Solstice DiskSuite 4.2 User's Guide

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

Part No: 805-5961–10
October 19, 1998

Contents

Tables

Figures

Preface

Solstice™ DiskSuite™ 4.2 is a software product that manages data and disk drives.

Solstice DiskSuite 4.2 runs on all SPARC™ systems running Solaris™ 2.6 or Solaris 7, and on all x86 systems running Solaris 2.6 or Solaris 7.

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

About This Book

Solstice DiskSuite 4.2 User's Guide replaces these two books previously released with DiskSuite:

- *Solstice DiskSuite 4.0 Administration Guide*
- *Solstice DiskSuite Tool 4.0 User's Guide*

Who Should Use This Book

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

How This Book Is Organized

This book is organized by groups of similar high-level tasks rather than by product features.

provides a quick way to begin using DiskSuite and provides an easy access to commonly performed tasks.

The remainder of this manual is organized as follows:

Chapter 1 describes how to get a new DiskSuite configuration up and running.

Chapter 2 describes how to create DiskSuite objects.

Chapter 3 describes maintenance-related tasks for DiskSuite objects, such as checking status and replacing an errored slice.

Chapter 4 describes how to change parameters on DiskSuite objects.

Chapter 5 describes how to delete DiskSuite objects from the system.

Chapter 6 describes how to manage SPARCstorage Arrays with DiskSuite Tool, get performance statistics, and how to integrate DiskSuite with other Solstice products.

Chapter 7 describes various types of problems that a DiskSuite administrator may encounter and how to solve those problems.

Chapter 8 provides some less obvious ways to put DiskSuite to work to maximize its potential.

Appendix A describes how to use the File System Manager and Disk Manager graphical tools to perform tasks such as creating a file system and partitioning a disk.

Related Books

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

- *Solstice AdminSuite 2.2 Administration Guide*
- *Solstice DiskSuite 4.2 Reference Guide*
- *System Administration Guide, Volume I*
- *System Administration Guide, Volume II*
- *SPARCstorage Array User's Guide* and *SPARCstorage Array Configuration Guide*

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	`#`

Finding Solstice DiskSuite 4.2 Information

Solstice DiskSuite 4.2 User's Guide describes how to set up and maintain systems using Solstice DiskSuite 4.2.

If you want to begin using the DiskSuite product right away, use the information presented in this chapter. Organized as a "roadmap," this chapter serves as a high-level guide to find information for certain DiskSuite tasks, such as setting up storage capacity. It does not address all the tasks that you will need to use DiskSuite. Instead, it provides an easy way to find procedures describing how to perform common tasks associated with the following DiskSuite concepts:

- Storage Capacity
- Availability
- I/O Performance
- Administration
- Troubleshooting

This chapter also provides a complete listing of tasks organized by feature. See "DiskSuite Task Summary" on page xxxiv.

Caution - If you do not use DiskSuite correctly, you can destroy data. DiskSuite provides a powerful way to manage your disks and data on them. As a minimum safety precaution, you should make sure you understand how DiskSuite works before attempting to use it.

TABLE P–1 DiskSuite Roadmap-Storage Capacity

Task	Description	For Instructions, Go To
Set Up Storage Capacity	You can create storage capacity that spans slices by creating a striped metadevice or a RAID5 metadevice. The stripe or RAID5 metadevice can then be used for a file system or any application, such as a database, that accesses the raw device.	"How to Create a Striped Metadevice (DiskSuite Tool)" on page 21 "How to Create a RAID5 Metadevice (DiskSuite Tool)" on page 48
Expand an Existing File System	Increase the capacity of an existing file system by creating a concatenation then adding additional slices.	"How to Expand a Slice Containing Existing Data (DiskSuite Tool)" on page 125
Expand an Existing Concatenation/Stripe	Use DiskSuite Tool to create a concatenated stripe to expand an existing concatenation or stripe.	"How to Expand an Existing Concat/Stripe (DiskSuite Tool)" on page 128
Expand a RAID5 Metadevice	If you need to expand the capacity of a RAID5 metadevice, you can concatenate additional slices to it.	"How to Expand a RAID5 Metadevice (DiskSuite Tool)" on page 133
Increase the Size of UFS	The `growfs(1M)` command expands the size of a UFS while it is mounted and without disrupting access to the data.	"How to Grow a File System (Command Line)" on page 138

TABLE P–1 DiskSuite Roadmap-Storage Capacity *(continued)*

Task	Description	For Instructions, Go To
Create a File System	You can create a file system on a stripe, concatenation, mirror, RAID5 metadevice, or trans metadevice.	"How to Create a File System on a Metadevice (File System Manager)" on page 73

TABLE P–2 DiskSuite Roadmap-Availability

Task	Description	For Instructions, Go To
Maximize Data Availability	If you want maximum availability of your data, use DiskSuite's mirroring feature to maintain multiple copies of your data. You can create a mirror from unused slices in preparation for data, or mirror an existing file system, including root (/) and /usr.	"How to Create a Mirror From Unused Slices (DiskSuite Tool)" on page 28 "How to Create a Mirror From a File System That Can Be Unmounted (DiskSuite Tool)" on page 31 "How to Create a Mirror From a File System That Cannot Be Unmounted (DiskSuite Tool)" on page 36
Add Data Availability With Minimum Hardware Cost	To increase data availability with a minimum of hardware, use DiskSuite's RAID5 Metadevice feature.	"How to Create a RAID5 Metadevice (DiskSuite Tool)" on page 48
Add Increased Data Availability to an Existing Mirror or RAID5 Metadevice	To increase data availability for a mirror or a RAID5 metadevice, create a hot spare pool then associate it with a mirror's submirrors, or a RAID5 metadevice.	"How to Create a Hot Spare Pool (DiskSuite Tool)" on page 60

TABLE P–2 DiskSuite Roadmap-Availability *(continued)*

Task	Description	For Instructions, Go To
Increase File System Availability After Reboot	To increase overall file system availability after reboot, add UFS logging (trans metadevice) to the system. Logging a file system reduces the amount of time `fsck(1M)` has to run when the system reboots.	"How to Create a Trans Metadevice for a File System That Can Be Unmounted (DiskSuite Tool)" on page 52 "How to Create a Trans Metadevice for a File System That Cannot Be Unmounted (DiskSuite Tool)" on page 55

TABLE P–3 DiskSuite Roadmap-I/O Performance

Task	Description	For Instructions, Go To
Increase Mirror Performance	Adding additional state database replicas before creating the mirror helps improve a mirror's performance.	"How to Create Additional State Database Replicas (DiskSuite Tool)" on page 18
Tune Mirror Read and Write Policies	The read and write policies for a mirror can be specified to improve performance for a given configuration.	"How to Change a Mirror's Options (DiskSuite Tool)" on page 164
Optimize Device Performance	Creating stripes optimizes performance of devices that make up the stripe. The stripe's interlace value can be optimized for random or sequential access.	"How to Create a Striped Metadevice (DiskSuite Tool)" on page 21

TABLE P–3 DiskSuite Roadmap-I/O Performance *(continued)*

Task	Description	For Instructions, Go To
Maintain Device Performance Within an Existing Stripe	A concatenated stripe expands a stripe or concatenation that has run out of space. A concatenation of stripes is better for performance than a concatenation of slices.	"How to Expand a Slice Containing Existing Data (Command Line)" on page 127
Improve System Performance	UFS logging (trans metadevice) helps performance by decreasing the number of synchronous disk writes.	"How to Create a Trans Metadevice for a File System That Can Be Unmounted (DiskSuite Tool)" on page 52 "How to Create a Trans Metadevice for a File System That Cannot Be Unmounted (DiskSuite Tool)" on page 55

TABLE P–4 DiskSuite Roadmap-Administration

Task	Description	For Instructions, Go To
Simplify Administration of Large Configurations	The DiskSuite Tool graphical interface makes working with many disks quick and easy. It supports drag-and-drop operations and provides physical and logical views of the system.	*Solstice DiskSuite 4.2 Reference Guide*, Chapter 4, "DiskSuite Tool"
Graphically Administer Slices/File Systems	DiskSuite is integrated with the Solstice Storage Manager graphical user interface. Use it to administer your disks and file systems, performing such tasks as partitioning disks and constructing UFS file systems.	Appendix A

TABLE P–4 DiskSuite Roadmap-Administration *(continued)*

Task	Description	For Instructions, Go To
Administer SPARCstorage Arrays	The DiskSuite Tool graphical interface enables you to perform a number of maintenance tasks on SPARCstorage Arrays, such as starting and stopping trays of disks, and working with NVRAM.	Chapter 6
Reconfigure Metadevices	Administering metadevices is made easier through the `metarename(1M)` command.	"How to Rename a Metadevice (Command Line)" on page 140
Optimize Solstice DiskSuite 4.2	DiskSuite performance is dependent on a well-designed configuration. Once created, the configuration needs monitoring and tuning.	"How to Graph Device Statistics (DiskSuite Tool)" on page 211
Plan for Future Expansion	Because file systems tend to run out of space, you can plan for future growth by putting a file system into a concatenation.	"How to Expand a Slice Containing Existing Data (DiskSuite Tool)" on page 125
Automate DiskSuite Monitoring	Use DiskSuite's SNMP features to integrate alerts with SunNet Manager.	"How to Enable SunNet Manager to Launch DiskSuite Tool (SunNet Manager)" on page 213 "How to Configure DiskSuite SNMP Support (Command Line)" on page 215

TABLE P–5 DiskSuite Roadmap-Troubleshooting

Task	Description	For Instructions, Go To
Replace a Failed Slice	The situation could arise when a failing slice in a metadevice needs replacing. In the case of stripes and concatenation, you have to use a new slice, delete and recreate the metadevice, then restore data from a backup. Slices in mirrors and RAID5 metadevices might be able to be replaced and resynced without loss of data.	"How to Recreate a Stripe or Concatenation After Slice Failure (DiskSuite Tool)" on page 102 "How to Enable a Slice in a Submirror (DiskSuite Tool)" on page 109 "How to Enable a Slice in a RAID5 Metadevice (DiskSuite Tool)" on page 113
Recover From Boot Problems	Special problems can arise when booting the system, due to a hardware problem or operator error.	"How to Recover From Improper `/etc/vfstab` Entries (Command Line)" on page 228 "How to Recover From Insufficient State Database Replicas (Command Line)" on page 231 "How to Recover From a Boot Device Failure (Command Line)" on page 234
Work With an SSA Disk Problem	For the most part, using DiskSuite with a SPARCstorage Array is transparent. Some procedures, such as resolving disk problems, have slightly different steps.	"How to Replace a Failed SPARCstorage Array Disk in a Mirror (DiskSuite Tool)" on page 246

TABLE P–5 DiskSuite Roadmap-Troubleshooting *(continued)*

Task	Description	For Instructions, Go To
Work With Trans Metadevice Problems	Problems with trans metadevices can occur on either the master or logging device, and they can either be caused by errored data or device problems. All trans metadevices sharing the same logging device must be fixed before they return to a usable state.	"How to Recover a Trans Metadevice With a File System Panic (Command Line)" on page 122 "How to Recover a Trans Metadevice With Hard Errors (Command Line)" on page 122

DiskSuite Task Summary

The information in this section, organized by DiskSuite feature, serves as a quick reference for all DiskSuite tasks. Keyboard accelerators (if available) and the command line equivalents are given for each task.

TABLE P–6 DiskSuite Feature/Task List

Feature/Task	DiskSuite Tool Menu or Action	Keyboard Command
Concatenations		
Checking status	Object -> Info	`metastat(1M)`
Creating	Edit -> Create -> Concat/Stripe	`metainit(1M)`
Expanding	Drag an unused slice to the object	`metattach(1M)`
Recreating after slice failure	Delete and recreate metadevice	`metaclear(1M)`, `metainit(1M)`
Removing	Edit -> Delete	`metaclear(1M)`

TABLE P–6 DiskSuite Feature / Task List *(continued)*

Feature/Task	DiskSuite Tool Menu or Action	Keyboard Command
Configurations		
Renaming a metadevice	Use the Info window	`metarename(1M)`
Restoring uncommitted configuration	File -> Restore From File	
Reversing all uncommitted operations	Edit -> Undo All	
Reversing an uncommitted operation	Edit -> Undo Last	
Saving uncommitted configuration	File -> Save To File	
DiskSuite Tool		
Collapsing an object	Object -> Collapse	
Committing an object	Object -> Commit	`metainit(1M)`
Configuration Log, viewing	Browse -> Configuration Log	
Deleting an object	Edit -> Delete	`metaclear(1M)`
Disks, viewing	Browse -> Disk View	
Displaying entire object	Object -> Expand	
Duplicating an object	Edit -> Duplicate	
Evaluating an object	Object -> Evaluate	
Exiting	File -> Exit	
Finding metadevices	Browse -> Find	
Metadevices, viewing	Browse ->Metadevices	
Object information	Object -> Info	`metastat(1M)`
Problems, viewing	Browse -> Problem List	
Putting away an object	Object -> Put Away	
Reorganizing objects on the canvas	Edit -> Cleanup Canvas	

TABLE P–6 DiskSuite Feature / Task List *(continued)*

Feature/Task	DiskSuite Tool Menu or Action	Keyboard Command
Slices, viewing	Browse -> Slices	`prtvtoc(1M)`, `format(1M)`
Hot Spares		
Adding a slice to a hot spare pool	Drag an available slice to the hot spare pool object	`metahs(1M)`
Associating a hot spare pool	Drag a hot spare pool object onto submirror or RAID5 object	`metaparam(1M)`
Changing the associated hot spare pool	Drag a replacement hot spare pool object onto submirror or RAID5 object	`metaparam(1M)`
Checking status	Object -> Info	`metahs(1M)`
Creating hot spare pool	Edit -> Create -> Spare Pool	`metainit(1M)`
Enabling a hot spare	Use the Hot Spare Pool Info window	`metahs(1M)`
Removing a hot spare	Use the Hot Spare Pool Info Window	`metahs(1M)`
Removing a hot spare pool	Edit -> Delete	`metaparam(1M)`, `metahs(1M)`
Replacing a component	Drag a replacement slice to the Hot Spare Pool object	`metahs(1M)`
Viewing a hot spare pool	Browse -> Hot Spare Pool	`metastat(1M)`
Mirrors		
Attaching a submirror	Drag a submirror to a Mirror object	`metattach(1M)`
Changing options	Use the Mirror Info window	`metaparam(1M)`
Checking status	Object -> Info	`metastat(1M)`
Creating	Edit -> Create -> Mirror	`metainit(1M)`

TABLE P–6 DiskSuite Feature / Task List *(continued)*

Feature/Task	DiskSuite Tool Menu or Action	Keyboard Command
Detaching a submirror	Use the Mirror Info window	`metadetach(1M)`
Expanding	Drag unused slices to the submirrors	`metattach(1M)`, `growfs(1M)`
Placing a submirror offline / online	Use the Mirror Info window	`metaonline(1M)`, `metaoffline(1M)`
Replacing failed components	Drag a replacement slice to the errored slice	`metareplace(1M)`
Removing	Edit -> Delete	`metadetach(1M)`, `metaclear(1M)`
Unmirroring a File System		
Performance Monitoring		
Displaying device statistics	Object -> Statistics	`iostat(1M)`
Displaying graphs	Browse -> Statistics Graphs	`iostat(1M)`
RAID5 Metadevices		
Checking Status	Object -> Info	`metastat(1M)`
Creating	Edit -> Create -> RAID	`metainit(1M)`
Expanding	Drag an unused slice to the RAID5 object	`metattach(1M)`
Replacing failed components	Drag a replacement slice to the errored slice	`metareplace(1M)`
Removing	Edit -> Delete	`metaclear(1M)`
SPARCstorage Arrays		
Checking status of fan and battery	Disk View -> Object -> Info	`ssaadm(1M)`

TABLE P–6 DiskSuite Feature / Task List *(continued)*

Feature/Task	DiskSuite Tool Menu or Action	Keyboard Command
Displaying controller information	Disk View -> Object -> Info	`ssaadm(1M)`
Disabling NVRAM	Disk View -> Object -> Fast Write Disable	`ssaadm(1M)`
Enabling NVRAM	Disk View -> Object -> Fast Write Enable	`ssaadm(1M)`
Enabling NVRAM (synchronous writes)	Disk View -> Object -> Fast Write Synchronous	`ssaadm(1M)`
Flushing outstanding writes from NVRAM	Disk View -> Object -> Sync NVRAM	`ssaadm(1M)`
Purging fast write data from NVRAM	Select the object, display its pop-up menu and choose Purge NVRAM	`ssaadm(1M)`
Reserving a Disk	Disk View -> Object -> Reserve Disks	`ssaadm(1M)`
Releasing a Disk	Disk View -> Object -> Release Disks	`ssaadm(1M)`
Starting a disk /tray/controller	Disk View -> Object -> Start Disks	`ssaadm(1M)`
Stopping a disk/tray/controller	Disk View -> Object -> Stop Disks	`ssaadm(1M)`
State Database Replicas		
Adding more	Drag slices to the initialized Metadevice State Database object	`metadb(1M)`
Attaching	Use the MetaDB Info window	`metadb(1M)`
Checking status	Object -> Info	`metadb(1M)`
Creating initial	Drag slices to the uninitialized Metadevice State Database object	`metadb(1M)`
Removing	Display the Metadevice State Database Information window and choose Remove.	`metadb(1M)`

TABLE P–6 DiskSuite Feature / Task List *(continued)*

Feature/Task	DiskSuite Tool Menu or Action	Keyboard Command
Restoring	Display the Metadevice State Database Information window and choose Restore.	
Viewing and modifying	Use the Metadevice State Database Info window	
Stripes		
Checking status	Object -> Info	`metastat(1M)`
Creating	Edit -> Create -> Concat / Stripe	`metainit(1M)`
Expanding	Drag an unused slice to the object	`metattach(1M)`
Recreating after slice failure	Delete and recreate metadevice	`metaclear(1M)`, `metainit(1M)`
Removing	Edit -> Delete	`metaclear(1M)`
Trans Metadevices		
Attaching logging device	Drag slice or metadevice to log rectangle of trans device.	`metattach(1M)`
Checking status	Object -> Info	`metastat(1M)`
Creating	Edit -> Create -> Trans	`metainit(1M)`
Detaching logging device	Drag logging device out of trans metadevice.	`metadetach(1M)`
Removing	Edit -> Delete	`metaclear(1M)`
Sharing a log among trans metadevices	Edit -> Create -> Trans; for the log, use the same slice already in use by another trans metadevice.	`metainit(1M)`; for the log, use the same slice already in use by another trans metadevice.

CHAPTER 1

Getting Started

This chapter introduces the DiskSuite software, and describes the steps to take to get a new configuration up and running. The high-level tasks for getting started are:

- “Planning Your Configuration” on page 2
- “Creating Initial State Database Replicas ” on page 3

If you are merely upgrading an existing DiskSuite installation, you don‘t need to read this chapter. Instead, refer to the *Solstice DiskSuite 4.2 Product Installation and Release Notes* to upgrade to Solstice DiskSuite 4.2.

Introduction

Solstice DiskSuite 4.2 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 center around increasing:

- Storage capacity
- Data availability

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.

If, for example, you want to create more storage capacity, you could use DiskSuite to “fool” the system into thinking that a collection of many small slices is one physical disk. After you have created a metadevice from these slices, you can immediately begin using it just as any “real” disk.

