as long as at least two of the three state database replicas are available.

To protect data, DiskSuite will not function if a majority (half + 1) of all state database replicas is not available. The algorithm, therefore, ensures against corrupt data.

The majority consensus algorithm guarantees the following:

- The system will stay running with exactly half or more state database replicas.
- The system will panic if more than half the state database replicas are not available.
- The system will not reboot without one more than half the total state database replicas.

Note - When the number of state database replicas is odd, DiskSuite computes the majority by dividing the number in half, rounding down to the nearest integer, then adding 1 (one). For example, on a system with seven replicas, the majority would be four (seven divided by two is three and one-half, rounded down is three, plus one is four).

During booting, DiskSuite ignores corrupted state database replicas. In some cases DiskSuite tries to rewrite state database replicas that are bad. Otherwise they are ignored until you repair them. If a state database replica becomes bad because its underlying slice encountered an error, you will need to repair or replace the slice and then enable the replica.

If all state database replicas are lost, you could, in theory, lose all data that is stored on your disks. For this reason, it is good practice to create enough state database replicas on separate drives and across controllers to prevent catastrophic failure. It is also wise to save your initial DiskSuite configuration information, as well as your disk partition information.

Refer to *Solstice DiskSuite 4.2.1 User's Guide* for information on adding additional state database replicas to the system, and on recovering when state database replicas are lost.

Metadevice State Database Conventions

- **What is the size of a state database replica?**

  By default, 517 Kbytes or 1034 disk blocks of a slice. Because your disk slices may not be that small, you may want to resize a slice to hold the state database replica. (See *Solstice DiskSuite 4.2.1 User's Guide* for more information on resizing a slice.)

- **What are the minimum number of state database replicas required?**

  Three (3), preferably spread out across at least three disks (to avoid a single point-of-failure). DiskSuite does not operate with less than a majority.

- **What are the maximum number of state database replicas possible?**

  50.

- **Where are state database replicas created?**

  You can create state database replicas on slices not in use.

  You cannot create state database replicas on existing file systems, root (`/`), `/usr`, and `swap`. If necessary, you can create a new slice (provided a slice name is available) by allocating space from `swap` and put state database replicas on that new slice. See *Solstice DiskSuite 4.2.1 User's Guide* for more information.

- **Can I create a state database replica on a slice that will be part of a metadevice?**

Yes, but you must create it before adding the slice to the metadevice. You can also create a state database replica on a logging device. DiskSuite reserves the starting part of the slice for the state database replica.

- **Can I place more than one state database replica on a single disk drive?**

  In general, it is best to distribute state database replicas across slices, drives, and controllers, to avoid single points-of-failure.

  If you have two disks, create two state database replicas on each disk.

- **What happens if a slice that contains a state database replica becomes errored?**

  The rest of your configuration should remain in operation. DiskSuite finds a good state database (as long as there are at least half + 1 valid state database replicas).

- **What happens when state database replicas are repaired?**

  When you manually repair or enable state database replicas, DiskSuite updates them with valid data.

Hot Spare Pools

A *hot spare pool* is a collection of slices (*hot spares*) reserved by DiskSuite to be automatically substituted in case of a slice failure in either a submirror or RAID5 metadevice. Hot spares provide increased data availability for mirrors and RAID5 metadevices. You can create a hot spare pool with either DiskSuite Tool or the command line interface.

How Do Hot Spare Pools Work?

When errors occur, DiskSuite checks the hot spare pool for the first available hot spare whose size is equal to or greater than the size of the slice being replaced. If found, DiskSuite automatically resyncs the data. If a slice of adequate size is not found in the list of hot spares, the submirror or RAID5 metadevice that failed is considered errored. For more information, see Chapter 3.

Metadevice and Disk Space Expansion

DiskSuite enables you to expand a metadevice by adding additional slices.

Mounted or unmounted UFS file systems contained within a metadevice can be expanded without having to halt or back up your system. (Nevertheless, backing up

your data is always a good idea.) After the metadevice is expanded, you grow the file system with the `growfs(1M)` command.

After a file system is expanded, it cannot be decreased. Decreasing the size of a file system is a UFS limitation.

Applications and databases using the raw metadevice must have their own method to “grow” the added space so that the application or database can recognize it. DiskSuite does not provide this capability.

You can expand the disk space in metadevices in the following ways:

1. Adding a slice to a stripe or concatenation.
2. Adding multiple slices to a stripe or concatenation.
3. Adding a slice or multiple slices to all submirrors of a mirror.
4. Adding one or more slices to a RAID5 device.

You can use either DiskSuite Tool or the command line interface to add a slice to an existing metadevice.

Note - When using DiskSuite Tool to expand a metadevice that contains a UFS file system, the `growfs(1M)` command is run automatically. If you use the command line to expand the metadevice, you must manually run the `growfs(1M)` command.

The `growfs(1M)` Command

The `growfs(1M)` command expands a UFS file system without loss of service or data. H‘owever, write-access to the metadevice is suspended while the `growfs(1M)` command is running. You can expand the file system to the size of the slice or the metadevice that contains the file system.

The file system can be expanded to use only part of the additional disk space by using the `-s` *size* option to the `growfs(1M)` command.

Note - When expanding a mirror, space is added to the mirror’s underlying submirrors. Likewise, when expanding a trans metadevice, space is added to the master device. The `growfs(1M)` command is then run on the mirror or the trans metadevice, respectively. The general rule is that space is added to the underlying devices(s), and the `growfs(1M)` command is run on the top-level device.

System and Startup Files

This section explains the files necessary for DiskSuite to operate correctly. For the most part, you do not have to worry about these files because DiskSuite accesses (updates) them automatically (with the exception of `md.tab`).

- `/etc/lvm/mddb.cf`

  A file that records the locations of state database replicas. When state database replica locations change, DiskSuite makes an entry in the `mddb.cf` file that records the locations of all state databases. Similar information is entered into the `/etc/system` file.

- `/etc/lvm/md.tab`

  An input file that you can use along with the command line utilities `metainit(1M)`, `metadb(1M)`, and `metahs(1M)` to create metadevices, state database replicas, or hot spares. A metadevice, group of state database replicas, or hot spare may have an entry in this file.

Note - The configuration information in the `/etc/lvm/md.tab` file may differ from the current metadevices, hot spares, and state database replicas in use. It is only used at metadevice creation time, not to recapture the DiskSuite configuration at boot.

- `/etc/lvm/md.cf`

  A backup file of a “local” diskset’s configuration. DiskSuite provides the `md.cf` file for recovery. When you change the DiskSuite configuration, DiskSuite automatically updates the `md.cf` file (except for hot sparing).

Caution - You should not directly edit either the `mddb.cf` or `md.cf` files.

- /kernel/drv/`md.conf`

  DiskSuite uses this configuration file at startup. You can edit two fields in this file: `nmd`, which sets the number of metadevices that the configuration can support, and `md_nsets`, which is the number of disksets. The default value for `nmd` is 128, which can be increased to 1024. The default value for `md_nsets` is 4, which can be increased to 32. The total number of disksets is always one less than the `md_nsets` value, because the local set is included in `md_nsets`.

- `/etc/lvm/mdlodg.cf`

  DiskSuite uses this file to control the behavior of the DiskSuite `mdlogd` SNMP trap generating daemon. It is an editable ASCII file that specifies where the SNMP trap data should be sent when the DiskSuite driver detects a specified condition.

- `/etc/rcS.d/S35lvm.init`

For automatic reloading of metadevice configuration at boot.

- `/etc/rc2.d/S95lvm.sync`

  For automatic resyncing of metadevices.

For more information on DiskSuite system files, refer to the man pages.

Disksets

A *shared diskset*, or simply *diskset*, is a set of shared disk drives containing metadevices and hot spares that can be shared exclusively but not at the same time by two hosts. Currently, disksets are only supported on SPARCstorage Array disks.

A diskset provides for data redundancy and availability. If one host fails, the other host can take over the failed host's diskset. (This type of configuration is known as a *failover configuration*.)

For more information, see Chapter 5.

CHAPTER 2

Metadevices

This chapter covers the different types of metadevices available in DiskSuite. Use the following table to proceed directly to the section that provides the information you need.

Simple Metadevices

A *simple metadevice* is a metadevice built only from slices, and is either used directly or as the basic building block for mirrors and trans metadevices. There are three kinds of simple metadevices: *concatenated metadevices*, *striped metadevices*, and *concatenated striped metadevices*.

In practice, people tend to think of two basic simple metadevices: concatenated metadevices and striped metadevices. (A concatenated stripe is simply a striped metadevice that has been "grown" from its original configuration by concatenating slices.)

Simple metadevices enable you to quickly and simply expand disk storage capacity. The drawback to a simple metadevice is that it does not provide any data

redundancy. A mirror or RAID5 metadevice can provide data redundancy. (If a single slice fails on a simple metadevice, data is lost.)

You can use a simple metadevice containing multiple slices for any file system except the following:

- Root (/)
- /usr
- swap
- /var
- /opt
- Any file system accessed during an operating system upgrade or installation

---

Note - When you mirror root (/), /usr, swap, /var, or /opt, you put the file system into a one-way concatenation (a concatenation of a single slice) that acts as a submirror. This is mirrored by another submirror, which is also a concatenation.

Concatenated Metadevice (Concatenation)

A *concatenated metadevice*, or *concatenation*, is a metadevice whose data is organized serially and adjacently across disk slices, forming one logical storage unit.

You would use a concatenated metadevice to get more storage capacity by logically combining the capacities of several slices. You can add more slices to the concatenated metadevice as the demand for storage grows.

A concatenated metadevice enables you to dynamically expand storage capacity and file system sizes online. With a concatenated metadevice you can add slices even if the other slices are currently active.

Note - To increase the capacity of a striped metadevice, you would have to build a concatenated stripe (see "Concatenated Stripe" on page 36).

A concatenated metadevice can also expand any active and mounted UFS file system without having to bring down the system. In general, the total capacity of a concatenated metadevice is equal to the total size of all the slices in the concatenated metadevice. If a concatenation contains a slice with a state database replica, the total capacity of the concatenation would be the sum of the slices less the space reserved for the replica.

You can also create a concatenated metadevice from a single slice. You could, for example, create a single-slice concatenated metadevice. Later, when you need more storage, you can add more slices to the concatenated metadevice.

Concatenations have names like other metadevices (d0, d1, and so forth). For more information on metadevice naming, see Table 1–4.

Concatenated Metadevice Conventions

- **When would I create a concatenated metadevice?**

  To expand the capacity of an existing data set, such as a file system.

  Concatenation is good for small random I/O and for even I/O distribution.

- **What are the limitations to concatenation?**

  Practically speaking, none. You must use a concatenation to encapsulate root (`/`), `swap`, `/usr`, `/opt`, or `/var` when mirroring these file systems.

- **How large can a concatenated metadevice be?**

  Up to one Terabyte.

Example — Concatenated Metadevice

Figure 2–1 illustrates a concatenated metadevice made of three slices (disks).

The data blocks, or chunks, are written sequentially across the slices, beginning with Disk A. Disk A can be envisioned as containing logical chunks 1 through 4. Logical chunk 5 would be written to Disk B, which would contain logical chunks 5 through 8. Logical chunk 9 would be written to Drive C, which would contain chunks 9 through 12. The total capacity of metadevice `d1` would be the combined capacities of the three drives. If each drive were 2 Gbytes, metadevice `d1` would have an overall capacity of 6 Gbytes.

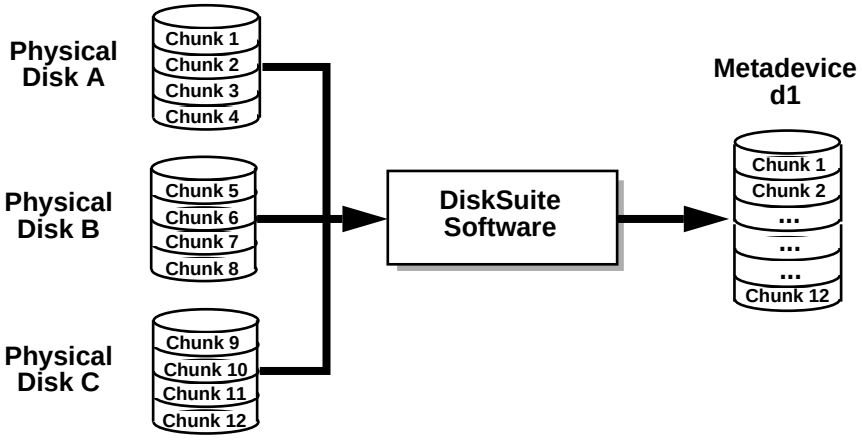

Figure 2–1 Concatenation Example

Striped Metadevice (Stripe)

A *striped metadevice*, or *stripe*, is a metadevice that arranges data across two or more slices. Striping alternates equally-sized segments of data across two or more slices, forming one logical storage unit. These segments are interleaved round-robin, so that the combined space is made alternately from each slice, in effect, shuffled like a deck of cards.

Note - Sometimes a striped metadevice is called a “stripe.” Other times, “stripe” refers to the component blocks of a striped concatenation. “To stripe” means to spread I/O requests across disks by chunking parts of the disks and mapping those chunks to a virtual device (a metadevice). Striping is also classified as RAID level 0, as is concatenation.

While striping and concatenation both are methods of distributing data across disk slices, striping alternates chunks of data across disk slices, while concatenation distributes data “end-to-end” across disk slices.

For sequential I/O operations on a concatenated metadevice, DiskSuite reads all the blocks on the first slice, then all the blocks of the second slice, and so forth.

For sequential I/O operations on a striped metadevice, DiskSuite reads all the blocks in a segment of blocks (called an *interlace*) on the first slice, then all the blocks in a segment of blocks on the second slice, and so forth.

On both a concatenation and a striped metadevice, all I/O occurs in parallel.

Striped Metadevice Conventions

- **Why would I create a striped metadevice?**

  To take advantage of the performance increases that come from accessing data in parallel and to increase capacity. Always use striped metadevices for new file systems or data sets.

  Striping enables multiple controllers to access data at the same time (parallel access). Parallel access can increase I/O throughput because all disks in the metadevice are busy most of the time servicing I/O requests.

  Striping is good for large sequential I/O and for uneven I/O.

- **What are the limitations to striping?**

  An existing file system cannot be directly converted to a striped metadevice. If you need to place a file system on a striped metadevice, you can back up the file system, create a striped metadevice, then restore the file system to the striped metadevice.

  When creating a stripe, do not use slices of unequal size, as this will result in unused disk space.

- **What is an interlace value?**

  The size, in Kbytes, Mbytes, or blocks, of the logical data chunks in a striped metadevice. Depending on the application, different interlace values can increase performance for your configuration. The performance increase comes from several disk arms doing I/O. When the I/O request is larger than the interlace size, you may get better performance.

- **What is DiskSuite's default interlace value?**

  16 Kbytes.

- **Can I set the interlace value?**

  Yes, when you create a new striped metadevice, using either the command line or DiskSuite Tool. Once you have created the striped metadevice, you cannot change the interlace value.

- **Can I set the interlace value on an existing striped metadevice?**

  No. (Though you could back up the data on it, delete the striped metadevice, create a new striped metadevice with a new interlace value, and then restore the data.)

Note - RAID5 metadevices also use an interlace value. See "RAID5 Metadevices" on page 44 for more information.

Example — Striped Metadevice

Figure 2–2 shows a striped metadevice built from three slices (disks).

When DiskSuite stripes data from the metadevice to the slices, it writes data from chunk 1 to Disk A, from chunk 2 to Disk B, and from chunk 3 to Disk C. DiskSuite then writes chunk 4 to Disk A, chunk 5 to Disk B, chunk 6 to Disk C, and so forth.

The interlace value sets the size of each chunk. The total capacity of the striped metadevice `d2` equals the number of slices multiplied by the size of the smallest slice. (If each slice in the example below were 2 Gbytes, `d2` would equal 6 Gbytes.)

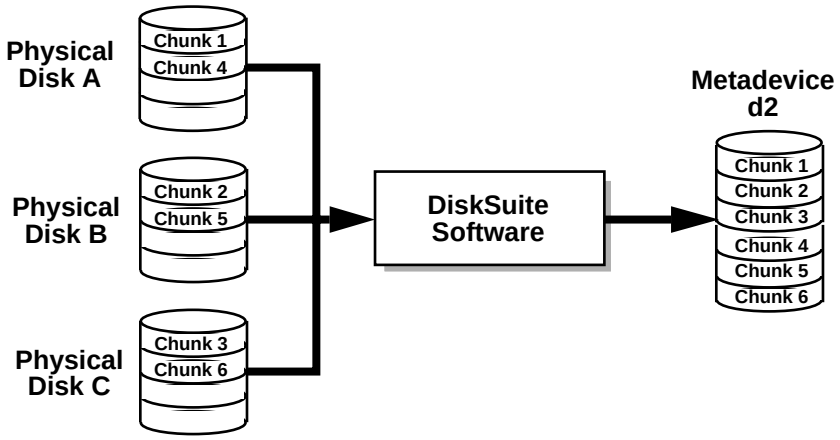

Figure 2–2 Striped Metadevice Example

Concatenated Stripe

A *concatenated stripe* is a striped metadevice that has been expanded by concatenating additional slices (stripes).

Concatenated Stripe Conventions

- **Why would I use a concatenated stripe?**

  This is the only way to expand an existing striped metadevice.

Note - If you use DiskSuite Tool to drag multiple slices into an existing striped metadevice, you are given the optional of making the slices into a concatenation or a stripe. If you use the `metattach(1M)` command to add multiple slices to an existing striped metadevice, they must be added as a stripe.

- **How do I set up the interlace value for a concatenated stripe?**

  At the stripe level, using either the Stripe Information window in DiskSuite Tool, or the `-i` option to the `metattach(1M)` command. Each stripe within the concatenated stripe can have its own interlace value. When you create a concatenated stripe from scratch, if you do not specify an interlace value for a particular stripe, it inherits the interlace value from the stripe before it.

Example — Concatenated Stripe

Figure 2–3 illustrates that `d10` is a concatenation of three stripes.

The first stripe consists of three slices, Disks A through C, with an interlace of 16 Kbytes. The second stripe consists of two slices Disks D and E, and uses an interlace of 32 Kbytes. The last stripe consists of a two slices, Disks F and G. Because no interlace is specified for the third stripe, it inherits the value from the stripe before it, which in this case is 32 Kbytes. Sequential data chunks are addressed to the first stripe until that stripe has no more space. Chunks are then addressed to the second stripe. When this stripe has no more space, chunks are addressed to the third stripe. Within each stripe, the data chunks are interleaved according to the specified interlace value.

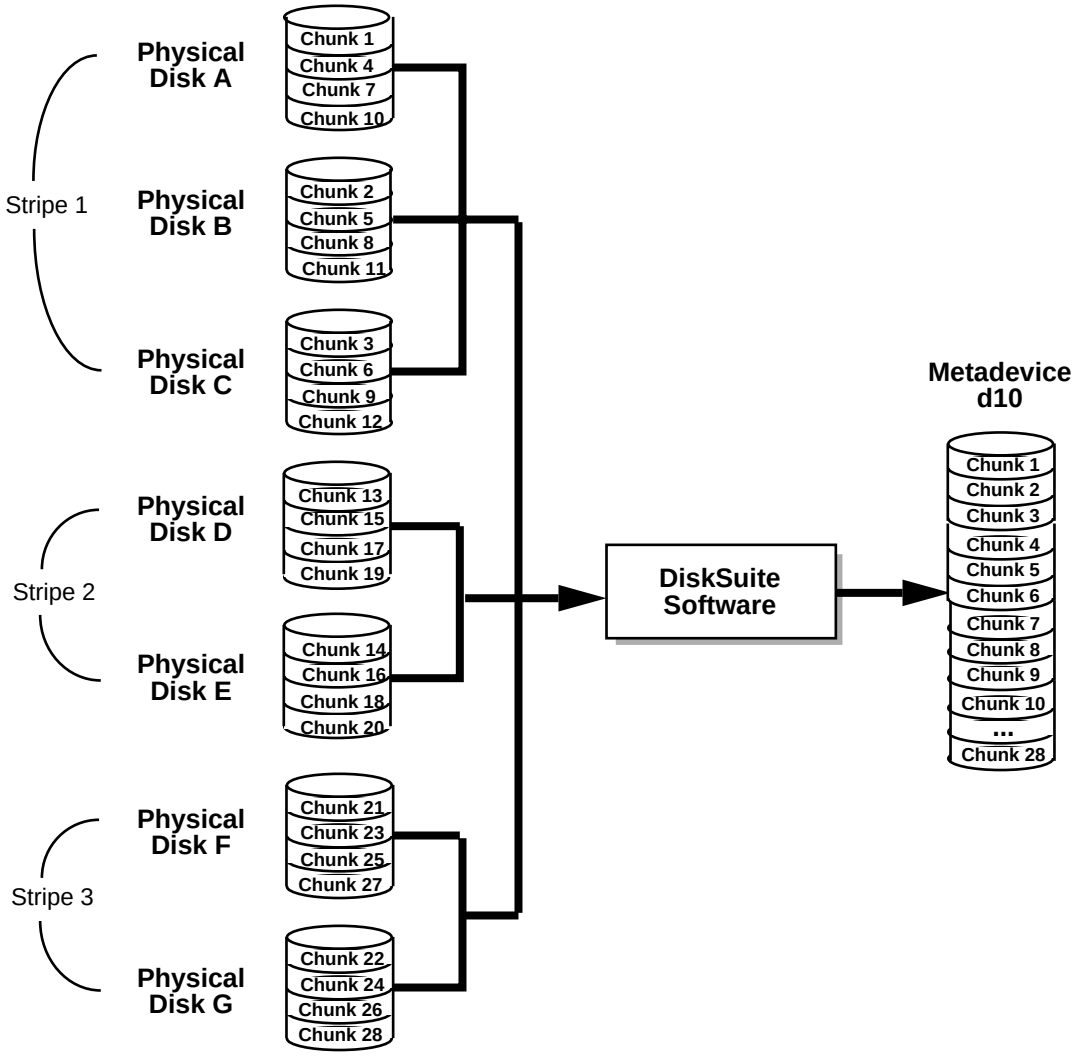

Figure 2–3 Concatenated Stripe Example

Simple Metadevices and Starting Blocks

When you create a simple metadevice of more than one slice, any slice except the first skips the first disk cylinder, if the slice starts at cylinder 0. For example, consider this output from the `metastat(1M)` command:

```
# metastat d0

d0: Concat/Stripe
    Size: 3546160 blocks
    Stripe 0: (interface: 32 blocks)
        Device              Start Block  Dbase
        c1t0d0s0                   0     No
        c1t0d1s0                   1520  No
        c1t0d2s0                   1520  No
        c1t0d2s0                   1520  No
        c1t1d0s0                   1520  No
        c1t1d1s0                   1520  No
        c1t1d2s0                   1520  No
```

In this example, stripe `d0` shows a start block for each slice except the first as block 1520. This is to preserve the disk label in the first disk sector in all of the slices except the first. The metadisk driver must skip at least the first sector of those disks when mapping accesses across the stripe boundaries. Because skipping only the first sector would create an irregular disk geometry, the entire first cylinder of these disks is skipped. This enables higher level file system software (UFS) to optimize block allocations correctly. Thus, DiskSuite protects the disk label from being overwritten, and purposefully skips the first cylinder.

The reason for not skipping the first cylinder on all slices in the concatenation or stripe has to do with UFS. If you create a concatenated metadevice from an existing file system, and add more space to it, you would lose data because the first cylinder is where the data is expected to begin.

Mirrors

A *mirror* is a metadevice that can copy the data in simple metadevices (stripes or concatenations) called *submirrors*, to other metadevices. This process is called *mirroring* data. (Mirroring is also known as RAID level 1.)

A mirror provides redundant copies of your data. These copies should be located on separate physical devices to guard against device failures.

Mirrors require an investment in disks. You need at least twice as much disk space as the amount of data you have to mirror. Because DiskSuite must write to all

submirrors, mirrors can also increase the amount of time it takes for write requests to be written to disk.

After you configure a mirror, it can be used just as if it were a physical slice.

You can also use a mirror for online backups. Because the submirrors contain identical copies of data, you can take a submirror offline and back up the data to another medium—all without stopping normal activity on the mirror metadevice. You might want to do online backups with a three-way mirror so that the mirror continues to copy data to two submirrors. Also, when the submirror is brought back online, it will take a while for it to sync its data with the other two submirrors.

You can mirror any file system, including existing file systems. You can also use a mirror for any application, such as a database. You can create a one-way mirror and attach another submirror to it later.

Note - You can use DiskSuite's hot spare feature with mirrors to keep data safe and available. For information on hot spares, see Chapter 3.

Mirrors have names like other metadevices (`d0`, `d1`, and so forth). For more information on metadevice naming, see Table 1–4. Each submirror (which is also a metadevice) has a unique device name.

Submirrors

A mirror is made of one or more stripes or concatenations. The stripes or concatenations within a mirror are called submirrors. (A mirror cannot be made of RAID5 metadevices.)

A mirror can consist of up to three (3) submirrors. (Practically, creating a two-way mirror is usually sufficient. A third submirror enables you to make online backups without losing data redundancy while one submirror is offline for the backup.)

Submirrors are distinguished from simple metadevices in that normally they can only be accessed by the mirror. The submirror is accessible only through the mirror when you attach it to the mirror.

If you take a submirror "offline," the mirror stops reading and writing to the submirror. At this point, you could access the submirror itself, for example, to perform a backup. However, the submirror is in a read-only state. While a submirror is offline, DiskSuite keeps track of all writes to the mirror. When the submirror is brought back online, only the portions of the mirror that were written (*resync regions*) are resynced. Submirrors can also be taken offline to troubleshoot or repair physical devices which have errors.

Submirrors have names like other metadevices (`d0`, `d1`, and so forth). For more information on metadevice naming, see Table 1–4.

Submirrors can be attached or detached from a mirror at any time. To do so, at least one submirror must remain attached at all times. You can force a submirror to be detached using the `-f` option to the `metadetach(1M)` command. DiskSuite Tool always “forces” a mirror detach, so there is no extra option. Normally, you create a mirror with only a single submirror. Then you attach a second submirror after creating the mirror.

Mirror Conventions

- **Why would I use a mirror?**

  For maximum data availability. The trade-off is that a mirror requires twice the number of slices (disks) as the amount of data to be mirrored.

- **How many submirrors can a mirror contain?**

  DiskSuite enables you to create up to a three-way mirror (a mirror of three submirrors). However, two-way mirrors usually provide sufficient data redundancy for most applications, and are less expensive in terms of disk drive costs.

- **Why should I always create a one-way mirror then attach additional submirrors?**

  This ensures that a mirror resync is performed so that data is consistent in all submirrors.

Example — Mirrored Metadevice

Figure 2–4 illustrates a mirror, `d2`, made of two metadevices (submirrors) `d20` and `d21`.

DiskSuite software takes duplicate copies of the data located on multiple physical disks, and presents one virtual disk to the application. All disk writes are duplicated; when reading, data only needs to be read from one of the underlying submirrors. The total capacity of mirror `d2` is the size of the smaller of the submirrors (if they are not equal sized).

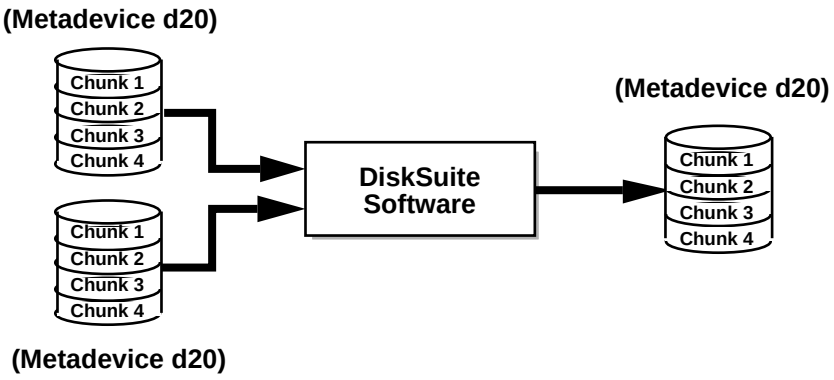

Figure 2–4 Mirror Example

Mirror Options

The following options are available to optimize mirror performance:

- Mirror read policy
- Mirror write policy
- The order in which mirrors are resynced (pass number)

You can define mirror options when you initially create the mirror, or after a mirror has been set up. For tasks related to changing these options, refer to *Solstice DiskSuite 4.2.1 User's Guide*.

Mirror Resync

Mirror resynchronization is the process of copying data from one submirror to another after submirror failures, system crashes, when a submirror has been taken offline and brought back online, or after the addition of a new submirror.

While the resync takes place, the mirror remains readable and writable by users.

A mirror resync ensures proper mirror operation by maintaining all submirrors with identical data, with the exception of writes in progress.

Note - A mirror resync is mandatory, and cannot be omitted. You do not need to manually initiate a mirror resync; it occurs automatically.

Full Mirror Resync

When a new submirror is attached (added) to a mirror, all the data from another submirror in the mirror is automatically written to the newly attached submirror. Once the mirror resync is done, the new submirror is readable. A submirror remains attached to a mirror until it is explicitly detached.

If the system crashes while a resync is in progress, the resync is started when the system reboots and comes back up.

Optimized Mirror Resync

During a reboot following a system failure, or when a submirror that was offline is brought back online, DiskSuite performs an optimized mirror resync. The metadisk driver tracks submirror regions and knows which submirror regions may be out-of-sync after a failure. An optimized mirror resync is performed only on the out-of-sync regions. You can specify the order in which mirrors are resynced during reboot, and you can omit a mirror resync by setting submirror pass numbers to 0 (zero). (See “Pass Number” on page 42 for information.)

Caution - A pass number of 0 (zero) should only be used on mirrors mounted as read-only.

Partial Mirror Resync

Following a replacement of a slice within a submirror, DiskSuite performs a partial mirror resync of data. DiskSuite copies the data from the remaining good slices of another submirror to the replaced slice.

Pass Number

The pass number, a number in the range 0-9, determines the order in which a particular mirror is resynced during a system reboot. The default pass number is one (1). Smaller pass numbers are resynced first. If 0 is used, the mirror resync is skipped. A 0 should be used only for mirrors mounted as read-only. Mirrors with the same pass number are resynced at the same time.

Mirror Read and Write Policies

DiskSuite enables different read and write policies to be configured for a mirror. Properly set read and write policies can improve performance for a given configuration.

TABLE 2–1 Mirror Read Policies

Read Policy	Description
Round Robin (Default)	Attempts to balance the load across the submirrors. All reads are made in a round-robin order (one after another) from all submirrors in a mirror.
Geometric	Enables reads to be divided among submirrors on the basis of a logical disk block address. For instance, with a two-way submirror, the disk space on the mirror is divided into two equally-sized logical address ranges. Reads from one submirror are restricted to one half of the logical range, and reads from the other submirror are restricted to the other half. The geometric read policy effectively reduces the seek time necessary for reads. The performance gained by this mode depends on the system I/O load and the access patterns of the applications.
First	Directs all reads to the first submirror. This should be used only when the device(s) comprising the first submirror are substantially faster than those of the second submirror.