DiskSuite can also increase the availability of data by using mirrors and RAID5 metadevices. Mirrors and RAID5 metadevices replicate data so that it is not destroyed if the disk on which it is stored fails.

For a more detailed discussion of metadevices, and for more information on all of DiskSuite’s tools, refer to *Solstice DiskSuite 4.2 Reference Guide*.

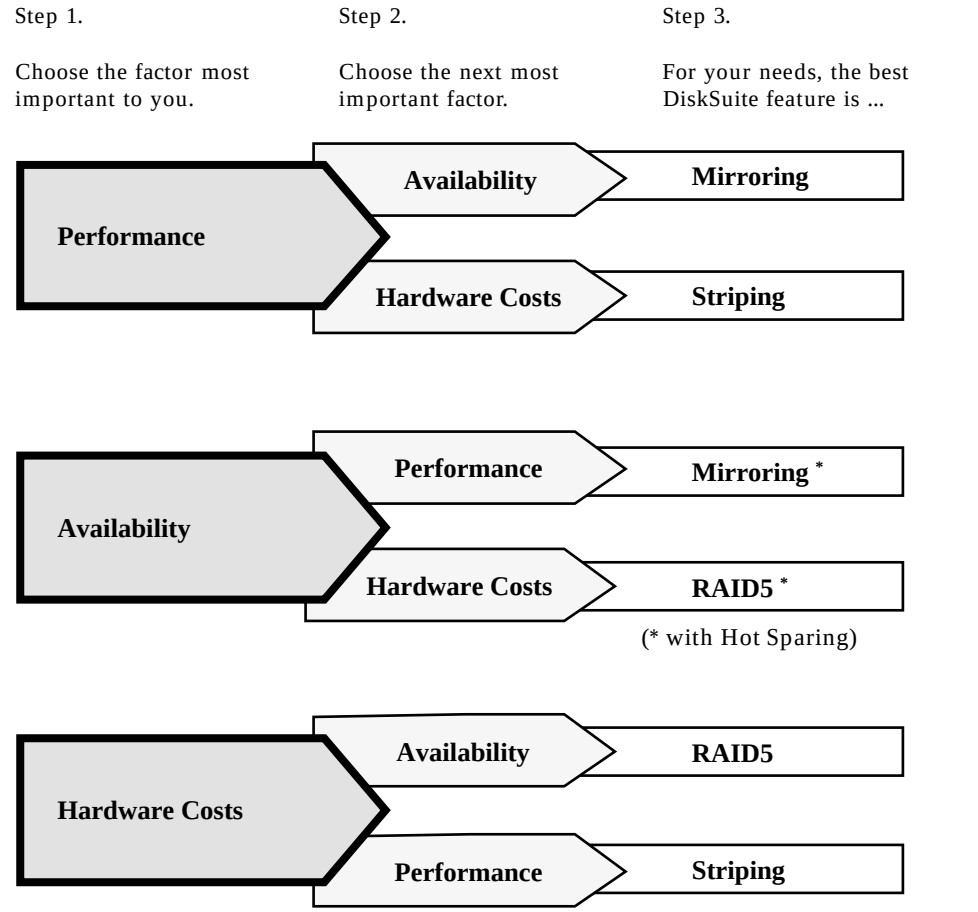
Planning Your Configuration

Reading *Solstice DiskSuite 4.2 Reference Guide* is an essential starting point for understanding how DiskSuite works, and how you can best use its features.

The Glossary located at the end of *Solstice DiskSuite 4.2 Reference Guide* explains DiskSuite terminology, which may be unfamiliar to you.

▼ How to Determine DiskSuite Use in Your Environment

Use the following high-level decision tree to determine the best fit for DiskSuite in your environment.



Example: If your most important factor is lower hardware costs, followed by availability, your needs would be best served by using RAID5 metadevices.

Note - For a complete discussion of how to plan for DiskSuite, refer to *Solstice DiskSuite 4.2 Reference Guide*.

Creating Initial State Database Replicas

This section describes how to create initial state database replicas based on your system configuration.

Preliminary Information for Creating State Database Replicas

- A state database replica stores DiskSuite configuration and state information. Before you can use DiskSuite, you must create state database replicas.
- At least three state database replicas should be created. The system will stay running with exactly half or more state database replicas available at any one time. The system will panic when less than half the state database replicas are available. The system will not reboot without one more than half the total state database replicas. Instead, it will go into single-user mode for administrative tasks.
- Use Table 1–1 to estimate the number of state database replicas required based on the number of hard drives in your system. Adjust it to meet your needs.

TABLE 1–1 Estimating Number of State Database Replicas Required

Number of Hard Drives	Number of State Database Replicas to Create
One	Three, all on one slice[1]
Two–Four	Two on each drive
Five or more	One on each drive

1. This configuration is not desirable, as it creates a single point of failure.

Note - In a two-drive configuration, always create two state database replicas on each drive. For example, assume you create two state database replicas on one slice and only one state database replica on the other. If the slice with two state database replicas fails, DiskSuite will not function because the remaining slice only has one state database replica. Refer to *Solstice DiskSuite 4.2 Reference Guide* for more information on the operation of state database replicas.

- You can create state database replicas on either a dedicated slice, or on a slice that will be used as part of a simple metadevice, RAID5 metadevice, or trans metadevice. Refer to *Solstice DiskSuite 4.2 Reference Guide* for guidelines on planning the location of state database replicas.
- The maximum number of state database replicas is 50. This also applies to replicas that are part of a diskset.

▼ How to Decide Which Method to Use for Creating Initial State Database Replicas

Use one of the following three methods to create your initial state database replicas. The methods depend on how the slices on your system are configured. If your system has no available slices, or if you cannot repartition existing slices, you will not be able to use DiskSuite software.

1. Starting from scratch: If you have a new system, the easiest way to create state database replicas is to put them on slices that will become part of a simple metadevice (stripe/concatenation), RAID5 metadevice, or trans metadevice. A state database replica cannot be part of root (/), `swap`, `/usr`, an existing file system, or a slice containing data.

   If you have a new system, refer to "How to Create Initial State Database Replicas From Scratch (DiskSuite Tool)" on page 6.

Note - When you combine state database replicas and metadevices on the same slice, DiskSuite detects the state database replica and adjusts the starting address and size of the metadevice accordingly. The advantage to this method is that you don't have to do anything extra, such as repartitioning a slice, and you don't have to worry about wasting space on a slice by having to dedicate it to just the state database replica.

---

2. Starting with existing unused slices: If your existing configuration has unused dedicated slices, put the state database replicas on those slices.

   You may find, however, that the state database replicas are relatively small (517 KB, or 1034 sectors), so you don't want to dedicate a large slice just to hold the small state database replica. You may have to make some slices smaller than they currently are.

   If you have existing, unused slices, refer to "How to Create Initial State Database Replicas on Existing Unused Slices (DiskSuite Tool)" on page 10.

3. Starting with no unused slices: If your existing configuration has no available slices, you need to "steal" space from another slice, such as `swap`, for the state database replicas. If this is the case, refer to "How to Create State Database Replicas by Allocating Space From the `swap` Partition" on page 11.

Caution - Never try to put a state database replica on a slice that is in use by a file system or database; this will destroy any existing data on that slice.

Prerequisites for Creating State Database Replicas

The prerequisites for the tasks in this section are:

- Have a current backup of all data.

- Have Solstice DiskSuite 4.2 software installed. (Refer to the Product Installation and Release Notes to install DiskSuite.)
- Have root privilege.

▼ How to Create Initial State Database Replicas From Scratch (DiskSuite Tool)

This procedure directs you to use three windows in DiskSuite Tool to identify and view your configuration, and then put state database replicas on slices.

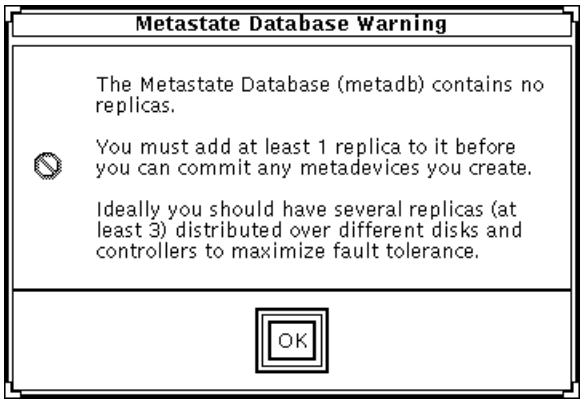
1. **Make sure you have met the prerequisites (“Prerequisites for Creating State Database Replicas” on page 5), and have read the preliminary information (“Preliminary Information for Creating State Database Replicas” on page 4).**

2. **Decide how many state database replicas you need, based on Table 1–1.**

3. **As root, start DiskSuite Tool to identify and view your configuration.**

```
# metatool &
```

4. **The first time you run DiskSuite, you see the following message. Click OK to continue.**



5. **From the Objects list, drag the MetaDB object onto the canvas.**
   Use the ADJUST mouse button (by default, the middle button) to drag the object.

Note - The MetaDB object represents or “contains” all the state database replicas in your configuration. Because you are starting from scratch, the MetaDB object has a “Critical” label on it to show you that you need to add state database replicas to it.

6. **Click Disk View to display the Disk View window.**

7. **Select View All Controllers from the View menu in the Disk View window.**

   This shows your storage configuration: controllers, disks, and slices. You may want to use the View menu on the Disk View window to set the view to 50, 100, or 200 percent viewing. The default is 100 percent.

8. **Click Slices to display the Slice Browser window.**

   This window shows the status and current use of slices. (The Disk View window’s status line also displays this information when you position the cursor over the slice.)

9. **Build a MetaDB object (create the state database replicas) by dragging available slices from the disks shown in the Disk View window, or from the Slice Browser window, over the MetaDB object in the canvas.**

   As long as the MetaDB object was selected when you opened the Disk View window, when a slice is put into the MetaDB object, the Disk View window colors the slice (blue on color screens, black on monochrome screens) that contains the state database replica. This helps you see where state replicas are located, for example, across controllers.

   Try to balance state database replicas across controllers to achieve redundancy. For configurations with many disks, put a state database replica on each disk. (You can also click the Set Filters button in the Disk View window to filter for slices that are not in use.)

10. **[Optional] To add multiple state database replicas to the same slice, display the MetaDB object’s Info window, type the slice in the Slice field, type the number of replicas to add in the Replicas field, click Attach, then click Close.**

    This method does not enable you to “change” the number of state database replicas on a slice, only to initially specify them.

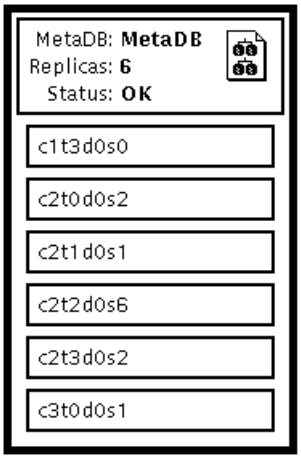
11. **Select the MetaDB object, then click Commit.**

12. **To verify that the MetaDB object was committed, display the Configuration Log.**

    Select Configuration Log from the Browse menu.

Example — Committed MetaDB Object Consisting of Six State Database Replicas

This example shows a committed MetaDB object consisting of six slices, each containing one state database replica. The status of the MetaDB object is “OK,” indicating that the state database replicas are spread across at least three controllers.



▼ How to Create Initial State Database Replicas From Scratch (Command Line)

This task is the equivalent of the previous one. It demonstrates the use of the command line utilities to create state database replicas.

After checking the prerequisites (“Prerequisites for Creating State Database Replicas” on page 5), and the preliminary information (“Preliminary Information for Creating State Database Replicas” on page 4), use the `metadb(1M)` command to create state database replicas. Refer to the `metadb(1M)` man page for more information.

Note - When you run the `metadb(1M)` command for the first time, the system displays a warning message that no state database replicas exist for this host. Ignore this message. It appears only when you create the state database replicas for the first time.

Example — Creating Initial State Database Replicas on a System With Five Disks

```
# metadb -a -f c0t1d0s3 c1t1d0s3 c2t1d0s3 c3t1d0s3 c4t1d0s3
# metadb
```

(continued)

```
     flags         first blk      block count
a        u          16               1034           /dev/dsk/c0t1d0s3
a        u          16               1034           /dev/dsk/c1t1d0s3
a        u          16               1034           /dev/dsk/c2t1d0s3
a        u          16               1034           /dev/dsk/c3t1d0s3
a        u          16               1034           /dev/dsk/c4t1d0s3
```

The –a and –f options are used together to create the initial state database replicas. Five initial state database replicas, one on each of five slices, are created. By spreading the state database replicas across controllers, you can increase metadevice performance and reliability. The metadb command checks that the replicas are active, as indicated by the –a flag.

Example — Creating Initial State Database Replicas on a System With Three Disks

```
# metadb -a -f -c 2 c0t1d0s3 c1t1d0s3 c2t1d0s3
# metadb
     flags         first blk      block count
a        u          16               1034           /dev/dsk/c0t1d0s3
a        u          1050             1034           /dev/dsk/c0t1d0s3
a        u          16               1034           /dev/dsk/c1t1d0s3
a        u          1050             1034           /dev/dsk/c1t1d0s3
a        u          16               1034           /dev/dsk/c2t1d0s3
a        u          1050             1034           /dev/dsk/c2t1d0s3
```

The –a and –f options are used together to create the initial state database replicas. The –c 2 option puts two state database replicas on each specified slice, creating a total of six replicas. By spreading the state database replicas across controllers, you can increase metadevice performance and reliability. The metadb command checks that the replicas are active, as indicated by the –a flag.

Example — Creating Initial State Database Replicas on a System With Only One Disk

```
# metadb -a -f -c 3 c0t0d0s3
# metadb
        flags           first blk      block count
     a        u          16             1034            /dev/dsk/c0t0d0s3
     a        u          1050           1034            /dev/dsk/c0t0d0s3
     a        u          2084           1034            /dev/dsk/c0t0d0s3
```

The system in this example consists of only one disk. (DiskSuite is being installed for its UFS logging feature.) The –a and –f options are used together to create the initial state database replicas. The –c 3 option creates three state database replicas on the same slice on the system's only disk. The metadb command checks that the replicas are active, as indicated by the –a flag.

Caution - Use a one-disk configuration for state database replicas as a last resort, as it creates a single point-of-failure.

▼ How to Create Initial State Database Replicas on Existing Unused Slices (DiskSuite Tool)

In this procedure, you will resize slices to hold state database replicas. Most slices are likely to be much larger than what is required for a state database replica. To prevent the waste of large amounts of disk space, you can take space from one slice and allocate it to an unused slice name or a new slice.

1. **Make sure you have met the prerequisites ("Prerequisites for Creating State Database Replicas" on page 5), and have read the preliminary information ("Preliminary Information for Creating State Database Replicas" on page 4). Also be sure that you know what slices are in your configuration and the names of the slices that are not currently in use.**

2. **Decide how many state database replicas you need, based on Table 1–1.**

3. **Create at least one new "small" slice. You can use the format(1M) command, the fmthard(1M) command, or Storage Manager. See *System Administration Guide, Volume I* for information on managing disks and slices.**

   Note - Make sure that the disk space assigned to the slice you are creating for the metadevice state database is not shared with another slice.

4. **Reboot.**

5. **Follow Step 3 on page 6 through Step 12 on page 7 of "How to Create Initial State Database Replicas From Scratch (DiskSuite Tool)" on page 6.**

▼ How to Create State Database Replicas by Allocating Space From the swap Partition

Use this procedure if you have no unused slices on which to put state database replicas. Even if you have no unused slices, you can create a new slice by taking space from the end of the swap partition and assigning it to an unused slice name.

Caution - Re-allocating space for a new slice can be done only on the swap partition. Attempting this process on a file system will cause the loss of data on that file system.

1. **As root, halt the system and boot into single-user mode.**

```
# halt
...
ok boot -s
...
INIT: SINGLE USER MODE

Type Ctrl-d to proceed with normal startup,
(or give root password for system maintenance):<root-password>
Entering System Maintenance Mode

Mar 12 16:52:38 su: 'su root' succeeded for root on /dev/syscon
Sun Microsystems Inc.   SunOS 5.5       Generic November 1995
```

2. **Use the** swap -l **command to identify a system's swap areas. Activated swap devices or files are listed under the** swapfile **column.**

```
# swap -l
swapfile           dev  swaplo blocks   free
/dev/dsk/c0t2d0s1 32,17      8 205624 192704
```

3. **Turn off swapping and verify that no slices are being used for** swap**.**

```
# swap -d /dev/dsk/slice
# swap -l
```

4. **Repartition the disk with the** `format(1M)` **command, the** `fmthard(1M)` **command, or Storage Manager, and reboot your system.**

   You will delete cylinders from the `swap` slice, then add the cylinders to a new slice name that is not being used.

5. **Verify that the new slice exists by logging back into your system and using the** `prtvtoc(1M)` **command.**

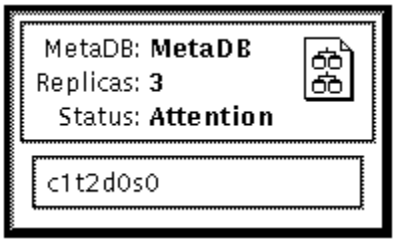
   ```
   # prtvtoc /dev/rdsk/slice
   ```

6. **Follow Step 3 on page 6 through Step 12 on page 7 of "How to Create Initial State Database Replicas From Scratch (DiskSuite Tool)" on page 6.**

   Using the Metadevice State Database Info window, create three state database replicas on the slice that you created by repartitioning `swap`.

Example — Committed MetaDB Object Consisting of Three State Database Replicas

This example shows a committed MetaDB object with three state database replicas on the same slice. Notice that the object indicates three state database replicas even though it contains only one slice. The status of the MetaDB object is "Attention," because the replicas are not spread across at least three controllers.



Where to Go From Here

Now you are ready to use DiskSuite to manage your configuration. Use the procedures in Chapter 2 to create DiskSuite objects, such as metadevices and hot spare pools.

CHAPTER 2

Creating DiskSuite Objects

This chapter describes how to create DiskSuite objects, both with the DiskSuite graphical user interface and the command line utilities.

Use the following to proceed directly to the section that provides step-by-step instructions using DiskSuite Tool.

Use the following to proceed directly to the section that provides step-by-step instructions using the command line interface.

Overview of Creating DiskSuite Objects

When you create a DiskSuite object, you assign physical slices to a logical DiskSuite name. The DiskSuite objects that you can create using the steps in this chapter include:

- Additional state database replicas
- Metadevices (stripes, concatenations, mirrors, RAID5 metadevices, and trans metadevices)
- Hot spare pools

- Disksets

For general information on DiskSuite, refer to *Solstice DiskSuite 4.2 Reference Guide*.

Note - For hints on how to name metadevices, refer to "Metadevice Naming Conventions" on page 277.

Prerequisites for Creating DiskSuite Objects

Here are the prerequisites for the steps in this chapter:

- Create initial state database replicas. If you have not done so, refer to Chapter 1.
- Identify slices that are available for use by DiskSuite. If necessary, use `format(1M)`, `fmthard(1M)`, or Disk Manager to repartition existing disks.
- Make sure you have root privilege.
- Have a current backup of all data.
- If using the graphical user interface, start DiskSuite Tool.

  To work with "local" metadevices (metadevices not in a diskset configuration), type:

```
# metatool &
```

  To work with metadevices in a diskset, make sure you are the diskset owner and type:

```
# metatool -s diskset_name &
```

Creating Additional State Database Replicas

This section describes how to create additional state database replicas on a system that is up and running.

Preliminary Information for Creating Additional State Database Replicas

- You can create state database replicas on either a dedicated slice, or on a slice that will be used as part of a simple metadevice, mirror, RAID5 metadevice, or trans metadevice. Refer to *Solstice DiskSuite 4.2 Reference Guide* for guidelines on planning the location of state database replicas.
- You can add additional state database replicas to the system at any time. The additional state database replicas can help ensure DiskSuite availability.
- Adding additional state database replicas before creating a mirror can increase the mirror's performance. As a general rule, add one additional replica for each mirror you add to the system.
- The maximum number of state database replicas is 50. This also applies to replicas that are part of a diskset.

▼ How to Create Additional State Database Replicas (DiskSuite Tool)

After the initial state database replicas are created, you can create additional state database replicas as needed.

1. **Make sure you have met the prerequisites ("Prerequisites for Creating DiskSuite Objects" on page 17), and have read the preliminary information ("Preliminary Information for Creating Additional State Database Replicas" on page 18).**

2. **Double-click the MetaDB object in the Objects list.**

   DiskSuite Tool displays the MetaDB object on the canvas.

3. **Click Slices to open the Slice Browser window.**

4. **Select the slice(s) for the additional state database replica(s). Drag the slice(s) to the top rectangle of the MetaDB object.**

   Use Control-click to select multiple slices.

5. **[Optional] To add multiple state database replicas to the same slice, display the MetaDB object's Info window, type the slice in the Slice field, type the number of replicas to add in the Replicas field, click Attach, then click Close.**

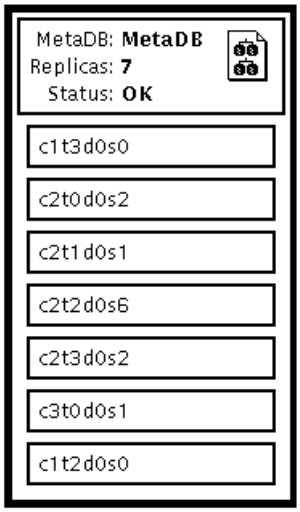
   This method does not enable you to "change" the number of state database replicas on a slice, only to initially specify them.

6. **Make sure that the MetaDB object is selected then click Commit.**

7. **To verify that the MetaDB object was committed, display the Configuration Log.**

Example — Committed MetaDB Object

This example shows a new state database replica added to a MetaDB object.



The added slice that contains the state database replica, `c1t2d0s0`, appears at the bottom of the list of slices in the committed MetaDB object.

▼ How to Create Additional State Database Replicas (Command Line)

After checking the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17), and the preliminary information (“Preliminary Information for Creating Additional State Database Replicas” on page 18), use the `metadb(1M)` command to create additional state database replicas. Refer to the `metadb(1M)` man page for more information.

Example — Adding a State Database Replica

```
# metadb -a c0t2d0s0
# metadb
        flags           first blk      block count
...
     a        u         16              1034            /dev/dsk/c0t2d0s0
```

The –a option adds the additional state database replica to the system. The `metadb` command checks that the replica is active, as indicated by the –a flag.

Example — Adding Two State Database Replicas to the Same Slice

```
# metadb -a -c 2 c0t2d0s0
# metadb
        flags           first blk      block count
...
     a        u         16              1034            /dev/dsk/c0t2d0s0
     a        u         1050            1034            /dev/dsk/c0t2d0s0
```

The –a option adds additional state database replicas to the system. The –c 2 option places two replicas on the specified slice. The metadb command checks that the replicas are active, as indicated by the –a flag.

Creating Stripes and Concatenations

This section describes how to create stripes and concatenations. To concatenate (add slices) to an existing stripe, refer to “How to Expand an Existing Concat/Stripe (DiskSuite Tool)” on page 128.

Preliminary Information for Creating Stripes and Concatenations

- In DiskSuite terms, *stripes* and *concatenations* are “simple metadevices” in that they are composed of slices. Both stripes and concatenations enable you to expand disk storage capacity. They can be used either directly or as the building blocks for mirrors and trans metadevices.
- The system must contain at least three state database replicas before you can create stripes and concatenations. (See Chapter 1.)
- You can use *multi-way* stripes and concatenations (stripes or concatenations that consist of more than one slice) for any file system except the following:
  - Root (/)
  - swap (You can create multiple, single-slice swap metadevices)
  - /usr
  - /var
  - /opt
  - Any other file system accessed during a Solaris installation or upgrade

- Do not stripe slices that are on the same physical disk. This practice eliminates simultaneous access and reduces performance.
- If possible create metadevices from disks consisting of the same disk geometries. The historical reason is that UFS uses disk blocks based on disk geometries. Today, the issue is centered around performance: a stripe composed of disks with different geometries will only be as fast as its slowest disk.
- When possible, distribute the slices of a stripe or concatenation across different controllers and busses. Using stripes that are each on different controllers increases the number of simultaneous reads and writes that can be performed.
- Do not create a striped metadevice from an existing file system or data. Doing so will destroy data. Instead, use a concatenation. (You can create a striped metadevice from existing data, but you must dump and restore the data to the metadevice.)
- Avoid striping slices with different sizes; this wastes disk space. If necessary, you can assign the unused portion to another slice. The slice must be repartitioned (using `format(1M)`, `fmthard(1M)`, or Storage Manager) to assign the unused disk space to another available slice name.

Note - Because DiskSuite Tool uses the Concat/Stripe object to represent both concatenated metadevices and striped metadevices, the only way to distinguish them by a glance is to study the pattern of rectangles in the Concat/Stripe object.

▼ How to Create a Striped Metadevice (DiskSuite Tool)

Caution - Do not create a striped metadevice from an existing file system or data. Doing so will destroy data. To create a striped metadevice from existing data, you must dump and restore the data to the metadevice.

---

1. **Make sure you have met the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17), and have read the preliminary information (“Preliminary Information for Creating Stripes and Concatenations” on page 20).**

2. **Click the Concat/Stripe template to display an unassigned and uncommitted Concat/Stripe object on the canvas. The metadevice name is automatically assigned.**

3. **[Optional] Change the default metadevice name.**

   Display the object’s pop-up menu and choose Info. Type the new metadevice name in the Device Name field and click the Attach button. Then click the Close button.

4. **Click Slices to open the Slice Browser window.**

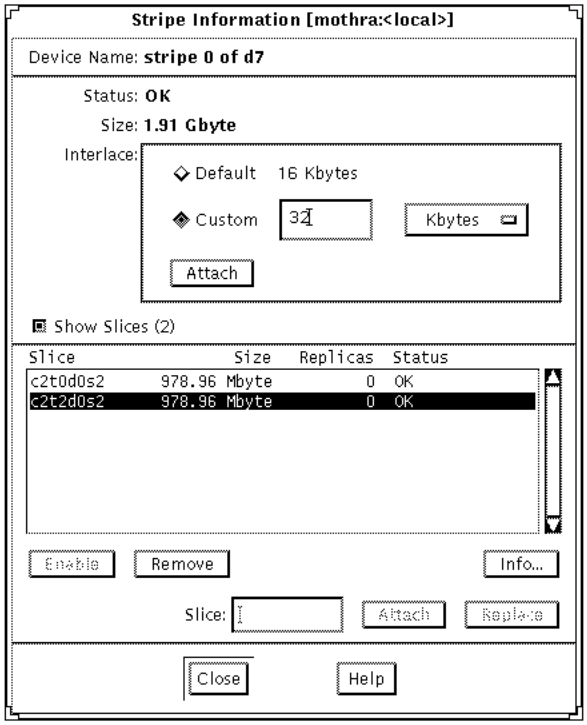
5. **Use Control-click to select multiple slices, and drag them into the Concat/Stripe object.**

6. **Click Stripe on the "Stripe or Concat" dialog box that appears.**

Note - If you drag slices one at a time, the "Stripe or Concat" dialog box does not appear, so be careful where you drop them on the Concat / Stripe object. To create a stripe, drop the slices on the rectangle labeled "stripe 0 of d*x*."

---

7. **[Optional] To change the default interlace value of 16 Kbytes, point the cursor inside the rectangle where "stripe 0 of d*x*" is displayed, and display the pop-up Stripe Information window for this striped metadevice.**



8. **To change the interlace value, click Custom and type the new value in the field next to the Custom button. This value can be in Kbytes, Mbytes, or sectors. You can configure it by using the drop-down menu beside this field. Click Attach to set the value, then click Close.**

For more information on setting the interlace value, refer to *Solstice DiskSuite 4.2 Reference Guide.*

Note - The interlace value cannot be changed after committing the metadevice.

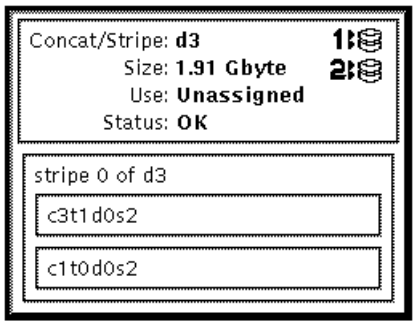
---

9. Click the top rectangle of the Concat/Stripe object, then click Commit.

10. To verify that striped metadevice was committed, display the Configuration Log.

Example — Committed Stripe

This example shows a newly created stripe made of two slices. The Stripe object displays the slices on top of each other in a single rectangle. Compare the presentation of a concatenated metadevice with "Example — Committed Concatenation Object" on page 25.



Where to Go From Here

To prepare the newly created stripe for a file system, refer to "How to Create a File System on a Metadevice (File System Manager)" on page 73. An application, such as a database, that uses the raw metadevice must have its own way of recognizing the metadevice.

▼ How to Create a Striped Metadevice (Command Line)

After checking the prerequisites ("Prerequisites for Creating DiskSuite Objects" on page 17), and the preliminary information ("Preliminary Information for Creating Stripes and Concatenations" on page 20), create the striped metadevice with the `metainit(1M)` command. Refer to the `metainit(1M)` man page for more information.

Caution - Do not create a striped metadevice from an existing file system or data. Doing so will destroy data. To create a striped metadevice from existing data, you must dump and restore the data to the metadevice.

Example — Creating a Striped Metadevice of Two Slices With a 32 Kbyte Interlace

```
# metainit d10 1 2 c0t1d0s2 c0t2d0s2 -i 32k
d10: Concat/Stripe is setup
```

The striped metadevice, `d10`, consists of a single stripe (the number `1`) made of two slices (the number `2`). The `–i` option sets the interlace to 32 Kbytes. (The interlace cannot be less than 8 Kbytes, nor greater than 100 Mbytes.) If interlace were not specified, the striped metadevice would use the default of 16 Kbytes. The system verifies that the Concat/Stripe object has been set up.

For more information on setting the interlace value, refer to *Solstice DiskSuite 4.2 Reference Guide*.

Example — Creating a Striped Metadevice of Three Slices

```
# metainit d20 1 3 c0t1d0s2 c0t2d0s2 c0t3d0s2
d20: Concat/Stripe is setup
```

The striped metadevice, `d20`, consists of a single stripe (the number `1`) made of three slices (the number `3`). Because no interlace is specified, the striped metadevice uses the default of 16 Kbytes. The system verifies that the Concat/Stripe object has been set up.

Where to Go From Here

To prepare the newly created striped metadevice for a file system, refer to “How to Create a File System on a Metadevice (Command Line)” on page 74. An application, such as a database, that uses the raw metadevice must have its own way of recognizing the metadevice.

▼ How to Create a Concatenation (DiskSuite Tool)

Use this procedure to create a concatenation from slices that do not contain any data. To concatenate existing data, such as a file system or a database, refer to “How to Expand a Slice Containing Existing Data (DiskSuite Tool)” on page 125.

1. **Make sure you have met the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17), and have read the preliminary information (“Preliminary Information for Creating Stripes and Concatenations” on page 20).**

2. **Click the Concat/Stripe template.**

   An unassigned and uncommitted Concat/Stripe object appears on the canvas. The metadevice name is automatically assigned.

3. **[Optional] Change the default metadevice name.**

   Display the object’s pop-up menu and choose Info. Type the new metadevice name in the Device Name field and click Attach. Then click Close.

4. **Click Slices to display the Slice Browser window.**

5. **Use Control-click to select the slices to be concatenated and drag them into the Concat/Stripe object.**

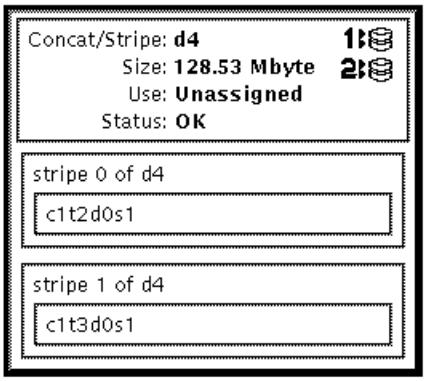
6. **Click Concat on the “Stripe or Concat?” dialog box that appears.**

7. **Click the top rectangle of the Concat/Stripe object, then click Commit.**

8. **To verify that concatenation was committed, display the Configuration Log.**

Example — Committed Concatenation Object

This example shows a newly created concatenation made of two slices. The object displays the slices in the concatenation so that each slice is in its own rectangle.



Where to Go From Here

To prepare the newly created concatenation for a file system, refer to "How to Create a File System on a Metadevice (File System Manager)" on page 73. An application, such as a database, that uses the raw metadevice must have its own way of recognizing the metadevice.

▼ How to Create a Concatenation (Command Line)

After checking the prerequisites ("Prerequisites for Creating DiskSuite Objects" on page 17), and the preliminary information ("Preliminary Information for Creating Stripes and Concatenations" on page 20), use the `metainit(1M)` command to create the concatenation. For more information, refer to the `metainit(1M)` man page.

Do not use this procedure on an existing file system or data. To concatenate existing data, such as a file system or a database, refer to "How to Expand a Slice Containing Existing Data (Command Line)" on page 127.

The following examples create concatenations that do not contain any existing data.

Example — Creating a Concatenation of Two Slices

```
# metainit d25 2 1 c0t1d0s2 1 c0t2d0s2
d25: Concat/Stripe is setup
```

This example creates a concatenation, `d25`, consisting of two "stripes" (the number `2`) each made of a single slice (the number `1` in front of each slice). The system verifies that the Concat/Stripe object has been set up.

Example — Creating a Concatenation of Four Slices

```
# metainit d40 4 1 c0t1d0s2 1 c0t2d0s2 1 c0t2d0s3 1 c0t2d1s3
d40: Concat/Stripe is setup
```

This example creates a concatenation called `d40` consisting of four “stripes” (the number 4) each made of a single slice (the number `1` in front of each slice). The system verifies that the Concat/Stripe object has been set up.

Where to Go From Here

To prepare the newly created concatenation for a file system, refer to “How to Create a File System on a Metadevice (Command Line)” on page 74. An application, such as a database, that uses the raw metadevice must have its own way of recognizing the metadevice.

Creating Mirrors

This section describes how to create a mirror from scratch, and how to mirror an existing file system, including root (/).

Note - In the past, when creating a mirror for a file system, the mount point would change. Now, in certain instances, you can create mirrors without having to change the mount point. Refer to “How to Create a Mirror From an Existing Concat/Stripe (Command Line)” on page 279, for more information.

Preliminary Information for Creating Mirrors

- A *mirror* is a metadevice composed of one or more *submirrors*. A submirror is made of one or more striped or concatenated metadevices. Mirroring data provides you with maximum data availability by maintaining multiple copies of your data.
- The system must contain at least three state database replicas before you can create mirrors.
- Before creating a mirror, create the striped metadevices or concatenated metadevices that will make up the mirror.
- Any file system including root (`/`), `swap`, and `/usr`, or any application such as a database, can use a mirror.

Caution - When creating a mirror for an existing file system, be sure that the initial submirror contains the data.

- When creating a mirror, first create a one-way mirror, then attach a second submirror. This starts a resync operation and ensures that data is not corrupted.
- To mirror an existing file system, use an additional slice of equal or greater size than the slice already used by the mirror. You can use a concatenated metadevice or striped metadevice of two or more slices that have adequate space to contain the mirror.
- You can create a one-way mirror for a future two- or three-way mirror.
- You can create up to a three-way mirror. However, two-way mirrors usually provide sufficient data redundancy for most applications, and are less expensive in terms of disk drive costs. A three-way mirror enables you to take a submirror offline and perform a backup while maintaining a two-way mirror for continued data redundancy.
- Use the same size slices when creating submirrors. Using different size slices creates unused space in the mirror.
- Avoid having slices of submirrors on the same disk. Also, when possible, use disks attached to different controllers to avoid single points-of-failure. For maximum protection and performance, place each submirror on a different physical disk and, when possible, on different disk controllers. For further data availability, use hot spares with mirrors.
- Adding additional state database replicas before creating a mirror can increase the mirror's performance. As a general rule, add one additional replica for each mirror you add to the system.
- If possible create mirrors from disks consisting of the same disk geometries. The historical reason is that UFS uses disk blocks based on disk geometries. Today, the issue is centered around performance: a mirror composed of disks with different geometries will only be as fast as its slowest disk.

▼ How to Create a Mirror From Unused Slices (DiskSuite Tool)

The high-level steps in this procedure are:

- Creating two striped metadevices or concatenated metadevices, which will be the submirrors. Refer to "How to Create a Striped Metadevice (DiskSuite Tool)" on page 21 or "How to Create a Concatenation (DiskSuite Tool)" on page 25.
- Creating a one-way mirror from one of the submirrors.
- Creating the two-way mirror from the second submirror.

1. **Make sure you have met the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17), and have read the preliminary information (“Preliminary Information for Creating Mirrors” on page 27).**

2. **Click the Mirror template.**

   An unassigned and uncommitted Mirror object appears on the canvas. The metadevice name is automatically assigned.

3. **[Optional] Change the default metadevice name.**

   Display the object’s pop-up menu and select Info. Type the new metadevice name in the Device Name field and click Attach. Then click Close.

4. **To create a one-way mirror, drag the first Concat/Stripe object (submirror) from the Objects list into the Mirror template. Click the top rectangle of the mirror, then click Commit.**

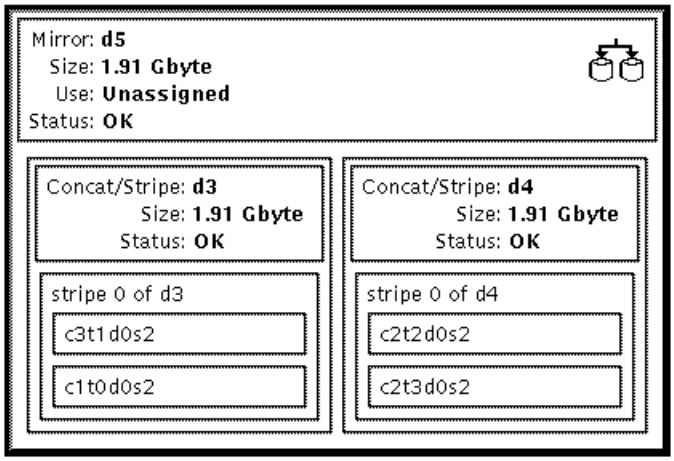
5. **To create a two-way mirror, drag the second Concat/Stripe object (submirror) from the Objects list into the Mirror. Click the top rectangle of the mirror then click Commit.**

   A resync of the second submirror begins. The Mirror object displays the resync progress.

6. **To verify that the mirror was committed, display the Configuration Log.**

Example — Committed Mirror Object

This example shows a committed mirror object, d5, consisting of two striped metadevices (submirrors), d3 and d4.