TABLE 2–2 Mirror Write Policies

Write Policy	Description
Parallel (Default)	A write to a mirror is replicated and dispatched to all of the submirrors simultaneously.
Serial	Performs writes to submirrors serially (that is, the first submirror write completes before the second is started). The serial option specifies that writes to one submirror must complete before the next submirror write is initiated. The serial option is provided in case a submirror becomes unreadable, for example, due to a power failure.

Mirror Robustness

DiskSuite cannot guarantee that a mirror will be able to tolerate multiple slice failures and continue operating. However, depending on the mirror's configuration, in many instances DiskSuite can handle a multiple-slice failure scenario. As long as multiple slice failures within a mirror do not contain the same logical blocks, the mirror continues to operate. (The submirrors must also be identically constructed.)

Consider this example:

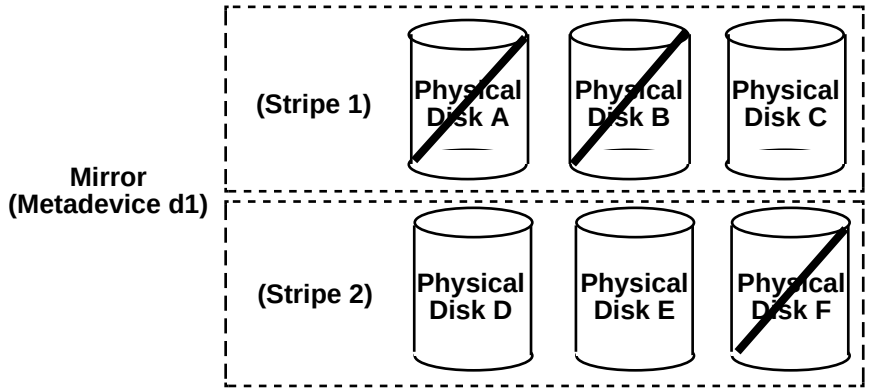

Figure 2–5 Mirror Robustness Example

Mirror `d1` consists of two stripes (submirrors), each of which consists of three identical physical disks and the same interlace value. A failure of three disks, A, B, and F can be tolerated because the entire logical block range of the mirror is still contained on at least one good disk.

If, however, disks A and D fail, a portion of the mirror's data is no longer available on any disk and access to these logical blocks will fail.

When a portion of a mirror's data is unavailable due to multiple slice errors, access to portions of the mirror where data is still available will succeed. Under this situation, the mirror acts like a single disk that has developed bad blocks; the damaged portions are unavailable, but the rest is available.

RAID5 Metadevices

RAID is an acronym for Redundant Array of Inexpensive Disks (or Redundant Array of Independent Disks).

There are seven RAID levels, 0-6, each referring to a method of distributing data while ensuring data redundancy. (RAID level 0 does not provide data redundancy, but is usually included as a RAID classification because it is the basis for the majority of RAID configurations in use.)

DiskSuite supports:

- RAID level 0 (concatenations and stripes)
- RAID level 1 (mirror)
- RAID level 5 (striped metadevice with parity information)

RAID level 5 is striping with parity and data distributed across all disks. If a disk fails, the data on the failed disk can be rebuilt from the distributed data and parity information on the other disks.

Within DiskSuite, a *RAID5 metadevice* is a metadevice that supports RAID Level 5.

DiskSuite automatically initializes a RAID5 metadevice when you add a new slice, or resyncs a RAID5 metadevice when you replace an existing slice. DiskSuite also resyncs RAID5 metadevices during rebooting if a system failure or panic took place.

RAID5 metadevices have names like other metadevices (`d0`, `d1`, and so forth). For more information on metadevice naming, see Table 1–4.

RAID5 Metadevice Conventions

- **Why should I use RAID5 metadevices?**

  RAID5 metadevices need fewer disks for data redundancy than mirrors, and therefore can cost less than a mirrored configuration.

- **What is the minimum number of slices that a RAID5 metadevice must have?**

  Three (3).

- **Is there a maximum number of slices a RAID5 metadevice can have?**

  No. The more slices a RAID5 metadevice contains, however, the longer read operations take when a slice fails. (By the nature of RAID5 metadevices, write operations are always slower.)

- **How do I expand a RAID5 metadevice?**

  By concatenating slices to the existing part of a RAID5 metadevice.

- **When I expand a RAID5 metadevice, are the new slices included in parity calculations?**

  Yes.

- **What are the limitations to RAID5 metadevices?**

  You cannot use a RAID5 metadevice for root (`/`), `/usr`, and `swap`, or existing file systems.

- **Is there a way to recreate a RAID5 metadevice without having to “zero out” the data blocks?**

  Yes. You can use the `metainit(1M)` command with the `-k` option. (There is no equivalent within DiskSuite Tool.) The `-k` option recreates the RAID5 metadevice without initializing it, and sets the disk blocks to the OK state. If any errors exist on disk blocks within the metadevice, DiskSuite may begin fabricating data. Instead of using this option, you may want to initialize the device and restore data from tape. See the `metainit(1M)` man page for more information.

Example — RAID5 Metadevice

Figure 2–6 shows a RAID5 metadevice, `d40`.

The first three data chunks are written to Disks A through C. The next chunk that is written is a parity chunk, written to Drive D, which consists of an exclusive OR of the first three chunks of data. This pattern of writing data and parity chunks results in both data and parity spread across all disks in the RAID5 metadevice. Each drive can be read independently. The parity protects against a single disk failure. If each disk in this example were 2 Gbytes, the total capacity of `d40` would be 6 Gbytes. (One drive's worth of space is allocated to parity.)

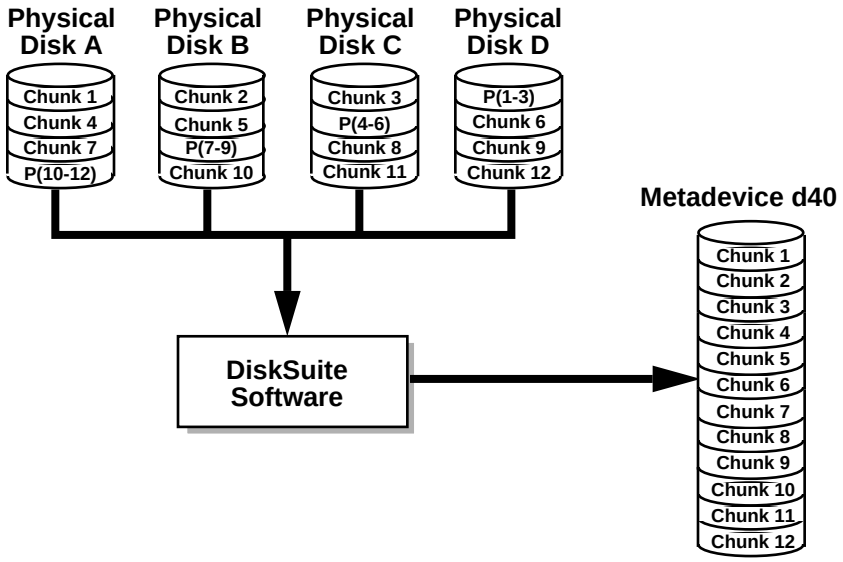

Figure 2–6 RAID5 Metadevice Example

Example — Concatenated (Expanded) RAID5 Metadevice

Figure 2–7 shows an example of an RAID5 metadevice that initially consisted of four disks (slices). A fifth disk has been dynamically concatenated to the metadevice to expand it.

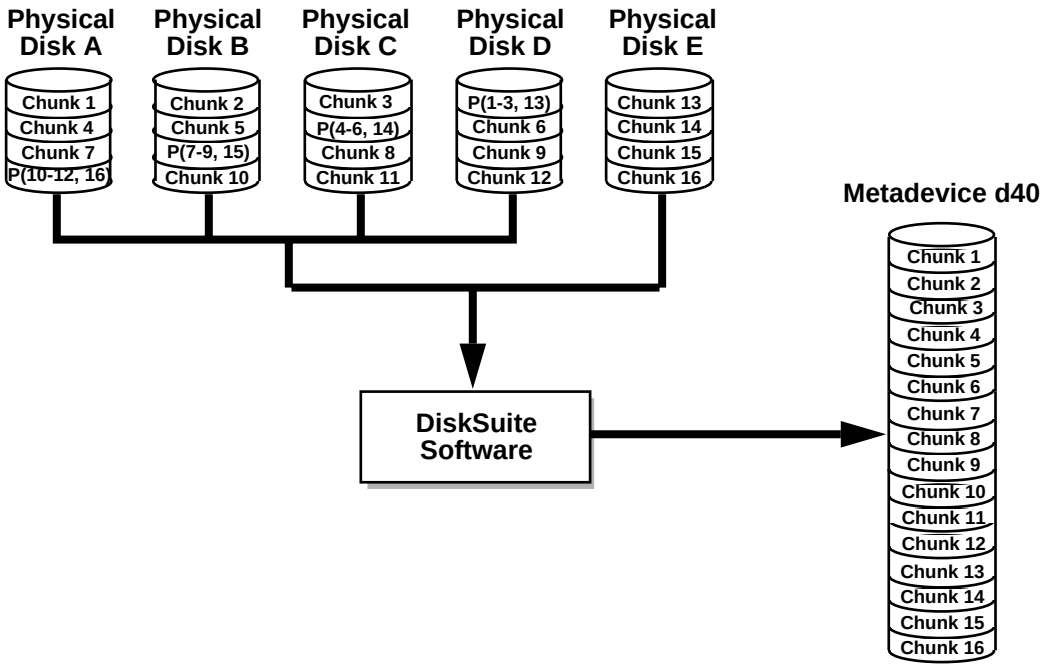

Figure 2–7 Expanded RAID 5 Metadevice Example

The parity areas are allocated when the initial RAID5 metadevice is created. One column's (slice's) worth of space is allocated to parity, although the actual parity blocks are distributed across all of the original columns to avoid hot spots. When you concatenate additional slices to the RAID, the additional space is devoted entirely to data; no new parity blocks are allocated. The data on the concatenated slices is, however, included in the parity calculations, so it is protected against single device failures.

Concatenated RAID5 metadevices are not suited for long-term use. Use a concatenated RAID5 metadevice unitl it is possible to reconfigure a larger RAID5 metadevice and copy the data to the larger metadevice.

Note - When you add a new slice to a RAID5 metadevice, DiskSuite "zeros" all the blocks in that slice. This ensures that the parity will protect the new data. As data is written to the additional space, DiskSuite includes it in the parity calculations.

UFS Logging or Trans Metadevices

UFS Logging

UFS logging is the process of writing file system “metadata” updates to a log before applying the updates to a UFS file system.

UFS logging records UFS transactions in a log. Once a transaction is recorded in the log, the transaction information can be applied to the file system later.

At reboot, the system discards incomplete transactions, but applies the transactions for completed operations. The file system remains consistent because only completed transactions are ever applied. Because the file system is never inconsistent, it does not need checking by `fsck(1M)`.

A system crash can interrupt current system calls and introduce inconsistencies into a UFS. If you mount a UFS without running `fsck(1M)`, these inconsistencies can cause panics or corrupt data.

Checking large file systems takes a long time, because it requires reading and verifying the file system data. With UFS logging, UFS file systems do not have to be checked at boot time because the changes from unfinished system calls are discarded.

DiskSuite manages UFS logging through trans metadevices.

UFS Logging Conventions

- **Can UFS logging improve performance?**

  UFS logging saves time when you reboot after a failure, because it eliminates the need to run the `fsck(1M)` command on file systems.

- **What are the drawbacks to UFS logging?**

  If the log fills up, performance can decrease because the UFS must empty the log before writing new information into it.

- **What versions of Solaris work with UFS logging?**

  UFS logging can only be used with Solaris 2.4 or later releases.

- **Which file systems cannot be logged?**

  Non-UFS file systems as well as the root (`/`) file system cannot be logged.

Trans Metadevices

A *trans metadevice* is a metadevice that manages UFS logging. A trans metadevice consists of two devices: a *master device* and a *logging device*.

A master device is a slice or metadevice that contains the file system that is being logged. Logging begins automatically when the trans metadevice is mounted, provided the trans metadevice has a logging device. The master device can contain an existing UFS file system (because creating a trans metadevice does not alter the master device), or you can create a file system on the trans metadevice later. Likewise, clearing a trans metadevice leaves the UFS file system on the master device intact.

A logging device is a slice or metadevice that contains the log. A logging device can be shared by several trans metadevices. The log is a sequence of records, each of which describes a change to a file system.

A trans metadevice has the same naming conventions as other metadevices: `/dev/md/dsk/d0`, `d1` ...,`d2`, and so forth. (For more information on metadevice naming conventions, see Table 1–4.)

Trans Metadevice Conventions

- **How do I use a trans metadevice?**

  After a trans metadevice is configured, it can be used just as if it were a physical slice. A trans metadevice can be used as a block device (up to 2 Gbytes) or a raw device (up to 1 Tbyte). A UFS file system can be created on the trans metadevice if the master device doesn't already have a file system.

Caution - A logging device or a master device can be a physical slice or a metadevice. For reliability and availability, however, use mirrors for logging devices. A device error on a physical logging device could cause data loss. You can also use mirrors or RAID5 metadevices as master devices.

- **How much disk space does a logging device need?**

  A minimum of 1 Mbyte. (Larger logs permit more simultaneous file-system transactions.) The maximum log size is 1 Gbyte. 1 Mbyte worth of log per 1 Gbyte of file system is a recommended minimum. 1 Mbyte worth of log per 100 Mbyte of file system is a recommended "average." Unfortunately, there are no hard and fast rules. The best log size varies with an individual system's load and configuration. However, a log larger than 64 Mbytes will rarely be used. Fortunately, log sizes can be changed without too much work.

- **Which file systems should I log?**

  Generally, log your largest UFS file systems and the UFS file system whose data changes most often. It is probably not necessary to log small file systems with mostly read activity.

- **Which file systems should always have separate logs?**

  All logged file systems can shared the same log. For better performance, however, file systems with the heaviest loads should have separate logs.

Caution - You must disable logging for `/usr`, `/var`, `/opt`, or any other file systems used by the system during a Solaris upgrade or installation when installing or upgrading software on a Solaris system.

- **Where should I place logs?**

  Place logs on mirrors, unused slices, or slices that contain the state database replicas. A device error on a physical logging device (a slice) can cause data loss.

- **What if no slice is available for the logging device?**

  You can still configure a trans metadevice. This may be useful if you plan to log exported file systems when you do not have a spare slice for the logging device. When a slice is available, you only need to attach it as a logging device. For instructions, see *Solstice DiskSuite 4.2.1 User's Guide*.

- **Can a logging device be shared between trans metadevices?**

  Yes, a logging device can be shared between file systems, though heavily-used file systems should have their own logging device. The disadvantage to sharing a logging device is that certain errors require that all file systems sharing the logging device must be checked with the `fsck(1M)` command.

Example — Trans Metadevice

Figure 2–8 shows a trans metadevice, `d1`, consisting of a mirrored master device, `d3`, and a mirrored logging device, `d30`

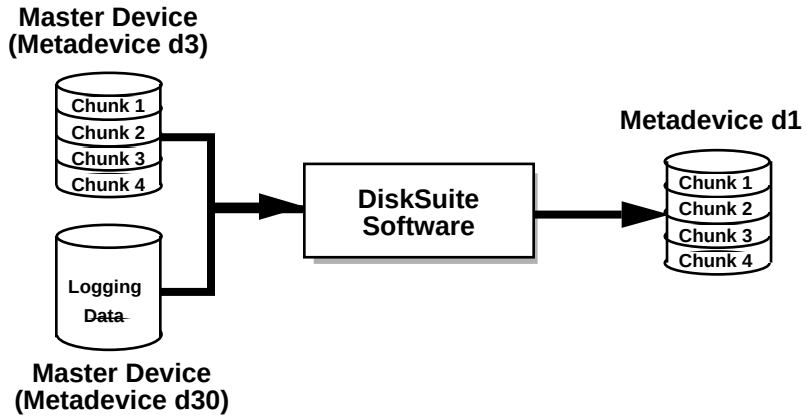

Figure 2–8 Trans Metadevice Example

Example — Shared Logging Device

Figure 2–9 shows two trans metadevices, `d1` and `d2`, sharing a mirrored logging device, `d30`. Each master device is also a mirrored metadevice, as is the shared logging device.

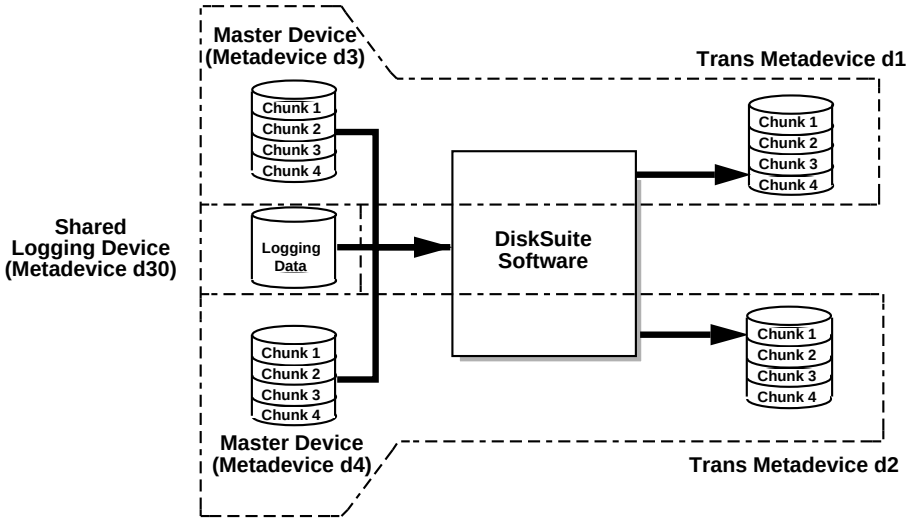

Figure 2–9 Shared Log Trans Metadevice Example

CHAPTER 3

Hot Spare Pools

This chapter explains hot spare pools. Use the following table to proceed directly to the section that provides the information you need.

Overview of Hot Spare Pools and Hot Spares

A *hot spare pool* is an ordered list (collection) of slices (*hot spares*) that DiskSuite uses to provide increased data availability for mirrors and RAID5 metadevices. A hot spare is reserved by DiskSuite to be automatically substituted in case of a slice failure in either a submirror or RAID5 metadevice.

A hot spare cannot be used to hold data while it is idle. A hot spare must remain ready for immediate use in the event of a slice failure in the metadevice with which it is associated. Because of the way in which hot spares operate, an additional investment in disks is necessary.

Hot Spares

A hot spare is a slice (not a metadevice) that is running but not in use. It is reserved, meaning that the hot spare stands ready to substitute for an errored slice in a submirror or RAID5 metadevice.

Because slice replacement and the resyncing of failed slices is automatic, hot spares provide protection from hardware failure. The hot spare can be used temporarily until a failed submirror or RAID5 metadevice slice is either fixed or replaced.

Hot spares remain idle most of the time, and so do not contribute to normal system operation. In addition, slices designated as hot spares cannot be used in any other metadevice, nor can they be used to hold data while idle.

You create hot spares within hot spare pools. Individual hot spares can be included in one or more hot spare pools. For example, you may have two submirrors and two hot spares. The hot spares can be arranged as two hot spare pools, with each pool having the two hot spares in a different order of preference. This enables you to specify which hot spare is used first. It also improves availability by having more hot spares available.

You cannot use hot spares within other metadevices, for example within a submirror. They must remain ready for immediate use in the event of a slice failure. A hot spare must be a physical slice. It cannot be a metadevice. In addition, hot spares cannot be used to hold state database replicas.

A submirror or RAID5 metadevice can use only a hot spare whose size is equal to or greater than the size of the failed slice in the submirror or RAID5 metadevice. If, for example, you have a submirror made of 1 Gbyte drives, a hot spare for the submirror must be 1 Gbyte or greater.

Note - A prerequisite for hot spares is that the metadevices with which they are associated have replicated data. When a hot spare takes over, any data on the failed slice must be recreated. For this reason, only mirrors and RAID5 metadevices use hot spares.

Hot Spare Pools

A hot spare pool is an ordered list (collection) of hot spares.

You can place hot spares into one or more pools to get the most security from the fewest slices. Then, a hot spare pool can be assigned to any number of submirror metadevices or RAID5 metadevices.

Note - You can assign a single hot spare pool to multiple submirrors or RAID5 metadevices. On the other hand, a submirror or a RAID5 metadevice can be associated with only one hot spare pool.

When errors occur, DiskSuite checks the hot spare pool for the first available hot spare whose size is equal to or greater than the size of the slice being replaced. If found, DiskSuite changes the hot spare's status to "*In-Use*" and automatically resyncs the data. In the case of a mirror, the hot spare is resynced with data from a good submirror. In the case of a RAID5 metadevice, the hot spare is resynced with the other slices in the metadevice. If a slice of adequate size is not found in the list of hot spares, the submirror or RAID5 metadevice that failed goes into an "errored" state. In the case of the submirror, it no longer replicates the data which that slice represented. In the case of the RAID5 metadevice, data redundancy is no longer available.

Hot Spare Pool Conventions

- **How are hot spare pools named?**

  Hot spare pools are named hsp*nnn*, where *nnn* is in the range 000-999.

- **What are "empty" hot spare pools?**

  You can create empty hot spare pools, enabling you to add hot spares when they become available.

- **How many hot spare pools are permitted?**

  1000.

- **Must a hot spare be a physical device?**

  Yes. It cannot be a metadevice.

- **How does the hot spare replacement algorithm work?**

  When a slice in a submirror or RAID5 metadevice goes into the "errored" state, a slice from the associated hot spare pool is used to replace it. DiskSuite selects the first hot spare (slice) that is large enough to replace the errored slice.

  DiskSuite searches a hot spare pool for a hot spare based on the order in which hot spares are added to a hot spare pool. The first hot spare found that is large enough is used as a replacement. When adding hot spares to a hot spare pool, it is best to add them from smallest to largest. This avoids potentially wasting "large" hot spares as replacements for small slices.

- **What are the size requirements for hot spares?**

  Hot spares must be equal to or greater than the smallest slice in the submirror or RAID5 metadevice with which the hot spare pool is associated. If DiskSuite cannot substitute an appropriately sized hot spare for a failed slice in a submirror or RAID5 metadevice, hot sparing will not occur for that metadevice.

- **Can a hot spare pool be associated with a “normal” stripe?**

  No. Hot sparing can only be used for mirrors and RAID5 metadevices. A hot spare pool must be associated with a submirror or a RAID5 metadevice.

- **What should I do if no hot spares are marked as Available?**

  Some hot spares must be marked as *Available*. If all hot spares are marked *In-Use*, you should either add more hot spares to the hot spare pool or repair the spared slices and return them to service.

- **Can I assign a hot spare pool to a one-way mirror?**

  Do not assign a hot spare pool to a submirror in a one-way mirror. Failed slices in a one-way mirror cannot be replaced by a hot spare. In these metadevices, no other copy of data is available to reconstruct on the hot spare.

- **Why should hot spares be defined on different controllers?**

  To maximize their availability in case of controller errors or failures.

Example — Hot Spare Pool

Figure 3–1 illustrates a hot spare pool, `hsp000`, that is associated with submirrors `d11` and `d12` in mirror `d1`. If a slice in either submirror were to become errored, a hot spare slice would automatically be substituted for the errored slice. The hot spare pool itself is associated with each submirror metadevice, not the mirror. The hot spare pool could also be associated with other submirrors or RAID5 metadevices if desired.

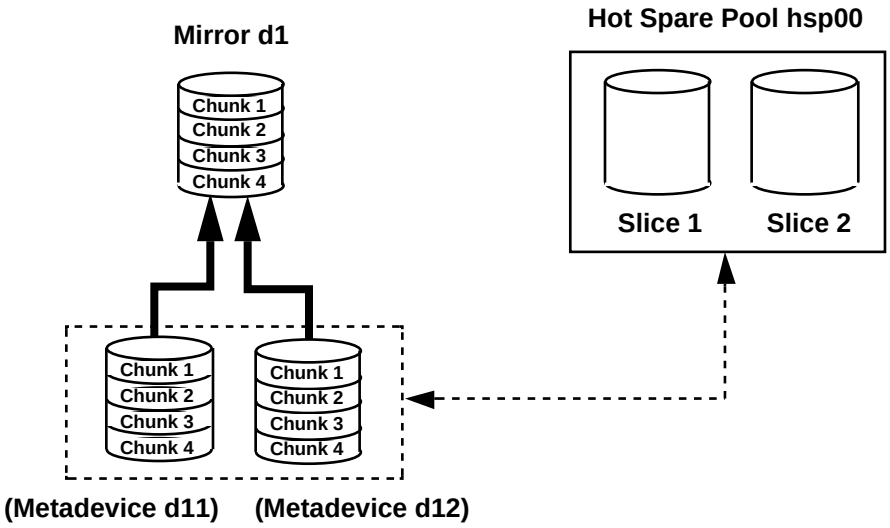

Figure 3–1 Hot Spare Pool Example

Administering Hot Spare Pools

DiskSuite enables you to dynamically add, delete, replace, and enable hot spares within hot spare pools. You can use either DiskSuite Tool or the command line utilities to administer hot spares and hot spare pools.

You can add a hot spare to one or more hot spare pools. When you add a hot spare to a hot spare pool, it is added to the end of the list of slices in the hot spare pool.

You can delete a hot spare from any or all of the hot spare pools to which it has been associated. Once the hot spare is deleted, the order of the remaining hot spares in the hot spare pool changes to reflect the new position. For example, if the second of three hot spares is deleted, the third hot spare moves to the second position. You cannot delete a hot spare that is currently in-use.

You can replace a hot spare in any or all of the hot spare pools to which it has been associated. The order of hot spares does not change after a replacement. You cannot replace a hot spare that is currently in use.

If a hot spare needs to be repaired, you make it available again by enabling it.

See *Solstice DiskSuite 4.2.1 User's Guide* for information on adding, deleting, replacing, and enabling hot spares.

CHAPTER 4

DiskSuite Tool

This chapter provides a high-level overview of DiskSuite's graphical user interface, DiskSuite Tool. For information on the command line interface, see the man pages.

Use the following table to proceed directly to the section that provides the information you need.

Overview of DiskSuite Tool

DiskSuite Tool is a graphical user interface for setting up and administering a DiskSuite configuration. DiskSuite Tool provides a graphical view of DiskSuite objects—metadevices, hot spares, and disk slices. DiskSuite Tool uses drag and drop manipulation of DiskSuite objects, enabling you to quickly configure your disks or change an existing configuration. It also provides performance information on metadevices and physical devices.

DiskSuite Tool provides graphical views of both physical devices and metadevices, helping simplify storage administration. You can also perform specific SPARCstorage Array maintenance tasks using DiskSuite Tool.

To start DiskSuite Tool, make sure you are root and enter the command:

```
# /usr/sbin/metatool [-sdiskset] &
```

For more information, see the `metatool(1M)` man page.

DiskSuite Tool and the Command Line Interface

In some instances, DiskSuite Tool and the command line utilities provide slightly different functionality. You must use the command line interface for some operations (for example, creating disksets), and DiskSuite Tool for others. Table 4–1 shows where you will need to use either DiskSuite Tool or the command line to perform certain functions.

TABLE 4–1 DiskSuite Tool vs. the Command Line

Functionality	Provided by DiskSuite Tool?	Provided by the DiskSuite Command Line?
Adding/removing disks to/from disksets	No	Yes
Adding/removing hosts to/from disksets	No	Yes
Creating/removing disksets	No	Yes
Switching metadevice names. (You can rename a metadevice with both DiskSuite Tool and the command line.)	No	Yes
Monitoring metadevice performance	Yes	No, but you could use `iostat(1M)`.
Maintaining SPARCstorage Arrays	Yes	No, but many functions can be accomplished with the `ssaadm(1M)` command.

Using the Mouse in DiskSuite Tool

Table 4–2 explains how the mouse works in DiskSuite Tool.

TABLE 4–2 DiskSuite Tool Mouse Model

This Button ...	Is Used To ...
SELECT (Default is Left)	Select objects with a single click. By holding down the Control key and clicking the left button, you can select multiple objects. By holding down the Control key and clicking the left button, you can deselect objects that are selected. You can also drag objects by holding down the left button.
ADJUST (Default is Middle)	Drag selected objects and keeps the objects selected, or, if an object is not selected, drag that object. You can drop the object on an appropriate target. If a target is not appropriate, the international “no” sign displays while the cursor is over the target.
MENU (Default is Right)	Display pull-down menus when the cursor is pointing at any title in the menu bar, or display a pop-up menu when the cursor is pointing inside an object on the canvas.

Screen Descriptions for DiskSuite Tool

Metadevice Editor Window

When you start DiskSuite Tool, the Metadevice Editor window is displayed, as shown in Figure 4–1.

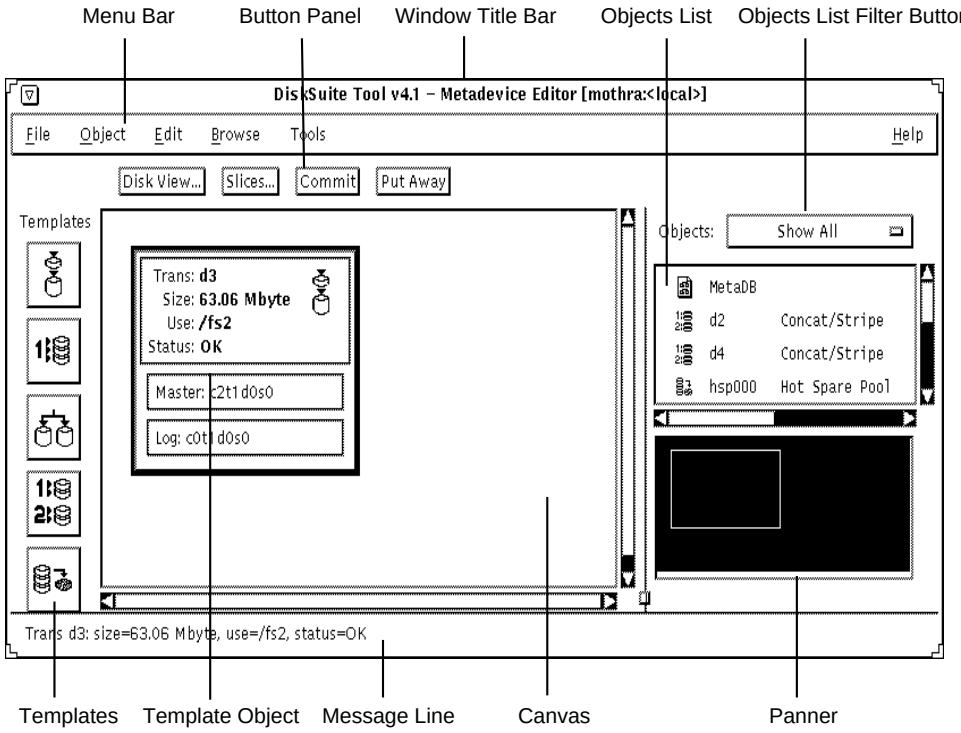

Figure 4–1 DiskSuite Tool Metadevice Editor Window

The Metadevice Editor window is the main window for DiskSuite Tool, enabling access to other parts of DiskSuite Tool. The following describes the areas within the Metadevice Editor window.