Where to Go From Here

To prepare the newly created mirror for a file system, refer to “How to Create a File System on a Metadevice (File System Manager)” on page 73. An application, such as a database, that uses the raw metadevice must have its own way of recognizing the metadevice.

▼ How to Create a Mirror From Unused Slices (Command Line)

The high-level steps in this procedure are:

- Creating two striped metadevices or concatenated metadevices, which will be the submirrors. Refer to “How to Create a Striped Metadevice (Command Line)” on page 23 or “How to Create a Concatenation (Command Line)” on page 26.
- Using the `metainit(1M)` `–m` command to create a one-way mirror from one of the submirrors.
- Using the `metattach(1M)` command to create the two-way mirror from the second submirror.

Check the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17), and the preliminary information (“Preliminary Information for Creating Mirrors” on page 27), before beginning. Refer to the `metainit(1M)` and `metattach(1M)` man pages for more information.

Example — Creating a Two-Way Mirror

```
# metainit d51 1 1 c0t0d0s2
d51: Concat/Stripe is setup
# metainit d52 1 1 c1t0d0s2
d52: Concat/Stripe is setup
# metainit d50 -m d51
d50: Mirror is setup
# metattach d50 d52
d50: Submirror d52 is attached
```

This example creates a two-way mirror, `d50`. The `metainit(1M)` command creates two submirrors (`d51` and `d52`), which are actually concatenations. The `metainit –m` command creates the one-way mirror from the `d51` concatenation. The `metattach(1M)` command attaches `d52`, creating a two-way mirror and causing a mirror resync. (Any data on the attached submirror is overwritten by the other submirror during the resync.) The system verifies that the objects are set up.

Where to Go From Here

To prepare a newly created mirror for a file system, refer to “How to Create a File System on a Metadevice (Command Line)” on page 74. An application, such as a database, that uses the raw metadevice must have its own way of recognizing the metadevice.

▼ How to Create a Mirror From a File System That Can Be Unmounted (DiskSuite Tool)

Use this procedure to mirror file systems that can be unmounted.

Note - Some file systems can be unmounted, while other file systems (like root (`/`), `/usr`, `/opt`, or `swap`) cannot be unmounted. To mirror these file systems, refer to “How to Create a Mirror From a File System That Cannot Be Unmounted (DiskSuite Tool)” on page 36.

The high-level steps in this procedure are:

- Identifying the slice that contains the existing file system to be mirrored
- Putting the file system’s slice in a Concat/Stripe object (which becomes submirror1)
- Creating a second Concat/Stripe object (which becomes submirror2)
- Dragging submirror1 to a mirror template and committing the mirror
- Unmounting the file system
- Dragging submirror2 to the mirror object and committing the mirror
- Remounting the file system

Caution - Do not create a multi-way mirror at first. Instead, commit a one-way mirror, then drag the additional submirror to the mirror template and commit the mirror again. This procedure initiates a mirror resync, ensuring that data is not corrupted.

---

1. **Make sure you have met the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17), and have read the preliminary information (“Preliminary Information for Creating Mirrors” on page 27).**

2. **Click Slices to display the Slice Browser window.**

   DiskSuite Tool should display the file system’s name next to the name of the slice upon which it is mounted. If you mount a file system after starting DiskSuite Tool, select Rescan Configuration from the File menu.

3. **Click the Concat/Stripe template.**

DiskSuite Tool displays an unassigned and uncommitted Concat/Stripe object on the canvas and gives it a metadevice name.

4. **[Optional] Change the default metadevice name.**

   Display the object's pop-up menu and select Info. Type the new metadevice name in the Device Name field and click Attach. Then click Close.

5. **Drag the file system's slice from the Slice Browser window into the Concat/Stripe object.**

   When the slice is dropped, a warning dialog box is displayed that the slice is mounted. Click Continue.

6. **Make sure the Concat/Stripe object is selected, then click Commit.**

   Click Really Commit on the warning dialog box that appears. This creates a metadevice that contains the file system, which will be used as the first submirror.

Note - If an entry exists in the `/etc/vfstab` file for the file system, and the file system is currently mounted, DiskSuite Tool automatically updates it to use the concatenation's name.

---

7. **Click the Concat/Stripe template.**

   DiskSuite Tool displays an unassigned and uncommitted Concat/Stripe object on the canvas and gives it a metadevice name.

8. **[Optional] Change the default metadevice name.**

   See Step 4 on page 32.

9. **Drag a slice from the Slice Browser window into the Concat/Stripe object.**

   Select an unused slice, a slice that contains a state database replica, or more than one slice that is the same size or greater as the existing file system.

10. **Make sure the Concat/Stripe object is selected, then click Commit.**

    This creates a metadevice, which will be used as the second submirror.

11. **Click the Mirror template.**

    DiskSuite Tool displays an unassigned and uncommitted mirror object on the canvas and gives it a metadevice name.

12. **[Optional] Change the default metadevice name.**

    See Step 4 on page 32.

13. **Drag the Concat/Stripe object containing the file system (created in Step 6 on page 32) into the Mirror template. A warning dialog box appears. Click Continue.**

Note - If an entry exists in the `/etc/vfstab` file for the file system, and the file system is currently mounted, DiskSuite Tool automatically updates it to use the mirror's metadevice name.

---

14. **Click the top rectangle of the Mirror object, then click Commit.**

    This creates the one-way mirror.

15. **Unmount the file system.**

    For example, use the `umount(1M)` command, or File System Manager.

16. **Select Rescan Configuration from the File menu.**

    DiskSuite Tool updates the file system's current mount status.

17. **Remount the file system on the mirror.**

    For example, use the `mount(1M)` command or File System Manager. (The mount point has now changed from the slice name to the mirror name.)

18. **Select Rescan Configuration from the File menu.**

    DiskSuite Tool updates the file system's mount point. The Mirror object shows that it is being used by the file system.

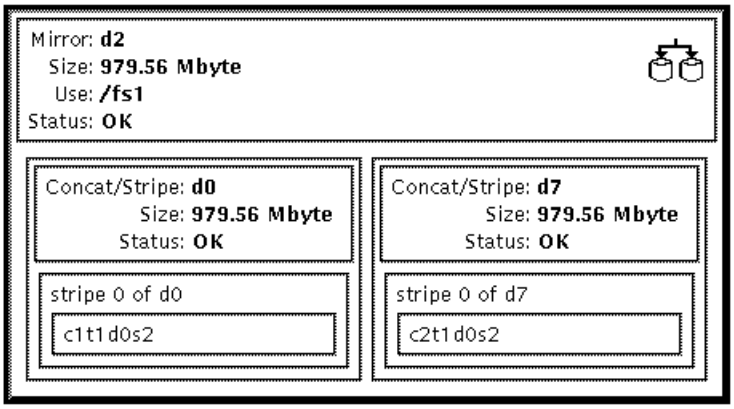
19. **Drag the second Concat/Stripe object (created in Step 10 on page 32) into the Mirror object. Click on the top rectangle of the Mirror object then click Commit.**

    Data from the first submirror is resynced to the second mirror automatically.

20. **To verify that the mirror was committed, display the Configuration Log.**

Example — Committed Mirror Object with a Mounted File System

This example shows a committed two-way mirror consisting of submirrors `d0` and `d7`, which in turn consist of slices `c1t1d0s2` and `c2t1d0s2`, respectively. The file system, `/fs1`, is mounted on the mirror, `d2`.



▼ How to Create a Mirror From a File System That Can Be Unmounted (Command Line)

Use this procedure to mirror an existing file system that can be unmounted.

Note - To use the command line to mirror `/usr`, `/opt`, or `swap`, refer to "How to Create a Mirror From a File System That Cannot Be Unmounted (Command Line)" on page 38. To use the command line to mirror root (`/`), refer to "SPARC: How to Create a Mirror From root (`/`) (Command Line)" on page 40, or "x86: How to Create a Mirror From root (`/`) (Command Line)" on page 42.

The high-level steps in this procedure are:

- Identifying the slice that contains the existing file system to be mirrored
- Using `metainit(1M) -f` to put the mounted file system's slice in a concat/stripe (submirror1)
- Creating a second concat/stripe (submirror2)
- Using `metainit(1M) -m` to create a one-way mirror with submirror1
- Unmounting the file system
- Editing the `/etc/vfstab` file so that the file system refers to the mirror
- Remounting the file system
- Using `metattach` to attach submirror2

Check the prerequisites ("Prerequisites for Creating DiskSuite Objects" on page 17), and the preliminary information ("Preliminary Information for Creating Mirrors" on page 27), before beginning. Refer to the `metainit(1M)` and `metattach(1M)` man pages for more information.

Caution - Do not create a multi-way mirror. Rather, create a one-way mirror with the `metainit(1M)` command then attach the additional submirrors with the `metattach(1M)` command. When the `metattach(1M)` command is not used, no resync operations occur and data could become corrupted. Also, do not create a two-mirror for a file system without first unmounting the file system and editing the `/etc/vfstab` file to reference the mirror metadevice before attaching the second submirror.

Example — Creating a Two-Way Mirror

```
# metainit -f d1 1 1 c1t0d0s0
d1: Concat/Stripe is setup
# metainit d2 1 1 c2t0d0s0
d2: Concat/Stripe is setup
# metainit d0 -m d1
d0: Mirror is setup
# umount /master
(Edit the /etc/vfstab file so that the file system references the mirror)
# mount /master
# metattach d0 d2
d0: Submirror d2 is attached
```

The –`f` option forces the creation of the first concatenation, `d1`, which contains the mounted file system `/master` on `/dev/dsk/c1t0d0s0`. The second concatenation, `d2`, is created from `/dev/dsk/c2t0d0s0`. (This slice must be the same size or greater than that of `d1`.) The `metainit` command with the –`m` option creates the one-way mirror, `d0`, from `d1`.

Next, `/master` is unmounted and its entry changed in the `/etc/vfstab` file to reference the mirror. For example, the following line:

```
/dev/dsk/c1t0d0s0 /dev/rdsk/c1t0d0s0 /master ufs 2 yes -
```

should be changed to:

```
/dev/md/dsk/d0 /dev/md/rdsk/d0 /master ufs 2 yes -
```

Finally, the `/master` file system is remounted and submirror `d2` attached to the mirror, causing a mirror resync. (The system verifies that the concatenations and the mirror are set up, and that submirror `d2` is attached.)

▼ How to Create a Mirror From a File System That Cannot Be Unmounted (DiskSuite Tool)

Use this procedure to mirror file systems, such as root (/), `/usr`, `/opt`, and `swap`, that cannot be unmounted during normal system usage.

The high-level steps in this procedure are:

- Putting the file system's slice in a single-slice Concat/Stripe object (submirror1)
- Creating a second Concat/Stripe object (submirror2)
- Dragging submirror1 to a mirror template and committing the mirror
- Rebooting
- Dragging submirror2 to the Mirror object and committing the mirror
- Recording the alternate boot path if mirroring root (/)

Note - When mirroring root (/), it is essential that you record the secondary root slice name to reboot the system if the primary submirror fails. This information should be written down, not recorded on the system, which may not be available. See Chapter 7 for details on recording the alternate boot device, and on booting from the alternate boot device.

---

1. **Make sure you have met the prerequisites ("Prerequisites for Creating DiskSuite Objects" on page 17), and have read the preliminary information ("Preliminary Information for Creating Mirrors" on page 27).**

2. **Click the Concat/Stripe template.**

   DiskSuite Tool displays an unassigned and uncommitted Concat/Stripe object on the canvas and gives it a metadevice name.

3. **[Optional] Change the default metadevice name.**

   Display the object's pop-up menu and select Info. Type the new metadevice name in the Device Name field and click the Attach button. Then click the Close button.

4. **Click Slices to display the Slice Browser window.**

5. **Drag the slice containing the file system you want to mirror to the top rectangle of the Concat/Stripe object.**

   When you drop the slice, DiskSuite Tool displays a warning dialog box. Click the Continue button.

6. **Click the top rectangle of the Concat/Stripe object, then click Commit.**

   Click the Really Commit button if DiskSuite Tool displays a warning dialog box. This creates a one-way concatenation that contains the file system, which will be used as the first submirror.

Note - If an entry exists in the `/etc/vfstab` file for the file system, and the file system is currently mounted, DiskSuite Tool automatically updates it to use the concatenation's metadevice name.

7. **Click the Concat/Stripe template.**

   DiskSuite Tool displays an unassigned and uncommitted Concat/Stripe object on the canvas and gives it a metadevice name.

8. **[Optional] Change the default metadevice name.**

   See Step 3 on page 36.

9. **Drag a slice from the Slice Browser window to this Concat/Stripe object.**

   When creating this one-way concatenated metadevice, use an unused slice that is the same size as or greater than the existing file system.

10. **Click the top rectangle of the Concat/Stripe object then click Commit.**

    This creates the second submirror.

11. **Click the Mirror template.**

    DiskSuite Tool displays an unassigned and uncommitted Mirror object on the canvas and gives it a metadevice name.

12. **[Optional] Change the default metadevice name.**

    See Step 3 on page 36.

13. **Drag the Concat/Stripe object that contains the file system into the Mirror template.**

    Click the Continue button if DiskSuite Tool displays a warning dialog box.

Note - If an entry exists in the `/etc/vfstab` file for the file system, and the file system is currently mounted, DiskSuite Tool automatically updates it to use the mirror's metadevice name.

14. **Click the top rectangle of the Mirror object then click Commit.**

15. **Reboot.**

16. **Restart DiskSuite Tool.**

17. **Drag the second Concat/Stripe object created in Step 10 on page 37 into the Mirror object to create the two-way mirror. Then click Commit.**

    Data from the first submirror is resynced to the second mirror.

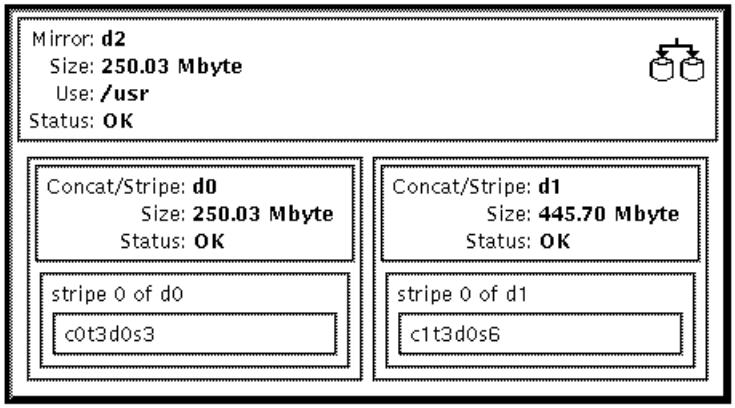
18. **If you are mirroring root (/), record the alternate boot path.**

    Refer to Chapter 7 for more information.

19. **To verify that the mirror was committed, display the Configuration Log.**

Example — Committed Mirror for /usr

This example shows a committed mirror containing the /usr file system.



▼ How to Create a Mirror From a File System That Cannot Be Unmounted (Command Line)

Use this procedure to mirror file systems, such as /usr, /opt, and swap, that cannot be unmounted during normal system usage.

Note - To use the command line to mirror root (/), refer to "SPARC: How to Create a Mirror From root (/) (Command Line)" on page 40, or "x86: How to Create a Mirror From root (/) (Command Line)" on page 42.

The high-level steps in this procedure are:

- Using metainit(1M) -f to put the file system's slice in a single slice (one-way) concat/stripe (submirror1)
- Creating a second concat/stripe (submirror2)
- Using metainit(1M)to create a one-way mirror from submirror1

- Editing the /etc/vfstab file so that the file system refers to the mirror
- Rebooting
- Using metattach(1M) to attach submirror2

Caution - Do not create a multi-way mirror. Rather, create a one-way mirror with the metainit(1M) command, then attach the additional submirrors with the metattach(1M) command. When the metattach(1M) command is not used, no mirror resync occurs and data could become corrupted.

After checking the prerequisites ("Prerequisites for Creating DiskSuite Objects" on page 17), and the preliminary information ("Preliminary Information for Creating Mirrors" on page 27), use the metainit(1M) and metattach(1M) commands to create the mirror.

Example — Creating a Mirror From /usr

```
# metainit -f d12 1 1 c0t3d0s6
d12: Concat/Stripe is setup
# metainit d22 1 1 c1t0d0s6
d22: Concat/Stripe is setup
# metainit d2 -m d12
d2: Mirror is setup
(Edit the /etc/vfstab file so that /usr references the mirror)
# reboot
...
# metattach d2 d22
d2: Submirror d22 is attached
```

The –f option forces the creation of the first concatenation, d12, which contains the mounted file system /usr on /dev/dsk/c0t3d0s6. The second concatenation, d22, is created from /dev/dsk/c1t0d0s6. (This slice must be the same size or greater than that of d12.) The metainit command with the –m option creates the one-way mirror d2 using the concatenation containing /usr. Next, the /etc/vfstab file must be edited to change the entry for /usr to reference the mirror. For example, the following line:

```
/dev/dsk/c0t3d0s6 /dev/rdsk/c0t3d0s6 /usr ufs 1 yes -
```

should be changed to:

```
/dev/md/dsk/d2 /dev/md/rdsk/d2 /usr ufs 1 yes -
```

After a reboot, the second submirror d22 is attached to the mirror, causing a mirror resync. (The system verifies that the concatenations and the mirror are set up, and that submirror d22 is attached.)

Example — Creating a Mirror From swap

```
# metainit -f d11 1 1 c0t0d0s1
d11: Concat/Stripe is setup
# metainit d21 1 1 c1t0d0s1
d21: Concat/Stripe is setup
# metainit d1 -m d11
d1: Mirror is setup
(Edit the /etc/vfstab file so that swap references the mirror)
# reboot
...
# metattach d1 d21
d1: Submirror d21 is attached
```

The `-f` option forces the creation of the first concatenation, `d11`, which contains the mounted file system `swap` on `/dev/dsk/c0t0d0s1`. The second concatenation, `d21`, is created from `/dev/dsk/c1t0d0s1`. (This slice must be the same size or greater than that of `d11`.) The `metainit` command with the `–m` option creates the one-way mirror `d1` using the concatenation containing `swap`. Next, if there is an entry for `swap` in the `/etc/vfstab` file, it must be edited to reference the mirror. For example, the following line:

```
/dev/dsk/c0t0d0s1 - - swap - no -
```

should be changed to:

```
/dev/md/dsk/d1 - - swap - no -
```

After a reboot, the second submirror `d21` is attached to the mirror, causing a mirror resync. (The system verifies that the concatenations and the mirror are set up, and that submirror `d21` is attached.)

▼ SPARC: How to Create a Mirror From root (/) (Command Line)

Use this task to mirror root (/) on a SPARC system.

Note - The task for using the command line to mirror root (/) on an x86 system is different from the task used for a SPARC system. To mirror root (/) on an x86 system, refer to “x86: How to Create a Mirror From root (/) (Command Line)” on page 42. When mirroring root (/), it is essential that you record the secondary root slice name to reboot the system if the primary submirror fails. This information should be written down, not recorded on the system, which may not be available. See Chapter 7 for details on recording the alternate boot device, and on booting from the alternate boot device.