Note - DiskSuite Tool grays out menu items and user interface elements when you cannot use them in a specific context.

- **Menu Bar –** Usually contains five menus: File, Object, Edit, Browse, and Help. For more information on these menus, see the online help (the section “Accessing and Using Help” on page 101 describes how to access help.

Note - You can configure DiskSuite Tool to display a “Tools menu” (see *Solstice DiskSuite 4.2.1 User’s Guide*, or `metatool-toolsmenu(4)`. The Tools menu can be used to launch other applications, such asAdminSuite™ Storage Manager, from DiskSuite Tool.

- **Button Panel –** Contains buttons that display windows, and act on DiskSuite objects.

Note - You must select an object before clicking either the Commit button or the Put Away button.

- **Window Title Bar –** Displays the window title and the name of the system upon which DiskSuite Tool is currently running. Also displays diskset information, either <local>, for a local diskset, or the name of a shared diskset.
- **Objects List –** Contains metadevices, hot spare pools, and the metadevice state database object.

  You can select and drag objects in the Objects List to the canvas. Or you can double-click an object in the Objects List to display it on the canvas.

  Colored objects indicate a problem:

  - Red=Critical
  - Orange=Urgent
  - Yellow=Attention

  Gray scale monitors display problem status information in gray scales.

  On monochrome monitors, you must horizontally scroll the device list to view the status associated with the objects.
- **Objects List Filter Button –** Enables you to filter the information that the Objects List displays. You can filter by:

  - Show All
  - Show Metadevices
  - Show Hot Spare Pools
  - Show Problems

- **Templates –** Contains template icons, on the left side of the Metadevice Editor window. For descriptions of the template icons, see the online help.

  The template icons are sources for empty DiskSuite objects (templates). Once you have a template displayed on the canvas, you can then build metadevices from it by dropping slices or other metadevices into it. To work with a template, you can either single-click it or drag it to the canvas.
- **Template Object –** Acts as a template for a DiskSuite object, such as a concatenation.
- **Message Line Area –** Displays messages about specific elements on the canvas.

  When you place the cursor over an area of the Metadevice Editor window, the message line displays a message about that area.
- **Canvas –** Enables you to create and manipulate DiskSuite objects.

You can drag DiskSuite objects from the Disk View window, the Objects list, and the Templates to the canvas. Clicking an object on the canvas selects the object.

- **Panner –** Shows the current view in the canvas. (See Figure 4–2.)

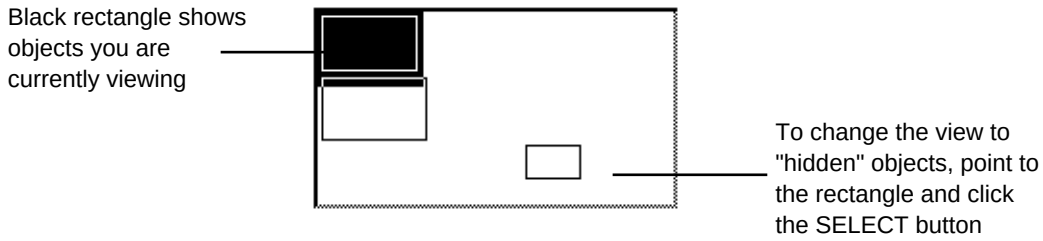

Figure 4–2 Panner

Pointing inside the Panner and clicking the SELECT button changes the current view. You can also point to the black rectangle, press and hold down the ADJUST button, and drag the view area to a new location.

Disk View Window

Figure 4–3, the Disk View window, shows how metadevices correspond to physical devices, and also serves as a drag-and-drop source for slices and a drag-and-drop target for metadevices. The following describes the areas within the Disk View window.

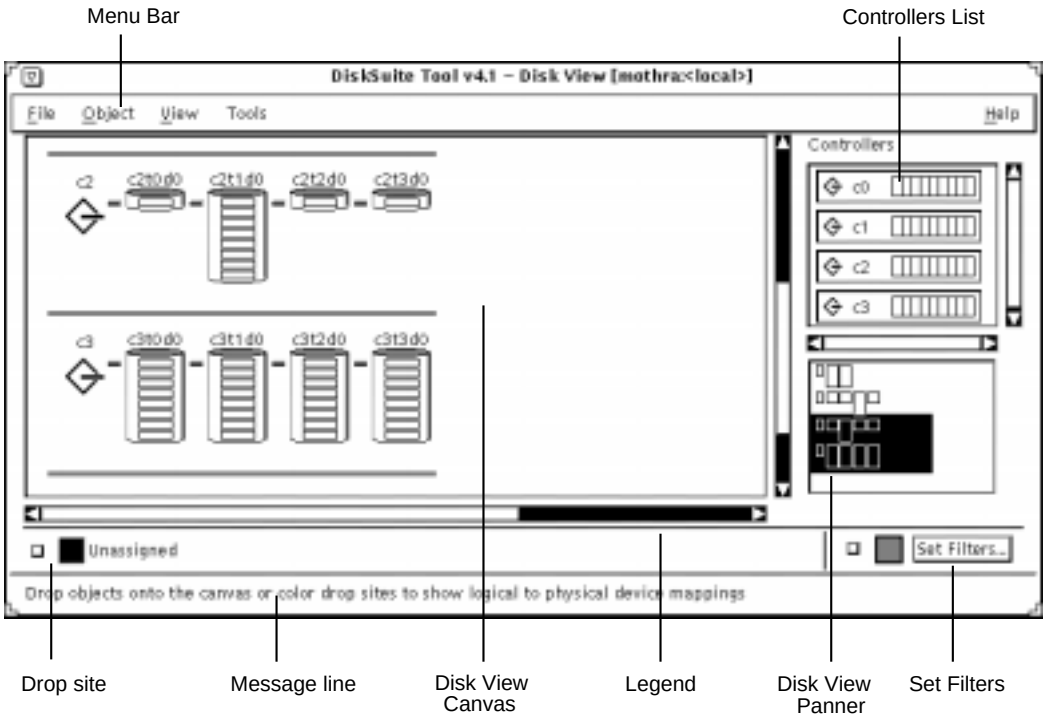

Figure 4–3 Disk View Window

- **Menu Bar –** Usually contains four menus: File, Object, View, and Help. For more information on these menus, see the online help (the section "Accessing and Using Help" on page 101 describes how to access help.)

Note - You can configure DiskSuite Tool to display a "Tools menu" (see *Solstice DiskSuite 4.2.1 User's Guide*, or `metatool-toolsmenu(4)`. The Tools menu can be used to launch other applications, such as Solstice Storage Manager, from DiskSuite Tool.

- *Controllers List* – Contains all the controllers currently in your configuration.

  Clicking a toggle button displays that controller on the Disk View canvas. Clicking again removes the controller from the canvas.

- **Color Drop Sites –** Show physical-to-logical device relations.

  Figure 4–4 shows the color drop sites.

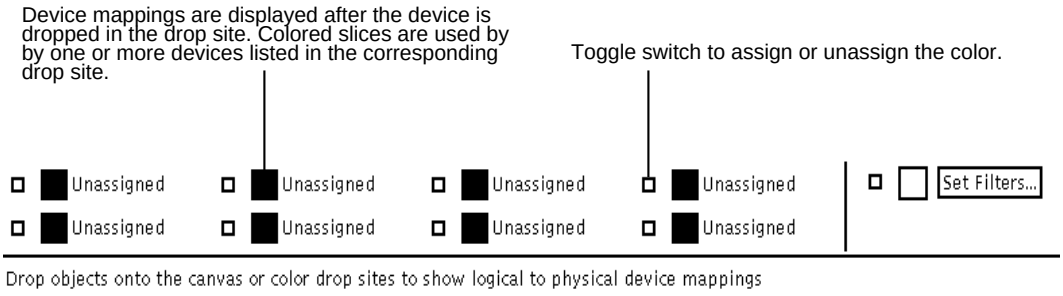

Figure 4–4 Color Drop Sites

Dropping a metadevice object onto a color drop site assigns a color to that metadevice object. The color, in turn, shows up on the Disk View window canvas, enabling you to see physical-to-logical device relations.

Each drop site has a pop-up menu that contains:

- **Info –** Displays the Information window for the object.
- **Clear –** Sets the color drop site to Unassigned.

You can change the colors for each of the eight color drop sites. Edit the X resource file, `/usr/lib/lvm/X11/app-defaults/Metatool`. It contains a list of all the X resources used by `metatool(1M)`. See *Solstice DiskSuite 4.2.1 User's Guide* for more information on editing this file.

A monochrome monitor will show only one drop site, black.

- **Message line –** Displays messages about specific elements on the canvas.

  When you place the cursor over an area of the Disk View Canvas, the message line displays a message about that area.

- **Disk View Canvas –** Displays the physical devices and mappings on the canvas.

  To select a disk on the Disk View canvas, click the top of the disk. To select a slice, click inside the slice rectangle. You can drag the object, whether selected or not, to a template on the Metadevice Editor canvas and add or replace slices in that template.

  The canvas is also a destination for drag and drop. When devices are dropped on the canvas from the Metadevice Editor window, they take on the next available color. If all drop sites are in use, a window is displayed that enables you to select a drop site. Also, if any object is selected on the editor canvas and the Disk View window is invoked, the objects will automatically take on the color of the next available drop site.

  The graphical representations of objects on the Disk View canvas are shown in Figure 4–5.

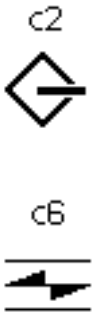

Figure 4–5 Disk View Objects

- **Legend –** On color systems, contains eight color drop sites that provide color cues for mappings. Each color can be hidden or exposed using the toggle button to the left of each color box. On monochrome systems, only one drop site is available, which is black.

  The legend region of the Disk View window can be turned on and off by choosing Show Legend from the View menu.

- **Disk View Panner –** Shows the current view in the canvas. See Figure 4–6.

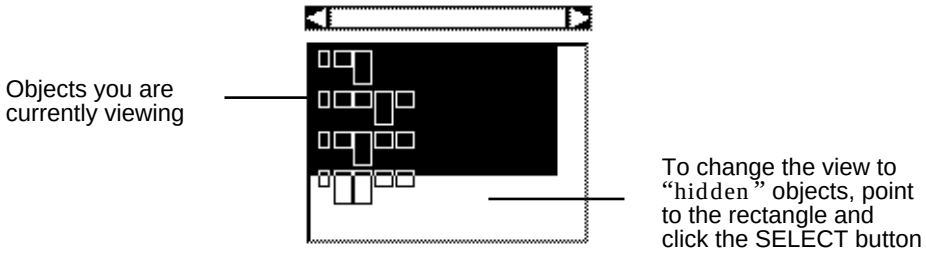

Figure 4–6 Disk View Panner

Pointing inside the Disk View Panner and clicking the SELECT button changes the current view. You can also point to the black rectangle, press and hold down the ADJUST button, and drag the view area to a new location.

- **Set Filters –** Enables you to filter slice information by usage criteria.

  You can view slices that are available to be Metadevice Components, Hot Spares, Replicas, Trans Logs, or Anything. The default value is Metadevice Components. You can also view slices currently in use as a file system, `swap`, Metadevices, Hot Spares, Replicas, Trans Logs, or Anything. Clicking the Don't Care button tells DiskSuite Tool not to filter by usage. Regular expressions can also be used to filter slices in this window.

Statistics Graphs Window (Grapher Window)

Figure 4–7 shows the Statistics Graph window (Grapher window). The Grapher window functions as a visual log of reported statistics utilizing a graph for each device. The Grapher window displays a subset of the information derived by the `iostat(1M)` interface. You can drag and drop metadevices and disks from any of DiskSuite Tool's windows to the Statistics Graphs window. This includes the Metadevice Editor canvas, the Metadevice Editor Objects list, the Slice window, and the Disk View window. An explanation of the Grapher window follows.

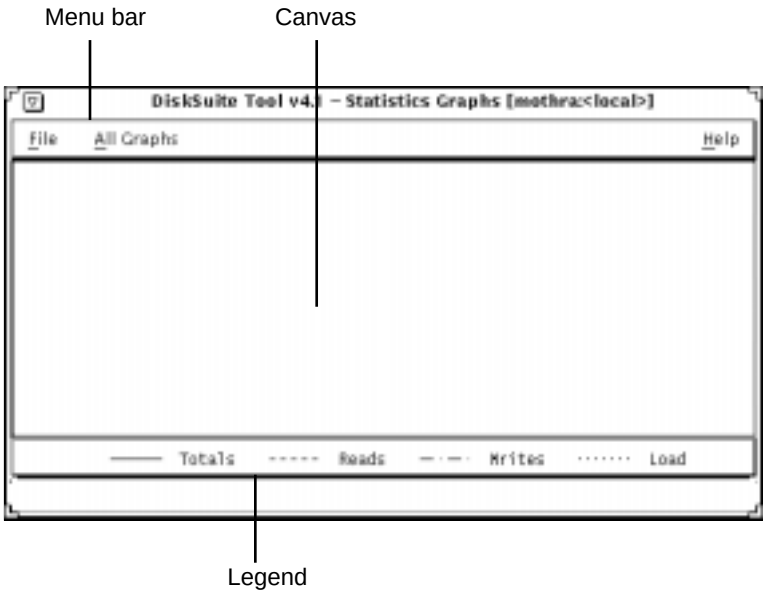

Figure 4–7 Statistics Graphs Window (Grapher Window)

- **Menu Bar –** Contains two menus titled File and All Graphs. For more information on these menus, see the online help (the section "Accessing and Using Help" on page 101 describes how to access help).
- **Canvas –** Shows instantaneous statistics, and has toggle buttons for controlling the information displayed.
- **Legend –** Contains a legend for all the graphs.

When you add a device to the Grapher window, a button bar appears. If you continue to add devices on the canvas, they appear in individual rows with a control area and graph. Figure 4–8 shows the Grapher window with a metadevice. An explanation of the buttons follows.

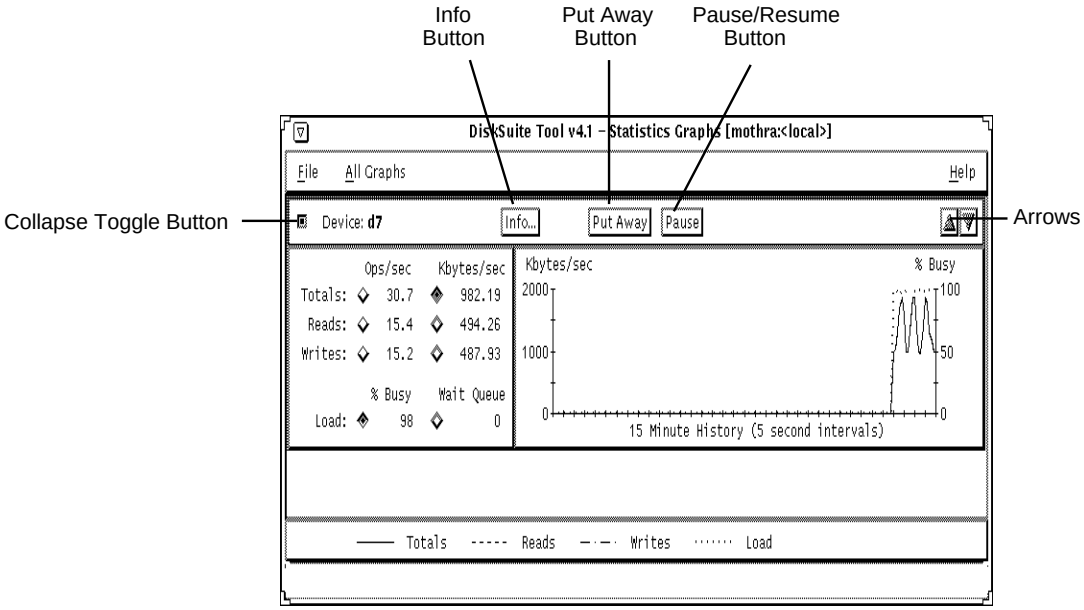

Figure 4–8 Grapher Window with Metadevice

- **Collapse Toggle Button –** Collapses a canvas row.
- **Info Button –** Displays the device's Information window.
- **Put Away Button –** Removes the device from the Grapher window.
- **Pause/Resume Button –** Suspends updates to the Grapher window (Pause), or alternately, resumes updates (Continue).
- **Arrows –** Reorder rows.

Information Windows

Several information windows are present in DiskSuite Tool. These information windows include:

- Disk Information Window (see "Disk Information Window" on page 70)
- Slice Information Window (see "Slice Information Window" on page 72)
- Device Statistics Sheet (see "Device Statistics Window" on page 74)
- Concat Information Window (see "Concat Information Window" on page 75)
- Stripe Information Window (see "Stripe Information Window" on page 77)
- Mirror Information Window (see "Mirror Information Window" on page 79)
- Trans Information Window (see "Trans Information Window" on page 82)
- Hot Spare Information Window (see "Hot Spare Information Window" on page 84)

- RAID Information Window (see “RAID Information Window” on page 86)
- Metadevice State Database Information Window (see “Metadevice State Database Info Window” on page 88)
- Tray Information Window (“Tray Information Window” on page 90)
- Controller Information Window (“Controller Information Window” on page 91)

Disk Information Window

By pointing to a disk on the Disk View canvas and pressing the MENU button, a menu enables you to bring up an information window. Shown in Figure 4–9, the read-only Disk Information window provides information about a disk and its slices.

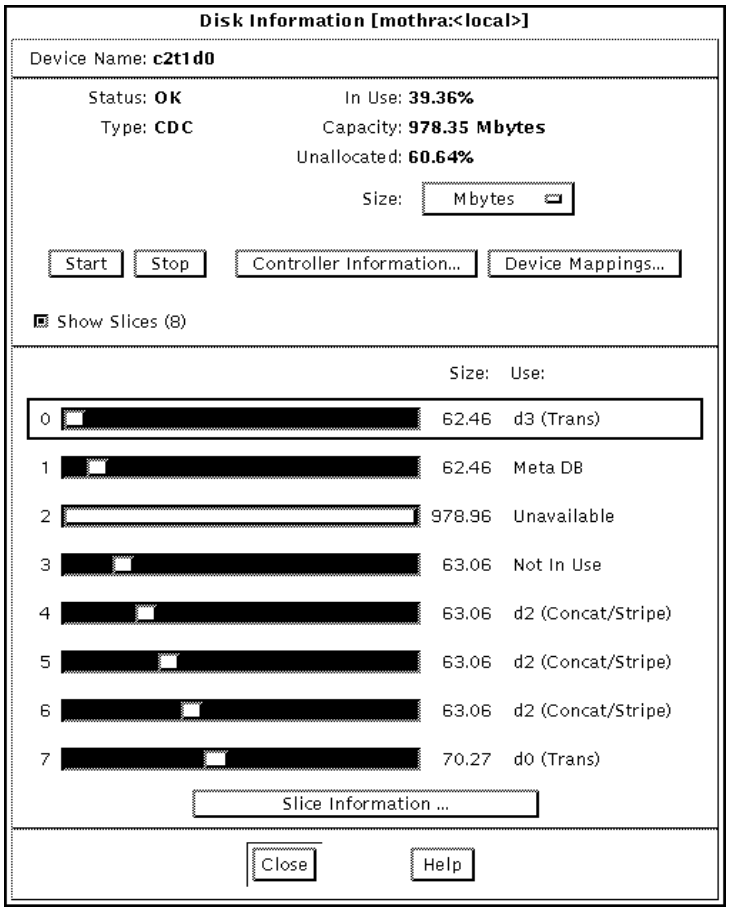

Figure 4–9 Disk Information Window

Table 4–3 lists the functionality provided by the Disk Information window.

TABLE 4–3 Disk Information Window Functionality

Field	Functions
Device name	The device name, for example, `c2t1d0`.
Status	The status is reported as OK, Reserved if the disk is reserved by another host, Failed, or Spun Down if this is a SPARCstorage Array disk and it is spun down.
Type	The type of the disk as listed in the `/etc/format.dat` file, for example SUN0535, SUN1.05, or DEFAULT.
In Use	The percentage of the disk that is currently in use as a metadevice, metadevice state database replica, or a mounted file system.
Capacity	Shows the usable capacity of the disk. The usable capacity of the disk is the disk capacity less any space taken by state database replicas or the disk label.
Unallocated	The percentage of the disk available for use.
Size	A pop-up menu that changes the size units represented in the capacity field and the information under the Size column in the Slice region. Choices include: Gbytes, Mbytes, Kbytes, Sectors, and Cylinders. The default is Mbytes.
Start	A button to start a stopped disk. (DiskSuite Tool shows the disk state only for SPARCstorage Array disks. A down arrow beneath a SPARCstorage Array disk indicates it is currently stopped.)
Stop	A button to stop a disk. (DiskSuite Tool shows the disk state only for SPARCstorage Array disks.)
Controller Information	A button that brings up the Controller Information window. See Figure 4–20.
Device Mappings	Displays the Physical to Logical Device Mappings window. (The Physical to Logical Device Mappings window is not dynamically updated when new mappings are created.)

TABLE 4–3 Disk Information Window Functionality *(continued)*

Field	Functions
Show Slices	A toggle button that expands and collapses the slice view. The number of non-zero size slices on the disk is shown in parentheses on the button.
Slice Information	A button that brings up the Slice Information window for each selected slice. Point to the slice area and click the SELECT button to select a slice. To select multiple slices, either press and hold down the Control key while pointing to the slices and clicking the SELECT button or hold down the SELECT button and drag the cursor over slices.

Table 4–4 lists additional functionality that appears for SPARCstorage Array disks.

TABLE 4–4 Disk Information Screen, SPARCstorage Array Functionality

Field	Functions
Vendor	Displays the vendor name.
Product ID	Displays the product identification number.
Firmware Rev.	Displays the product firmware revision information.
Fast Write	Radio buttons that enable fast writes or synchronous fast writes, or disable fast writes.

Slice Information Window

The Slice Information window, shown in Figure 4–10, displays information about a specific slice. There are three ways to display this window:

- Select a slice on the Disk Information window by pointing to it and pressing the SELECT button. Then click the Slice Information button.
- Point to a slice of a disk that is displayed on the Disk View window's canvas. Press and hold down the MENU button to display the pop-up menu for the slice then select the Info option.
- Point to a slice inside any metadevice displayed on the Metadevice Editor's canvas. Press and hold down the MENU button to display the pop-up menu for the slice then select the Info option.

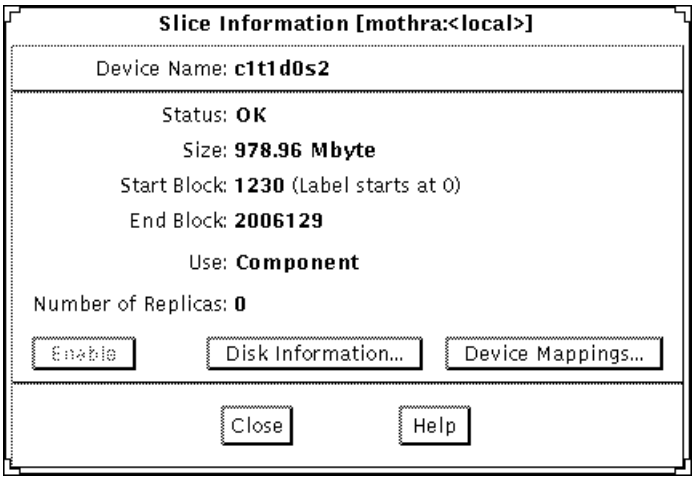

Figure 4–10 Slice Information Window

Table 4–5 explains the Slice Information window.

TABLE 4–5 Slice Information Window Functionality

Field	Functions
Device name	The device name, for example, `c1t1d0s2`.
Status	The status is reported as OK, Resyncing, Enabled, Critical, Spared, Urgent, or Attention.
Size	The total size of the slice.
Start block	The block on which the slice begins. If the slice has a label, there will be additional information about the label in this field.
End block	The block on which the slice ends.
Use	The current use of the slice, for example, file system or `swap`. If the use is hot spare, a Show Hot Spare Pools button is available on the right side of the Use field. This button opens a dialog that shows a list of Hot Spare Pools with which the slice is associated.
Number of Replicas	The number of replicas, if any, contained in the slice.
Enable	This button enables the slice. The button is available only if the data on the slice is replicated in a mirror or RAID5 metadevice, or if the slice is used as a hot spare that is currently “broken.”

TABLE 4–5 Slice Information Window Functionality *(continued)*

Field	Functions
Disk Information	Opens the Disk Information window.
Device Mappings	Displays the Physical to Logical Device Mappings window. (The Physical to Logical Device Mappings window is not dynamically updated when new mappings are created.)

Device Statistics Window

The Device Statistics window, shown in Figure 4–11, displays a “snap-shot” of statistical information about a specific metadevice or physical disk. There are two ways to display this window:

- Select a metadevice on the Metadevice Editor window’s canvas, or a disk on the Disk View canvas, by pointing to it and pressing the SELECT button. Select Statistics from the Object menu.
- Point to a metadevice displayed on the Metadevice Editor window’s canvas, or a disk displayed on the Disk View canvas. Press and hold down the MENU button to display the pop-up menu for the metadevice or disk then select the Statistics option.

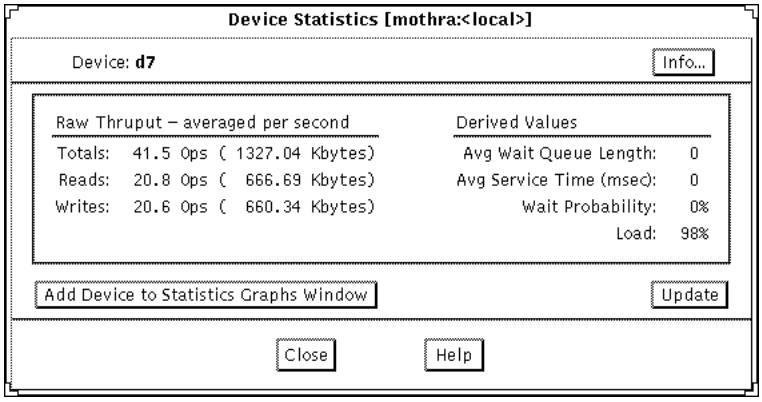

Figure 4–11 Device Statistics Window

Table 4–6 explains the Device Statistics Window.

TABLE 4–6 Device Statistics Window Functionality

Field	Functions
Device	This field displays the device name, for example, `d63`.
Info	This button brings up the device's Information window.
Raw Thruput	This information displays reads, writes, total reads and writes, averaged per second.
Derived Values	This information displays average wait queue length, average service time, wait probability, and load.
Add Device to Statistics Graphs Window	This button adds the device to the Statistics Graphs window. The graph area is blank until you select which statistics to graph. The default is Percent Busy.
Update	This button presents a new snap-shot of the statistical information.

Concat Information Window

There are three ways to display the Concat Information window shown in Figure 4–12:

- Double-click the Concat/Stripe object in the Objects list. The Concat/Stripe object is opened on the Metadevice Editor's canvas. Select Info from the Objects menu.
- If the Concat/Stripe object is on the Metadevice Editor's canvas, point inside the template. Press and hold down the MENU button to display the pop-up menu for the concatenation then select the Info option.
- If the Concat/Stripe object is on the Metadevice Editor's canvas, point inside the top rectangle of the object and double-click.

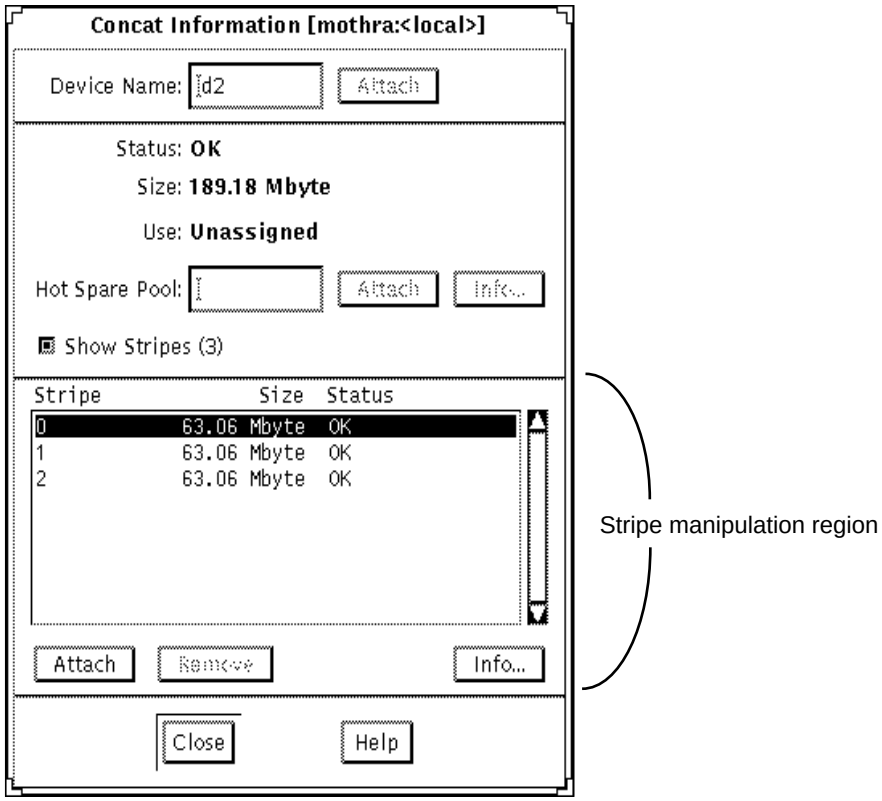

Figure 4–12 Concat Information Window

Table 4–7 lists the functionality associated with the regions of the Concat Information window.

TABLE 4–7 Concat Information Window Functionality

Field	Functions
Device Name	The metadevice name of the concatenation. As long as the device is not "open," you can change the name by typing a new one and clicking Attach.
Status	Description of the concatenation's status.
Size	The size of the concatenation.
Use	How the concatenation is currently used (for example, submirror).

TABLE 4–7 Concat Information Window Functionality *(continued)*

Field	Functions
Hot Spare Pool	The entry field for specifying the name of a Hot Spare Pool to be associated with the concatenation. To attach a hot spare pool enter the name in the field and click on the Attach button. The Hot Spare Pool Information window is displayed when you enter a hot spare pool name and click on the Info button.
Show Stripes	This toggle button enables you to turn on and off the stripe manipulation region. The number of stripes in the concatenation are shown in parentheses on the button.
Stripe manipulation region	The following functionality is available in this region: ■ **List of stripes –** Provides the size and status of each stripe included in the concatenation. ■ **Attach –** Attaches a new and empty stripe to the concatenation. ■ **Remove –** Removes the selected stripe from the concatenation. ■ **Info –** Brings up the Stripe Information window for the selected (highlighted) stripes.

Stripe Information Window

There are three ways to display the Stripe Information window shown in Figure 4–13:

- Double-click the Concat / Stripe object in the Objects list. The Concat / Stripe object is opened on the Metadevice Editor's canvas. Point to the stripe rectangle. Select Info from the Objects menu.
- If the Concat / Stripe object is on the Metadevice Editor's canvas, point inside the stripe rectangle of the Concat / Stripe object and double-click.
- If the Concat / Stripe object is on the Metadevice Editor's canvas, point inside the stripe rectangle. Press and hold down the MENU button to display the pop-up menu then select the Info option.

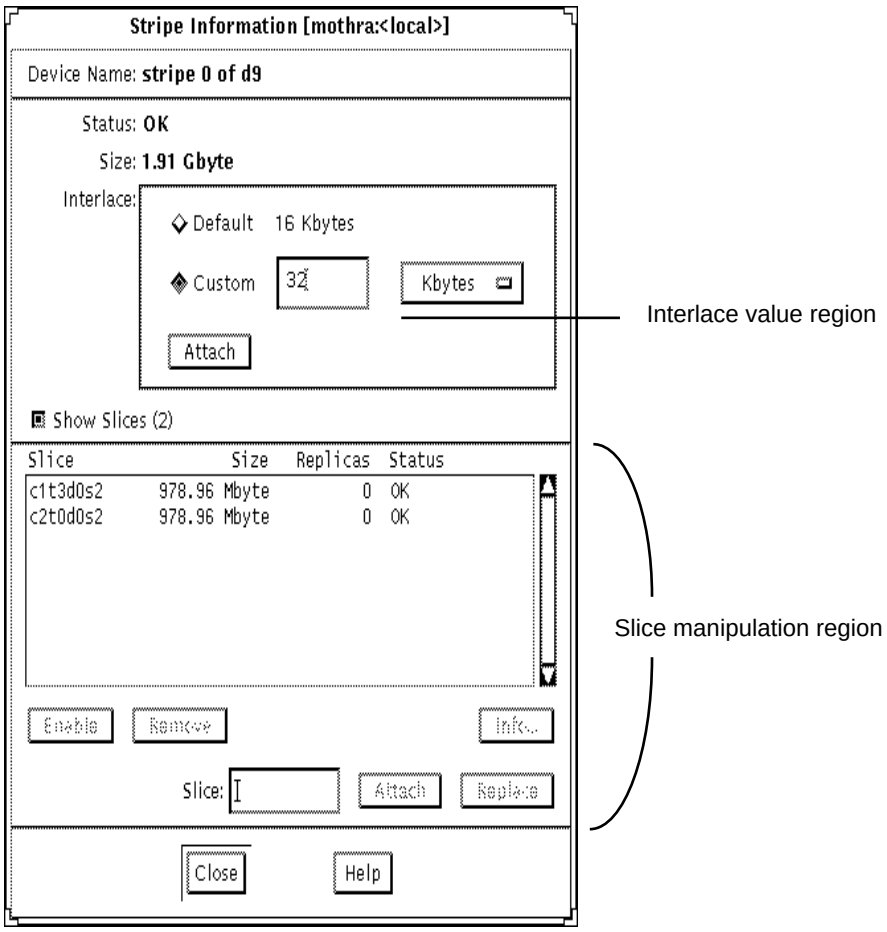

Figure 4–13 Stripe Information Window

Table 4–8 lists the functionality associated with the regions of the Stripe Information window.

TABLE 4–8 Stripe Information Window Functionality

Field	Functions
Device Name	The metadevice name of the stripe, such as d5.
Status	Description of the stripe's status.
Size	The size of the stripe.

TABLE 4–8 Stripe Information Window Functionality *(continued)*

Field	Functions
Interlace value region	The default interlace value is 16 Kbytes. To change the interlace value, click the Custom button and type the value in the field. The menu button to the right of the field enables you to specify the units used. The values on the menu are Gbytes, Mbytes, Kbytes, and Sectors. The default is Kbytes. After the Custom field is complete, the Attach button is used to assign the interlace value to the stripe. After a stripe is committed, the interlace value cannot be changed.
Show Slices	This toggle button enables you to turn on and off the slice manipulation region. The number of slices in the stripe are shown in parentheses on the button.
Slice manipulation region	The following functionality is available in this region: ■ **Scrolling List –** Shows slices included in the stripe. The information in this region includes the name of the slice, size, number of state database replicas on the slice, and the status. ■ **Enable –** Enables the selected slices if they are disabled. ■ **Remove –** Removes the selected slices. ■ **Slice –** Specifies a new slice to be attached to the stripe or replaces the selected slice. If no slice is selected, the button is unavailable. ■ **Attach –** Attaches the slice specified in the Slice field to the stripe. This button is active only when a slice name is entered in the field. ■ **Replace –** Replaces the selected stripe with the slice entered in the Slice field. This button is active only when a slice name has been entered in the field and a slice is selected on the scrolling list. ■ **Info –** Displays the Slice Information window for the selected (highlighted) slice.

Mirror Information Window

DiskSuite provides several options to optimize mirror performance. These options deal with the read and write policy for mirrors, and the order in which mirrors are resynced during reboot. You set these and other options using choices on the Mirror Information window, as shown in Figure 4–14. There are three ways to display the Mirror Information window:

- Double-click the mirror object in the Objects list. The mirror is opened on the Metadevice Editor's canvas. Select Info from the Objects menu.
- If the mirror object is on the Metadevice Editor's canvas, point inside the mirror rectangle. Press and hold down the MENU button to display the pop-up menu then select the Info option.
- If the mirror object is on the Metadevice Editor's canvas, double-click inside the mirror rectangle.

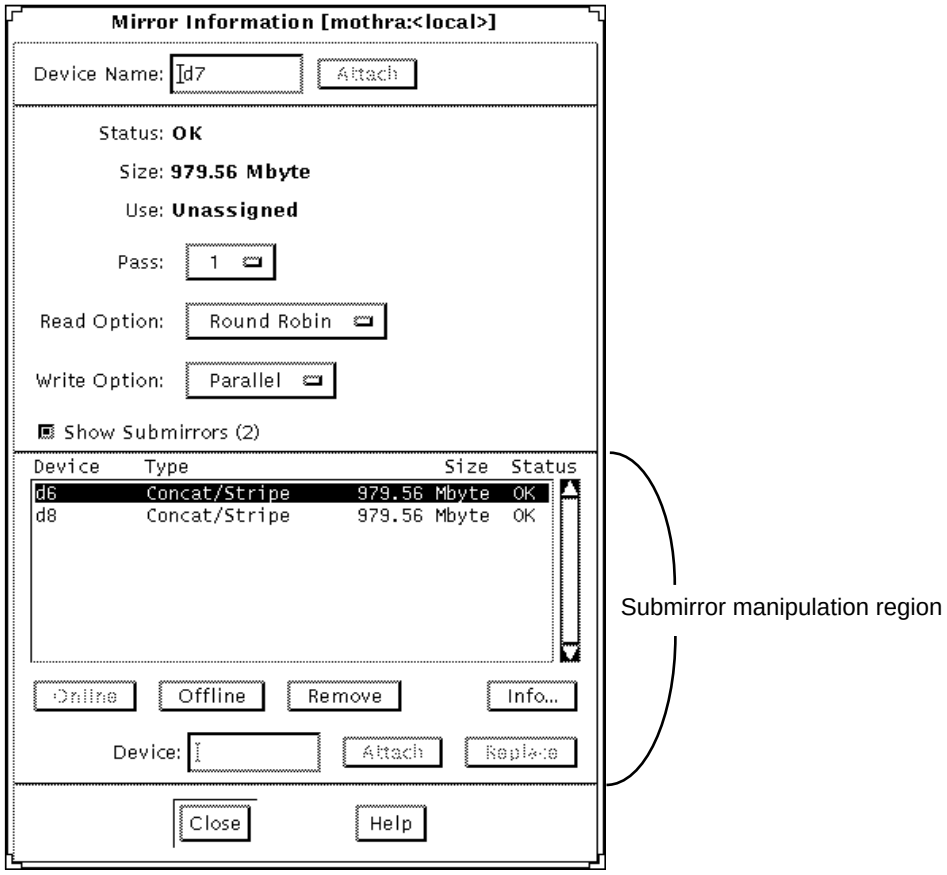

Figure 4–14 Mirror Information Window

The Mirror object must be committed before the policy changes take effect.

Table 4–9 lists the functionality associated with the regions of the Mirror Information window.

TABLE 4–9 Mirror Information Window Functionality

Field	Functions
Device Name	The metadevice name of the mirror. As long as the device is not “open,” you can change the name by typing a new one and clicking Attach.
Status	Description of the mirror’s status.
Size	The total size of the mirror.

TABLE 4–9 Mirror Information Window Functionality *(continued)*

Field	Functions
Use	Shows how the mirror is currently used, for example, file system, swap, or shared log. If the use is shared log, a button labeled Show Trans is displayed. The Show Trans button opens a Sharing Information window that shows the Trans devices that share the Mirror.
Show Submirrors	This toggle button enables you to turn on and off the submirror manipulation region. The number of submirrors in the mirror are shown in parentheses on the button.
Pass	A pass number in the range 0-9 can be assigned to a mirror using the Pass button menu. The pass (resync) number determines the order in which that mirror is resynced during a system reboot. The default is 1. Smaller pass numbers are resynced first. If 0 is chosen, the resync is skipped. A 0 should only be used for mirrors mounted as read-only. If different mirrors have the same pass number, they are resynced concurrently.
Read Option	There are three kinds of read options associated with mirrors: Round Robin, Geometric, and First. The default read option is Round Robin, also called balanced load. When set to Round Robin, all reads are made in a round robin order from all the submirrors in the mirror. That is, the first read comes from the first submirror, the next read comes from the second submirror, and so forth. The Geometric option provides faster performance on sequential reads or when you are using disks with track buffering. Geometric reads allow read operations to be divided among submirrors on the basis of a logical disk block address. For instance, with a three-way mirror the disk space on the mirror is divided into three (equally sized) logical address ranges. Reads from the three regions are then performed by separate submirrors (for example, reads to the first region are performed by the first submirror). The First option specifies reading from only the first submirror. This would be specified only if you have a second submirror that has poor read I/O characteristics.

TABLE 4–9 Mirror Information Window Functionality *(continued)*

Field	Functions
Write Option	A button that enables you to set parallel or serial writes to the submirror. Parallel writes are the default action of the metadisk driver, meaning the writes are dispatched to all submirrors simultaneously. Serial writes specify that writes to one submirror must complete before the next submirror write is started.
Submirror manipulation region	The following functionality is available in this region: ■ **Show Submirrors –** This toggle button enables showing or hiding the list of submirrors. ■ **Scrolling List –** Shows submirrors included in the mirror. The information in this region includes the name, type, size, and status. Click on the submirror to select it. When submirrors are selected, actions can be performed on them. ■ **Online –** Brings selected submirrors back online. This button is active only when the selected submirror is offline. ■ **Offline –** Takes selected submirrors offline. This button is active only when the selected submirror is online. ■ **Remove –** Detaches the selected submirrors. ■ **Info –** Opens the Concat Information window for the selected submirror. ■ **Device –** Specifies a new submirror in the field to attach or replace. The field is cleared when you click on the Attach or Replace buttons. ■ **Attach –** Adds the specified submirror. This button is active only when a submirror or device is entered in the Device field. ■ **Replace –** Replaces the selected submirror with the submirror entered in the field. This button is active only when a submirror or device is entered in the field and one in the list is selected.

Trans Information Window

The Trans Information window enables you to view and modify the attributes and components of a specific trans metadevice. Figure 4–15 shows the Trans Information window. There are three ways to display the Trans Information window:

- Double-click the Trans object in the Objects list. The object is opened on the Metadevice Editor's canvas. Select Info from the Objects menu.
- If the Trans Metadevice object is on the Metadevice Editor's canvas, point inside the Trans rectangle. Press and hold down the MENU button to display the pop-up menu then select the Info choice.

- If the Trans Metadevice object is on the Metadevice Editor's canvas, point inside the Trans rectangle and double-click.

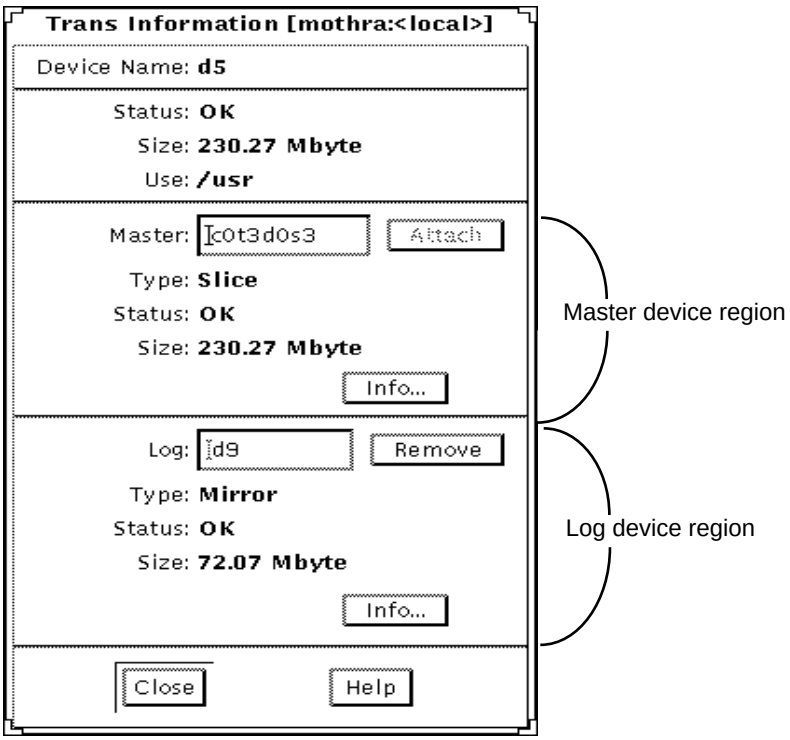

Figure 4–15 Trans Information Window

The Trans object must be committed before the changes take effect.

Table 4–10 lists the functionality associated with the regions of the Trans Information window.

TABLE 4–10 Trans Information Window Functionality

Field	Functions
Device Name	The metadevice name of the Trans device. As long as the device is not "open," and no logging device is attached, you can change the name by typing a new one and clicking Attach.
Status	Description of the Trans device status.
Size	The total size of the Trans device.
Use	How the Trans device is currently used (for example, file system).

TABLE 4–10 Trans Information Window Functionality *(continued)*

Field	Functions
Master device region	A region that contains the device name of the master device. The Attach button toggles between Attach and Remove. Other information in the region includes: ■ **Type –** The type of device used as the master. ■ **Status –** Shows the description of the master's status. ■ **Size –** Displays the size of the master device. ■ **Info –** Displays the information form for the master device.
Log device region	A region that contains the device name where the log device is located. The Remove button toggles between Attach and Remove. Other information in the region includes: ■ **Type –** The type of device used as the log. ■ **Status –** Shows the description of the log's status. ■ **Size –** Displays the size of the log device. ■ **Info –** Displays the information form for the log device.

Hot Spare Information Window

The Hot Spare Information window enables you to view and modify the attributes and components of a specific hot spare. Figure 4–16 shows the Hot Spare Information window. There are three ways to display the Hot Spare Information window:

- Double-click the Hot Spare Pool in the Objects list. The hot spare pool object is opened on the Metadevice Editor's canvas. Select Info from the Object menu.
- If the Hot Spare Pool object is on the Metadevice Editor's canvas, point inside the top of the Hot Spare Pool rectangle. Press and hold the MENU button to display the pop-up menu then select the Info option.
- If the Hot Spare Pool object is on the Metadevice Editor's canvas, point inside the top of the Hot Spare Pool rectangle and double-click.

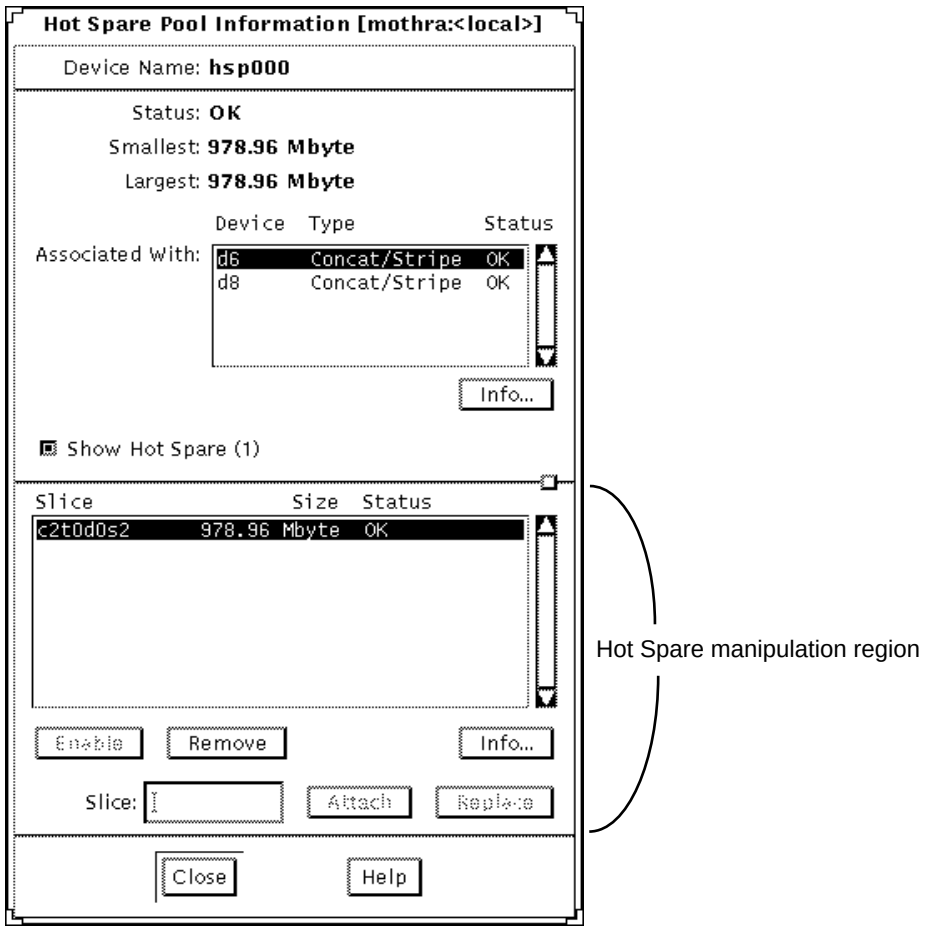

Figure 4–16 Hot Spare Information Window

The Hot Spare Pool object must be committed before the changes take effect.

Table 4–11 lists the functionality associated with the regions of the Hot Spare Pool Information window.

TABLE 4–11 Hot Spare Pool Information Window Functionality

Field	Functions
Device Name	The name of the Hot Spare Pool, such as `hsp000`.
Status	Description of the Hot Spare Pool's status.
Smallest	The size of the smallest slice in the Hot Spare Pool.

TABLE 4–11 Hot Spare Pool Information Window Functionality *(continued)*

Field	Functions
Largest	The size of the largest slice in the Hot Spare Pool.
Associated With	A scrolling list that displays the device names, types, and status of all metadevices associated with the Hot Spare Pool. To display information about the object either click the object then click Info or point to the object and double-click.
Info	Displays the Concatenation Information window for the selected (highlighted) Concat/Stripe in the Associated With region.
Hot Spare manipulation region	Contains a list of all the slices included in the Hot Spare Pool. New slices can be added. Existing slices can be manipulated. The functions of the buttons include: ■ **Show Hot Spare –** A toggle button that shows or hides the bottom portion of the window. ■ **List of slices –** A scrolling list of the slices included in the Hot Spare Pool. ■ **Enable –** Enables selected slices that are disabled. ■ **Remove –** Removes the selected slices from the Hot Spare Pool. ■ **Info –** Displays the Slice Information window for the selected (highlighted) slice. ■ **Slice –** Specifies a new slice to attach or replace the selected slice. ■ **Attach –** Attaches the slice specified in the Slice field to the Hot Spare Pool. This button is active only when a slice name has been entered in the field. ■ **Replace –** Replaces the selected spare slice with the slice entered in the field. This button is active only when a slice name has been entered in the field and a slice is selected on the list of slices.

RAID Information Window

These methods display the RAID Information window (see Figure 4–17):

- Double-click the RAID5 metadevice in the Objects list. The RAID5 metadevice is opened on the Metadevice Editor's canvas. Select Info from the Object menu.
- If the RAID5 metadevice is on the Metadevice Editor's canvas, point inside the top of the rectangle. Press and hold the MENU button to display the pop-up menu then select the Info choice.
- If the RAID5 metadevice is on the Metadevice Editor's canvas, point inside the top of the rectangle and double-click.

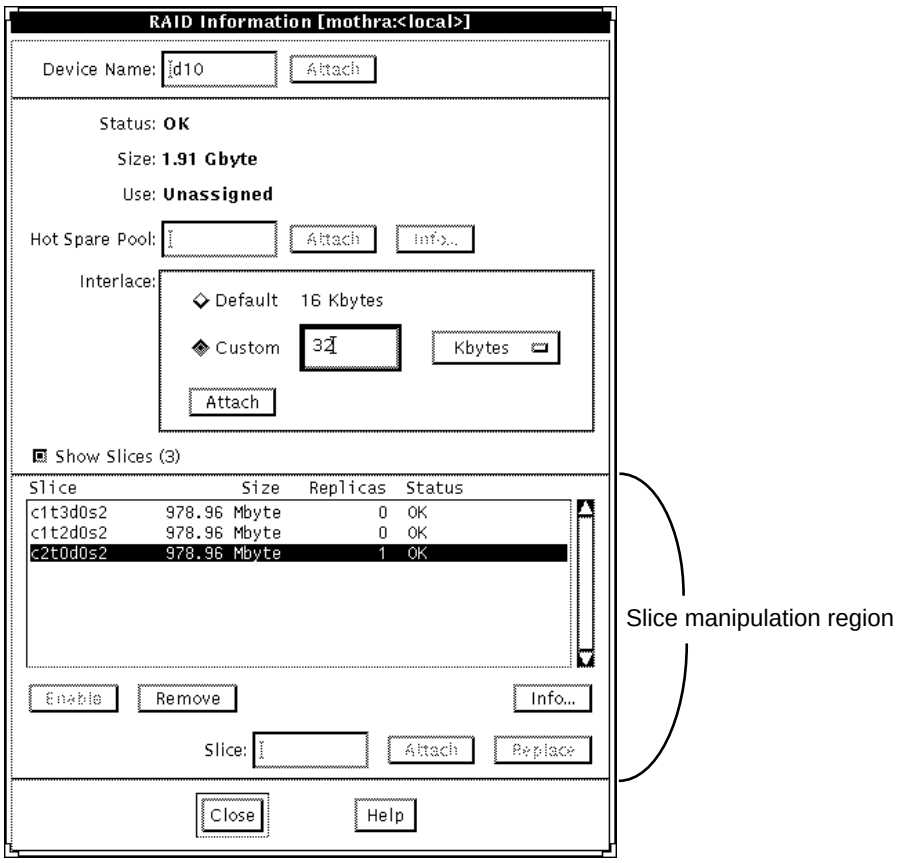

Figure 4–17 RAID Information Window

The RAID5 metadevice must be committed before the changes take effect.

Table 4–12 lists the functionality associated with the regions of the RAID Information window.

TABLE 4–12 RAID Information Window Functionality

Field	Functions
Device Name	The metadevice name of the RAID5 metadevice. As long as the device is not “open,” you can change the name by typing a new one and clicking Attach.
Status	Description of the RAID5 metadevice’s status.
Size	The size of the usable disk space. This does not include the size set aside for parity.

TABLE 4–12 RAID Information Window Functionality *(continued)*

Field	Functions
Use	The use of the RAID5 metadevice, for example, file system or swap. If the use of the RAID5 metadevice is a Trans Log, a Show Trans button is positioned to the right of the field.
Hot Spare Pool	This field enables assigning a Hot Spare Pool to the RAID5 metadevice. It has the following functions: ■ **Attach/Detach –** Attaches or detaches the specified Hot Spare Pool to the RAID5 metadevice. ■ **Info –** Displays the Hot Spare Pool Information window for the specified Hot Spare Pool.
Interlace value region	The default interlace value is 16 Kbytes. To change the interlace value, click on the Custom button and type the value in the field. The menu button to the right of the field enables you to specify the units used. The values on the menu are Gbytes, Mbytes, Kbytes, and Sectors. The default is Kbytes. After the Custom field is complete, the Attach button is used to assign the interlace value to the RAID5 metadevice. After a RAID5 metadevice is committed, the interlace value cannot be changed.
Slice manipulation region	The following functionality is available in this region: ■ **Show Slices –** A toggle button that shows or hides the scrolling list of components at the bottom of the window. ■ **Scrolling List –** A list of the slices included in the RAID5 metadevice. The information in this region includes the name of the slice, size, number of state database replicas on the slice and the status. ■ **Enable –** Enables the selected slices if they are disabled. ■ **Remove –** Removes the selected slices. ■ **Slice –** Specifies a new slice to attach to the RAID5 metadevice or replaces the selected slice. ■ **Attach –** Attaches the slice specified in the Slice field to the RAID5 metadevice. This button is active only when a slice name is entered in the field. ■ **Replace –** Replaces the selected RAID5 slice with the slice entered in the Slice field. This button is active only when a slice name has been entered in the field and a slice is selected from the scrolling list. ■ **Info –** Displays the Slice Information window for the selected (highlighted) slice.

Metadevice State Database Info Window

There are three ways to display the Metadevice State Database Information window shown in Figure 4–18:

- Double-click the MetaDB object in the Objects list. The MetaDB object is opened on the Metadevice Editor's canvas. Select Info from the Object menu.
- If the MetaDB object is on the Metadevice Editor's canvas, point inside the top of the rectangle. Press and hold the MENU button to display the pop-up menu then select the Info choice.
- If the MetaDB object is on the Metadevice Editor's canvas, point inside the top of the rectangle and double-click.

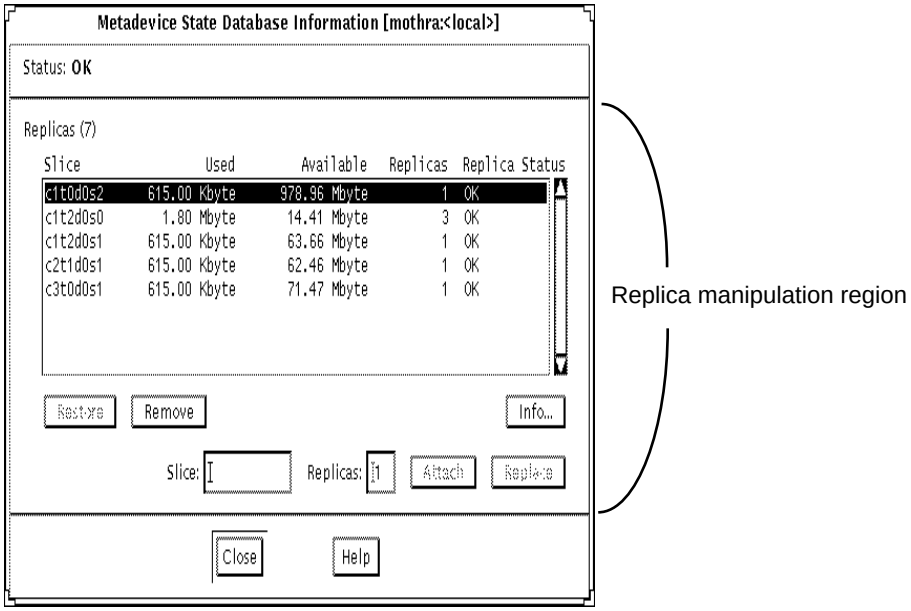

Figure 4–18 Metadevice State Database Information Window

The MetaDB object must be committed before the changes take effect.

Table 4–13 lists the functionality associated with the regions of the Metadevice State Database Information window.

TABLE 4–13 Metadevice State Database Information Window Functionality

Field	Functions
Status	Description of the metadevice state database's status.
Replica manipulation region	This region shows the following information and allows for manipulation. ■ **Replicas –** Shows the number of replicas. ■ **Scrolling List –** A scrolling list of the slices that contain replicas. The information in this field includes the name of the slice, the amount of space used, space available, number of replicas on the slice and the replica status. ■ **Restore –** Restores the selected slices if they are in error. ■ **Remove –** Removes the selected slices. ■ **Info –** Displays the Slice Information window for the selected (highlighted) slice. ■ **Slice –** This field is used to specify a new slice to attach to the MetaDB or replace the selected slice. ■ **Replicas –** Shows the number of replicas that will be created on the slice. This value is set to one by default. ■ **Attach –** Adds the slice entered in the slice field to the Replica list. This button is available only when a slice name is entered. ■ **Replace –** Replaces the selected slice with the slice entered in the Slice field.

Tray Information Window

The Tray Information window, as seen in Figure 4–19, displays information about a specific SPARCstorage Array tray. To display the Tray Information window, point to a SPARCstorage Array Tray on the Disk View canvas. Press and hold the MENU button to display the pop-up menu then select the Info option.

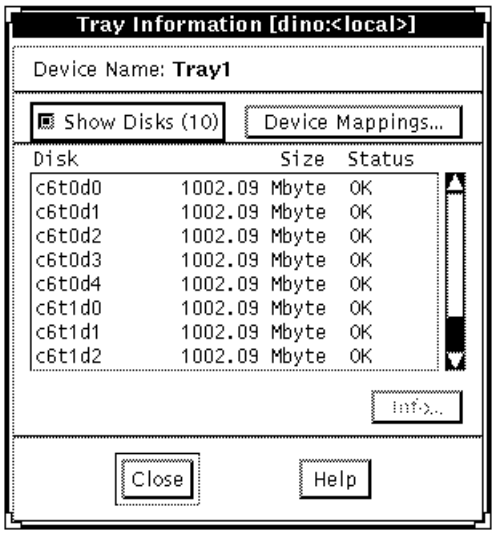

Figure 4–19 Tray Information Window

Table 4–14 lists the functionality associated with the Tray Information window.

TABLE 4–14 Tray Information Window Functionality

Field	Functions
Device Name	Names the tray (for example, Tray1).
Show Disks	Toggles on and off the disk information pane.
Device Mappings	Displays the Physical to Logical Device Mapping window. (The Physical to Logical Device Mappings window is not dynamically updated when new mappings are created.)
Disk information region	Contains a scrolling list of all disks, their size and status, on the tray.
Info	Selecting a disk in the disk information pane and clicking the Info button displays the Disk Information window for that disks.