The high-level steps in this procedure are:

- Using `metainit(1M) -f` to put the root (/) slice in a single slice (one-way) concat (submirror1)
- Creating a second concat (submirror2)
- Using `metainit(1M)` to create a one-way mirror with submirror1
- Running the `metaroot(1M)` command
- Running the `lockfs(1M)` command
- Rebooting
- Using `metattach(1M)` to attach submirror2
- Recording the alternate boot path

Make sure you have met the prerequisites ("Prerequisites for Creating DiskSuite Objects" on page 17), and have read the preliminary information ("Preliminary Information for Creating Mirrors" on page 27). For more information refer to the `metainit(1M)`, `metaroot(1M)`, and `metattach(1M)` man pages.

Example — SPARC: Creating a Mirror From root (/)

```
# metainit -f d11 1 1 c0t3d0s0
d11: Concat/Stripe is setup
# metainit d12 1 1 c1t3d0s0
d12: Concat/Stripe is setup
# metainit d10 -m d11
d10: Mirror is setup
# metaroot d10
# lockfs -fa
# reboot
...
# metattach d10 d12
d10: Submirror d12 is attached
# ls -l /dev/rdsk/c1t3d0s0
lrwxrwxrwx   1 root     root          88 Feb  8 15:51 /dev/rdsk/c1t3d0s0 ->
../../devices/iommu@f,e0000000/vme@f,df010000/SUNW,pn@4d,1080000/ipi3sc@0,0/i
d@3,0:a,raw
```

The –f option forces the creation of the first concatenation, `d11`, which contains the mounted file system root (/) on `/dev/dsk/c0t3d0s0`. The second concatenation, `d12`, is created from `/dev/dsk/c1t3d0s0`. (This slice must be the same size or greater than that of `d11`.) The `metainit` command with the –m option creates the one-way mirror `d10` using the concatenation containing root (/). Next, the `metaroot` command edits the `/etc/vfstab` and `/etc/system` files so that the system may be booted with the root file system (/) on a metadevice. (It is a good idea to run `lockfs -fa` before rebooting.) After a reboot, the submirror `d12` is attached to the mirror, causing a mirror resync. (The system verifies that the concatenations and the mirror are set up, and that submirror `d12` is attached.) The

`ls -l` command is run on the root raw device to determine the path to the alternate root device in case the system needs to be booted from it.

▼ x86: How to Create a Mirror From root (/) (Command Line)

Use this task to mirror root (/) on an x86 system.

Note - When mirroring root (/), it is essential that you record the secondary root slice name to reboot the system if the primary submirror fails. This information should be written down, not recorded on the system, which may not be available. See Chapter 7 for details on recording the alternate boot device, and on booting from the alternate boot device.

The high-level steps in this task are:

- Creating a Solaris partition large enough for the root (/) mirror
- Installing the boot information on the alternate boot disk
- Using `metainit(1M)` –`f` to put the root (/) slice in a single slice (one-way) concat (submirror1)
- Creating a second concat (submirror2)
- Using `metainit(1M)` to create a one-way mirror with submirror1
- Running the `metaroot(1M)` command
- Running the `lockfs(1M)` command
- Rebooting
- Using `metattach(1M)` to attach submirror2
- Recording the alternate boot path

Note - You cannot mirror root (/) on an IDE drive.

For the purpose of the following procedures, assume that the alternate disk is `c0t1d0`.

Create a Solaris Partition

Use these steps to create a Solaris partition on an x86 system.

1. **Create the disk partition using the `fdisk(1M)` command.**

```
# fdisk /dev/rdsk/c0t1d0p0
```

   a. **If this is the first time you have run `fdisk(1M)`, you see the following:**

```
The recommended default partitioning for your disk is:

a 100% ‘‘SOLARIS System’’ partition

To select this, please type ‘‘y’’. To partition your disk
differently, type ‘‘n’’ and the ‘‘fdisk’’ program will let you
select other partitions.
```

b. If you have previously run `fdisk(1M)`, you see a menu similar to the following:

```
        Total disk size is 1855 cylinders
        Cylinder size is 1110 (512 byte) blocks

Partition    Status    Type        Start    End    Length    %
=========    ======    ========    =====    ===    ======   ===
    1        Active    SOLARIS         1   1854     1854    100


SELECT ONE OF THE FOLLOWING:

     1.  Create a partition
     2.  Change Active (Boot from) partition
     3.  Delete a partition
     4.  Exit (Update disk configuration and exit)
     5.  Cancel (Exit without updating disk configuration)
Enter Selection:
```

2. Select menu items to ensure that you have a Solaris partition large enough for the root (/) mirror. The Solaris partition should be five cylinders larger than the size needed to hold the root (/) slice.

Make sure that the Solaris partition is active. Otherwise, you will be unable to boot from it.

3. Run the `format(1M)` command on the Solaris partition to create a slice for the root (/) mirror:

```
# format
Searching for disks...done

AVAILABLE DISK SELECTIONS:
       0. c0t0d0 <DEFAULT cyl 2676 alt 2 hd 9 sec 85>
          /eisa/ncrs@8000,0/cmdk@0,0
       1. c1t1d0 <DEFAULT cyl 1865 alt 2 hd 7 sec 80>  ABCDEFG
          /eisa/eha@1000,0/cmdk@1,0
       2. c1t2d0 <DEFAULT cyl 1461 alt 2 hd 9 sec 64>
          /eisa/eha@1000,0/cmdk@2,0
       3. c1t3d0 <DEFAULT cyl 1461 alt 2 hd 9 sec 64>
          /eisa/eha@1000,0/cmdk@3,0
       4. c1t4d0 <DEFAULT cyl 1865 alt 2 hd 7 sec 80>
          /eisa/eha@1000,0/cmdk@4,0
Specify disk (enter its number): 1

selecting c1t1d0: ABCDEFG
[disk formatted]

FORMAT MENU:
   disk      - select a disk
   type     - select (define) a disk type
   partition     - select (define) a partition table
   current     - describe the current disk
   format     - format and analyze the disk
   fdisk     - run the fdisk program
   repair     - repair a defective sector
   label     - write label to the disk
   analyze     - surface analysis
   defect     - defect list management
   backup     - search for backup labels
   verify     - read and display labels
   save     - save new disk/partition definitions
   inquiry     - show vendor, product and revision
   volname     - set 8-character volume name
   quit
```

4. Select `partition` **to define a partition:**

```
format> partition

PARTITION MENU:
        0      - change ‘0’ partition
        1      - change ‘1’ partition
        2      - change ‘2’ partition
        3      - change ‘3’ partition
        4      - change ‘4’ partition
        5      - change ‘5’ partition
        6      - change ‘6’ partition
        7      - change ‘7’ partition
        select - select a predefined table
```

(continued)

```
        modify - modify a predefined partition table
        name   - name the current table
        print  - display the current table
        label  - write partition map and label to the disk
        quit
partition> 0
Part    Tag    Flag   Cylinders       Size    Blocks
  0   unassigned   wm      0              0    (0/0/0)

Enter partition id tag [unassigned]: root
Enter partition permission flags [wm]: wm
Enter new starting cyl [0]: 4
Enter partition size [0b, 0c, 0.00mb]: 400mb
partition> label
Ready to label disk, continue? y
partition>
```

5. Exit from the `partition` **menu and the** `format(1M)` **program by typing the** `quit` **command twice.**

```
partition> quit

FORMAT MENU:
        disk       - select a disk
        type       - select (define) a disk type
        partition  - select (define) a partition table
        current    - describe the current disk
        format     - format and analyze the disk
        repair     - repair a defective sector
        label      - write label to the disk
        analyze    - surface analysis
        defect     - defect list management
        backup     - search for backup labels
        verify     - read and display labels
        save       - save new disk/partition definitions
        inquiry    - show vendor, product and revision
        volname    - set 8-character volume name
        quit
format> quit
```

Note the following important information about the Solaris root (`/`) partition:

- Its I.D. tag must be "`root`"
- Its size must be greater than or equal to the size of the original `root` partition

- It should not use cylinders 0-2

Install Boot Information

Use the `installboot` command to install the boot information on the alternate boot disk:

```
# installboot /usr/lib/fs/ufs/pboot \ /usr/lib/fs/ufs/bootblk \ /dev/rdsk/c0t1d0s0
```

Example — x86: Creating a Mirror From root (/)

This example assumes that a Solaris partition has been created and installed with the boot information using the tasks above.

```
# metainit -f d10 1 1 c0t0d0s0
d10:Concat/Stripe is setup
# metainit d20 1 1 c1t0d0s0
d20: Concat/Stripe is setup
# metainit d0 -m d10
d10: Mirror is setup
# metaroot d0
# lockfs -fa
# reboot
...
# metattach d0 d20
d0: Submirror d20 is attached
# ls -l /dev/rdsk/c1t0d0s0
lrwxrwxrwx 1  root root  55 Mar 5 12:54  /dev/rdsk/c1t0d0s0 -> ../.
devices/eisa/eha@1000,0/cmdk@1,0:a
```

The –`f` option forces the creation of the first concatenation, `d10`, which contains the mounted file system root (/) on `/dev/dsk/c0t0d0s0`. The second concatenation, `d20`, is created from `/dev/dsk/c1t0d0s0`. (This slice must be the same size or greater than that of `d10`.) The `metainit` command with the –`m` option creates the one-way mirror `d0` using the concatenation containing root (/). Next, the `metaroot` command edits the `/etc/vfstab` and `/etc/system` files so that the system may be booted with the root file system (/) on a metadevice. After a reboot, the submirror `d20` is attached to the mirror, causing a mirror resync. (The system verifies that the concatenations and the mirror are set up, and that submirror `d20` is attached.) Using the `ls -l` command on the root raw device determines the path to the alternate root device in case the system needs to be booted from it.

Creating RAID5 Metadevices

This section describes how to create RAID5 metadevices.

Preliminary Information for Creating RAID5 Metadevices

- A *RAID5 metadevice* uses storage capacity equivalent to one slice in the metadevice to store redundant information about user data stored on the remainder of the RAID5 metadevice's slices. The redundant information is distributed across all slices in the metadevice. Like a mirror, a RAID5 metadevice increases data availability, but with a minimum of cost in terms of hardware.
- The system must contain at least three state database replicas before you can create RAID5 metadevices.
- A RAID5 metadevice can only handle a single slice failure.
- Follow the 20-percent rule when creating a RAID5 metadevice: because of the complexity of parity calculations, metadevices with greater than about 20 percent writes should probably not be RAID5 metadevices. If data redundancy is needed, consider mirroring.
- There are drawbacks to a slice-heavy RAID5 metadevice: the more slices a RAID5 metadevice contains, the longer read and write operations will take if a slice fails.
- A RAID5 metadevice must consist of at least three slices.
- A RAID5 metadevice can be grown by concatenating additional slices to the metadevice. The new slices do not store parity information, however they are parity protected. The resulting RAID5 metadevice continues to handle a single slice failure.
- The interlace value is key to RAID5 performance. It is configurable at the time the metadevice is created; thereafter, the value cannot be modified. The default interlace value is 16 Kbytes. This is reasonable for most applications.
- Use the same size disk slices. Creating a RAID5 metadevice from different size slices results in unused disk space in the metadevice.
- Do not create a RAID5 metadevice from a slice that contains an existing file system. Doing so will erase the data during the RAID5 initialization process.
- RAID5 metadevices cannot be striped, concatenated, or mirrored.

▼ How to Create a RAID5 Metadevice (DiskSuite Tool)

1. **Make sure you have met the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17) and have read the preliminary information (“Preliminary Information for Creating RAID5 Metadevices” on page 47).**

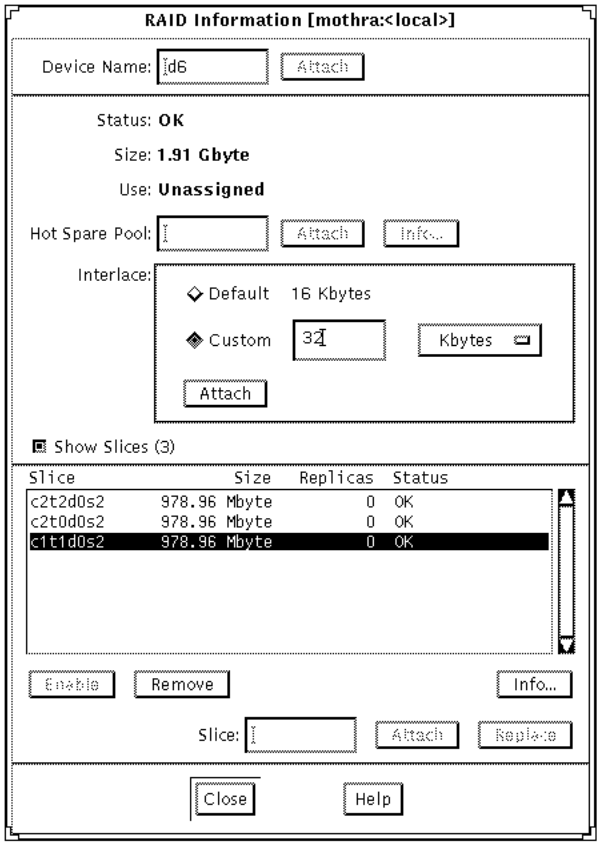
2. **Click the RAID5 template.**

   An unassigned and uncommitted RAID5 object appears on the canvas. The metadevice name is automatically assigned.

3. **[Optional] Change the default metadevice name.**

4. **[Optional] Change the default interlace size of 16 Kbytes.**

   Display the object’s pop-up menu and choose Info. The RAID Information window for this RAID5 metadevice appears.



5. **To change the interlace value, click Custom and type the new value in the field next to Custom. This value can be in Kbytes, Mbytes, or sectors, and is configurable via the drop-down menu beside this field. Click Attach to set the value, then click Close.**

   See *Solstice DiskSuite 4.2 Reference Guide* for more information on setting the interlace value.

6. **Click Slices to open the Slice Browser.**

7. **Select the slices and drag them to the top of the RAID5 object.**

   Use Control-click to select multiple slices. You need at least three slices.

8. **Click the top rectangle of the RAID5 object then click Commit.**

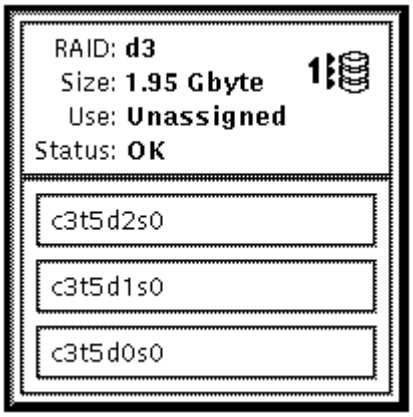
   DiskSuite starts initializing the RAID5 metadevice.

9. **To verify that the RAID5 metadevice was committed, display the Configuration Log.**

Note - You must wait for the initialization to finish before you can use the RAID5 metadevice.

Example

This example shows a RAID5 metadevice. Notice that the size displayed is 1.95 GBytes even though the metadevice is made of three 999.63 MByte slices. (One slice's worth of storage capacity is used for parity.)



Where to Go From Here

To prepare the newly created RAID5 metadevice for a file system, refer to “How to Create a File System on a Metadevice (File System Manager)” on page 73. An application, such as a database, that uses the raw metadevice must have its own way of recognizing the metadevice.

To associate a hot spare pool with a RAID5 metadevice, refer to “How to Associate a Hot Spare Pool (DiskSuite Tool)” on page 61.

▼ How to Create a RAID5 Metadevice (Command Line)

After checking the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17), and the preliminary information (“Preliminary Information for Creating RAID5 Metadevices” on page 47), use the `metainit(1M)` command to create the RAID5 metadevice. For more information, refer to the `metainit(1M)` man page.

Example — Creating a RAID5 Metadevice of Three Slices

```
# metainit d45 -r c2t3d0s2 c3t0d0s2 c4t0d0s2
d45: RAID is setup
```

The RAID5 metadevice `d45` is created with the `–r` option from three slices. Because no interlace is specified, `d45` uses the default of 16 Kbytes. The system verifies that the RAID5 metadevice has been set up, and begins initializing the metadevice.

Note - You must wait for the initialization to finish before you can use the RAID5 metadevice.

Where to Go From Here

To prepare the newly created RAID5 metadevice for a file system, refer to “How to Create a File System on a Metadevice (Command Line)” on page 74. An application, such as a database, that uses the raw metadevice must have its own way of recognizing the metadevice.

To associate a hot spare pool with a RAID5 metadevice, refer to “How to Associate a Hot Spare Pool (Command Line)” on page 62.

Creating Trans Metadevices

This section describes how to create trans metadevices (UFS logging).

Preliminary Information for Creating Trans Metadevices

- A *trans metadevice* enables UFS logging, which is the process of recording UFS updates in a log before the updates are applied to the UNIX file system. A trans metadevice can increase overall file system availability after reboot, because it reduces the amount of time `fsck(1M)` has to run when the system reboots.
- The system must contain at least three state database replicas before you can create a trans metadevice.
- The trans metadevice normally has two devices: the *master device* and the *logging device*. The master contains the file system that is being logged. The logging device contains the log and can be shared by several file systems. It is a sequence of records, each of which describes a change to a file system. Both the master device and the logging device can be a slice or a metadevice.
- Though logs can be shared between file systems, heavily-used file systems should have their own logging device.
- Small file systems with mostly read operations probably do not need to be logged.
- Any UFS, except root (`/`), can be logged.
- Even if you don't have an available slice for a logging device, you can still set up a trans metadevice without a logging device. This is useful if you plan to enable logging on exported file systems, but do not have an available slice for the logging device at this time.
- Before creating trans metadevices, identify the slices or metadevice to be used as the master devices and logging devices.
- Avoid placing logs on heavily-used disks.
- Do not use a RAID5 metadevice as a logging device. Instead, use a mirror for data redundancy.
- Logs (the logging device) can be placed on a slice that already contains a state database replica.
- Plan on using one megabyte of log space as a minimum, and an additional one megabyte of log space per 100 megabytes of file system data, up to a maximum log size of 64 Mbytes. Logs greater than 64 Mbytes waste space.
- The master devices and logging devices of the same trans metadevice should be located on separate drives and possibly separate controllers.

Caution - Mirroring logging devices is strongly recommended. Losing the data in a logging device because of device errors can leave a file system in an inconsistent state which `fsck(1M)` may not be able to fix without user intervention. Using a mirror for the master device is a good idea to ensure data redundancy.

▼ How to Create a Trans Metadevice for a File System That Can Be Unmounted (DiskSuite Tool)

Before beginning, identify the slice or metadevice that contains the file system. You'll need this when creating the master device. Also, decide if you want to mirror the logging device. If so, use "How to Create a Trans Metadevice Using Mirrors (DiskSuite Tool)" on page 57. If you are mirroring the logging device, it is a good idea that the master device be a mirror also.

You can create a file system on the trans metadevice later if the master device doesn't already have a file system.

To create a trans metadevice for a file system, such as `/usr`, that cannot be unmounted during normal system operation, refer to "How to Create a Trans Metadevice for a File System That Cannot Be Unmounted (DiskSuite Tool)" on page 55.

1. **Make sure you have met the prerequisites ("Prerequisites for Creating DiskSuite Objects" on page 17) and have read the preliminary information ("Preliminary Information for Creating Trans Metadevices" on page 51).**

2. **Click the trans metadevice template.**

   An unassigned and uncommitted Trans Metadevice object appears on the canvas. The metadevice name is automatically assigned.

3. **[Optional] Change the default metadevice name.**

   Display the object's pop-up menu and choose Info. Type the new metadevice name in the Device Name field and click Attach. Then click Close.

4. **Drag the slice or metadevice that will contain the master device to the master rectangle in the trans metadevice template.**

   When you can drag a slice from the Slice Browser, or drag a metadevice from the Objects list, a warning dialog box is displayed. Click Continue.