Controller Information Window

The Controller Information window, as seen in Figure 4–20, displays information about a disk's controller. To display the Controller Information window, point to a

controller on the Disk View canvas. Press and hold the MENU button to display the pop-up menu then select the Info option.

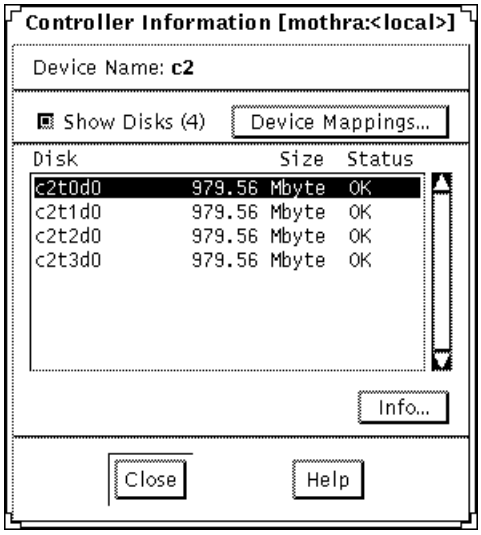

Figure 4–20 Controller Information Window

Table 4–15 lists the functionality associated with the Controller Information window.

TABLE 4–15 Controller Information Window Functionality

Field	Functions
Device Name	Names the controller (for example, c2).
Show Disks	Toggles on and off the disk information pane.
Device Mappings	Displays the Physical to Logical Device Mappings window. (The Physical to Logical Device Mappings window is not dynamically updated when new mappings are created.)
Disk information scrolling list	Contains a scrolling list of all disks, their size and status, on the controller.
Info	Selecting a disk in the disk information pane and clicking the Info button displays the Disk Information window for that disk.

Table 4–16 lists additional functionality for SPARCstorage Arrays.

TABLE 4–16 Controller Information Window, SPARCstorage Array Functionality

Field	Functions
Fan Status	Displays the current fan status (for example, Failed).
Battery Status	Displays the current battery status.
Vendor	Displays the vendor name.
Product ID	Displays the product identification number.
Product Rev	Displays the product revision number.
Firmware Rev	Displays the product firmware revision information.

Browsers

Three browsers can be accessed from the Browse menu on the Metadevice Editor window. These include:

- Slice Browser
- Metadevice Browser
- Hot Spare Pool Browser

The browsers provide similar functionality, enabling you to view all information about the slices, metadevices, and hot spare pools and drag these objects to the Metadevice Editor's canvas for manipulation. The only noticeable differences are found in some of the information displayed and in the Set Filter windows.

The Slice Browser window is show in Figure 4–21.

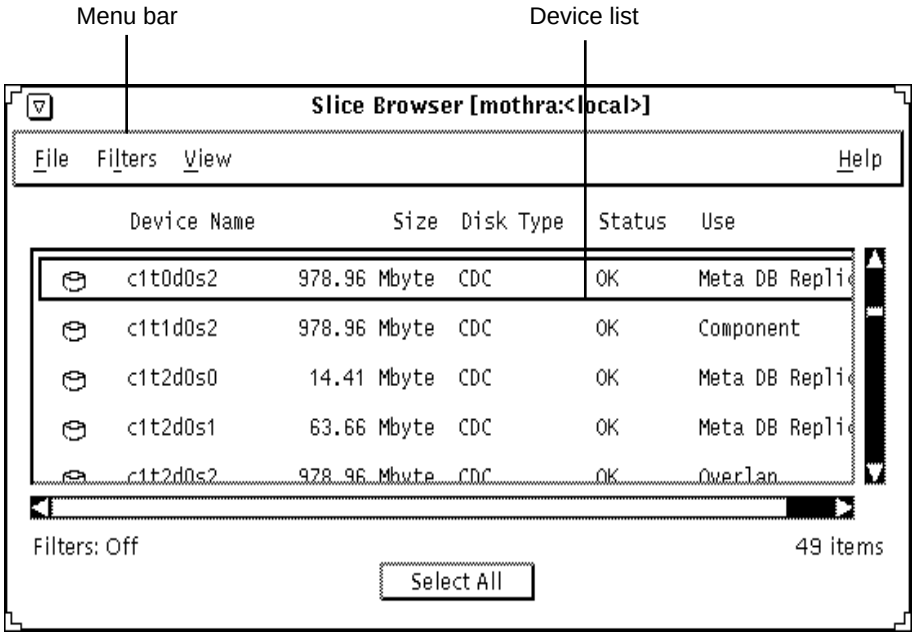

Figure 4–21 Slice Browser Window

The Slice, Metadevice, and Hot Spare Pool browsers all have the same window title bar and choices on the menu bar. The File menu enables you to exit the browser. The Filters menu enables you to set the filters and turn them on and off. The View menu enables you to change the order in which information is displayed in the device list. However, there are some subtle differences in the dialog boxes used to set the filters.

The device list varies in the following ways:

- **Slice Browser Device List –** To view additional information about the slices listed here, point to a slice and double-click the SELECT button. The Slice Information window displays information about the slice and provides access to the Disk Information and Associations windows. The Slice Browser device list contains the information shown in Table 4–17.

TABLE 4–17 Slice Browser Device List Information

Field	Function
Device Name	The device name, for example, c1t0d0s2.
Size	The total size of the device.
Disk Type	The type of the disk as listed in the `/etc/format.dat` file, for example SUN0535, SUN1.05, or DEFAULT.

TABLE 4–17 Slice Browser Device List Information *(continued)*

Field	Function
Status	Reported as OK, Resyncing, Enabled, Critical, Spared, Urgent, or Attention.
Use	Contains one of the following values: Unassigned, Trans Log, Trans Master, MetaDB Replica, Component, File System currently mounted on slice, Overlap, or Hot Spare.

- **Metadevice Browser Device List –** To view additional information about the metadevices listed, point to a metadevice and double-click the SELECT button. An information window is displayed. The Metadevice Browser device list contains the information shown in Table 4–18.

TABLE 4–18 Metadevice Browser Device List Information

Field	Function
Name	The metadevice is represented as d*n*, where the default value for n is a number in the range 0 to 127.
Status	The status is reported as OK, Critical, Urgent, or Failed.
Size	The total size of the metadevice.
Use	The use is either Unassigned, Submirror of d*n*, name of a file system, Master of d*n*, or Trans Log.
Type	The type is reported as either Trans, Concat/Stripe, Mirror, or RAID.

- **Hot Spare Pool Device List –** To view additional information about the hot spare pools listed, point to a hot spare pool and double-click the SELECT mouse button. The Hot Spare Information window is displayed, showing a list of the metadevices that have an association with the hot spare pool. It also shows information about the disks in the pool. The Hot Spare Pool device list contains the information shown in Table 4–19.

TABLE 4–19 Hot Spare Pool Device List Information

Field	Function
Name	The name of the hot spare pool is displayed as `hsp`nnn, where nnn is a number in the range 000 to 999.
Status	The status is OK, Broken (if all slices in the hot spare pool are broken), or Attention (if one or more slices are in use).
Smallest	The size of the smallest slice in the hot spare pool.
Largest	The size of the largest slice in the hot spare pool.
Spares	The number of hot spares in the pool.
Spares in Use	The number of hot spares currently in use.

Accessing Objects in the Browsers

All objects in the device list of any of the three browsers can be moved to the Metadevice Editor's canvas for manipulation.

Setting Browser Filters

The three browsers have configurable Filter windows that are available using the Set Filters choice in the Filter item on the menu bar. The filters are used to change the way information is displayed in the device list. Figure 4–22 shows the Slice Filters window.

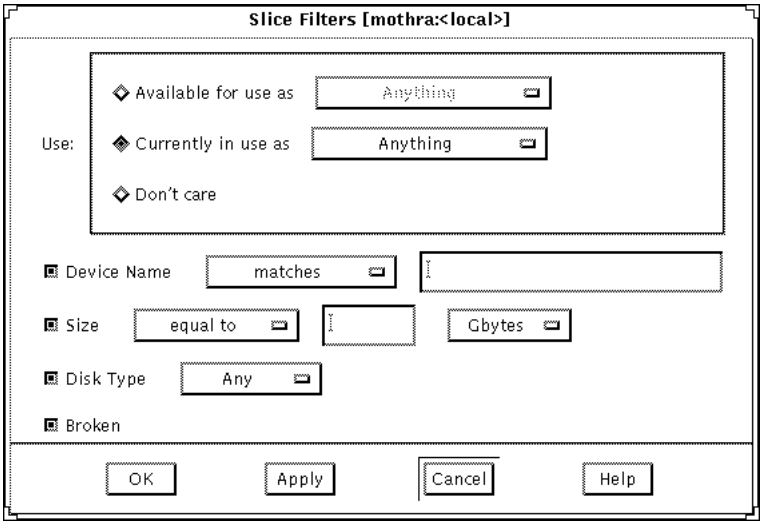

Figure 4–22 Slice Filter Window

Table 4–20 explains the items in the Slice Filter window.

TABLE 4–20 Slice Filter Window Items

Window Region	Function
Use	Three radio buttons in this region enable you to filter the display to only show the following: ■ **Available for use as –** The menu selections here include: Anything, Hot Spare, Replica, Metadevice, or Trans Log. Anything is the default. ■ **Currently in use as –** The menu selections here include: Anything, File System, Swap, Replica, Metadevice, Hot Spare, or Trans Log. Anything is the default. ■ **Don't care –** The filter is ignored.
Device Name toggle button	Turning on the name toggle button enables you to specify a device name. The two choices on the menu enable you to search for names that match or do not match. Wildcard character support includes both the asterisk (*) and question mark (?), which match any number of characters and any single character respectively. Matches is the default.
Size toggle button	Turning on the size toggle button enables you to specify a size for the filter. The menu button selections include: from (a field is added for specifying a "to" range), greater than, less than, equal to, and not equal to. The default is equal to. A size menu button enables you to specify Gbytes, Mbytes, Kbytes, and Sectors.

TABLE 4–20 Slice Filter Window Items *(continued)*

Window Region	Function
Disk Type toggle button	Turning on the Disk Type toggle button enables you to select the types of disks you wish to have displayed in the browser. The menu always enables you to select Any, but the other selections depend on the types of disks attached to your system.
Broken toggle button	Searches only for slices that have a “broken” status.

The Finder

The Finder is used to locate an object in the Metadevice Editor Window, or to locate the device associated with a specified mount point. The Finder is accessed from the Browse menu on the Metadevice Editor window.

- To locate an object inside the Metadevice Editor window, select the Find choice and either type the device name, or click the radio button beside Mount Point and type the mount point to find (see Figure 4–23). If the object is anywhere on the canvas, it is placed in the upper left corner. The object will become the current selection (any previously selected objects will be deselected.) If the object is in the Device List, it is opened and placed in the upper left corner of the canvas. The text fields are not case sensitive. Wildcard character support includes both the asterisk (*) and question mark (?). The asterisk matches zero or more characters and the question mark matches one character.

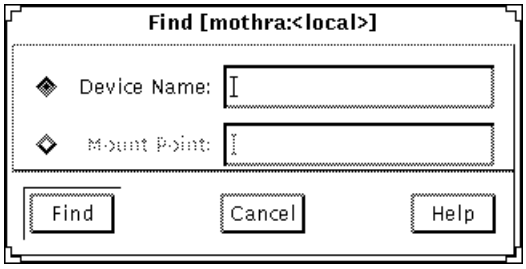

Figure 4–23 Finder Window

Dialog Boxes

DiskSuite Tool displays feedback via four different types of dialog boxes at various times. You must respond to a dialog box before you can perform any other action in DiskSuite Tool.

Caution - Read and understand the dialog boxes before responding. You can inadvertently lose data.

An example of a warning dialog box is shown in Figure 4–24.

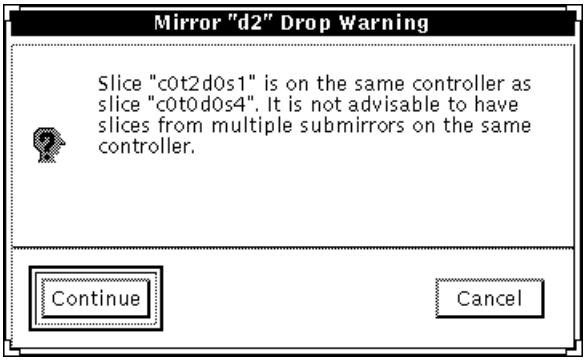

Figure 4–24 Example Dialog Box

The types of dialog boxes and the information they display are shown in Table 4–21.

TABLE 4–21 Dialog Boxes

Type	Information Presented
Error	When you attempt to perform an action that will result in an error, an error dialog box appears with a notification of the error.
Warning	When you attempt to perform an action that results in a warning, you are given the opportunity to cancel the action. Appendix A offers a listing of the error messages and the corrective action.
Confirmation	These provide a way for you to confirm an action that has been selected. These will appear when an action you initiated cannot be undone. The message string in each dialog varies according to the operation.
Information	These provide a helpful message. These dialog boxes appear with a large “i” on the left side of the message.

Configuration Log Window

The Configuration Log window, as shown in Figure 4–25, provides a history of all top-level DiskSuite operations. Each item on the list is given a time stamp.

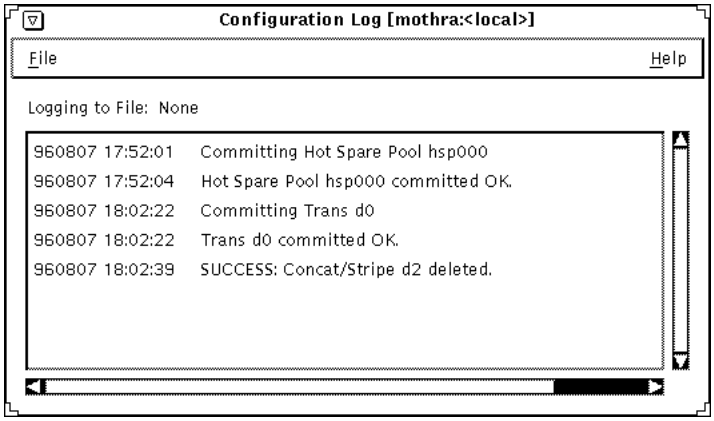

Figure 4–25 Configuration Log Window

Selections on the Configuration Log window's File menu enable you to clear the scrolling list, log the messages to a user-designated file, and close the window. Double-clicking an entry in the list brings up the information dialog window for the device and opens the device on the Metadevice Editor's canvas.

Problem List Window

The Problem List window contains a scrolling list of the current metadevice problems. The list does not provide a history of the problems that have been encountered. The list is updated each time DiskSuite Tool learns of a change in status. Each item on the list is given a time stamp.

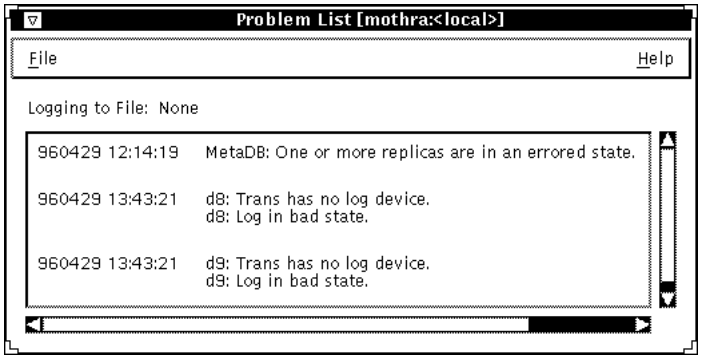

Figure 4–26 Problem List Window

Selections on the Problem List window's File menu enable you to log the messages to a user-designated file and close the window. The text field on the right side of the button displays the date and time of the most recent update.

Double-clicking an entry in the list brings up the information window for the device and places the device on the Metadevice Editor's canvas.

Note - When DiskSuite Tool is minimized, its icon flashes when there is a critical problem.

Accessing and Using Help

The DiskSuite Tool online help program provides detailed information about the DiskSuite Tool and the functionality it provides.

- To access online help, click Help on the menu bar. Then select either "On Help" or "On Window" from the menu.
- To access the online help from within a window, click the Help button.

The DiskSuite Tool help utility is shown in Figure 4–27.

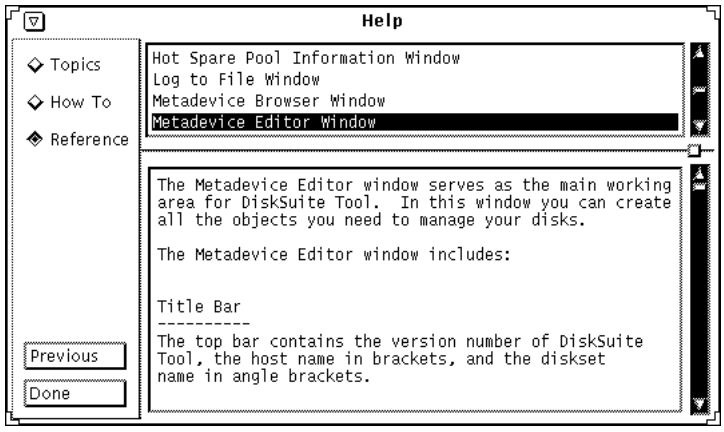

Figure 4–27 DiskSuite Tool Help Utility

The Help titles displayed in the top window pane identify the list of subjects available for each level of help.

The text in the bottom window pane describes information about using the current menu or command.

Use the scrollbars to the right of each pane to scroll through the help information displayed.

On the left side of the Help utility are buttons used to find information and navigate through the help system. The buttons are described in Table 4–22.

TABLE 4–22 DiskSuite Tool Help Buttons

Button	Click On This Button To ...	Then ...
Topics	Display a glossary of DiskSuite terms	Click on a title in the top window pane to view the accompanying help text.
How To	Display instructions for using the help	
Reference	Display screen-level help	
Previous	Return to the last-accessed help topic	The help viewer returns to the previous screen.
Done	Exit the online help system	The help system is closed.

Tool Registry

This is an application registry file used by DiskSuite Tool to initialize its “Tools” menu selection. Refer to the `metatool-toolsmenu(4)` man page for more information.

Event Notification

Event notification is a feature that keeps you aware of dynamic state changes, such as creation of a metadevice, a change in a metadevice status, or device errors. Event notification takes care of the following:

- More than one administrator at a time, if necessary, can run DiskSuite Tool on the same host with the assurance that state changes are propagated to each instance of DiskSuite Tool.

- When running multiple instances of DiskSuite Tool on the same host, event notification ensures that proper locking occurs to prevent one instance of DiskSuite Tool from overwriting the changes made by another. When one DiskSuite Tool has an uncommitted action, it has a “lock” until a commit occurs or the device is removed.

Note - Though you can run multiple instances of DiskSuite Tool on the same host, it is best to avoid doing so.

- You can run both DiskSuite Tool and the command line utilities together. Event notification is able to pass state changes from the command line to DiskSuite Tool.

Note - DiskSuite Tool provides the same functionality as the `ssaadm(1M)` command to start and stop a disk. However, do not use DiskSuite Tool and the `ssaadm(1M)` together. Doing so could cause DiskSuite Tool to incorrectly display a disk‘s status. Always use one or the other to both stop and start a disk.

CHAPTER 5

Disksets

This chapter explains shared disksets. Use the following table to proceed directly to the section that provides the information you need.

What Do Disksets Do?

A *shared diskset*, or simply *diskset*, is a set of shared disk drives containing metadevices and hot spares that can be shared exclusively but not at the same time by two hosts. Currently, disksets are only supported on SPARCstorage Array disks.

How Does DiskSuite Manage Disksets?

A diskset provides for data redundancy and availability. If one host fails, the other host can take over the failed host's diskset. (This type of configuration is known as a *failover configuration*.)

Though each host can control the set of disks, neither host has access to the set of disks at the same time that the other host controls the set of disks.

Note - Disksets are intended for use with Solstice HA, or another supported third-party HA framework. DiskSuite by itself does not provide all the functionality necessary to implement a failover configuration.

In addition to the shared diskset, each host has a *local diskset*. This consists of all of the disks on a host not in a shared diskset. A local diskset belongs to a specific host. The local diskset contains the metadevice state database for that specific host's configuration.

Metadevices and hot spare pools in a shared diskset can only consist of drives from within that diskset. Once you have created a metadevice within the diskset, you can use it just as you would a physical slice. However, disksets do not support the mounting of file systems from the `/etc/vfstab` file.

Similarly, metadevices and hot spare pools in the local diskset can only consist of drives from within the local diskset.

When you add disks to a diskset, DiskSuite automatically creates the state database replicas on the diskset. When a drive is accepted into a diskset, DiskSuite repartitions it so that the state database replica for the diskset can be placed on the drive. Drives are repartitioned when they are added to a diskset only if Slice 7 is not set up correctly. A small portion of each drive is reserved in Slice 7 for use by DiskSuite. The remainder of the space on each drive is placed into Slice 0. Any existing data on the disks is lost by repartitioning. After adding a drive to a diskset, it may be repartitioned as necessary, with the exception that Slice 7 is not altered in any way.

Unlike local diskset administration, you do not need to create or delete diskset metadevice state databases by hand. DiskSuite tries to place a reasonable number of state database replicas (on Slice 7) on across all drives in the diskset. If necessary, however, you can manually administer the replicas. (See *Solstice DiskSuite 4.2.1 User's Guide*)

Note - Disksets are not intended for "local" (not dual-connected) use.

Diskset Conventions

- **What are the diskset naming conventions?**

  Disksets use this naming convention:

  `/dev/md/`*setname*

  Metadevices within the shared diskset use these naming conventions:

  `/dev/md/`*setname*`/{dsk | rdsk}/d`*number*

  where *setname* is the name of the diskset, and *number* is the metadevice number (usually between 0-127).

Hot spare pools use *setname*/hsp*xxx*, where *xxx* is in the range 000-999.

Metadevices within the local diskset have the standard DiskSuite metadevice naming conventions. (See Table 1–4.)

- **What are the maximum number of disksets possible?**

  32 (though the default is 4). The actual number of shared disksets is always one less than the number configured, to account for the local diskset.

- **What are the hardware requirements for a diskset?**

  Currently, disksets are only supported on SPARCstorage Array drives. Disksets do not support SCSI disks.

  Three or more SPARCstorage Arrays are recommended to avoid losing one-half of the configuration, which would result in the diskset being inaccessible.

  The two hosts connected to the shared disks must be "symmetric." The shared disk drives must be the same. (Refer to the next question for specifics.)

- **What are the requirements for shared disk drive device names?**

  A shared disk drive must be seen on both hosts at the same device number (c#t#d#). The disk drive must also have the same major/minor number. If the minor numbers are not the same on both hosts, typically you see the message "drive c#t#d# is not common with host xxxx" when adding drives to the diskset. Finally, the shared disks must use the same driver name (`ssd`). See *Solstice DiskSuite 4.2.1 User's Guide* for more information on setting up shared disk drives in a diskset.

- **Are disksets supported on single-host configurations?**

  Disksets are supported in single-host configurations, but the disksets still must be manually "taken" and "released." (See "Administering Disksets" on page 108.) Usually, this is too much trouble for non-HA use.

- **Are disksets supported on the x86 platform?**

  No.

- **What are the requirements for creating a diskset?**

  To create a diskset, you need `root` in group 14, or you need a `/.rhosts` file entry containing the other hostname (on each host).

- **Can a file system that resides on a metadevice in a diskset be mounted automatically at boot via the `/etc/vfstab` file?**

  No. The necessary diskset RPC daemons (`rpc.metad` and `rpc.metamhd`) do not start early enough in the boot process to permit this. Additionally, the ownership of a diskset is lost during a reboot.

Example — Two Shared Disksets

Figure 5–1 shows an example configuration using two disksets.

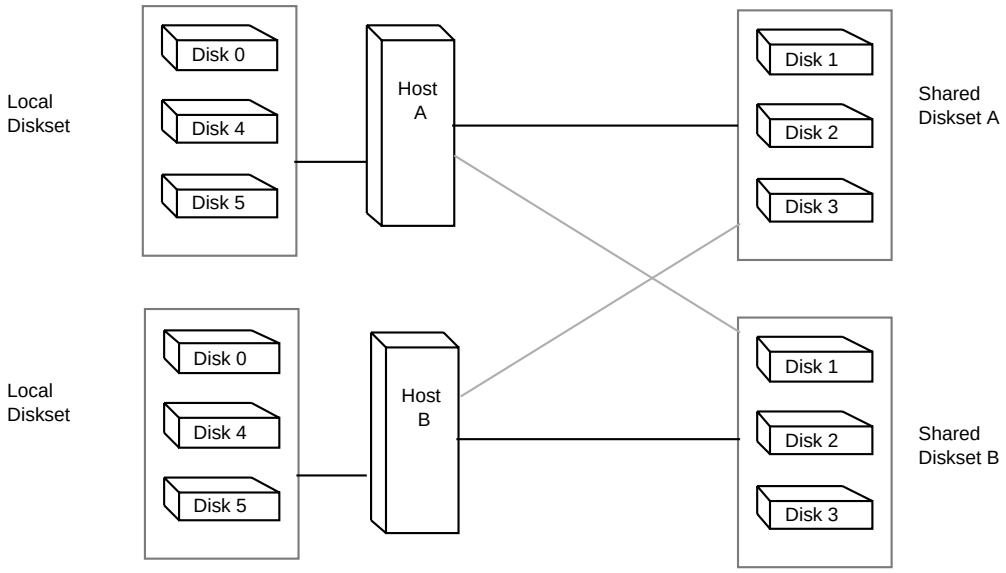

Figure 5–1 Disksets Example

In this configuration, Host A and Host B share disksets A and B. They each have their own local diskset, which is not shared. If Host A fails, Host B can take over control of Host A's shared diskset (Diskset A). Likewise, if Host B fails, Host A can take control of Host B's shared diskset (Diskset B).

Administering Disksets

Disksets must be created and configured using the DiskSuite command line interface (the `metaset(1M)` command). After you have created a diskset, you can administer state database replicas, metadevices, and hot spare pools within a diskset using either DiskSuite Tool or the command line utilities.

After drives are added to a diskset, the diskset can be *reserved* (or *taken*) and *released* by hosts in the diskset. When a diskset is reserved by a host, the other host in the diskset cannot access the data on the drives in the diskset. To perform maintenance on a diskset, a host must be the owner of the diskset or have reserved the diskset. A host takes implicit ownership of the diskset by putting the first drives into the set.

The SCSI `reserve` command is issued to each drive in the diskset to reserve it for exclusive use by the current host. Each drive in the diskset is probed once every second to determine that it is still reserved.

Note - If a drive has been determined unexpectedly not to be reserved, the host will panic. This behavior helps to minimize data loss which would occur if two hosts were to simultaneously access the same drive.

Reserving a Diskset

Before a host can use drives in a diskset, the host must reserve the diskset. There are two methods of reserving a diskset:

- **Safely -** When you safely reserve a diskset, DiskSuite checks to see if another host currently has the set reserved. If another host has the diskset reserved, your host will not be allowed to reserve the set.
- **Forcibly -** When you forcibly reserve a diskset, DiskSuite reserves the diskset whether or not another host currently has the set reserved. This method is generally used when a host in the diskset is down or not communicating. All disks within the set are taken over and FailFast is enabled. The metadevice state database is read in on the host performing the reservation and the shared metadevices configured in the set become accessible. If the other host had the diskset reserved at this point, it would panic due to reservation loss.

  Normally, two hosts in a diskset cooperate with each other to ensure that drives in a diskset are reserved by only one host at a time. A normal situation is defined as both hosts up and communicating with each other.

Releasing a Diskset

Sometimes it may be desirable to release a diskset. Releasing a diskset can be useful when performing maintenance on the drives in the set. When a diskset is released, it cannot be accessed by the host. If both hosts in a diskset release the set, neither host in the diskset can access the drives in the set.

CHAPTER 6

The md.tab and md.cf Files

This chapter describes how to use the /ect/lvm/md.tab file. It also explains the purpose of the /ect/lvm/md.cf file. Use the following table to locate specific information in this chapter.

Overview of the md.tab File

The /ect/lvm/md.tab file is an ASCII file that the metainit(1M), metahs(1M), and metadb(1M) commands use as an input file. You can use the /ect/lvm/md.tab file to create metadevices, hot spare pools, and state database replicas in a way similar to batch processing (as opposed to using the command line or DiskSuite Tool). Once you have edited the file, the metainit(1M), metahs(1M), and metadb(1M) commands then activate the metadevices, hot spare pools, and state database replicas defined in the file.

When you edit the /ect/lvm/md.tab file, you specify one complete configuration entry for each line using the syntax of the metainit(1M), metadb(1M), and metahs(1M) commands.

You then run the metainit(1M) command with either the –a option, to activate all metadevices in the /ect/lvm/md.tab file, or with the metadevice name that corresponds to a specific entry in the file.

Note - DiskSuite does not write to or store configuration information in the /ect/lvm/md.tab file. You must edit the file by hand and run the metainit(1M), metahs(1M), or metadb(1M) commands to create DiskSuite objects.

The following sections describe how to create state database replicas, metadevices, and hot spare pools using the md.tab file. For more information refer to the md.tab(4) man page.

Creating Initial State Database Replicas in the md.tab File

This example shows how to set up an initial state database on a server with three disks.

```
#
# (state database and replicas)
#
mddb01 -c 3 c0t1d0s0 c0t2d0s0 c0t3d0s0
```

This file entry creates three state database replicas on each of the three slices. mddb01 identifies the metadevice state database. –c 3 specifies that three state database replicas are placed on each slice. The metadb(1M) command activates this entry in the /ect/lvm/md.tab file.

Creating a Striped Metadevice in the md.tab File

This example shows a striped metadevice, /dev/md/dsk/d15, with two slices.

```
#
# (stripe consisting of two disks)
#
d15 1 2 c0t1d0s2 c0t2d0s2 -i 32k
```

The number 1 specifies to create a single stripe (a striped metadevice consisting of one stripe). The number 2 specifies how many slices to stripe. The –i 32k specifies a 32 Kbytes interlace value. (The default interlace value is 16 Kbytes.)

Creating a Concatenated Metadevice in the md.tab File

This example shows a metadevice, /dev/md/dsk/d7, that has a four-slice concatenated metadevice.

```
#
# (concatenation of four disks)
#
d7 4 1 c0t1d0s0 1 c0t2d0s0 1 c0t3d0s0 1 c0t4d0s0
```

The number 4 specifies to create four striped slices in the concatenated metadevice. Each stripe is made of one slice; therefore, you specify the number 1 for each slice.

Note - The first disk sector in the concatenated metadevice contains a disk label. To preserve the disk labels on devices /dev/dsk/c0t2d0s0, /dev/dsk/c0t3d0s0, and /dev/dsk/c0t4d0s0, DiskSuite skips the entire first cylinder of these disks. This permits higher-level file system software to optimize block allocations correctly.

Creating a Concatenated Stripe in the md.tab File

This example shows a metadevice, /dev/md/dsk/d75, that consists of two stripes that are concatenated together.

```
#
# (concatenation of two stripes, each made of three disks)
#
d75 2 3 c0t1d0s2 c0t2d0s2 c0t3d0s2 -i 16k \

     3 c1t1d0s2 c1t2d0s2 c1t3d0s2 -i 32k
```

The –i 16k specifies a 16 Kbytes interlace value for the first stripe. The –i 32k specifies a 32 Kbyte for the second stripe. The address blocks for each set of three disks are interlaced across three disks.

Creating a Mirror in the md.tab File

This example shows a one-way mirror, /dev/md/dsk/d50, and two other concatenations that would be attached later to create a three-way mirror. The mirror does not contain any data.

```
#
#
# (mirror)
#
d50 -m d51
d51 1 1 c0t1d0s2
d52 1 1 c0t2d0s2
d53 1 1 c0t3d0s2
```

The –m creates a one-way mirror consisting of submirror d51. The other two submirrors, d52 and d53, must be attached later with the metattach(1M) command. The default read and write options in this example are a round-robin read policy and parallel write policy.

Creating a Trans Metadevice in the md.tab File

This example shows a trans metadevice, /dev/md/dsk/d1, with logging device and master device.

```
#
# (trans)
#
d1 -t d10 d20
d10 -m d11
d11 1 1 c0t1d0s2
d12 1 1 c0t2d0s2
d20 -m d21
d21 1 1 c1t1d0s2
d22 1 1 c1t2d0s2
```

The –m creates the two one-way mirrors, d10 and d20. The –t creates d10 as the master device and d20 as the logging device. The submirrors d12 and d22 are attached later by using the metattach(1M) command on the d10 and d20 mirrors.

Creating a RAID5 Metadevice in the md.tab File

This example shows a RAID5 metadevice, d80, with three slices.

```
#
# (RAID devices)
#
d80 -r c0t1d0s1 c1t0d0s1 c2t0d0s1 -i 20k
```

The –r creates the RAID5 metadevice. The –i specifies an interlace value of 20 Kbytes. DiskSuite will stripe the data and parity information across the slices c0t1d0s1, c1t0d0s1, and c2t0d0s1. If you wanted to concatenate more slices to the original RAID5 metadevice, you could do so later by using the metattach(1M) command.

Creating a Hot Spare Pool in the md.tab File

This example shows a one-way mirror, /dev/md/dsk/d10, and two other concatenations that would be attached later to create a three-way mirror. Three hot spare pools are created and associated with the submirrors.

```
#
# (mirror and hot spare)
#
d10 -m d20
d20 1 1 c1t0d0s2 -h hsp001
d30 1 1 c2t0d0s2 -h hsp002
d40 1 1 c3t0d0s2 -h hsp003
hsp001 c2t2d0s2 c3t2d0s2 c1t2d0s2
hsp002 c3t2d0s2 c1t2d0s2 c2t2d0s2
hsp003 c1t2d0s2 c2t2d0s2 c3t2d0s2
```

The –m creates a one-way mirror consisting of submirror d20. You attach the other submirrors, d30 and d40, later by using the metattach(1M) command. The –h specifies which hot spare pools belong to the submirrors. There are three disks used as hot spares, each associated with three separate hot spare pools, hsp001, hsp002, and hsp003.

Note - The /ect/lvm/md.tab file can be used to both create and associate hot spare pools with metadevices at the same time.

Overview of the md.cf File

The /etc/lvm/md.cf file is a backup file of the DiskSuite configuration for a "local" diskset. You use the md.cf file for recovering from failures. Whenever you make a configuration change, the md.cf file is automatically updated (except for hot sparing). You never edit the md.cf file directly. If your system loses the information maintained in the metadevice state database, and as long as no metadevices were changed or created in the meantime, you can use the md.cf file to recover. Refer to *Solstice DiskSuite 4.2.1 User's Guide* for more information.

CHAPTER 7

Configuration Guidelines

Introduction

This appendix describes some ways to set up your configuration. Use the following table to locate specific information in this chapter.

Configuration Planning Overview

When planning a configuration, the main point to keep in mind is that for any given application there are trade-offs in *performance*, *availability*, and *hardware* costs. Experimenting with the different variables is necessary to figure out what works best for your configuration.

- **What are the performance trade-offs?**

Striping generally has the best performance, but it offers no data protection. For write intensive applications, mirroring generally has better performance than RAID5.

- **What are the availability trade-offs?**

  Mirroring and RAID5 metadevices both increase data availability, but they both generally have lower performance, especially for write operations. Mirroring does improve random read performance.

- **What are the hardware cost trade-offs?**

  RAID5 metadevices have a lower hardware cost than mirroring. Both striped metadevices and concatenated metadevices have no additional hardware cost.

Configuration Planning Guidelines

This section provides a list of guidelines for working with concatenations, stripes, mirrors, RAID5 metadevices, state database replicas, and file systems constructed on metadevices.

Concatenation Guidelines

- A concatenated metadevice uses less CPU time than striping.
- Concatenation works well for small random I/O.
- Avoid using physical disks with different disk geometries.

  Disk geometry refers to how sectors and tracks are organized for each cylinder in a disk drive. The UFS organizes itself to use disk geometry efficiently. If slices in a concatenated metadevice have different disk geometries, DiskSuite uses the geometry of the first slice. This fact may decrease the UFS file system efficiency.

Note - Disk geometry differences do not matter with disks that use Zone Bit Recording (ZBR), because the amount of data on any given cylinder varies with the distance from the spindle. Most disks now use ZBR.

- When constructing a concatenation, distribute slices across different controllers and busses. Cross-controller and cross-bus slice distribution can help balance the overall I/O load.

Striping Guidelines

- Set the stripe's interlace value correctly.

- The more physical disks in a striped metadevice, the greater the I/O performance. (The MTBF, however, will be reduced, so consider mirroring striped metadevices.)
- Don't mix differently sized slices in the striped metadevice. A striped metadevice's size is limited by its smallest slice.
- Avoid using physical disks with different disk geometries.
- Distribute the striped metadevice across different controllers and busses.
- Striping cannot be used to encapsulate existing file systems.
- Striping performs well for large sequential I/O and for random I/O distributions.
- Striping uses more CPU cycles than concatenation. However, it is usually worth it.
- Striping does not provide any redundancy of data.

Mirroring Guidelines

- Mirroring may improve read performance; write performance is always degraded.
- Mirroring improves read performance only in threaded or asynchronous I/O situations; if there is just a single thread reading from the metadevice, performance will not improve.
- Mirroring degrades write performance by about 15-50 percent, because two copies of the data must be written to disk to complete a single logical write. If an application is write intensive, mirroring will degrade overall performance. However, the write degradation with mirroring is substantially less than the typical RAID5 write penalty (which can be as much as 70 percent). Refer to Figure 7–1.

  Note that the UNIX operating system implements a file system cache. Since read requests frequently can be satisfied from this cache, the read/write ratio for physical I/O through the file system can be significantly biased toward writing.

  For example, an application I/O mix might be 80 percent reads and 20 percent writes. But, if many read requests can be satisfied from the file system cache, the physical I/O mix might be quite different—perhaps only 60 percent reads and 40 percent writes. In fact, if there is a large amount of memory to be used as a buffer cache, the physical I/O mix can even go the other direction: 80 percent reads and 20 percent writes might turn out to be 40 percent reads and 60 percent writes.

Figure 7–1 Mirror Performance Matrix

RAID5 Guidelines

- **RAID5 can withstand only a single device failure.**

  A mirrored metadevice can withstand multiple device failures in some cases (for example, if the multiple failed devices are all on the same submirror). A RAID5 metadevice can only withstand a single device failure. Striped and concatenated metadevices cannot withstand any device failures.

- **RAID5 provides good read performance if no error conditions, and poor read performance under error conditions.**

  When a device fails in a RAID5 metadevice, read performance suffers because multiple I/O operations are required to regenerate the data from the data and parity on the existing drives. Mirrored metadevices do not suffer the same degradation in performance when a device fails.

- **RAID5 can cause poor write performance.**

  In a RAID5 metadevice, parity must be calculated and both data and parity must be stored for each write operation. Because of the multiple I/O operations required to do this, RAID5 write performance is generally reduced. In mirrored metadevices, the data must be written to multiple mirrors, but mirrored performance in write-intensive applications is still much better than in RAID5 metadevices.

- **RAID5 involves a lower hardware cost than mirroring.**

  RAID5 metadevices have a lower hardware cost than mirroring. Mirroring requires twice the disk storage (for a two-way mirror). In a RAID5 metadevice, the amount required to store the parity is: 1/#-disks.

- **RAID5 can't be used for existing file systems.**

  You can't encapsulate an existing file system in a RAID5 metadevice (you must backup and restore).

State Database Replica Guidelines for Performance

- All replicas are written when the configuration changes.
- Only two replicas (per mirror) are updated for mirror dirty region bitmaps.
- A good average is two replicas per three mirrors.
- Use two replicas per one mirror for write intensive applications.
- Use two replicas per 10 mirrors for read intensive applications.

File System Guidelines

- The default inode density value (`–i` option) for the `newfs(1M)` command is not optimal for large file systems. When creating a new file system with the `newfs` command, you should set the inode density to 1 inode per 8 Kbyte of file space (`–i 8192`), rather than the default 1 inode per 2 Kbyte. Typical files today are approaching 64 Kbyte or larger in size, rather than the 1 Kbyte which typified files in 1980.
- For large metadevices (greater than 8 Gbyte), it may be necessary to increase the size of a cylinder group to as many as 256 cylinders as in:

```
# newfs -c 256 /dev/md/rdsk/d114
```

Note - The man page in Solaris 2.3 and 2.4 incorrectly states that the maximum size is 32 cylinders.)

- If possible, set your file system cluster size equal to some integral of the stripe width.

  For example, try the following parameters for sequential I/O:

  `maxcontig = 16` (16 * 8 Kbyte blocks = 128 Kbyte clusters)

  Using a four-way stripe with a 32 Kbyte interlace value results in a 128 Kbyte stripe width, which is a good performance match:

  interlace size = 32 Kbyte (32 Kbyte stripe unit size * 4 disks = 128 Kbyte stripe width)

- You can set the `maxcontig` parameter for a file system to control the file system I/O cluster size. This parameter specifies the maximum number of blocks, belonging to one file, that will be allocated contiguously before inserting a rotational delay.

  Performance may be improved if the file system I/O cluster size is some integral of the stripe width. For example, setting the `maxcontig` parameter to 16 results in 128 Kbyte clusters (16 blocks * 8 Kbyte file system block size).

Note - The options to the `mkfs(1M)` command can be used to modify the default `minfree`, inode density, cylinders/cylinder group, and `maxcontig` settings. You can also use the `tunefs(1M)` command to modify the `maxcontig` and `minfree` settings.

See the man pages for `mkfs(1M)`, `tunefs(1M)`, and `newfs(1M)` for more information.

General Performance Guidelines

- Assign data to physical drives to evenly balance the I/O load among the available disk drives.
- Identify the most frequently accessed data, and increase access bandwidth to that data with mirroring or striping.
- Both striped metadevices and RAID5 metadevices distribute data across multiple disk drives and help balance the I/O load. In addition, mirroring can also be used to help balance the I/O load.
- Use DiskSuite Tool performance monitoring capabilities, and generic OS tools such as `iostat(1M)`, to identify the most frequently accessed data. Once identified, the "access bandwidth" to this data can be increased using mirroring, striping, or RAID5.

RAID5 Metadevices and Striped Metadevices

This section compares performance issues for RAID5 metadevices and striped metadevices.

- **How does I/O for a RAID5 metadevice and a striped metadevice compare?**
  - The striped metadevice performance is better than the RAID5 metadevice, but it doesn't provide data protection (redundancy).
  - RAID5 metadevice performance is lower than striped metadevice performance for write operations, because the RAID5 metadevice requires multiple I/O operations to calculate and store the parity.
  - For raw random I/O reads, the striped metadevice and the RAID5 metadevice are comparable. Both the striped metadevice and RAID5 metadevice split the data across multiple disks, and the RAID5 metadevice parity calculations aren't a factor in reads except after a slice failure.

- For raw random I/O writes, the striped metadevice performs better, since the RAID5 metadevice requires multiple I/O operations to calculate and store the parity.
- For raw sequential I/O operations, the striped metadevice performs best. The RAID5 metadevice performs lower than the striped metadevice for raw sequential writes, because of the multiple I/O operations required to calculate and store the parity for the RAID5 metadevice.

Optimizing for Random I/O and Sequential I/O

This section explains the differences between random I/O and sequential I/O, and DiskSuite strategies for optimizing your particular configuration.

Random I/O

- **What is random I/O?**

  Databases and general-purpose file servers are examples of random I/O environments. In random I/O, the time spent waiting for disk seeks and rotational latency dominates I/O service time.

- **Why do I need to know about random I/O?**

  You can optimize the performance of your configuration to take advantage of a random I/O environment.

- **What is the general strategy for configuring for a random I/O environment?**

  You want all disk spindles to be busy most of the time servicing I/O requests. Random I/O requests are small (typically 2-8 Kbytes), so it's not efficient to split an individual request of this kind onto multiple disk drives.

  The interlace size doesn't matter, because you just want to spread the data across all the disks. Any interlace value greater than the typical I/O request will do.

  For example, assume you have 4.2 Gbytes DBMS table space. If you stripe across four 1.05-Gbyte disk spindles, and if the I/O load is truly random and evenly dispersed across the entire range of the table space, then each of the four spindles will tend to be equally busy.

  The target for maximum random I/O performance on a disk is 35 percent or lower as reported by DiskSuite Tool's performance monitor, or by `iostat(1M)`. Disk use in excess of 65 percent on a typical basis is a problem. Disk use in excess of 90 percent is a major problem.

If you have a disk running at 100 percent and you stripe the data across four disks, you might expect the result to be four disks each running at 25 percent (100/4 = 25 percent). However, you will probably get all four disks running at greater than 35 percent since there won't be an artificial limitation to the throughput (of 100 percent of one disk).

Sequential Access I/O

- **What is sequential I/O?**

  While most people think of disk I/O in terms of sequential performance figures, only a few servers—DBMS servers dominated by full table scans and NFS servers in very data-intensive environments—will normally experience sequential I/O.

- **Why do I need to know about sequential I/O?**

  You can optimize the performance of your configuration to take advantage of a sequential I/O environment.

  The goal in this case is to get greater sequential performance than you can get from a single disk. To achieve this, the stripe width should be "small" relative to the typical I/O request size. This will ensure that the typical I/O request is spread across multiple disk spindles, thus increasing the sequential bandwidth.

- **What is the general strategy for configuring for a sequential I/O environment?**

  You want to get greater sequential performance from an array than you can get from a single disk by setting the interlace value small relative to the size of the typical I/O request.

- `max-io-size / #-disks-in-stripe`

  Example:

  Assume a typical I/O request size of 256 Kbyte and striping across 4 spindles. A good choice for stripe unit size in this example would be:

  256 Kbyte / 4 = 64 Kbyte, or smaller

Note - Seek and rotation time are practically non-existent in the sequential case. When optimizing sequential I/O, the internal transfer rate of a disk is most important.

The most useful recommendation is: `max-io-size / #-disks`. Note that for UFS file systems, the `maxcontig` parameter controls the file system cluster size, which defaults to 56 Kbyte. It may be useful to configure this to larger sizes for some sequential applications. For example, using a `maxcontig` value of 12 results in 96 Kbyte file system clusters (12 * 8 Kbyte blocks = 96 Kbyte clusters). Using a 4-wide stripe with a 24 Kbyte interlace size results in a 96 Kbyte stripe width (4 * 24 Kbyte = 96 Kbyte) which is a good performance match.

Example: In sequential applications, typical I/O size is usually large (greater than 128 Kbyte, often greater than 1 Mbyte). Assume an application with a typical I/O request size of 256 Kbyte and assume striping across 4 disk spindles. Do the arithmetic: 256 Kbyte / 4 = 64 Kbyte. So, a good choice for the interlace size would be 32 to 64 Kbyte.

Number of stripes: Another way of looking at striping is to first determine the performance requirements. For example, you may need 10.4 Mbyte/sec performance for a selected application, and each disk may deliver approximately 4 Mbyte/sec. Based on this, then determine how many disk spindles you need to stripe across:

```
10.4 Mbyte/sec / 4 Mbyte/sec = 2.6
```

Therefore, 3 disks would be needed.

Striping Trade-offs

- Striping cannot be used to encapsulate existing file systems.
- Striping performs well for large sequential I/O and for uneven I/O distributions.
- Striping uses more CPU cycles than concatenation, but the trade-off is usually worth it.
- Striping does not provide any redundancy of data.

To summarize the trade-offs: Striping delivers good performance, particularly for large sequential I/O and for uneven I/O distributions, but it does not provide any redundancy of data.

Write intensive applications: Because of the read-modify-write nature of RAID5, metadevices with greater than about 20 percent writes should probably not be RAID5. If data protection is required, consider mirroring.

RAID5 writes will never be as fast as mirrored writes, which in turn will never be as fast as unprotected writes. The NVRAM cache on the SPARCstorage Array closes the gap between RAID5 and mirrored configurations.

Full Stripe Writes: RAID5 read performance is always good (unless the metadevice has suffered a disk failure and is operating in degraded mode), but write performance suffers because of the read-modify-write nature of RAID5.

In particular, when writes are less than a full stripe width or don't align with a stripe, multiple I/Os (a read-modify-write sequence) are required. First, the old data and parity are read into buffers. Next, the parity is modified (XOR's are performed between data and parity to calculate the new parity—first the old data is logically subtracted from the parity and then the new data is logically added to the parity), and the new parity and data are stored to a log. Finally, the new parity and new data are written to the data stripe units.

Full stripe width writes have the advantage of not requiring the read-modify-write sequence, and thus performance is not degraded as much. With full stripe writes, all new data stripes are XORed together to generate parity, and the new data and parity are stored to a log. Then, the new parity and new data are written to the data stripe units in a single write.

Full stripe writes are used when the I/O request aligns with the stripe and the I/O size exactly matches:

```
interlace_size * (num_of_columns - 1)
```

For example, if a RAID5 configuration is striped over 4 columns, in any one stripe, 3 chunks are used to store data, and 1 chunk is used to store the corresponding parity. In this example, full stripe writes are used when the I/O request starts at the beginning of the stripe and the I/O size is equal to: `stripe_unit_size * 3`. For example, if the stripe unit size is 16 Kbyte, full stripe writes would be used for aligned I/O requests of size 48 Kbyte.

Performance in degraded mode: When a slice of a RAID5 metadevice fails, the parity is used to reconstruct the data; this requires reading from every column of the RAID5 metadevice. The more slices assigned to the RAID5 metadevice, the longer read and write operations (including resyncing the RAID5 metadevice) will take when I/O maps to the failed device.

Logging Device Trade-offs

- Logs (logging devices) are typically accessed frequently. For best performance, avoid placing them on heavily-used disks. You may also want to place logs in the middle of a disk, to minimize the average seek times when accessing the log.
- The log device and the master device of the same trans metadevice should be located on separate drives and possibly separate controllers to help balance the I/O load.

  Sharing logs: trans metadevices can share log devices. However, if a file system is heavily used, it should have a separate log. The disadvantage to sharing a logging device is that certain errors require that all file systems sharing the logging device must be checked with the `fsck(1M)` command.
- The larger the log size, the better the performance. Larger logs allow for greater concurrency (more simultaneous file system operations per second).
- The absolute minimum size for a logging device is 1 Mbyte. A good average for performance is 1 Mbyte of log space for every 100 Mbyte of file system space. A recommended minimum is 1 Mbyte of log for every 1 Gbyte of file system space.

  Assume you have a 4 Gbyte file system. What are the recommended log sizes?

- For good performance, a size of 40 Mbyte is recommended (1 Mbyte log / 100 Mbyte file system).
- A recommended minimum is 4 Mbyte (1 Mbyte log/1 Gbyte file system).
- The absolute minimum is 1 Mbyte.

- It is strongly recommended that you mirror all logs. It is possible to lose the data in a log because of device errors. If the data in a log is lost, it can leave a file system in an inconsistent state which `fsck` may not be able to repair without user intervention.

State Database Replicas

- State database replicas contain configuration and status information of all metadevices and hot spares. Multiple copies (replicas) are maintained to provide redundancy. Multiple copies also prevent the database from being corrupted during a system crash (at most, only one copy if the database will be corrupted).
- State database replicas are also used for mirror resync regions. Too few state database replicas relative to the number of mirrors may cause replica I/O to impact mirror performance.
- At least three replicas are recommended. DiskSuite allows a maximum of 50 replicas. The following guidelines are recommended:
  - For a system with only a single drive: put all 3 replicas in one slice.
  - For a system with two to four drives: put two replicas on each drive.
  - For a system with five or more drives: put one replica on each drive.
- In general, it is best to distribute state database replicas across slices, drives, and controllers, to avoid single points-of-failure.
- Each state database replica occupies 517 Kbyte (1034 disk sectors) of disk storage by default. Replicas can be stored on: a dedicated disk partition, a partition which will be part of a metadevice, or a partition which will be part of a logging - device.

Note - Replicas cannot be stored on the root (`/`), `swap`, or `/usr` slices, or on slices containing existing file systems or data.

Summary of State Database Replicas

- **Why do I need at least three state database replicas?**

Three or more replicas are required. You want a majority of replicas to survive a single component failure. If you lose a replica (for example, due to a device failure), it may cause problems running DiskSuite or when rebooting the system.

- **How does DiskSuite handle failed replicas?**

  The system will stay running with exactly half or more replicas. The system will panic when fewer than half the replicas are available to prevent data corruption.

  The system will not reboot without one more than half the total replicas. In this case, you must reboot single-user and delete the bad replicas (using the `metadb` command).

  As an example, assume you have four replicas. The system will stay running as long as two replicas (half the total number) are available. However, in order for the system to reboot, three replicas (half the total plus 1) must be available.

  In a two-disk configuration, you should always create two replicas on each disk. For example, assume you have a configuration with two disks and you only created three replicas (two on the first disk and one on the second disk). If the disk with two replicas fails, DiskSuite will stop functioning because the remaining disk only has one replica and this is less than half the total number of replicas.

Note - If you created two replicas on each disk in a two-disk configuration, DiskSuite will still function if one disk fails. But because you must have one more than half of the total replicas available in order for the system to reboot, you will be unable to reboot in this state.

- **Where should I place replicas?**

  If multiple controllers exist, replicas should be distributed as evenly as possible across all controllers. This provides redundancy in case a controller fails and also helps balance the load. If multiple disks exist on a controller, at least two of the disks on each controller should store a replica.

  Replicated databases have an inherent problem in determining which database has valid and correct data. To solve this problem, DiskSuite uses a majority consensus algorithm. This algorithm requires that a majority of the database replicas agree with each other before any of them are declared valid. This algorithm requires the presence of at least three initial replicas which you create. A consensus can then be reached as long as at least two of the three replicas are available. If there is only one replica and the system crashes, it is possible that all metadevice configuration data may be lost.

  The majority consensus algorithm is conservative in the sense that it will fail if a majority consensus cannot be reached, even if one replica actually does contain the most up-to-date data. This approach guarantees that stale data will not be accidentally used, regardless of the failure scenario. The majority consensus algorithm accounts for the following: the system will stay running with exactly half or more replicas; the system will panic when fewer than half the replicas are available; the system will not reboot without one more than half the total replicas.

APPENDIX **A**

DiskSuite Error Messages

Introduction

The first part of this appendix, “DiskSuite Tool Messages” on page 130, contains the status, error, and log messages displayed by DiskSuite’s graphical user interface, DiskSuite Tool. The second part of this appendix, “DiskSuite Command Line Messages” on page 157, contains the error and log messages displayed by the command line utilities.

Use the following table to locate DiskSuite Tool status, error, and log information.

Use the following table to locate DiskSuite command line error and log information.

DiskSuite Tool Messages

This part of this appendix contains the status, error, and log messages displayed by DiskSuite's graphical user interface, DiskSuite Tool.

State Information Terms

The dialog boxes shown in DiskSuite Tool use the following terms when the state of the components is reported.

- **OK –** The component is operating properly.
- **Resyncing –** The component is in the process of resyncing (copying) the data.
- **Erred –** The slice has encountered an I/O error or an open error. All reads and writes to and from this slice have been discontinued. See the *Solstice DiskSuite 4.2.1 User's Guide* for information on slice replacement.
- **Last erred –** The slice has encountered an I/O error or an open error. However, the data is not replicated elsewhere due to another slice failure. I/O is still performed on the slice. If I/O errors result, the mirror or RAID5 I/O will fail. See the *Solstice DiskSuite 4.2.1 User's Guide* for information on slice replacement.

Metadevice Editor Messages

The Metadevice Editor has a message line at the bottom of the window that displays information about objects and actions. There are two types of feedback displayed:

- When you are pointing the cursor at an object, the message line has the following format:

```
object_type object_name: attribute=value,...
```

- When you are dragging an object that is not yet populated, the message line has the form:

```
Drop requirement comp_type into new_object_type new_object_name
```

  Once the object is sufficiently populated, the message line has the form:

```
Drop comp_type into new_object_type new_object_name or commit
```

Dialog Box Error Messages

DiskSuite displays the following messages in Error dialog boxes. When these messages are displayed, the only choice on the dialog box is OK. The action you attempted will not be performed. Use the information here to correct the error.

```
A RAID can only have one operation that causes a resync per commit
```

You have tried to commit two separate changes to a RAID device at the same time. While the changes may be valid, only one can be performed at a time. For example, if you replace a slice and add a new slice, this message is displayed. You must perform one change and click on the Commit button, then perform the other change and click on the Commit button.

```
Concat dn has no stripes
```

You have tried to commit a concatenation that has no stripes. You must add stripes to the concatenation.

```
You cannot delete a metadevice that is in use.
```

You have tried to delete a metadevice that contains a mounted file system, is being swapped on, or is open as a raw device.

```
dn has no components.
```

You have tried to commit a Concat/Stripe template that has no slices. You must add slices to the object before clicking on the Commit button.

```
Mirror dn has no submirrors
```

You have tried to commit a mirror that has no submirrors. You must add submirrors before clicking on Commit.

```
RAID dn must have at least three slices.
```

You have tried to commit a RAID metadevice that has fewer than three slices. Add the necessary slices and commit the RAID metadevice.

```
Slices added to a RAID device must be at least
as large as the smallest original slice.
```

You have tried to add a slice to a RAID device that is smaller than the slices that are already part of the RAID device.

```
Slice slice is mounted. You cannot add it to Concat/Stripe dn, it
is not the first mounted slice to be added.
```

You have tried to add a slice that has a mounted file system to a Concat/Stripe and there is already at least one slice in the Concat/Stripe. The slice with the mounted file system must be the first one added to the Concat/Stripe.

```
Slice slice is mounted. You cannot add it to dn, it already has a
mounted slice.
```

You have tried to add a slice that contains a mounted file system to a Concat/Stripe template. The slice that contains the mounted file system must be the first slice added.

```
Slice slice is mounted. You cannot add a mounted slice to a RAID
device, doing so would corrupt the file system.
```

You have tried to add a slice that contains a mounted file system to a RAID template. Choose another slice that does not contain any data.

```
Slice slice is too small to be used in a RAID device.
```

You have tried to add a slice that is too small. The slice being added is either smaller than the slices already in the RAID device or is too small to be used in a RAID device.

```
Submirror dn has a mounted file system, it should be the first
submirror added.
```

You have tried to add a submirror that contains a mounted file system to an existing mirror. To mirror this file system, create a one-way mirror using this submirror then attach another submirror that does not contain data.

```
Submirror dn is too small.
```

You have tried to add a new submirror that is smaller than the current size of the Mirror. The submirror must be as large as the existing submirror.

```
Mirror dn has a component with a file system mounted. You cannot
add another submirror.
```

You have tried to add a submirror that contains a mounted file system and the Mirror already has a mounted file system on the other submirror. You must add an unassigned slice.

```
The root file system may not be mounted on
a concat with more than one stripe.
```

You have tried to drop the slice that contains the root file system into a Concat/Stripe template. Remove one of the existing stripes.

```
The root file system may not be mounted on a trans device.
```

You have tried to drop the slice that contains the root file system into a Trans device template. The root file system cannot be placed in a Trans device.

```
Trans dn has no master
```

You have tried to commit a Trans device that has no master device. Add the master device and commit the device.

```
You cannot add device to a RAID device while it is initializing
```

You have tried to add a slice or hot spare pool to a RAID device that has been committed and is initializing. Wait until the device is initialized.

```
You cannot replace a slice in a RAID device while it is
initializing.
```

You have tried to add a slice to a RAID device that has been committed and is initializing. Wait until the device is initialized.

```
The value you entered value is too large.
You should use a value less than new value, which is the
maximum possible device size.
```

You tried to enter an unacceptably large value in one of the Slice Filter window‘s size fields.

```
Your attempt to change the name of Hot Spare Pool hspnnn
to hspnnn failed for the following reason:
```

You tried to change the name of a hot spare pool to a name that already exists, or the name is not valid.

```
RAID component component is not the same size as component
component. Extra space on the larger component will be wasted.
```

You tried to add a slice to a RAID5 metadevice that is not the same size as the existing slices in the RAID device.

```
You cannot change the hot spare pool for a RAID device while it
is initializing.
```

You tried to change the current hot spare pool for a RAID5 metadevice during its initialization. Wait for the initialization to complete before attempting to change the hot spare pool again.

```
The RAID device has failed to initialize. It cannot be repaired
and should be deleted.
```

There was an error when trying to initialize the RAID5 metadevice. The only recourse left is to delete the device and recreate it, after repairing any errored slices.

```
A slice in a created stripe may not be replaced unless the stripe
is part of a submirror with redundancy.
```

You can only replace a slices if data redundancy exists.

```
A slice in a stripe may not be enabled unless the stripe is part
of a submirror with redundancy.
```

You can only enable slices if data redundancy exists.

```
The metadevice state database has not been committed since slice
slice was added. You cannot restore replicas on the slice.
```

You need to commit the MetaDB object before enabling any broken slices.

```
There is no device with a mounted file system which matches
the path name path.
```

You tried to drag a file name from Storage Manager to the Metadevice Editor canvas and DiskSuite Tool could not locate the device containing the file system.

```
Disk Set Released
host no longer owns the setname disk set.
setname cannot continue; you must exit.
```

```
Disk Set Changed
An external change has occurred to the setname disk set.
```

```
Disk Set Load Failed
Unable to load disk set setname.
```

The above three messages indicate that changes were made to the diskset from the command line while DiskSuite Tool was running on that diskset.

```
Statistics sheets are not available for controllers, trays or
slices.
```

You tried to display the Device Statistics window for a controller, tray, or slice. The Device Statistics window is only available for metadevices or disks.

```
Sync NVRAM is only available for SPARCstorage Array controllers,
trays and disks with working batteries.
```

You tried to sync NVRAM on a non-SPARCstorage Array device, or you tried to sync NVRAM on a SPARCstorage Array whose battery has failed.

```
Fast Write is only available for SPARCstorage Array controllers,
trays and disks with working batteries.
```

You tried to enable Fast Write on a SPARCstorage Array device whose battery has failed.

```
Reserve Disks is only available for SPARCstorage Array
controllers, trays and disks.
```

You tried to perform a disk reservation on a non-SPARCstorage Array device.

```
Release Disks is only available for SPARCstorage Array
controllers, trays and disks.
```

You tried to release a disk reservation on a non-SPARCstorage Array device.

```
Start Disks is not available for slices.
```

Slices by themselves cannot be started, only disks.

```
Stop Disks is not available for slices.
```

Slices by themselves cannot be stopped, only disks.

```
You may not detach a submirror while it is resyncing.
```

You cannot detach a submirror in the process of resyncing.

```
Error while trying to reserve lock for metaset setname.
```

The lock for *setname* could not be acquired. Either another instance of DiskSuite Tool or the command line currently has the lock.

```
Interlace value is out of range.
```

You entered an invalid interlace value for the striped metadevice or RAID5 metadevice.

```
Failed trying to exchange metadevices dn and dn.
```

You cannot exchange metadevice names of a metadevice that is in use.

```
Failure trying to rename dn to dn.
```

You cannot rename a metadevice that is in-use, nor can you rename a metadevice to a name that already exists.

```
Metadevice name not in range dn - dn.
```

You tried to give a name to a metadevice outside the current defined range. If necessary, increase the value of `nmd` in the `/kernel/drv/md.conf` file.

```
The hot spare pool name not in the range hsp000 - hsp999.
```

Hot spare pools must be named `hsp`*nnn*, where *nnn* is a number between 000 and 999.

```
The hot spare pool hspnnn already exists.
```

You tried to create a hot spare pool with an existing hot spare pool's name.

```
You cannot delete a mounted trans device that has an attached
logging device.
```

To delete a trans metadevice, first detach the logging device.

```
The metadevice dn is currently either opened, swapped on, or
mounted. Deleting it will cause the name to be pushed down.
```

The mirror or trans metadevice you are trying to delete is currently in-use. The deleted device name will be switched with one of its subdevices names. In the case of a mirror, the mirror name is switched with one of its submirror names. In the case of a trans metadevice, the trans name and the master device name are switched.

```
You cannot delete a mounted mirror with more than one submirror.
```

To delete the mirror, make it into a one-way mirror.

```
You cannot delete a mounted trans device whose master is not a
metadevice.
```

You are attempting to delete a mounted trans device whose master device is a slice. Unmount the trans metadevice to be able to delete it.

```
Cannot purge the NVRAM for device. Disk is reserved by another host.

Cannot sync the NVRAM for device. Disk is reserved by another host.

Cannot reserve device. Disk is reserved by another host.

Cannot release device. Disk is reserved by another host.

Cannot start device. Disk is reserved by another host.

Cannot stop device. Disk is reserved by another host.

Cannot disable fast write for device. Disk is reserved by another
host.

Cannot enable fast write for device. Disk is reserved by another
host.

Cannot enable fast write for synchronous writes for device. Disk
is reserved by another host.
```

The above messages indicate that another host has a reservation on *device*. To perform the desired action, first release the reservation.

```
You cannot detach an existing submirror while a replace, enable
or attach is pending.
```

You tried to detach a submirror that is currently be replaced, enabled, or attached. Wait for the operation to complete before attempting the detach again.

```
You cannot enable a slice in a mirror while the mirror is
resyncing.
```

You tried to enable a slice in a submirror while the mirror is being resynced. Wait for the operation to complete before attempting the enable again.