   Note - If an entry exists in the `/etc/vfstab` file for the file system, and the file system is currently mounted, DiskSuite Tool automatically updates it to use the trans metadevice's name.

5. **Select the slice or metadevice that will contain the logging device, and drag it to the log rectangle of the trans metadevice template.**

   A Warning dialog box appears if the logging device is not mirrored.

6. **Click the top rectangle of the Trans Metadevice object then click Commit.**

7. **Unmount then remount the file system on the trans metadevice.**

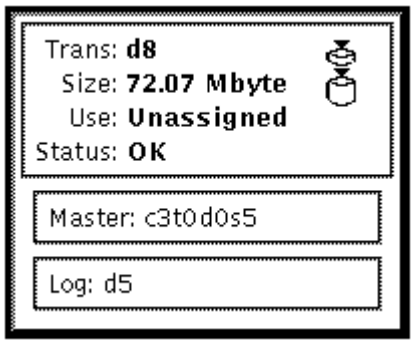
   Logging becomes effective for the file system when you remount the system. On subsequent reboots, instead of checking the file system, `fsck(1M)` displays this message:

   ```
   /dev/md/rdsk/dx: is logging.
   ```

8. **To verify that the trans metadevice was committed, display the Configuration Log.**

Example — Committed Trans Metadevice Object

This example shows a committed trans metadevice, `d8`, consisting of slice `c3t0d0s5` for the master device, and a mirror, `d5`, for the logging device. Notice how the master and logging devices are displayed within the trans object.



▼ How to Create a Trans Metadevice for a File System That Can Be Unmounted (Command Line)

After checking the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17), and the preliminary information (“Preliminary Information for Creating Trans Metadevices” on page 51), create the trans metadevice with the `metainit(1M)` command. Refer to the `metainit(1M)` man page for more information.

Example — Creating a Trans Metadevice With a Slice for the Master Device

```
# umount /home1
# metainit d63 -t c0t2d0s2 c2t2d0s1
d63: Trans is setup
(Edit the /etc/vfstab file so that the file system references
the trans metadevice)
# mount /home1
```

Slice `/dev/dsk/c0t2d0s2` contains a file system mounted on `/home1`. The slice to contain the logging device is `/dev/dsk/c2t2d0s1`. First, the file system is unmounted. The `metainit` command with the `–t` option creates the trans metadevice, `d63`.

Next, the `/etc/vfstab` file must be edited to change the entry for the file system to reference the trans metadevice. For example, the following line:

```
/dev/dsk/c0t2d0s2 /dev/rdsk/c0t2d0s2 /home1 ufs 2 yes -
```

should be changed to:

```
/dev/md/dsk/d63 /dev/md/rdsk/d63 /home1 ufs 2 yes -
```

Logging becomes effective for the file system when it is remounted.

On subsequent reboots, instead of checking the file system, `fsck(1M)` displays a logging message for the trans metadevice:

```
# reboot
...
/dev/md/rdsk/d63: is logging
```

Example — Creating a Trans Metadevice With a Stripe for the Master Device

```
# umount /home2
# metainit d40 -t d2 c1t2d0s0
d40: Trans is setup
(Edit the /etc/vfstab file so that the file system references
the trans metadevice)
# mount /home2
```

Stripe `d2` contains a file system mounted on `/home2`. The slice to contain the logging device is `/dev/dsk/c1t2d0s0`. First, the file system is unmounted. The `metainit` command with the `–t` option creates the trans metadevice, `d40`.

Next, the `/etc/vfstab` file must be edited to change the entry for the file system to reference the trans metadevice. For example, the following line:

```
/dev/md/dsk/d2 /dev/md/rdsk/d2 /home2 ufs 2 yes -
```

should be changed to:

```
/dev/md/dsk/d40 /dev/md/rdsk/d40 /home2 ufs 2 yes -
```

Logging becomes effective for the file system when it is remounted.

On subsequent reboots, instead of checking the file system, `fsck(1M)` displays a logging message for the metadevice:

```
# reboot
...
/dev/md/rdsk/d40: is logging
```

▼ How to Create a Trans Metadevice for a File System That Cannot Be Unmounted (DiskSuite Tool)

Use this procedure to log a file system, such as `/usr`, that cannot be unmounted during normal system operation.

1. **Make sure you have met the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17), and have read the preliminary information (“Preliminary Information for Creating Trans Metadevices” on page 51).**

2. **Click the trans metadevice template.**

   An unassigned and uncommitted Trans Metadevice object appears on the canvas. The metadevice name is automatically assigned.

3. **[Optional] Change the default metadevice name.**

   Display the object’s pop-up menu and choose Info. Type the new metadevice name in the Device Name field and click Attach. Then click Close.

4. **Click Slices to open the Slice Browser.**

5. **Select the slice or metadevice that contains the file system to be logged, and drag it to the master rectangle in the Trans Metadevice object.**

   This must be the slice or metadevice that contains the file system.

6. **Confirm the device that will be the master.**

Click the Continue button on the dialog box that appears.

7. **Select the slice or metadevice that will contain the logging device, and drag it to the log rectangle of the Trans Metadevice object.**

8. **Confirm the slice that will be the log.**
   Click the Continue button on the dialog box that appears.

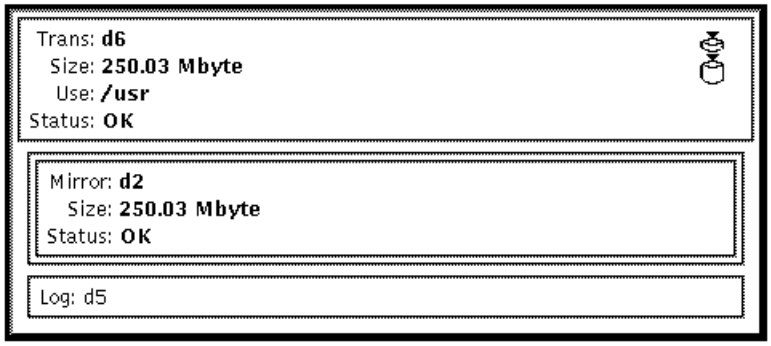
9. **Click the top rectangle of the Trans Metadevice object then click Commit.**
   Click the Really Commit button on the confirmation dialog box.

10. **To verify that the trans metadevice was committed, display the Configuration Log.**

11. **Reboot.**
    After the reboot, logging becomes effective for the file system.

Example — Committed Trans Object for `/usr`

This example shows a trans metadevice that contains the `/usr` file system.



▼ How to Create a Trans Metadevice for a File System That Cannot Be Unmounted (Command Line)

After checking the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17), and the preliminary information (“Preliminary Information for Creating Trans Metadevices” on page 51), use this procedure to log a file system, such as `/usr`, that cannot be unmounted during normal system operation. Refer to the `metainit(1M)` man page for more information.

Example — Creating a Trans Metadevice for /usr

```
# metainit -f d20 -t c0t3d0s6 c1t2d0s1
d20: Trans is setup
(Edit the /etc/vfstab file so that the file system references
the trans metadevice)
# reboot
```

Slice /dev/dsk/c0t3d0s6 contains the /usr file system. The slice to contain the logging device is /dev/dsk/c1t2d0s1. Because /usr cannot be unmounted, the metainit command is run with the –f option to force the creation of the trans device, d20. Next, the line in the /etc/vfstab file that mounts the file system must be changed to reference the trans metadevice. For example, the following line:

```
/dev/dsk/c0t3d0s6 /dev/rdsk/c0t3d0s6 /usr ufs 1 no -
```

should be changed to:

```
/dev/md/dsk/d20 /dev/md/rdsk/d20 /usr ufs 1 no -
```

Logging becomes effective for the file system when the system is rebooted.

▼ How to Create a Trans Metadevice Using Mirrors (DiskSuite Tool)

You can increase data availability of a trans metadevice by using mirrors for the master and logging devices. Failure to mirror the logging device could result in significant data loss if the logging slice experiences errors. If you are mirroring the logging device, it is a good idea that the master device be a mirror also.

1. **Make sure you have met the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17), and have read the preliminary information (“Preliminary Information for Creating Trans Metadevices” on page 51).**

2. **Refer to the tasks on creating mirrors (“How to Create a Mirror From Unused Slices (DiskSuite Tool)” on page 28to “How to Create a Mirror From a File System That Cannot Be Unmounted (Command Line)” on page 38).**

   Create two mirrors, one for the master device, and one for the logging device. The mirror for the master device must contain the file system.

3. **Refer to “How to Create a Trans Metadevice for a File System That Can Be Unmounted (DiskSuite Tool)” on page 52.**

   Use the mirrors created in step 2 for the master and logging devices.

▼ How to Create a Trans Metadevice Using Mirrors (Command Line)

After checking the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17), and the preliminary information (“Preliminary Information for Creating Trans Metadevices” on page 51), use the `metainit(1M)` to create the trans metadevice.

Example — Creating a Trans Metadevice Using Mirrors

```
# umount /home1
# metainit d64 -t d30 d12
d64: Trans is setup
(Edit the /etc/vfstab file so that the file system references
the trans metadevice)
# mount /home1
```

Mirror `d30` contains a file system mounted on `/home1`. The mirror to contain the logging device is `d12`. First, the file system is unmounted. The `metainit` command with the `–t` option creates the trans metadevice, `d64`.

Next, the line in the `/etc/vfstab` file that mounts the file system must be changed to reference the trans metadevice. For example, the following line:

```
/dev/md/dsk/d30 /dev/md/rdsk/d30 /home1 ufs 2 yes -
```

should be changed to:

```
/dev/md/dsk/d64 /dev/md/rdsk/d64 /home1 ufs 2 yes -
```

Logging becomes effective for the file system when the file system is remounted.

On subsequent file system remounts or system reboots, instead of checking the file system, `fsck(1M)` displays a logging message for the metadevice:

```
# reboot
...
/dev/md/rdsk/d64: is logging
```

Creating Hot Spare Pools

This section describes how to create hot spare pools, add slices to hot spare pools, and assign hot spare pools to metadevices.

Preliminary Information for Creating 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.
- A hot spare must be a slice; it cannot be a metadevice.
- Hot spare pools are used by mirrors and RAID5 metadevices. In the case of a mirror, the hot spare pool is associated with the submirrors.
- A hot spare slice can be assigned to one or more hot spare pools.
- A hot spare pool can be associated with multiple submirrors or RAID5 metadevices. However, a submirror or RAID5 metadevice can only be associated with one hot spare pool.
- Replacement is based on a first fit for the failed slice, and follows the order specified in the hot spare pool.
- Hot spares are not intended to remain a permanent part of your configuration. They need to be replaced with repaired or new slices.
- Hot spares cannot contain state database replicas.
- Ideally, slices added to the hot spare pool should be attached to different controllers. This ensures availability of data due to controller error or failure.
- Do not associate hot spares of the wrong size with submirrors or RAID5 metadevices.
- Do not assign a hot spare pool to a submirror in a one-way mirror.
- Do not have all hot spares within a hot spare pool marked as in-use. If this happens, either add additional hot spares to the hot spare pool, or repair the slices that have been hot spare replaced.
- Hot spare pools may be allocated, deallocated, or reassigned at any time unless a slice in the hot spare pool is being used to replace an errored slice in a metadevice it is associated with.

▼ How to Create a Hot Spare Pool (DiskSuite Tool)

You can create an empty hot spare pool, and add hot spares later if necessary. If you choose to do so, skip Step 4 on page 60 and Step 5 on page 60 in this task.

1. **Make sure you have met the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17) and have read the preliminary information (“Preliminary Information for Creating Hot Spare Pools” on page 59).**

2. **Click the Hot Spare Pool template.**

   An unassigned and uncommitted Hot Spare Pool object appears on the canvas. The metadevice name is automatically assigned.

3. **[Optional] Change the default hot spare pool name.**

   Display the object’s pop-up menu and choose Info. Type the new metadevice name in the Device Name field and click Attach. Then click Close.

4. **Click Slices to open the Slice Browser.**

5. **Select the slices and drag them into the Hot Spare Pool object.**

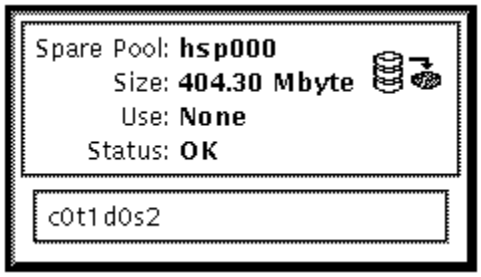
   Use Control-click to select multiple slices.

6. **Click the top rectangle of the Hot Spare Pool object then click Commit.**

7. **To verify that the hot spare pool was committed, display the Configuration Log.**

Example — Committed Hot Spare Pool Object

This example shows a committed hot spare pool object, `hsp000`, consisting of slice `c0t1d0s2`.



Where to Go From Here

To add hot spares to the hot spare pool, refer to “How to Add a Hot Spare Slice to a Hot Spare Pool (DiskSuite Tool)” on page 63. After creating the hot spare pool, you need to associate it with a submirror or RAID5 metadevice. See “How to Associate a Hot Spare Pool (DiskSuite Tool)” on page 61.

▼ How to Create a Hot Spare Pool (Command Line)

After checking the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17), and the preliminary information (“Preliminary Information for Creating Hot Spare Pools” on page 59), use the `metainit(1M)` command to create a hot spare pool. Refer to the `metainit(1M)` man page for more information.

Example — Creating a Hot Spare Pool

```
# metainit hsp001 c2t2d0s2 c3t2d0s2
hsp001: Hotspare pool is setup
```

The hot spare pool `hsp001` contains two disks as the hot spares. The system verifies that the hot spare pool has been set up.

Where to Go From Here

To add more hot spares to the hot spare pool, refer to “How to Add a Hot Spare Slice to a Hot Spare Pool (Command Line)” on page 64. After creating the hot spare pool, you need to associate it with a submirror or RAID5 metadevice. See “How to Associate a Hot Spare Pool (Command Line)” on page 62.

▼ How to Associate a Hot Spare Pool (DiskSuite Tool)

A hot spare pool must exist before it can be associated with a submirror or RAID5 metadevice.

1. **Make sure you have met the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17) and have read the preliminary information (“Preliminary Information for Creating Hot Spare Pools” on page 59).**

2. **Double-click the mirror or RAID5 metadevice in the Objects list.**

   The object appears on the canvas.

3. **Choose Hot Spare Pools from the Browse menu then choose a hot spare pool object from the list.**

   The hot spare pool should have slices the same size or larger than the submirrors in the mirror, or the slices in the RAID5 metadevice. Ideally, the Hot Spare Pool contains hot spares that are on a different controller than the slices in the metadevice.

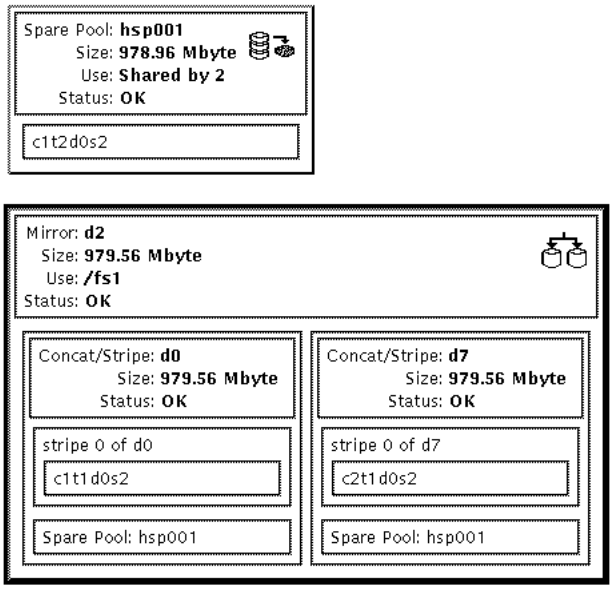
4. **Drag the Hot Spare Pool object to the top of the metadevice object.**

5. **Click the top rectangle of the metadevice object then click Commit.**

6. **To verify that the hot spare pool was associated, display the Configuration Log.**

Example — Hot Spare Pool Associations

This example shows a hot spare pool associated with two submirrors.



▼ How to Associate a Hot Spare Pool (Command Line)

After checking the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17), and the preliminary information (“Preliminary Information for Creating Hot Spare Pools” on page 59), use the `metaparam(1M)` command to associate a hot spare pool. Refer to the `metaparam(1M)` man page for more information.

Example — Associating a Hot Spare Pool with Submirrors

```
# metaparam -h hsp100 d10
# metaparam -h hsp100 d11
# metastat d0
d0: Mirror
    Submirror 0: d10
      State: Okay
    Submirror 1: d11
      State: Okay
...

d10: Submirror of d0
    State: Okay
    Hot spare pool: hsp100
...


d11: Submirror of d0
    State: Okay
    Hot spare pool: hsp100
...
```

The –h option associates a hot spare pool, `hsp100`, with two submirrors, `d10` and `d11`, of a mirror, `d0`. The `metastat` command shows that the hot spare pool is associated with the submirrors.

Example — Associating a Hot Spare Pool with a RAID5 Metadevice

```
# metaparam -h hsp001 d10
# metastat d10
d10: RAID
    State: Okay
    Hot spare pool: hsp001
...
```

The –h option associates a hot spare pool named `hsp001` with a RAID5 metadevice named `d10`. The `metastat` command shows that the hot spare pool is associated with the RAID5 metadevice.

▼ How to Add a Hot Spare Slice to a Hot Spare Pool (DiskSuite Tool)

You can add a slice to one or more hot spare pools. When a hot spare is added, the existing order of the hot spares is preserved. The new hot spare is added at the end of the list of hot spares in the hot spare pool that is specified.

1. **Make sure you have met the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17) and have read the preliminary information (“Preliminary Information for Creating Hot Spare Pools” on page 59).**

2. **Double-click an existing Hot Spare Pool object in the Objects list.**
   The object appears on the canvas.

3. **Click Slices to open the Slice Browser.**

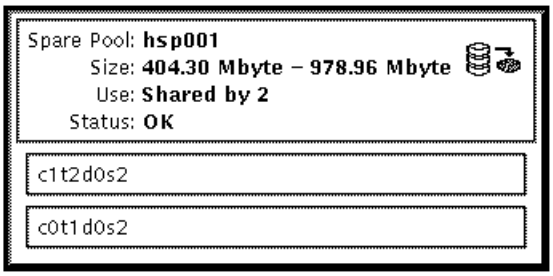
4. **Select a slice then drag it into the Hot Spare Pool object.**
   If the hot spare pool already contains slices, locate a slice of the same or greater size than the slice(s) in the hot spare pool.

5. **Click the top rectangle of the Hot Spare Pool object then click Commit.**

6. **To verify that the hot spare pool was committed, display the Configuration Log.**

Example — Hot Spare Pool with Two Slices

This example shows a hot spare pool that initially consisted of one slice, `c1t2d0s2`. Slice `c0t1d0s2` was added and the hot spare pool was committed.



▼ How to Add a Hot Spare Slice to a Hot Spare Pool (Command Line)

After checking the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17), and the preliminary information (“Preliminary Information for Creating Hot Spare Pools” on page 59), use the `metahs(1M)` command to add a slice to a hot spare pool. Refer to the `metahs(1M)` man page for more information.

Note - The added hot spare follows whatever hot spares already exist in the hot spare pool.