Dialog Box Warning Messages

DiskSuite displays the following messages in Warning Dialog Boxes. When these messages are displayed you can either continue or undo the action. The information here will help you determine which action to select.

```
All of the state database replicas are on the same controller.
```

You have populated the MetaDB template with slices that are all attached to the same controller. If the controller fails, you will not have access to any of the metadevices.

```
The new Concat/Stripe device dn has a slice with a mounted file
system. If an entry for its file system exists in
''/etc/vfstab'' it will be updated when the Concat/Stripe is
committed so that the next mount of the file system will use the
new device. The system must be rebooted for this device mount to
take effect.
```

You have tried to add a slice that contains a mounted file system to a Concat/Stripe template. The slice that contains the mounted file system must be the first slice added. You cannot add a mounted file system to a RAID device.

```
Metadevice device_type dn will be deleted.
Data could be lost.
Really delete it?
```

This message displays when you attempt to delete any committed metadevice. You should continue only if you are sure your data is protected.

```
Stripe component dn is not the same size as component
dn. Extra space on the larger component will be wasted.
```

You have tried to add slices to a Concat/Stripe (stripe) that are a different size than the slices already in the stripe. Adding slices of a different sizes to a stripe causes wasted space.

```
Slice dn is on the same controller as slice dn.It is not advisable
to have slices from multiple submirrors on the same controller.
```

You have tried to create a Mirror with submirrors that are made up of slices attached to the same controller. If the controller fails, the mirror will not protect against lost data.

```
Slice dn is on the same disk as slice dn.It is not advisable to
have slices from multiple submirrors on the same disk.
```

You have tried to create a Mirror with submirrors that are made up of slices from the same disk. If the disk fails, the mirror will not protect against lost data.

```
Submirror dn is not the same size as submirror dn.
Extra space on the larger submirror will be wasted.
```

You have tried to create a Mirror that has differently sized submirrors. The extra space on the larger submirror cannot be used.

```
Submirror dn has an erred component.
Its data will not be valid after it is detached.
```

You have tried to detach or offline a submirror that has a slice reporting errors.

```
The file system mounted on metadevice dn has
been unmounted since the last status update.
```

You have tried to delete a metadevice that was unmounted. The device does not display the unmounted information. Select Rescan Configuration from the Metadevice Editor window's File menu to update this information.

```
The following components are in the erred state: dn
You may not replace RAID component dn until they are fixed.
```

You are replacing a component of a RAID metadevice that has reported errors (in the last-errored state). This cannot be performed if there are any other components in the RAID metadevice that have reported errors.

```
The following components are in the erred state: dn
The data for the component replacing RAID component dn may be
compromised.
```

You are replacing or enabling a RAID component that has reported errors. This action is dangerous if there is another component that has reported errors (in the last-errored state). The data on the new component may not be completely accurate.

```
The following components are in the last_erred state: dn
The data for RAID component dn may be compromised.
```

You are replacing or enabling a RAID component that is reporting errors. This action is dangerous if there is another component that has reported errors (in the last-errored state). The data on the new component may not be completely accurate.

```
The following components have erred: dn
The data for RAID component dn WILL NOT BE RESYNCED.
```

You have tried to replace a component in a RAID metadevice and there are two or more components reporting errors. It is not possible to replace the components because there is no way to recreate the data. If you proceed with the replacement, you must obtain the data from a backup copy.

```
The format of disk dn has changed.
You must restart metatool to incorporate the changes.
```

You have reformatted a disk that used to have a metadevice, file system, or database replica and selected the Rescan Configuration option from the Metadevice Editor window's File menu. If the disk is not being used, the new information is read by DiskSuite and displayed in the appropriate windows (for example, Slice Browser and Disk View).

```
The log device for Trans dn cannot be detached until
the Trans is unmounted or the system is rebooted.
```

You have tried to detach a log and commit the Trans object. The detach will not be performed as long as the log master is mounted. The Trans device is actually in a "detach pending state."

```
The master device dn for Trans dn has a mounted file system.

In order for logging of
this file system to be activated, the file
/etc/vfstab must be updated with the
new device name and the system rebooted.

Committing Trans dn will update
/etc/vfstab automatically if an entry exists
for the file system.
```

You have tried to add a metadevice that has a mounted file system to a Trans master. DiskSuite will automatically change the entry for the file system in the `/etc/vfstab` file. If an entry for the file system does not exist in the `/etc/vfstab` file, you must create one. The message also tells you to reboot the system.

```
The master device dn for Trans dn has a mounted file system.

If an entry for its file system exists in ''/etc/vfstab,'' it will
be updated with the new device to mount for the file system.

The system must be rebooted for this
device mount to take effect.
```

You have tried to add a master device that has a mounted file system to a Trans. DiskSuite will automatically change the entry for the file system in the `/etc/vfstab` file. If an entry for the file system does not exist in the `/etc/vfstab` file, you must create one. The message also tells you to reboot the system.

```
The metadevice dn has been removed as a swap device
since the last status update.
```

You have tried to delete a device that is the `swap` device. The device still says it is `swap`. To update the device's status, select Rescan Configuration from the Metadevice Editor window's File menu.

```
The new Mirror device dn
has a submirror with a mounted file system.

If an entry for its file system exists in /etc/vfstab, it will
be updated with the new device to mount for the file system.

The system must be rebooted for this
device mount to take effect.
```

You have tried to add a Concat/Stripe that has a mounted file system to a Mirror. DiskSuite will automatically change the entry for the file system in the `/etc/vfstab` file. If an entry for the file system does not exist in the `/etc/vfstab` file, you must create one. The message also tells you to reboot the system.

```
The state database will have no replicas.
If the system reboots, all metadevices will
be corrupted.
```

You have tried to remove the state database and all replicas from the MetaDB template. If you commit, you will not have access to any metadevices after the next reboot.

```
The submirror dn has a slice with a mounted
file system.

In order for mirroring of
this file system to be activated, the file
/etc/vfstab must be updated with the
new device name and the system rebooted.

Committing Mirror dn will update
/etc/vfstab automatically if an entry exists
for the file system.
```

You have tried to add a submirror that has a mounted file system to a Mirror. DiskSuite will automatically change the entry for the file system in the `/etc/vfstab` file. If an entry for the file system does not exist in the `/etc/vfstab` file, you must create one. The message also tells you to reboot the system.

```
This log is not mirrored. It is recommended that you mirror logs
whenever possible to avoid single points of failure.
```

You have tried to create a Trans device with a log that is not mirrored. If the log is not mirrored, the data could be lost or unavailable.

```
Trans d n has no log device.
```

You have tried to commit a Trans device that has no Trans log. You should add the log before committing the device. Until you add the log, the logging feature is disabled.

Dialog Box Information Messages

DiskSuite displays the following messages in information dialog boxes. These messages tell you a common mistake has been made or provide a helpful message. These dialog boxes appear with a large “i” on the left side of the message. There is only one button on the bottom of this dialog: OK.

```
You added additional capacity to the metadevice metadevice_type on
which file system file_system is mounted. You have the
option of growing this file system to take advantage of
this additional space either now or later (by hand).
If you choose to grow it now, the application will be
disabled until the growth process completes.

The command that will be run is:

    ‘‘growfs -M file_system /dev/md/rdsk/dn’’

Do you want to grow the metadevice now or later?
```

You have tried to add slices to a Concat/Stripe metadevice. Following a commit, you can expand the file system, as documented in the *Solstice DiskSuite 4.2.1 User’s Guide*.

```
The file system file_system mounted on metadevice dn
is now being grown.
```

You are growing a file system.

```
Statistic sheets are not available for the Metastate
database (metadb), Hot Spare Pools or slices.
```

You cannot display a Device Statistics window for the metadevice state database, hot spare pools, or slices.

Metadevice Editor Window Messages

The following messages are displayed when you are pointing to an object inside the Metadevice Editor window. An explanation of the message follows the sample output.

```
Click or drag to create a new object device
```

You are pointing at any of the five Template icons. *object* is either Trans, RAID, Mirror, Concat/Stripe, or Hot Spare Pool.

```
component_type dn: size=size, use=use, status=status
```

You are pointing at an object (*component_type*) on the canvas. The *component_type* is either Trans, RAID, Mirror, Concat/Stripe, or Hot Spare. The metadevice name is reported as d*n*, where the default size for *n* is a number in the range 0 to 127. The *size* is the capacity of the metadevice (for example, 500 Mbytes). The *use* is either Unassigned, Submirror, or */filesystem*. The *status* is reported as OK, Attention, Urgent, or Critical.

```
Drag objects into this work area to look at or modify them...
```

You are pointing at an empty canvas or the device list in the Metadevice Editor window.

```
hspnnn: status=status
```

You are pointing at a Hot Spare Pool on the canvas. The Hot Spare Pool name is reported as hsp*nnn*, where *nnn* is a number in the range 000 to 999. The *status* is reported as OK, Attention, Urgent, or Critical.

```
Slice cntndnsn: size=size, use=use, status=status
```

You are pointing at a disk slice on the canvas. The name of the slice appears in the format, c*n*t*n*d*n*s*n*. The *size* is the capacity of the slice (for example, 5 Mbytes). The *use* is either Unassigned or Component. The *status* is reported as OK, Attention, Urgent, or Critical.

```
Use Button2 to pan the viewport over the work area ...
```

You are pointing at the Panner. By pressing the middle button and moving the cursor, you move the canvas to a new view area.

Messages Displayed When Dragging an Object

The following messages are displayed when you are dragging an object inside the Metadevice Editor window. An explanation of the message follows the sample output.

```
Concatenations must be part of a mirror
for hot spare pools to function
```

You are dragging a Hot Spare Pool over a concatenation. This message is telling you that the Concat/Stripe must be part of a Mirror or the Hot Spare Pool you are dropping will not work.

```
Drop a concatenation into mirror dn
```

You are dragging a Concat/Stripe over the specified Mirror. If you drop the Concat/Stripe, it will become part of that Mirror.

```
Drop a concatenation to replace submirror dn
```

You are dragging a Concat/Stripe over the specified submirror. Drop the Concat/Stripe inside the rectangle that contains the submirror to make the replacement.

```
Drop a hot spare pool to assign it to concatenation dn
```

You are dragging a Hot Spare Pool over the specified concatenation. By dropping the Hot Spare Pool into the Concat/Stripe, it becomes associated with that concatenation.

```
Drop here to associate a hot spare pool with this RAID
```

You are dragging a Hot Spare Pool over the specified RAID device. If you drop the Hot Spare Pool, it will become associated with the RAID device.

```
Drop a metadevice or slice into trans log
```

You are dragging a metadevice or slice over a Trans device. If you drop the metadevice or slice, it will become part of the Trans device.

```
Drop a metadevice or slice into master
```

You are dragging a metadevice or slice over the Master of a Trans device. Drop the object into the Master to add it to the device.

```
Drop a slice into hot spare pool dn
```

You are dragging a slice over the specified Hot Spare Pool. Drop the slice to add it to the Hot Spare Pool.

```
Drop a slice into RAID dn
```

You are dragging an unused slice over the specified RAID device. If you drop the slice, it will become part of the RAID device.

```
Drop a slice of the same size to replace the current slice
```

You are dragging a slice either over a committed RAID device or over a submirror that has more than one submirror. You can drop the new slice on the existing slice to make a replacement.

```
Drop a slice to replace current slice
```

You are dragging an unused slice over a Concat/Stripe, RAID, or Trans device. To replace the slice you are over, release the middle button and drop the slice.

```
Drop a slice into stripe dn or commit
```

You are dragging an unused slice over a Concat/Stripe that has one or more slices. You can populate the Concat/Stripe with additional slices or select the Concat/Stripe (stripe) and execute a commit.

```
Drop a slice to add new replicas; you should have at least
three replicas.
```

You are dragging a slice over the MetaDB object. Drop the slice to create another replica. DiskSuite requires the configuration have a minimum of three slices in the MetaDB object.

```
Drop at least one concatenation into mirror dn
```

You are dragging a Concat/Stripe over the specified Mirror. You must drop a minimum of one Concat/Stripe into the specified Mirror.

```
Drop at least one slice into stripe dn
```

You are dragging an unused slice over a Concat/Stripe that has zero slices. You must populate the Concat/Stripe (stripe) with a minimum of one slice.

```
Drop at least three to create the RAID parity group
```

You are dragging an unused slice over the specified RAID device. You must drop a minimum of three slices into the RAID device.

```
Drop a slice to add a new replica.
```

You are dragging a slice over the MetaDB object. Drop the slice to create another replica.

```
You cannot add more concatenations;
mirror dn already has three submirrors
```

You are dragging a Concat/Stripe over the specified Mirror. You cannot add another Concat/Stripe (submirror) to a Mirror that already has three submirrors.

```
You cannot add slices to committed stripe dn
```

You are dragging an unused slice over a committed Concat/Stripe. DiskSuite does not permit you to add slices to a committed Concat/Stripe (stripe).

```
You cannot replace in-use slices in a hot spare pool
```

You are dragging an unused slice over a slice that is in use in a committed Hot Spare Pool. You cannot drop the new slice on a slice in a Hot Spare Pool that is currently in use.

```
You cannot replace objects in committed RAID dn
```

You are dragging an object over the specified RAID device. Because the device is committed, you cannot make replacements.

```
You cannot replace slices in a committed stripe unless
it is part of a submirror
```

You are dragging a slice over a a committed Concat/Stripe. You cannot make changes to this metadevice, unless it is part of a submirror.

```
You cannot replace submirror dn when
mirror dn has only one submirror
```

You are dragging a submirror over the specified submirror. You cannot drop the submirror into the Mirror when there is only one submirror present.

```
You cannot replace slices in a committed trans device
```

You are dragging an unused slice over a slice that is in use as a Trans master or log. You cannot replace slices in a committed Trans device.

Disk View Window Messages

The Disk View window has a message line at the bottom that displays information about objects and actions. There are two types of message line feedback displayed:

- When you are pointing the cursor at an object, the message line has the following format:

```
object_type object_name: attribute=value,...
```

  If you are pointing at a disk or slice that has a status problem, the message has the form:

```
object_type object_name: problem_description, affected_device
```

- When you are pointing the cursor at an empty portion of the canvas, the following message displays:

```
Drop object onto color drop sites to show mappings
```

  You can select a disk or slice and drag it to the color map at the bottom of the Disk View window. On a color monitor, you have four colors available as drop sites. On a monochrome monitor, you have one color drop site.

Messages Displayed When Pointing at an Object

The following messages are displayed when you are pointing to an object inside the Disk View window:

```
Slice cntndnsn: size=size, use=use, status=status
```

You are pointing at a disk slice on the canvas. The name of the slice appears in the format, c*n*t*n*d*n*s*n*. The size is the capacity of the slice (for example, 5 Mbytes). The use is either Unassigned, Component, Hot Spare, MetaDB Replica, Reserve, *mount_point*, swap, Trans Log, or Overlap. The *status* is reported as OK, Attention, Urgent, or Critical.

Messages Displayed When Dragging an Object

The following messages are displayed when you are dragging an object inside the Disk View window.

```
Drag slices onto object templates in the metadevice editor canvas
```

You are dragging an object from the Disk View window. You can drop the slices inside an object or on the canvas of the Metadevice Editor window.

Log Messages

Log messages are those passed by `syslog(3)` to the `syslogd(1M)`. These messages are appended to a file and written to the console window. These messages will not appear in any DiskSuite error or problem list.

The log messages are divided into the following categories:

- Notice log messages
- Warning log messages
- Panic log messages

The log messages displayed by DiskSuite are listed in alphabetical order below. Each message is always preceded with md: The variables in these messages indicate the following:

- *dev* is a device name.
- d*num* is a metadevice name.
- *num* is a number.
- *state* is a Trans device state.
- *trans* is either “logging” or “master.”

Note - When the initial portion of a message begins with a variable, the message is alphabetized by the first word following the variable.

Notice Log Messages

```
Could not load misc /dev
```

The named `misc` module is not loadable. It is possibly missing, or something else has been copied over it.

```
db: Parsing error on ‘dev’
```

The set command in `/etc/system` for the `mddb.bootlist`<*number*> is not in the correct format. Run `metadb` –`p` to place the correct set commands into the `/etc/system` file.

```
dnum: Hotspared device dev with dev
```

The first device name listed has been hot spare replaced with the second device name listed.

```
dnum: Hotspared device dev(num, num) with dev(num, num)
```

The first device number listed has been hot spare replaced with the second device number listed.

```
dnum: no mem for property dev
```

Memory could not be allocated in the `prop_op` entry point.

Warning Log Messages

```
dnum: not configurable, check /kernel/drv/md.conf
```

This error occurs when the number of metadevices as specified by the `nmd` parameter in the `/kernel/drv/md.conf` file is lower than the number of configured metadevices on the system. It can also occur if the `md_nsets` parameter for disksets is lower than the number of configured disksets on the system. To fix this problem, examine the `md.conf` file and increase the value of either `nmd` or `md_nsets` as needed.

```
dnum: Cannot load dev driver
```

The underlying named driver module is not loadable (for example, `sd`, `id`, or a third-party driver). This could indicate that the driver module has been removed.

```
Open error of hotspare dev
Open error of hotspare dev(num, num)
```

The named hot spare cannot be opened, or the underlying driver is not loadable.

```
dnum: read error on dev
dnum: write error on dev
```

A read or write error has occurred on the specified metadevice at the specified device name. This happens if any read or write errors occur on a metadevice.

```
dnum: read error on dev(num, num)
dnum: write error on dev(num, num)
```

A read or write error has occurred on the specified metadevice at the specified device number. This happens if any read or write errors occur on a metadevice.

```
dnum: read error on dnum
dnum: write error on dnum
```

A read or write error has occurred on the specified metadevice at the specified device number. This happens if any read or write errors occur on a metadevice.

```
State database commit failed
State database delete failed
```

These messages occur when there have been device errors on components where the state database replicas reside. These errors only occur when more than half of the replicas have had errors returned to them. For example, if you have three components with state database replicas and two of the components report errors, these errors may occur. The state database commit or delete is retried periodically. If a replica is added, the commit or delete will finish and the system will be operational. Otherwise, the system will time out and panic.

```
State database is stale
```

This message occurs when there are not enough usable replicas for the state database to be able to update records in the database. All accesses to the metadevice driver will fail. To fix this problem, add more replicas or delete inaccessible replicas.

```
trans device: read error on dnum
trans device: write error on dnum
```

A read or write error has occurred on the specified logging or master device at the specified metadevice. This happens if any read or write errors occur on a logging or master device.

```
trans device: read error on dev
trans device: write error on dev
```

A read or write error has occurred on the specified logging or master device at the specified device name. This happens if any read or write errors occur on a logging or master device.

```
trans device: read error on dev(num, num)
trans device: write error on dev(num, num)
```

A read or write error has occurred on the specified logging or master device at the specified device number. This happens if any read or write errors occur on a logging or master device.

```
logging device: dnum changed state to state
logging device: dev changed state to state
logging device: dev(num, num) changed state to state
```

The logging device and its associated master device(s) have changed to the specified state(s).

Panic Log Messages

```
State database problem
```

A failed metadevice state database commit or deletion has been retried the default 100 times.

```
dnum: Unknown close type
dnum: Unknown open type
```

A metadevice is being opened/closed with an unknown open type (OTYP).

DiskSuite Command Line Messages

This part of this appendix contains the error and log messages displayed by the command-line metadevice utilities of DiskSuite.

Errors that deal with command usage and other simple error messages are not documented in this appendix. All DiskSuite command line error messages are displayed in the following format:

```
program name: host: [optional1:] name: [optional2]:
error message...
```

where:

- *program name*: is the name of the application name and version being used (for example, *DiskSuite 4.2.1*).
- *host*: is the host name of the machine on which the error occurred (for example, *blue*).
- [*optional1*]: is an optional field containing contextual information for the specific error displayed for example. mountpoint or which daemon returned the error).
- *name*: is the command name which generated the error message (for example, `metainit`).
- [*optional2*]: is a second optional field containing additional contextual information for the specific error displayed
- *error message...* is the error message itself (as listed in this appendix).

For the purpose of this appendix, only the final portion (*error message...*) of each error message is listed.

The log messages listed near the back of this appendix are divided into three categories:

- Notice log messages
- Warning log messages
- Panic log messages

Error Messages

The command line error messages displayed by DiskSuite are listed in alphabetical order below. The message is preceded by some or all of the variables described in the previous section. Other variables included in these messages indicate the following:

- *nodename* is the name of a specific host.
- *drivename* is the name of a specific drive.
- *metadevice* is the number of a specific metadevice device or hot spare pool.
- *setname* is the name of a specific diskset.
- *num* is a number.

```
add or replace failed, hot spare is already in use
```

The hot spare that is being added or replaced is already in the hot spare pool.

```
administrator host nodename can't be deleted, other hosts still in
set. Use -f to override
```

The host which owns the diskset cannot be deleted from the diskset without using the –f option to override this restriction. When the –f option is used, all knowledge of the diskset is removed from the local host. Other hosts within the diskset are unaware of this change.

```
administrator host nodename deletion disallowed in one host admin
mode
```

The administrator host is the host which has executed the command. This host cannot be deleted from the diskset if one or more host in the diskset are unreachable.

```
already has log
```

The specified trans metadevice already has an attached logging device.

```
already used in metadevice
```

The specified component is currently being used in the *metadevice*.

```
attempt to detach last running submirror
```

An attempt was made to detach the last submirror. The operation would result in an unusable mirror. DiskSuite does not allow a `metadetach` to be performed on the last submirror.

```
attempt an operation on a submirror that has erred components
```

An attempt was made to take a submirror offline or detach a submirror that contains the data. The other submirrors have erred components. If this operation were allowed, the mirror would be unusable.

```
attempt an operation on a submirror in illegal state
```

An attempt was made to take a submirror offline that is not in the OKAY state or to online a submirror that is not in the offlined state. Use the `–f` option if you really need to offline a submirror that is in a state other than OKAY.

```
attempt to replace a component on the last running submirror
```

An attempt was made to replace a component in a one-way mirror.

```
attempted to clear mirror with submirror(s) in invalid state
```

The user attempted to use the `metaclear` command on a metamirror that contained submirrors that weren't in the OKAY state (Needs maintenance state). If the metamirror must be cleared, the submirrors must also be cleared. Use `–r` (recursive) to clear all the submirrors, or use `–f` (force) to clear a metamirror containing submirrors in the Needs maintenance state.

```
can't attach labeled submirror to unlabeled mirror
```

An attempt was made to attach a labeled submirror to an unlabeled mirror. A labeled metadevice is a device whose first component starts at cylinder 0. To prevent the submirror's label from being corrupted, DiskSuite does not allow labeled submirrors to be attached to unlabeled mirrors.

```
can't find component in unit
```

An attempt was made to replace or enable a component that did not exist in the specified metadevice.

```
can't find submirror in mirror
```

An attempt was made to either `metaonline(1M)`, `metaoffline(1M)`, or `metadetach(1M)` the submirror, d*num*. The submirror is not currently attached to the specified metamirror causing the command to fail.

```
can’t include device dev, it already exists in dnum
```

An attempt was made to use the device *dev* in a new metadevice and it already is used in the metadevice d*num*.

```
can’t include device dev, it overlaps with a device in dnum
```

The user has attempted to use device *dev* in a new metadevice which overlaps an underlying device in the metadevice, d*num*.

```
cannot delete the last database replica in the diskset
```

An attempt was made to delete the last database replica in a diskset. To remove all database replicas from a diskset, delete all drives from the diskset.

```
cannot enable hotspared device
```

An attempt was made to perform a `metareplace –e` (enable) on an underlying device which is currently hot spared. Try enabling the hot spare component instead.

```
can’t modify hot spare pool, hot spare in use
```

An attempt was made to modify the associated hot spare pool of a submirror, but the submirror is currently using a hot spare contained within the pool.

```
checksum error in mddb.cf file
```

The `/etc/lvm/mddb.cf` file has probably been corrupted or user-edited. The checksum this file contains is currently invalid. To remedy this situation: delete the `mddb.cf` file, delete a database replica, and add back the database replica.

```
component in invalid state to replace \
     - replace ‘‘Maintenance’’ components first
```

An attempt was made to replace a component that contains the only copy of the data. The other submirrors have erred components. If this operation were allowed, the mirror would be unusable.

```
data not returned correctly from disk
```

After a replica of the state database is first created, it is read to make sure it was created correctly. If the data read does not equal the data written this message is returned. This results from unreported device errors.

```
device not in set
```

An attempt was made to use a component for a shared metadevice or shared hot spare pool whose drive is not contained within the diskset.

```
device in shared set
```

An attempt was made to use a component for a local metadevice or local hot spare pool whose drive is contained within the diskset. The drives in the local diskset are all those which are not in any shared disksets.

```
device is too small
```

A component (*dev*) in stripe *num* is smaller than the interlace size specified with the –i flag in the `md.tab` file.

```
device size num is too small for metadevice database replica
```

An attempt was made to put a database replica on a partition that is not large enough to contain it.

```
devices were not RAIDed previously or are specified in the wrong
order
```

An attempt was made to metainit a RAID device using the –k option. Either some of the devices were not a part of this RAID device, or the devices were specified in a different order than they were originally specified.

```
drive drivename is in set setname
```

An attempt was made to add the drive *drivename* to a diskset which is already contained in the diskset *setname*.

```
drive drivename is in use
```

An attempt was made to add the drive *drivename* to a diskset, however a slice on the drive is in use.

```
drive drivename is not common with host nodename
```

An attempt was made to add the drive *drivename* to a diskset, however, the device name or device number is not identical on the local host and the specified *nodename*; or the drive is not physically connected to both hosts.

```
drive drivename is not in set
```

An attempt was made to delete the drive *drivename* from a diskset and the diskset does contain the specified drive.

```
drive drivename is specified more than once
```

The same drive (*drivename*) was specified more than once in the command line.

```
driver version mismatch
```

The utilities and the drivers are from different versions of the DiskSuite package. It is possible that either the last DiskSuite package added did not get fully installed (try running `pkgchk(1M)`), or the system on which DiskSuite was recently installed has not been rebooted since the installation.

```
failed to take ownership of a majority of the drives
```

Reservation of a majority of the drives was unsuccessful. It is possible that more than one host was concurrently attempting to take ownership of the same diskset. One host will succeed, and the other will receive this message.

```
growing of metadevice delayed
```

The attempted growth of a submirror has been delayed until a mirror resync finishes. The metamirror will be grown automatically upon completion of the resync operation.

```
has a metadevice database replica
```

An attempt was made to use a component (i.e., for a hot spare) which contains a database replica.

```
host nodename already has a set numbered setnumber
```

An attempt was made to add a host *nodename* to a diskset which has a conflicting *setnumber*. Either create a new diskset with both hosts in the diskset, or delete one of the conflicting disksets.

```
host nodename already has set
```

An attempt was made to add a host *nodename* to a diskset which has a different diskset using the same name. Delete one of the disksets and recreate the diskset using a different name.

```
host nodename does not have set
```

An attempt was made to delete a host or drive from a set, but the host *nodename* has an inconsistent view of the diskset. This host should probably be forcibly (`–f`) deleted.

```
host nodename is already in the set
```