Example — Adding a Hot Spare Slice to One Hot Spare Pool

```
# metahs -a hsp001 /dev/dsk/c3t0d0s2
hsp001: Hotspare is added
```

The –a option adds the slice /dev/dsk/c3t0d0s2 to hot spare pool hsp001. The system verifies that the slice has been added to the hot spare pool.

Example — Adding a Hot Spare Slice to All Hot Spare Pools

```
# metahs -a -all /dev/dsk/c3t0d0s2
hsp001: Hotspare is added
hsp002: Hotspare is added
hsp003: Hotspare is added
```

The –a and –all options add the slice /dev/dsk/c3t0d0s2 to all hot spare pools configured on the system. The system verifies that the slice has been added to all hot spare pools.

▼ How to Change the Associated Hot Spare Pool (DiskSuite Tool)

You can change a hot spare pool's association at any time as long as none of its slices are currently in use as hot spares.

1. **Make sure you have met the prerequisites ("Prerequisites for Creating DiskSuite Objects" on page 17) and have read the preliminary information ("Preliminary Information for Creating Hot Spare Pools" on page 59).**

2. **Double-click the Mirror or RAID5 object with the hot spare pool association to be changed in the Objects list.**

   The object appears on the canvas.

3. **Double-click the Hot Spare Pool object in the Objects list to use as the replacement.**

   The object appears on the canvas.

4. **Drag the Hot Spare Pool object onto the rectangle in the submirror or RAID5 object containing the hot spare pool to be replaced.**

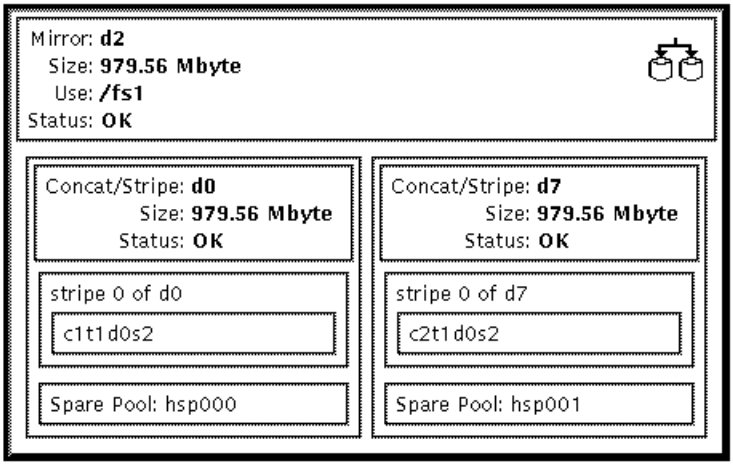
5. **Click inside the top rectangle of the object then click Commit.**

6. **To verify that the hot spare pool was committed, display the Configuration Log.**

Note - To avoid data fabrication, DiskSuite does not allow hot sparing of a metadevice if it contains slices in the “Last Erred” state. For more information, refer to “Overview of Replacing and Enabling Slices in Mirrors and RAID5 Metadevices” on page 105.

Example — Mirror Object with Hot Spare Pool Associations

This example shows a mirror that started with hot spare pool `hsp000` associated with each submirror. Submirror `d7` is now associated with `hsp001`.



▼ How to Change the Associated Hot Spare Pool (Command Line)

After checking the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17), and the preliminary information (“Preliminary Information for Creating Hot Spare Pools” on page 59), use the `metaparam(1M)` command to change the metadevice’s associated hot spare pool. For more information, refer to the `metaparam(1M)` man page.

Example — Changing the Hot Spare Pool Association

```
# metastat d4
d4: RAID
    State: Okay
```

(continued)

```
    Hot spare pool: hsp001
...
# metaparam -h hsp002 d4
# metastat d4
d4: RAID
    State: Okay
    Hot spare pool: hsp002
...
```

In this example, the hot spare pool `hsp001` is currently associated with a RAID5 metadevice named `d4`. The hot spare pool association is changed to `hsp002`. The `metastat` command shows the hot spare pool association before and after.

Creating Disksets

This section describes how to create disksets, and populate them with hosts and disk drives.

Note - Currently, disksets are only supported on SPARCstorage Array disks.

Preliminary Information for Creating Disksets

- A *diskset* is a set of shared disk drives containing DiskSuite objects that can be shared exclusively (but not concurrently) by one or two hosts. Disksets are enablers for host fail-over scenarios.
- DiskSuite must be installed on each host that will be connected to the diskset.
- There is one metadevice state database per shared diskset and one on the “local” diskset. Each host must have its local metadevice state database set up before you can create disksets.
- Unlike local metadevice administration, it is not necessary to create or delete state database replicas manually on the diskset. DiskSuite tries to balance a reasonable number of replicas across all drives in a diskset.
- Each host in a diskset must have a local diskset that is separate from the shared diskset. The local diskset for a host consists of all drives that are not part of a shared diskset.

- All disks that you plan to share between hosts in the diskset must be connected to each host and must have the same name on each host. Configuring the hardware for use in diskset configuration can be problematic. The disk drives must be symmetric; that is, the shared drives must have the same device number, which implies the same device name / number (controller / target / drive). See "How to Configure Disk Drive Device Names for a Diskset (Command Line)" on page 290.
- You can create a diskset with one host and add the second host later.
- You cannot add a drive that is in use to a diskset. Before adding a drive, make sure it is not currently being used for a file system, database, or any other application.
- Do not add a drive with existing data that you want to preserve to a diskset; the process of adding it to the diskset repartitions the disk, destroying any data.
- When a drive is accepted into a diskset, DiskSuite repartitions it so that the metadevice 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 the 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. If Slice 7 starts at cylinder 0, and is large enough to contain a state database replica, the disk is not repartitioned.
- When drives are added to a diskset, DiskSuite re-balances the state database replicas across the remaining drives. Later, if necessary, you can change the replica layout with the `metadb(1M)` command.
- To create a diskset, `root` must be a member of Group 14, or the `./rhosts` file must contain an entry for the other hostname (on each host).

Note - Disksets must be created and configured using the DiskSuite command line interface. 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.

▼ How to Create a Diskset (Command Line)

The high-level steps in creating a diskset are:

- Adding hosts to a diskset (this creates the diskset)
- Adding drives to the diskset

Before creating a diskset:

- Configure disk drive device names so they have the same name on each host in the diskset.
- Connect drives to be shared to both hosts.
- Configure local metadevice state database replicas on each host.

Note - If you are not familiar with how to configure the same device names for the shared drives in the diskset, refer to "How to Configure Disk Drive Device Names for a Diskset (Command Line)" on page 290.

1. **Make sure you have met the prerequisites ("Prerequisites for Creating DiskSuite Objects" on page 17) and have read the preliminary information ("Preliminary Information for Creating Disksets" on page 67).**

2. **Create the diskset by defining the host(s) that have access to the devices.**

```
# metaset -s setname -a -h host...
```

In this command,

–s *setname*	Specifies the name of a diskset on which `metaset(1M)` will work.
–a	Adds hosts to the named diskset. DiskSuite supports a maximum of two hosts per diskset.
–h *host...*	Specifies one or more hosts to be added to a diskset. Adding the first host creates the set. The second host can be added later, but it is not accepted if all the drives within the set cannot be found on the specified *host*. *host* is the same name found in `/etc/nodename`.

3. **Check the status of the new diskset with the `metaset(1M)` command.**

```
# metaset
```

Example — Creating Two Disksets

```
red# metaset -s relo-red -a -h red blue
red# metaset -s relo-blue -a -h red blue
red# metaset
Set name = relo-red, Set number = 1

Host                Owner
  red
  blue

Set name = relo-blue, Set number = 2
```

(continued)

```
Host                Owner
  red
  blue
```

In this example, you create two shared disksets, `relo-red` and `relo-blue`, from the host `red`. The host names are `red` and `blue`, each with their own local disksets. The `metaset` command shows the status. At this point, neither set has an owner. The host that adds disks to the set will become the owner by default.

▼ How to Add Drives to a Diskset (Command Line)

The following conditions must be satisfied for a drive to be accepted into a diskset:

- It must not be in use within a metadevice, or contain a state database replica.
- It must not be currently mounted, swapped on, or otherwise opened for use by an application.

1. **Make sure you have met the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17) and have read the preliminary information (“Preliminary Information for Creating Disksets” on page 67).**

2. **Add drives to the diskset.**

```
# metaset -s setname -a drivename...
```

In this command,

–s *setname*	Specifies the name of a diskset on which `metaset(1M)` will work.
–a	Adds drives to the named diskset.
drivename...	Specifies the drives to add to the diskset. Drive names are in the form c*x*t*x*d*x*; no “s*x*” slice identifiers are at the end of the name. They need to be the same on all hosts in the diskset.

The first host to add a drive to a diskset becomes the implicit owner of the diskset.

Caution - Do not add a disk with data; the process of adding it to the diskset repartitions the disk, destroying any data.

3. Use the `metaset(1M)` **command to verify the status of the diskset and drives.**

```
# metaset
```

Example — Adding Drives to a Diskset

```
red# metaset -s relo-red -a c1t2d0 c1t3d0 c2t2d0 c2t3d0 c2t4d0 c2t5d0
red# metaset
Set name = relo-red, Set number = 1

Host                Owner
  red                Yes
  blue

Drive               Dbase
  c1t2d0             Yes
  c1t3d0             Yes
  c2t2d0             Yes
  c2t3d0             Yes
  c2t4d0             Yes
  c2t5d0             Yes

Set name = relo-blue, Set number = 2

Host                Owner
  red
  blue
```

In this example, the host names are `red` and `blue`, each with their own local disksets. The two shared disksets are `relo-red` and `relo-blue`. The disks in the set `relo-red` are normally accessed by host `red`, but may be accessed by host `blue` if `red` fails. At this point, no disks have been added to the diskset `relo-blue`.

Creating DiskSuite Objects in a Diskset

After you create a diskset, you can create metadevices and hot spare pools using the drives you added to the diskset. You can use either DiskSuite Tool or the command line utilities.

▼ How to Create a DiskSuite Object in a Diskset (DiskSuite Tool)

To use DiskSuite Tool to create DiskSuite objects in a diskset, make sure you are the diskset owner and enter the command line:

```
# metatool -s diskset_name &
```

Use the DiskSuite Tool tasks in this chapter to create DiskSuite objects, such as mirrors and trans metadevices.

▼ How to Create a DiskSuite Object in a Diskset (Command Line)

After checking the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17), use the `metainit(1M)` command to create DiskSuite objects in a diskset. Refer to the `metainit(1M)` man page for more information.

Example — Creating a Mirror in a Diskset

```
# metainit -s relo-red d51 1 1 /dev/dsk/c0t0d0s2
relo-red/d51: Concat/Stripe is setup
# metainit -s relo-red d52 1 1 /dev/dsk/c1t0d0s2
relo-red/d52: Concat/Stripe is setup
# metainit -s relo-red d50 -m d51
relo-red/d50: mirror is setup
# metattach -s relo-red d50 d52
relo-red/d50: Submirror d52 is attached
```

This example creates a mirror, `d50`, in diskset `relo-red`.

Creating File Systems on Metadevices

This section describes how to create a new file system on a metadevice.

Preliminary Information for Creating File Systems on Metadevices

- You can use either the `newfs(1M)` command or File System Manager to create a file system on a newly created metadevice.
- You can create a file system on a stripe, concatenation, mirror, RAID5, or trans metadevice.

▼ How to Create a File System on a Metadevice (File System Manager)

This task assumes you have installed Solstice Storage Manager on the host you are working with, and that you have launched the Storage Manager application. For more information, see Appendix A.

1. **Make sure you have met the prerequisites (“Prerequisites for Creating DiskSuite Objects” on page 17) and have read the preliminary information (“Preliminary Information for Creating File Systems on Metadevices” on page 73).**

2. **Start File System Manager.**

   If you have appropriately configured DiskSuite Tool, select File System Manager from the DiskSuite Tool “Tools” menu.

   If you have not configured DiskSuite Tool as such, you must first launch Storage Manager from the Solstice Launcher.

   Note - To configure DiskSuite to work with Storage Manager, refer to “How to Enable DiskSuite to Launch Storage Manager (Command Line)” on page 217.

   ---

3. **Within the File System Manager main window, select Create from the Object menu, then select File System to display the Property Book for a new file system.**

4. **Click Device in the Property Viewer portion of the window.**

5. **Select and drag a metadevice from DiskSuite Tool's Metadevice Editor window to the Device Name text field.**

6. **Click Apply.**

   File System Manager begins building the new file system on the metadevice.

▼ How to Create a File System on a Metadevice (Command Line)

After checking the prerequisites ("Prerequisites for Creating DiskSuite Objects" on page 17), and the preliminary information ("Preliminary Information for Creating File Systems on Metadevices" on page 73), use the `newfs(1M)` command to create a new file system on a metadevice. Refer to the `newfs(1M)` man page for more information.

Example — Creating a File System on a Concatenation

```
# newfs /dev/md/rdsk/d3
newfs: construct a new file system /dev/md/rdsk/d3: (y/n)? y
/dev/md/rdsk/d3:        917280 sectors in 1638 cylinders of 7 tracks, 80 sectors
        447.9MB in 103 cyl groups (16 c/g, 4.38MB/g, 2112 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 9072, 18112, 27152, 36192, 45232, 54272, 63312, 71712, 80752, 89792,
 98832, 107872, 116912, 125952, 134992, 143392, 152432, 161472, 170512, 179552,
 188592, 197632, 206672, 215072, 224112, 233152, 242192, 251232, 260272,
 269312, 278352, 286752, 295792, 304832, 313872, 322912, 331952, 340992,
 350032, 358432, 367472, 376512, 385552, 394592, 403632, 412672, 421712,
 430112, 439152, 448192, 457232, 466272, 475312, 484352, 493392, 501792,
 510832, 519872, 528912, 537952, 546992, 556032, 565072, 573472, 582512,
 591552, 600592, 609632, 618672, 627712, 636752, 645152, 654192, 663232,
 672272, 681312, 690352, 699392, 708432, 716832, 725872, 734912, 743952,
 752992, 762032, 771072, 780112, 788512, 797552, 806592, 815632, 824672,
 833712, 842752, 851792, 860192, 869232, 878272, 887312, 896352, 905392,
 914432,
# fsck /dev/md/rdsk/d3
** /dev/md/rdsk/d3
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups
2 files, 9 used, 942654 free (14 frags, 117830 blocks,  0.0% fragmentation)
```

This example creates a file system on a concatenation named `d3`. The `fsck(1M)` command verifies the new file system.

Example — Creating a File System on a Concatenation Within a Diskset

```
# newfs /dev/md/relo-red/rdsk/d33
newfs: construct a new file system /dev/md/relo-red/rdsk/d33: (y/n)? y
/dev/md/relo-red/rdsk/d33:        917280 sectors in 1638 cylinders of 7 tracks,
80 sectors
        447.9MB in 103 cyl groups (16 c/g, 4.38MB/g, 2112 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 9072, 18112, 27152, 36192, 45232, 54272, 63312, 71712, 80752, 89792,
 98832, 107872, 116912, 125952, 134992, 143392, 152432, 161472, 170512, 179552,
 188592, 197632, 206672, 215072, 224112, 233152, 242192, 251232, 260272,
 269312, 278352, 286752, 295792, 304832, 313872, 322912, 331952, 340992,
 350032, 358432, 367472, 376512, 385552, 394592, 403632, 412672, 421712,
 430112, 439152, 448192, 457232, 466272, 475312, 484352, 493392, 501792,
 510832, 519872, 528912, 537952, 546992, 556032, 565072, 573472, 582512,
 591552, 600592, 609632, 618672, 627712, 636752, 645152, 654192, 663232,
 672272, 681312, 690352, 699392, 708432, 716832, 725872, 734912, 743952,
 752992, 762032, 771072, 780112, 788512, 797552, 806592, 815632, 824672,
 833712, 842752, 851792, 860192, 869232, 878272, 887312, 896352, 905392,
 914432,
```

This example creates a file system on a concatenation name `d33` in diskset `relo-red`.

Note - A file system that resides on a metadevice in a diskset cannot be automatically mounted at boot via the `/etc/vfstab` file. 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.

Where to Go From Here

If You Want to Make the File System Available ...	Then ...
Now	Mount the file system with the `mount(1M)` command, specifying the metadevice name as the mount device, such as `/dev/md/dsk/d30`.
Automatically when the system boots	Create or modify the file system's entry in `/etc/vfstab file`, using the metadevice block and raw device names.
As a shared (exported) resource	Create a file system entry in the `/etc/dfs/dfstab` file.

CHAPTER 3

Maintaining DiskSuite Objects

This chapter describes how to maintain DiskSuite objects, both with the DiskSuite Tool graphical interface and with the command line utilities.

Use the following to proceed directly to the section that provides step-by-step instructions using DiskSuite Tool.

- "How to Unmirror a File System (DiskSuite Tool)" on page 141
- "How to Attach a Submirror (DiskSuite Tool)" on page 146
- "How to Detach a Submirror (DiskSuite Tool)" on page 147
- "How to Place a Submirror Offline and Online (DiskSuite Tool)" on page 148

Use the following to proceed directly to the section that provides step-by-step instructions using the command line interface.

- "How to Check the Status of State Database Replicas (Command Line)" on page 83
- "How to Check the Status of Metadevices and Hot Spare Pools (Command Line)" on page 91
- "How to Check the Status of a Diskset (Command Line)" on page 99
- "How to Recreate a Stripe or Concatenation After Slice Failure (Command Line)" on page 104
- "How to Enable a Slice in a Submirror (Command Line)" on page 109
- "How to Replace a Slice in a Submirror (Command Line)" on page 111
- "How to Replace a Submirror (Command Line)" on page 113
- "How to Enable a Slice in a RAID5 Metadevice (Command Line)" on page 114
- "How to Replace a Hot Spare in a Hot Spare Pool (Command Line)" on page 118
- "How to Enable a Hot Spare (Command Line)" on page 120
- "How to Recover a Trans Metadevice With a File System Panic (Command Line)" on page 122
- "How to Recover a Trans Metadevice With Hard Errors (Command Line)" on page 122
- "How to Expand a Slice Containing Existing Data (Command Line)" on page 127
- "How to Expand an Existing Stripe (Command Line)" on page 129
- "How to Expand a Mirror (Command Line)" on page 132
- "How to Expand a RAID5 Metadevice (Command Line)" on page 134
- "How to Expand a Trans Metadevice (Command Line)" on page 136
- "How to Grow a File System (Command Line)" on page 138
- "How to Rename a Metadevice (Command Line)" on page 140
- "How to Unmirror a File System (Command Line)" on page 142
- "How to Attach a Submirror (Command Line)" on page 146
- "How to Detach a Submirror (Command Line)" on page 148
- "How to Place a Submirror Offline and Online (Command Line)" on page 150
- "How to Reserve a Diskset (Command Line)" on page 151
- "How to Release a Diskset (Command Line)" on page 152

Overview of Maintaining DiskSuite Objects

This chapter describes maintenance tasks you may need to perform after creating a DiskSuite object, such as:

- Checking object status
- Replacing or repairing an errored object
- Expanding a metadevice and growing a file system
- Renaming an object
- Unmirroring a file system

For general information on DiskSuite, see *Solstice DiskSuite 4.2 Reference Guide*. Refer to Chapter 2 for information on creating metadevices and hot spare pools.

Prerequisites for Maintaining DiskSuite Objects

Here are the prerequisites for the steps in this chapter:

- Make sure you have a current backup of all data.
- Make sure you have root privilege.
- If using the graphical user interface, start DiskSuite Tool.