An attempt was made to add a host *nodename* which already exists within the diskset.

```
host nodename is modifying set - try later or restart rpc.metad
```

Either an attempt was made to perform an operation on a diskset at the same time as someone else, or a previous operation dropped core and the `rpc.metad` daemon should be restarted on host *nodename*.

```
host nodename is not in the set
```

An attempt was made to delete the host *nodename* from a diskset which does not contain the host.

```
host nodename is specified more than once
```

The same host (*nodename*) was specified more than once in the command line.

```
host name nodename is too long
```

The name used for the host nodename is longer than DiskSuite accepts.

```
hotspare doesn't exist
```

An attempt was made to perform an operation on the hot spare *dev* and the specified hot spare does not exist.

```
hotspare in use
```

An attempt was made to perform an operation on the hot spare *dev* and the specified hot spare is in use.

```
hotspare isn't broken, can't enable
```

An attempt was made to enable a hot spare that is not in the broken state.

```
hotspare database create failure
```

An attempt to create a hot spare record in the metadevice state database failed. Run `metadb –i` to determine the cause of the failure.

```
hotspare pool database create failure
```

An attempt to create a hot spare pool record in the metadevice state database failed. Run `metadb –i` to determine the cause of the failure.

```
hotspare pool is busy
```

An attempt was made to delete the hot spare pool hsp*nnn* before removing all the hot spares associated with the specified hot spare pool.

```
hotspare pool is referenced
```

An attempt was made to delete the hot spare pool, hsp*nnn*, that is associated with a metadevice.

```
hotspare pool in use
```

An attempt was made to `metaclear(1M)` a hotspare pool without first removing its association with metadevices.

```
hotspare pool is already setup
```

An attempt was made to create a hot spare pool which already exists.

```
illegal option
```

An attempt was made to use an option which is not valid in the context of the specified metadevice or command.

```
in Last Erred state, errored components must be replaced
```

An attempt was made to replace or enable a component of a mirror in the “Last Erred” state when other components are in the “Erred” state. You must first replace or enable all of the components in the “Erred” state.

```
invalid RAID configuration
```

An invalid RAID device configuration entry was supplied to `metainit`, either from the command line or via the `md.tab` file.

```
invalid argument
```

An attempt was made to use an argument which is not valid in the context of the specified metadevice or command.

```
invalid column count
```

An invalid RAID configuration entry was supplied to `metainit`, either from the command line or via the `md.tab` file. Specifically, an invalid argument was provided with the `–o` option.

```
invalid interlace
```

An unsupported interlace value follows the `–i` option on a metadevice configuration line. The `–i` specifies the interlace size. The interlace size is a number (8, 16, 32) followed by either `–k` for kilobytes, `–m` for megabytes, or `–b` for blocks. The units can be either uppercase or lowercase. This message will also appear if the interlace size specified is greater than 100 Mbytes.

```
invalid mirror configuration
```

An invalid mirror configuration entry was supplied to `metainit`, either from the command line or via the `md.tab` file.

```
invalid pass number
```

An attempt was made to use a pass number for a mirror that is not within the 0 - 9 range.

```
invalid stripe configuration
```

An invalid stripe configuration entry was supplied to `metainit`, either from the command line or via the `md.tab` file.

```
invalid trans configuration
```

An invalid trans configuration entry was supplied to `metainit`, either from the command line or via the `md.tab` file.

```
invalid write option
```

An attempt was made to change the write option on a mirror using an invalid option. The legal strings are “serial” and “parallel.”

```
invalid hotspare pool
```

The metadevice configuration entry in the `md.tab` file has a `–h` hsp*nnn* and a `metainit` has not been performed on the hot spare pool.

```
invalid read option
```

The user has specified both the `–r` and `–g` options on the same metamirror.

```
invalid unit
```

The metadevice (submirror) passed to `metattach` is already a submirror. The metadevice may already be a submirror for another metamirror.

```
is a metadevice
```

The device *dev* being used is a metadevice and it should be a physical component.

```
is mounted on
```

The device *dev* in the metadevice configuration has a file system mounted on it.

```
hostname is not a nodename, but a network name
```

An attempt was made to add a host to a diskset without using the nodename found in the `/etc/nodename` file.

```
is swapped on
```

The device in the metadevice configuration is currently being used as a `swap` device.

```
maximum number of nodenames exceeded
```

An attempt was made to add more nodenames than DiskSuite allows in a diskset.

```
maxtransfer is too small
```

An attempt was made to add a component to a RAID device whose maxtransfer is smaller than the other components in the RAID device.

```
metadevice in use
```

An attempt was made to `metaclear(1M)` a submirror without first running metaclear on the metamirror in which it is contained.

```
metadevice is open
```

The metadevice (submirror) passed to metattach is already open (in use) as a metadevice.

```
num1 metadevice database replicas is too many; the maximum is num2
```

An attempt was made to add more databases (*num*1) than the maximum allowed (*num*2).

```
metadevice database has too few replicas, can't create new
records
```

An attempt to create a metadevice record in the metadevice state database failed. Run `metadb –a` to add more database replicas.

```
metadevice database is full, can't create new records
```

An attempt to create a metadevice record in the metadevice state database failed. Run `metadb –a` (and `–s`) to add larger database replicas. Then delete the smaller replicas.

```
metadevice database replica exists on device
```

An attempt was made to use a component (that is, for a hot spare) which contains a database replica.

```
metadevice is temporarily too busy for renames
```

An attempt was made to rename a metadevice that is open. An open metadevice is either mounted on, swapped on, or being used as the raw device by an application or database. To rename the metadevice, first make sure it is not open. This error can also appear if the –f option is not used when switching trans metadevice members, or when trying to switch trans metadevice members with a logging device still attached.

```
mirror has maximum number of submirrors
```

An attempt was made to attach more than the supported number of submirrors. The maximum supported number of submirrors is three.

```
must be owner of the set for this command
```

An attempt was made to perform an operation on a diskset or a shared metadevice on a host which is not the owner of the diskset.

```
must have at least 2 databases (-f overrides)
```

An attempt was made to delete database replicas, reducing the number of database replicas to a number less than two. To override this restriction, use the –f option.

```
must replace errored component first
```

An attempt was made to replace or enable a component of a mirror in the “Last Erred” state when other components are in the “Erred” state. You must first replace or enable all of the components in the “Erred” state.

```
no available set numbers
```

An attempt was made to create more disksets than DiskSuite allows.

```
no hotspare pools found
```

An `metahs` operation was attempted using the “–`all`” argument when no hot spare pools meet the criteria for the operation.

```
no metadevice database replica on device
```

An attempt was made to delete non-existent database replicas.

```
no such set
```

An attempt was made to perform an operation on a diskset or a shared metadevice using a non-existent set name.

```
nodename of host nodename creating the set must be included
```

An attempt was made to create a diskset on the local host without adding the name of the local host to the diskset.

```
not a disk device
```

The component name specified is not a disk device name. For example, a CD-ROM device doesn't have the characteristics of a disk device.

```
not enough components specified
```

An invalid stripe configuration entry was supplied to `metainit`, either from the command line or via the `md.tab` file.

```
not enough stripes specified
```

Invalid stripe configuration entry was supplied to `metainit`, either from the command line or via the `md.tab` file.

```
not enough submirrors specified
```

Invalid mirror configuration entry was supplied to `metainit`, either from the command line or via the `md.tab` file.

```
not in local set
```

An attempt was made to create a local metadevice or local hot spare pool with a component whose drive is contained in a shared diskset.

```
not a metadevice
```

The specified device is not a metadevice. DiskSuite expected a metadevice name.

```
only slice 7 is usable for a diskset database replica
```

An attempt was made to add a database replica for a shared diskset on a component other than Slice 7.

```
only the current owner nodename may operate on this set
```

An attempt was made to perform an operation on a diskset or a shared metadevice on a host which is not the owner of the diskset.

```
only valid action is metaclear
```

The initialization of a RAID device has failed. Use the `metaclear` command to clear the RAID device.

```
operation would result in no readable submirrors
```

An operation was attempted on a component or submirror that contains the only copy of the data. The other submirrors have erred components. If this operation were allowed, the mirror would be unusable.

```
operation requires -f (force) flag
```

Due to the components within the RAID device being in the “Maintenance” or “Last Erred” state, the force flag (`–f`) is required to complete the operation.

```
overlaps with device in metadevice
```

Overlapping slices are not allowed in metadevices or hot spare pools.

```
replace failure, new component is too small
```

An attempt to use `metareplace` failed because the new component is too small to replace the old component.

```
reserved by another host
```

An attempt was made to add a currently reserved drive to a diskset.

```
resync in progress
```

The mirror operation failed because a resync is being performed on the specified metamirror. Retry this operation when the resync is finished.

```
rpc.metad: permission denied
```

The user does not have permission to run a remote process on the other systems in the diskset. The remote access permissions need to be set up.

```
set setname is out of date - cleaning up - take failed
```

The diskset *setname* is out of data with respect to the other host’s view. This error should occur only after one-host administration.

```
set lock failed - bad key
```

Either another DiskSuite command is running and has locked the diskset or a DiskSuite command has aborted without unlocking the diskset on one of the hosts in the diskset. Check to see if there are other DiskSuite commands running on the hosts in the diskset. Check all the hosts and allow other commands to complete on all hosts before retrying the failed command. If the error message appears when no other DiskSuite commands are running, kill and restart `rpc.metad` on all the hosts in the diskset. Make sure `rpc.metad` is running on all the hosts before trying the command again.

```
set name contains invalid characters
```

An attempt was made to use illegal characters to name a diskset.

```
set name is in-use or invalid on host nodename
```

The diskset name selected is already in use on host *nodename* or contains characters not considered valid in a diskset name.

```
set name is too long
```

An attempt was made to create a diskset using more characters in the diskset name than DiskSuite will accept.

```
set unlock failed - bad key
```

The diskset is locked and the user does not have the key. Either another DiskSuite command is running and has locked the diskset or a DiskSuite command has aborted without unlocking the diskset on one of the hosts in the diskset. Check to see if there are other DiskSuite commands running on the hosts in the diskset. Check all the hosts and allow other commands to complete before retrying the failed command. If the error message appears when no other DiskSuite commands are running, kill and restart `rpc.metad` on all the hosts in the diskset. Make sure `rpc.metad` is running on all the hosts before trying the command again.

```
side information missing for host nodename
```

The diskset is incomplete. Kill `rpc.metad` on all hosts and then retry the operation.

```
slice 7 is not usable as a metadevice component
```

An attempt was made to use Slice 7 in a shared metadevice or shared hot spare pool. Slice 7 is reserved for database replicas only.

```
submirror too small to attach
```

The metadevice passed to `metattach` is smaller than the metamirror to which it is being attached.

```
stale databases
```

The user attempted to modify the configuration of a metadevice when at least half the metadevice state database replicas were not accessible.

```
syntax error
```

An invalid metadevice configuration entry was provided to `metainit` from the command line or via the `md.tab` file.

```
target metadevice is not able to be renamed
```

An attempt was made to switch metadevices that do not have a child-parent relationship. For example, you cannot rename a trans metadevice with a stripe that is part of a mirrored master device for the trans.

```
there are no existing databases
```

To create any metadevices or hot spare pools, database replicas must exist. See `metadb(1M)` for information on the creation of database replicas.

```
unable to delete set, it still has drives
```

An attempt was made to delete the last remaining host from a diskset while drives still exist in the diskset.

```
unit already set up
```

The user requested that a metadevice d*num* be initialized when d*num* is already set up.

```
unit is not a concat/stripe
```

An attempt was made to perform a concat/stripe specific operation on a metadevice that is not a concat/stripe.

```
unit is not a mirror
```

An attempt was made to perform a mirror specific operation on a metadevice that is not a mirror.

```
unit is not a RAID
```

An attempt was made to perform a RAID specific operation on a metadevice that is not a RAID device.

```
unit is not a trans
```

An attempt was made to perform a metatrans specific operation on a metadevice that is not a metatrans device.

```
unit not found
```

An attempt was made to perform an operation on a non-existent metadevice.

```
unit not set up
```

An attempt was made to perform an operation on a non-existent metadevice.

```
waiting on /tmp/.mdlock
```

Some other metadevice utility is currently in progress and the lock cannot be accessed at this time. DiskSuite utilities are serialized using the `/tmp/.mdlock` file as a lock. If you determine that there are no other utilities currently running, you may want to remove this lock file.

Log Messages

The command line log messages displayed by DiskSuite are listed in alphabetical order below. Each message is always preceded with "`md:`" The variables in these messages indicate the following:

- *dev* is a device name.
- d*num* is a metadevice name.
- *num* is a number.
- *state* is a metatrans device state.
- *trans* is either "logging" or "master."

Note - When the initial portion of a message begins with a variable, the message is alphabetized by the first word following the variable.

Notice Log Messages

```
Could not load misc /dev
```

The named `misc` module is not loadable. It is possibly missing, or something else has been copied over it.

```
db: Parsing error on 'dev'
```

The set command in `/etc/system` for the `mddb.bootlist`<*number*> is not in the correct format. Run `metadb –p` to place the correct set commands into the `/etc/system` file.

```
dnum: Hotspared device dev with dev
```

The first device name listed has been hot spare replaced with the second device name listed.

```
dnum: Hotspared device dev(num, num) with dev(num, num)
```

The first device number listed has been hot spare replaced with the second device number listed.

```
dnum: no mem for property dev
```

Memory could not be allocated in the `prop_op` entry point.

Warning Log Messages

```
dnum: Cannot load dev driver
```

The underlying named driver module is not loadable (for example, `sd`, `id`, `xy`, or a third-party driver). This could indicate that the driver module has been removed.

```
Open error of hotspare dev
Open error of hotspare dev(num, num)
```

The named hot spare is not openable, or the underlying driver is not loadable.

```
dnum: read error on dev
dnum: write error on dev
```

A read or write error has occurred on the specified metadevice at the specified device name. This happens if any read or write errors occur on a metadevice.

```
dnum: read error on dev(num, num)
dnum: write error on dev(num, num)
```

A read or write error has occurred on the specified metadevice at the specified device number. This happens if any read or write errors occur on a metadevice.

```
dnum: read error on dnum
dnum: write error on dnum
```

A read or write error has occurred on the specified metadevice at the specified device number. This happens if any read or write errors occur on a metadevice.

```
State database commit failed
State database delete failed
```

These messages occur when there have been device errors on components where the state database replicas reside. These errors only occur when more than half of the replicas have had errors returned to them. For example, if you have three components with state database replicas and two of the components report errors, than these errors may occur. The state database commit or delete is retried periodically. If the replica is added, the commit or delete will finish and the system will be operational. Otherwise, the system will time out and panic.

```
State database is stale
```

This message occurs when there are not enough usable replicas for the state database to be able to update records in the database. All accesses to the metadevice driver will fail. To fix this problem, add more replicas or delete unaccessible replicas.

```
trans device: read error on dnum
trans device: write error on dnum
```

A read or write error has occurred on the specified logging or master device at the specified metadevice. This happens if any read or write errors occur on a logging or master device.

```
trans device: read error on dev
trans device: write error on dev
```

A read or write error has occurred on the specified logging or master device at the specified device name. This happens if any read or write errors occur on a logging or master device.

```
trans device: read error on dev(num, num)
trans device: write error on dev(num, num)
```

A read or write error has occurred on the specified logging or master device at the specified device number. This happens if any read or write errors occur on a logging or master device.

```
logging device: dnum changed state to state
logging device: dev changed state to state
logging device: dev(num, num) changed state to state
```

The logging device and its associated master device(s) have changed to the specified state(s).

Panic Log Messages

```
State database problem
```

A failed metadevice state database commit or deletion has been retried the default 100 times.

```
dnum: Unknown close type
dnum: Unknown open type
```

A metadevice is being opened/closed with an unknown open type (OTYP).

APPENDIX **B**

Upgrading to Other Solaris Versions

Introduction

Upgrading to later versions of the Solaris environment while using metadevices requires steps not currently outlined in the Solaris documentation. The current Solaris upgrade procedure is incompatible with DiskSuite. The following supplemental procedure is provided as an alternative to completely reinstalling the Solaris and DiskSuite packages.

Note - You must have the media to upgrade Solaris (and DiskSuite if necessary).

Upgrading Solaris With Solstice DiskSuite

Caution - Before you begin this procedure, back up all file systems. See the `ufsdump(1M)` man page for details.

How to Upgrade Solaris With Solstice DiskSuite

1. Repair any mirrors that have errors.

2. **Save** `/etc/vfstab` **for later use.**

3. **Clear any trans metadevices that may be used during the Solaris upgrade (for example,** `/usr`**,** `/var`**, and** `/opt`**).**

   See *Solstice DiskSuite 4.2.1 User's Guide* for information on clearing (removing logging from) trans metadevices. If you are uncertain which trans metadevices should be cleared, clear all trans metadevices.

4. **Comment out file systems in** `/etc/vfstab` **mounted on metadevices that are not simple metadevices or simple mirrors.**

   A simple metadevice is composed of a single component with a `Start Block` of 0. A simple mirror is composed of submirrors, all of which are simple metadevices.

5. **Convert the remaining (simple) mirrors to one-way mirrors with the** `metadetach` **command.**

   Upgrade will be performed on a single submirror of each mirror. The other submirrors will be synced up with `metattach` after the upgrade.

6. **If root (/) is mounted on a metadevice or mirror, set the root (/) file system to be mounted on the underlying component of the metadevice or the underlying component of the remaining attached submirror.**

   Use the `metaroot` command to do this safely.

7. **Edit the** `/etc/vfstab` **file to change any file systems or** `swap` **devices still mounted on metadevices or mirrors after Step 3 on page 178.**

   Mount the file systems on the underlying component of the metadevices or the underlying component of the remaining attached submirrors.

8. **Remove symbolic links to the DiskSuite startup files so that it is no longer initialized at boot time.**

```
demo# rm /etc/rcS.d/S35lvm.init /etc/rc2.d/S95lvm.sync
```

   These links will be added back later by reinstalling DiskSuite after the Solaris upgrade.

9. **Halt the machine and upgrade Solaris, then reboot the machine.**

10. **Reinstall DiskSuite, then reboot the machine.**

    This will re-establish the symbolic links removed in Step 8 on page 178.

Note - Make certain that the version of Solaris you are installing is compatible with Solstice DiskSuite 4.1.1.1.

11. **If root (/) was originally mounted on a metadevice or mirror, set the root (/) file system to be mounted back on the original metadevice or mirror.**

    Use the `metaroot` command to do this safely.

12. **Edit the `/etc/vfstab` file to change any file systems or `swap` devices edited in Step 7 on page 178 to be mounted back on their original metadevice or mirror.**

13. **Edit the `/etc/vfstab` file to uncomment the file systems commented out in Step 4 on page 178.**

14. **Reboot the machine to remount the file systems.**

15. **Use the `metattach` command to reattach and resync any submirrors broken off in Step 5 on page 178.**

16. **Recreate the cleared trans metadevices. See *Solstice DiskSuite 4.2.1 User's Guide* for information on creating trans metadevices.**

Glossary

attach logging device — To add a logging device to an existing trans metadevice. If the trans metadevice is mounted, DiskSuite attaches the log when the file system is unmounted or the system is rebooted.

attach submirror — To add a submirror to an existing mirror. DiskSuite automatically resyncs the submirror with other submirrors.

block — A unit of data that can be transferred by a device, usually 512 bytes long.

boot — To start a computer program that clears memory, loads the operating system, and otherwise prepares the computer.

browser — In DiskSuite Tool, a window for browsing through DiskSuite objects in list form. There is a separate browser for slices, metadevices, and hot spare pools.

byte — A group of adjacent binary digits (bits) operated on by the computer as a unit. The most common size byte contains eight binary digits.

canvas — In DiskSuite Tool, the main region where DiskSuite objects are displayed and manipulated.

collapse — A DiskSuite Tool command that decreases (minimizes) the size of DiskSuite objects, as shown on the canvas.

commit — A DiskSuite Tool command that commits changes that have been made to DiskSuite objects. The changes are stored in the `md.cf` file.

concatenated metadevice — See *concatenation*.

concatenated stripe — A metadevice made of concatenated groups of striped slices.

concatenation In its simplest meaning, concatenation refers to the combining of two or more data sequences to form a single data sequence. In DiskSuite:

(1) Another word for *concatenated metadevice*.

(2) Creating a single logical device (metadevice) by sequentially distributing disk addresses across disk slices.

The sequential (serial) distribution of disk addresses distinguishes a concatenated metadevice from a striped metadevice.

configuration The complete set of hardware and software that makes up a storage system. Typically, a configuration will contain disk controller hardware, disks (divided into slices), and the software to manage the flow of data to and from the disks.

configuration log A history (log) kept by DiskSuite Tool of all top-level operations and input-validation errors during a session.

controller Electronic circuitry that acts as a mediator between the CPU and the disk drive, interpreting the CPU's requests and controlling the disk drive.

cylinder In a disk drive, the set of tracks with the same nominal distance from the axis about which the disk rotates. See also *sector*.

detach logging device To remove a logging device from a trans metadevice.

detach submirror To remove a submirror's logical association from a mirror.

Disk View window In DiskSuite Tool, a graphical view of the physical devices attached to the system. It can be used to show the relationship between the logical and physical devices.

diskset A set of disk drives containing logical devices (metadevices) and hot spares that can be shared exclusively (but not concurrently) by two hosts. Used in host fail-over solutions.

DiskSuite objects In DiskSuite Tool, a graphical representation for the state database, metadevice or part of a metadevice, or hot spare pool.

driver Software that translates commands between the CPU and the disk hardware.

drop site In DiskSuite Tool, the region of the Disk View window where any metadevice, group of metadevices, or physical device can be

dragged and dropped. The physical layout of the device mappings is displayed after the metadevice is dropped on a specific color in the drop site.

encapsulate To put an existing file system into a one-way concatenation. A one-way concatenation consists of a single slice.

Evaluate A DiskSuite Tool command that displays errors and warning messages in the configuration log for the selected metadevice.

Expand A DiskSuite Tool command that increases (magnifies) the view of DiskSuite objects.

fault tolerance A computer system's ability to handle hardware failures without interrupting system performance or data availability.

formatting Preparing a disk to receive data. Formatting software organizes a disk into logical units, like blocks, sectors, and tracks.

full mirror resync See *resyncing*.

Gbyte (*Gigabyte*), 1024 Mbytes (or 1,073,741,824 bytes).

head In a magnetic disk drive, an electromagnet that stores and reads data to and from the platter. Controlled by a disk *controller*.

high-availability A term describing systems that can suffer one or more hardware failures and rapidly make data access available.

hot spare A slice reserved to substitute automatically for a failed slice in a submirror or RAID5 metadevice. A hot spare must be a physical slice, not a metadevice.

hot spare pool A group of hot spares. A hot spare pool is associated with submirrors or RAID5 metadevices.

icon well In DiskSuite Tool, the region containing icons that are the source for new DiskSuite objects. Icons are used as templates to create metadevices and hot spare pools. See also *templates*.

interlace (1) To distribute data in non-contiguous logical data units across disk slices.

(2) A value: the size of the logical data segments in a striped metadevice or RAID5 metadevice.

interleave See *interlace*.

Kbyte (*Kilobyte*), 1024 bytes.

latency The time it takes for a disk drive's platter to come around to a specific location for the read / write head. Usually measured in milliseconds. Latency does not include the time it takes for the read / write head to position itself (head seek time).

local diskset A diskset that is not in a shared diskset and that belongs to a specific host. The local diskset contains the metadevice state database for that specific host's configuration. Each host in a diskset must have a local diskset to store its own local metadevice configuration.

logical An abstraction of something real. A logical disk, for example, can be an abstraction of a large disk that is really made of several small disks.

logging Recording UFS updates in a log (the logging device) before the updates are applied to the UNIX file system (the master device).

logging device The slice or metadevice that contains the log for a trans metadevice.

master device The slice or metadevice that contains an existing or newly created UFS file system for a trans metadevice.

Mbyte (*Megabyte*), 1024 Kbytes.

`md.cf` A backup file of the DiskSuite configuration which can be used for disaster recovery. This file should not be edited or removed. It should be backed up on a regular basis.

`md.conf` A configuration file used by DiskSuite while loading. It can be edited to increase the number of metadevices and disksets supported by the metadisk driver.

`mddb.cf` A file to track the locations of state database replicas. This file should not be edited or removed.

`md.tab` An input file that you can use with the command line interface utilities `metainit(1M)`, `metadb(1M)`, and `metahs(1M)` to administer metadevices and hot spare pools.

MetaDB object — The graphical object in DiskSuite Tool that represents the metadevice state database. The MetaDB object administers the metadevice state database and its copies (the state database replicas).

metadevice — A group of physical slices accessed as a single logical device by concatenation, striping, mirroring, setting up RAID5 metadevices, or logging physical devices. After they are created, metadevices are used like slices.

The metadevice maps logical block addresses to the correct location on one of the physical devices. The type of mapping depends on the configuration of the particular metadevice.

Also known as pseudo, or virtual device in standard UNIX terms.

Metadevice Editor window — The main window for DiskSuite Tool. It provides a view of metadevices and hot spare pools in which you can graphically create, display, or edit your configuration.

metadevice state database — A database, stored on disk, that records configuration and state of all metadevices and error conditions. This information is important to the correct operation of DiskSuite and it is replicated. See also *state database replica*.

metadisk driver — A UNIX pseudo device driver that controls access to metadevices, enabling them to be used like physical disk slices. The metadisk driver operates between the file system and application interfaces and the device driver interface. It interprets information from both the UFS or applications and the physical device drivers.

mirror — A metadevice made of one or more other metadevices called submirrors. It replicates data by maintaining multiple copies.

mirroring — Writing data to two or more disk drives at the same time. In DiskSuite, mirrors are logical storage objects that copy their data to other logical storage objects called submirrors.

multi-way mirror — A mirror that has at least two submirrors.

Objects list — In DiskSuite Tool, a pseudo-browser in the Metadevice Editor window that displays metadevices, hot spares, and configuration problems.

one-way mirror — A mirror that consists of only one submirror. You create a one-way submirror, for example, when mirroring slices that contain existing data. A second submirror is then attached.

online backup A backup taken from a mirror without unmounting the entire mirror or halting the system. Only one of the mirror's submirrors is taken offline to complete the backup.

optimized mirror resync A resync of only the submirror regions that are out of sync at a system reboot. The metadisk driver tracks submirror regions and can determine which submirror regions are out of sync after a failure. See *resyncing*.

Panner In DiskSuite Tool, the region where a miniature view of the canvas shows small representations of the DiskSuite objects currently displayed on the canvas.

parity A way for RAID5 configurations to provide data redundancy. Typically, a RAID5 configuration stores data blocks and parity blocks. In the case of a missing data block, the missing data can be regenerated using the other data blocks and the parity block.

partial mirror resync A resync of only a replacement part of a submirror or RAID5 metadevice, rather than the entire submirror or RAID5 metadevice. See *full mirror resync* and *optimized mirror resync*.

partition See *slice*.

On a SPARC system, a slice and partition are the same.

On an x86 system, a slice and partition are distinct. A partition is a part of a disk set aside for use by a particular operating system using the `fdisk` program. Thus partitioning the disk enables it to be shared by several different operating systems. Within a Solaris partition, you can create normal Solaris slices.

platter The spinning disk that stores data inside a disk drive.

Put Away A DiskSuite Tool command that returns DiskSuite objects on the Metadevice Editor window canvas to the Objects list.

RAID *Redundant Array of Inexpensive Disks.* A classification of different ways to back up and store data on multiple disk drives. There are seven levels of RAID:

Level 0: Nonredundant disk array (striping)

Level 1: Mirrored disk array

Level 2: Memory-style Error Code Correction (ECC)

Level 3: Bit-Interleaved Parity

Level 4: Block-Interleaved Parity

Level 5: Block-Interleaved Distributed-Parity

Level 6: P + Q Redundancy

DiskSuite implements RAID levels 0, 1, and 5.

resync region A division of a mirror that enables tracking changes by submirror regions rather than over the entire mirror. Dividing the mirror into resync regions can reduce resync time.

resyncing The process of preserving identical data on mirrors or RAID5 metadevices.

Mirrors are resynced by copying data from one submirror to another after submirror failures, system crashes, or after adding a new submirror.

RAID5 metadevices are resynced during reboot if any operations that may have been halted from a system panic, a system reboot, or a failure to complete are restarted.

SCSI Small Computer Systems Interface. An interface standard for peripheral devices and computers to communicate with each other.

sector The smallest divisions of a disk platter's tracks. Usually 512 bytes. See *block*.

seek time The time it takes for a disk drive's read / write head to find a specific track on the disk platter. Seek time does not include *latency* nor the time it takes for the controller to send signals to the read / write head.

shared diskset See *diskset*.

simple metadevice A term usually reserved for a concatenated metadevice, striped metadevice, or concatenated stripe metadevice.

slice A part of each physical disk that is treated as a separate area for storage of files in a single file system, or for an application such as a database. Before you can create a file system on disk, you must partition it into slices.

Slice Filter window In DiskSuite Tool, a menu available from the Disk View window and the Slice Browser that filters the slices to view those available to be parts of metadevices, hot spares, state database replicas, and trans metadevice logs.

state database replica — A copy of the metadevice state database. Keeping copies of the metadevice state database protects against the loss of state and configuration information critical to metadevice operations.

stripe — (1) A metadevice created by striping (also called a *striped metadevice*).

(2) An interlaced slice that is part of a striped metadevice.

(3) To create striped metadevices by interlacing data across slices.

striping — Creating a single logical device (metadevice) by transparently distributing logical data segments across slices. The logical data segments are called stripes.

Striping is sometimes called interlacing because the logical data segments are distributed by interleaving them across slices.

Striping is generally used to gain performance, enabling multiple controllers to access data at the same time.

Compare striping with concatenation, where data is mapped sequentially on slices.

submirror — A metadevice that is part of a mirror. See also *mirror*.

`system` **(**`/etc/system`**)** — A file used to set system specifications. DiskSuite uses this file, for example, when mirroring the root (`/`) file system.

templates — In DiskSuite Tool, the template icons create new, empty metadevices. The new metadevices cannot be used until they are "populated" with their necessary parts. Templates can also be combined to build additional metadevices.

Tbyte — (*Terabyte*), 1,024 Gbytes, or 1 trillion bytes (1,099,511,627,776 bytes).

three-way mirror — A mirror made of three submirrors. This configuration enables a system to tolerate a double-submirror failure. You can also do online backups with the third submirror.

trans metadevice — A metadevice for UFS logging. A trans metadevice includes one or more other metadevices or slices: a master device, containing a UFS file system, and a logging device. After they are created, trans metadevices are used like slices.

two-way mirror — A mirror made of two submirrors. This configuration enables a system to tolerate a single-submirror failure.

UFS UNIX file system.

UFS logging The process of recording UFS updates in a log (the logging device) before the updates are applied to the UNIX file system (the master device).

Index

E

F

G

H

I

K

L

M

T

U

V

W