  To work with "local" metadevices (metadevices not in a diskset configuration), type:

```
# metatool &
```

To work with metadevices in a diskset, make sure you are the diskset owner and type:

```
# metatool -s diskset_name &
```

Checking Status of DiskSuite Objects

This section contains the tasks that check the status of DiskSuite objects, including state database replicas, metadevices, hot spares, and disksets. Check an object's status before performing any of the tasks below, which are described later in this chapter:

- Replacing an errored slice in a metadevice
- Repairing a trans metadevice with a hard error
- Expanding a metadevice
- Renaming a metadevice
- Unmirroring a mirrored file system

Using DiskSuite Tool to Check Status

DiskSuite Tool gives you three ways to check the status of a DiskSuite object:

- Double-clicking the object in the Objects list, which displays it on the canvas, and checking the object's status field.
- Viewing the object's Information window by double-clicking the object on the canvas, or by displaying the object's pop-up menu and selecting Info.
- Viewing the DiskSuite Problem List by selecting Problem List from the Browse menu. Existing problems (if any) are described in the list.

Using the Command Line to Check Status

Two commands, `metadb(1M)` and `metastat(1M)`, check the status of DiskSuite objects.

- To check state database replicas, use the `metadb(1M)` command.

```
# metadb [-s setname] [-i]
```

In this command,

-s *setname*	Specifies the name of the diskset on which the `metadb` command will work.
-i	Displays a legend that describes the status flags.

- To check metadevices and hot spare pools, use the `metastat(1M)` command.

```
# metastat [-s setname] [-p] [-t] [object]
```

In this command,

`-s` *setname*	Specifies the name of the diskset on which `metastat` will work.
`-p`	Displays the status in a format similar to that of the `md.tab` file.
`-t`	Displays the time of the last state change.
object	Is the name of the stripe, concatenation, concatenated stripe, mirror, RAID5 metadevice, trans metadevice, or hot spare pool. If you omit a specific object, the status of all metadevices and hot spare pools is displayed.

▼ How to Check the Status of State Database Replicas (DiskSuite Tool)

1. **Make sure you have met the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79).**

2. **Check the status of the MetaDB object by displaying the object’s Information window.**

   For other ways of checking status, see “Using DiskSuite Tool to Check Status” on page 80.

3. **Refer to Table 3–1 for explanations of the Status fields of the MetaDB object, and possible actions to take.**

TABLE 3–1 MetaDB Object Status Keywords

Keyword	Meaning	Action
OK	The MetaDB object (state database) has no errors and is functioning correctly.	None.
Attention	The number of good state database replicas is less than three, or at least one replica is broken. This status is also displayed if the metadevice state database replicas have been created on fewer than three different controllers.	Add more replicas, preferably spread across different controllers, or fix the broken replicas. If possible, add another controller and create state database replicas on drives attached to the new controller. See "How to Create Additional State Database Replicas (DiskSuite Tool)" on page 18 to add more state database replicas. See "How to Enable a State Database Replica (DiskSuite Tool)" on page 102 to fix broken replicas.
Urgent	The number of good state database replicas is less than two, or one or more state database replicas are broken.	Add more replicas, preferably spread across different controllers, or fix the broken replicas. See "How to Create Additional State Database Replicas (DiskSuite Tool)" on page 18 to add more state database replicas. See "How to Enable a State Database Replica (DiskSuite Tool)" on page 102 to fix broken replicas.
Critical	There are no good state database replicas.	Create at least three state database replicas from scratch, before rebooting. Otherwise the system will not boot properly. See "How to Create Initial State Database Replicas From Scratch (DiskSuite Tool)" on page 6.

▼ How to Check the Status of State Database Replicas (Command Line)

After checking the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79), use the `metadb(1M)` command with the `-i` option to view the status of state database replicas. Refer to the `metadb(1M)` man page for more information.

Example — Checking Status of All State Database Replicas

```
# metadb -i
        flags           first blk       block count
     a        u         16              1034            /dev/dsk/c4t3d0s2
     a        u         16              1034            /dev/dsk/c3t3d0s2
     a        u         16              1034            /dev/dsk/c2t3d0s2
 o - state database replica active prior to last mddb configuration change
 u - state database replica is up to date
 l - locator for this state database replica was read successfully
 c - state database replica's location was in /etc/opt/SUNWmd/mddb.cf
 p - state database replica's location was patched in kernel
 m - state database replica is master, this is state database replica selected as input
 W - state database replica has device write errors
 a -
 state database replica is active, commits are occurring to this state database replica
 M - state database replica had problem with master blocks
 D - state database replica had problem with data blocks
 F - state database replica had format problems
 S - state database replica is too small to hold current data base
 R - state database replica had device read errors
```

The characters in the front of the device name represent the status. All of the state database replicas in this example are active, as indicated by the `a` flag. A legend of all the flags follows the status.

Uppercase letters indicate a problem status. Lowercase letters indicate an “Okay” status.

▼ How to Check the Status of Metadevices and Hot Spare Pools (DiskSuite Tool)

Use this procedure to view and interpret metadevice and hot spare pool status information.

1. **Make sure you have met the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79).**

2. **Check the status of a metadevice or hot spare pool by displaying the object’s Information window.**

For other ways of checking status, see “Using DiskSuite Tool to Check Status” on page 80.

3. **Refer to Table 3–2 for explanations of the status keywords used by metadevices and hot spare pools.**

TABLE 3–2 General Status Keywords

Keyword	Meaning	Used By ...
OK	The metadevice or hot spare pool has no errors and is functioning correctly.	All metadevice types and hot spare pools
Attention	The metadevice or hot spare pool has a problem, but there is no immediate danger of losing data.	All metadevice types and hot spare pools
Urgent	The metadevice is only one failure away from losing data.	Mirrors / submirrors, RAID5 metadevices, and trans metadevices
Critical	Data potentially has been corrupted. For example, all submirrors in a mirror have errors, or a RAID5 metadevice has errors on more than one slice. Template objects, except the hot spare pool template, also show a Critical status if the metadevice configuration is invalid.	Mirrors / submirrors, RAID5 metadevices, trans metadevices, and all template objects

Note - If the fan fails on a SPARCstorage Array, all metadevices and slices on that SPARCstorage Array are marked “Critical.”

4. **Use the following to find the appropriate section about a specific DiskSuite object’s status and possible actions to take.**
   - “Stripe and Concatenation Status (DiskSuite Tool)” on page 85
   - “Mirror and Submirror Status (DiskSuite Tool)” on page 85
   - “RAID5 Metadevice Status (DiskSuite Tool)” on page 87
   - “Trans Metadevice Status (DiskSuite Tool)” on page 88

- “Hot Spare Pool and Hot Spare Status (DiskSuite Tool)” on page 90

Stripe and Concatenation Status (DiskSuite Tool)

DiskSuite does not report a state change for a concatenation or stripe that experiences errors, unless the concatenation or stripe is used as a submirror. If there is a slice error, or other device problem, DiskSuite returns an error to the requesting application, and outputs it to the console, such as:

```
WARNING: md d4: read error on /dev/dsk/c1t3d0s6
```

Note - DiskSuite can send SNMP trap data (alerts), such as the message above, to any network management console capable of receiving SNMP messages. Refer to “How to Configure DiskSuite SNMP Support (Command Line)” on page 215, for more information.

Because concatenations and stripes do not contain replicated data, to recover from slice errors on simple metadevices you must replace the physical disk, recreate the metadevice, and restore data from backup. Refer to “How to Recreate a Stripe or Concatenation After Slice Failure (DiskSuite Tool)” on page 102, or “How to Recreate a Stripe or Concatenation After Slice Failure (Command Line)” on page 104.

Mirror and Submirror Status (DiskSuite Tool)

A Mirror object has two Status fields: one for the mirror device itself, and individual Status fields for each submirror. The Status field for a mirror, as explained in Table 3–3, gives a high-level status.

TABLE 3–3 Mirror Status Keywords

Keyword	Meaning
OK	The mirror has no errors and is functioning correctly.
Attention	A submirror has a problem, but there is no immediate danger of losing data. There are still two copies of the data (the mirror is three-way mirror and only one submirror failed), or a hot spare has kicked in.
Urgent	The mirror contains only a single good submirror, providing only one copy of the data. The mirror is only one failure away from losing data.
Critical	All submirrors have errors and data has potentially been corrupted.

Table 3–4 shows the Status fields of submirrors, and possible actions to take.

TABLE 3–4 Submirror Status Keywords

Keyword	Meaning	Action
OK	The submirror has no errors and is functioning correctly.	None.
Resyncing	The submirror is actively being resynced.	None. An error has occurred and been corrected, the submirror has just been brought back online, or a new submirror has been added.
Component Resyncing	A slice in the submirror is actively being resynced.	None. Either a hot spare slice or another slice has replaced an errored slice in the submirror.
Attaching	The submirror is being attached.	None.
Attached (resyncing)	The entire submirror is being resynced after the attach occurred.	None.
Online (scheduled)	The submirror will be brought online the next time you click Commit.	Click the Commit button to enable the submirror.
Offline (scheduled)	The submirror will be brought offline the next time you click Commit.	Click the Commit button to offline the submirror.
Offlined	The submirror is offline.	When appropriate, bring the submirror back online, for example, after performing maintenance. See “How to Place a Submirror Offline and Online (DiskSuite Tool)” on page 148.

TABLE 3–4 Submirror Status Keywords *(continued)*

Keyword	Meaning	Action
Maintenance	The submirror has an error.	Repair the submirror. You can fix submirrors in the “Errored” state in any order. See “How to Enable a Slice in a Submirror (DiskSuite Tool)” on page 109, or “How to Replace a Slice in a Submirror (DiskSuite Tool)” on page 110.
Last Erred	The submirror has errors, and data for the mirror has potentially been corrupted.	Fix submirrors in the “Maintenance” state first, then fix the submirror in the “Last Erred” state. See “How to Enable a Slice in a Submirror (DiskSuite Tool)” on page 109, or “How to Replace a Slice in a Submirror (DiskSuite Tool)” on page 110. After fixing the error, validate the data.

Note - DiskSuite does not retain state and hot spare information for simple metadevices that are not submirrors.

RAID5 Metadevice Status (DiskSuite Tool)

Table 3–5 explains the keywords in the Status fields of RAID5 objects, and possible actions to take.

TABLE 3–5 RAID5 Status Keywords

Keyword	Meaning	Action
OK	The RAID5 metadevice has no errors and is functioning correctly.	None.
Attached / initialize (resyncing)	The RAID5 metadevice is being resynced after an attach occurred, or after being created.	Normally none. During the initialization of a new RAID5 metadevice, if an I/O error occurs, the device goes into the "Maintenance" state. If the initialization fails, the metadevice is in the "Initialization Failed" state and the slice is in the "Maintenance" state. If this happens, clear the metadevice and recreate it.
Attention	There is a problem with the RAID5 metadevice, but there is no immediate danger of losing data.	Continue to monitor the status of the device.
Urgent	The RAID5 metadevice has a slice error and you are only one failure away from losing data.	Fix the errored slice. See "How to Enable a Slice in a RAID5 Metadevice (DiskSuite Tool)" on page 113, or "How to Replace a RAID5 Slice (DiskSuite Tool)" on page 115.
Critical	The RAID5 metadevice has more than one slice with an error. Data has potentially been corrupted.	To fix the errored slices, see "How to Enable a Slice in a RAID5 Metadevice (DiskSuite Tool)" on page 113, or "How to Replace a RAID5 Slice (DiskSuite Tool)" on page 115. You may need to restore data from backup.

Trans Metadevice Status (DiskSuite Tool)

Table 3–6 explains the keywords in the Status fields of Trans Metadevice objects, and possible actions to take.

TABLE 3–6 Trans Metadevice Status Keywords

Keyword	Meaning	Action
OK	The device is functioning properly. If mounted, the file system is logging and will not be checked at boot (that is, the file system will not be checked by `fsck` at boot).	None.
Detach Log (in progress)	The trans metadevice log will be detached when the Trans metadevice is unmounted or at the next reboot.	None.
Detach Log (scheduled)	The trans metadevice log will be detached the next time you click the Commit button.	Click Commit to detach the log. The detach takes place at the next reboot, or when the file system is unmounted and remounted.
Attention	There is a problem with the trans metadevice, but there is no immediate danger of losing data.	Continue to monitor the status of the trans metadevice.
Urgent	There is a problem with the trans metadevice and it is only one failure away from losing data. This state can only exist if the trans metadevice contains a RAID5 metadevice or mirror.	Fix the errored mirror or RAID5 master device. See “Overview of Replacing and Enabling Slices in Mirrors and RAID5 Metadevices” on page 105.
Critical (log missing)	The trans metadevice does not have a logging device attached.	Attach a logging device. Logging for the file system cannot start until a logging device is attached.

TABLE 3–6 Trans Metadevice Status Keywords *(continued)*

Keyword	Meaning	Action
Critical (log hard error)	A device error or file system panic has occurred while the device was in use. An I/O error is returned for every read or write until the device is closed or unmounted. The first open causes the device to transition to the Error state.	Fix the trans metadevice. See "How to Recover a Trans Metadevice With a File System Panic (Command Line)" on page 122, or "How to Recover a Trans Metadevice With Hard Errors (Command Line)" on page 122.
Critical (error)	The device can be read and written. The file system can be mounted read-only. However, an I/O error is returned for every read or write that actually gets a device error. The device does not transition back to the Hard Error state, even when a later device error of file system panic occurs.	Fix the trans metadevice. See "How to Recover a Trans Metadevice With a File System Panic (Command Line)" on page 122, or "How to Recover a Trans Metadevice With Hard Errors (Command Line)" on page 122. Successfully completing `fsck(1M)` or `newfs(1M)` transitions the device into the Okay state. When the device is in the Hard Error or Error state, `fsck` automatically checks and repairs the file system at boot time. `newfs` destroys whatever data may be on the device.

Hot Spare Pool and Hot Spare Status (DiskSuite Tool)

Table 3–7 explains the keywords in the Status fields of Hot Spare Pool objects, and possible actions to take.

TABLE 3–7 Hot Spare Pool Status Keywords

Keyword	Meaning	Action
OK	The hot spares are running and ready to accept data, but are not currently being written to or read from.	None.
In-use	Hot spares are currently being written to and read from.	Diagnose how the hot spares are being used. Then repair the slice in the metadevice for which the hot spare is being used.
Attention	There is a problem with a hot spare or hot spare pool, but there is no immediate danger of losing data. This status is also displayed if there are no hot spares in the Hot Spare Pool, or if all the hot spares are in use or any are broken.	Diagnose how the hot spares are being used or why they are broken. You can add more hot spares to the hot spare pool if necessary.

▼ How to Check the Status of Metadevices and Hot Spare Pools (Command Line)

Make sure you have met the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79). Use the `metastat(1M)` command to view metadevice or hot spare pool status. Refer to the `metastat(1M)` man pages for more information.

Use the following to find an explanation of the command line output and possible actions to take.

- “Stripe and Concatenation Status (Command Line)” on page 92
- “Mirror and Submirror Status (Command Line)” on page 92
- “RAID5 Metadevice Status (Command Line)” on page 94
- “Trans Metadevice Status (Command Line)” on page 97
- “Hot Spare Pool and Hot Spare Status (Command Line)” on page 98

Note - Refer to Table 3–2 for an explanation of DiskSuite’s general status keywords.

Stripe and Concatenation Status (Command Line)

DiskSuite does not report a state change for a concatenation or a stripe, unless the concatenation or stripe is used as a submirror. Refer to “Stripe and Concatenation Status (DiskSuite Tool)” on page 85 for more information.

Mirror and Submirror Status (Command Line)

Running `metastat(1M)` on a mirror displays the state of each submirror, the pass number, the read option, the write option, and the size of the total number of blocks in the mirror. Refer to “How to Change a Mirror’s Options (Command Line)” on page 165 to change a mirror’s pass number, read option, or write option.

Here is sample mirror output from `metastat`.

```
# metastat
d0: Mirror
    Submirror 0: d1
      State: Okay
    Submirror 1: d2
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 5600 blocks

d1: Submirror of d0
    State: Okay
    Size: 5600 blocks
    Stripe 0:
        Device              Start Block  Dbase State        Hot Spare
        c0t2d0s7                  0     No    Okay

...
```

For each submirror in the mirror, `metastat` shows the state, an “invoke” line if there is an error, the assigned hot spare pool (if any), size in blocks, and information about each slice in the submirror.

Table 3–8 explains submirror states.

TABLE 3–8 Submirror States (Command Line)

State	Meaning
Okay	The submirror has no errors and is functioning correctly.
Resyncing	The submirror is actively being resynced. An error has occurred and been corrected, the submirror has just been brought back online, or a new submirror has been added.
Needs Maintenance	A slice (or slices) in the submirror has encountered an I/O error or an open error. All reads and writes to and from this slice in the submirror have been discontinued.

Additionally, for each stripe in a submirror, `metastat` shows the “Device” (device name of the slice in the stripe); “Start Block” on which the slice begins; “Dbase” to show if the slice contains a state database replica; “State” of the slice; and “Hot Spare” to show the slice being used to hot spare a failed slice.

The slice state is perhaps the most important information when troubleshooting mirror errors. The submirror state only provides general status information, such as “Okay” or “Needs Maintenance.” If the submirror reports a “Needs Maintenance” state, refer to the slice state. You take a different recovery action if the slice is in the “Maintenance” or “Last Erred” state. If you only have slices in the “Maintenance” state, they can be repaired in any order. If you have a slices in the “Maintenance” state and a slice in the “Last Erred” state, you must fix the slices in the “Maintenance” state first then the “Last Erred” slice. Refer to “Overview of Replacing and Enabling Slices in Mirrors and RAID5 Metadevices” on page 105.

Table 3–9 explains the slice states for submirrors and possible actions to take.

TABLE 3–9 Submirror Slice States (Command Line)

State	Meaning	Action
Okay	The slice has no errors and is functioning correctly.	None.
Resyncing	The slice is actively being resynced. An error has occurred and been corrected, the submirror has just been brought back online, or a new submirror has been added.	If desired, monitor the submirror status until the resync is done.

TABLE 3–9 Submirror Slice States (Command Line) *(continued)*

State	Meaning	Action
Maintenance	The slice has encountered an I/O error or an open error. All reads and writes to and from this slice have been discontinued.	Enable or replace the errored slice. See "How to Enable a Slice in a Submirror (Command Line)" on page 109, or "How to Replace a Slice in a Submirror (Command Line)" on page 111. Note: The `metastat(1M)` command will show an `invoke` recovery message with the appropriate action to take with the `metareplace(1M)` command. You can also use the `metareplace -e` command.
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 I/O will fail.	First, enable or replace slices in the "Maintenance" state. See "How to Enable a Slice in a Submirror (Command Line)" on page 109, or "How to Replace a Slice in a Submirror (Command Line)" on page 111. Usually, this error results in some data loss, so validate the mirror after it is fixed. For a file system, use the `fsck(1M)` command to validate the "metadata" then check the user-data. An application or database must have its own method of validating the metadata.

RAID5 Metadevice Status (Command Line)

Running the `metastat(1M)` command on a RAID5 metadevice shows the status of the metadevice. Additionally, for each slice in the RAID5 metadevice, `metastat` shows the "Device" (device name of the slice in the stripe); "Start Block" on which the slice begins; "Dbase" to show if the slice contains a state database replica; "State" of the slice; and "Hot Spare" to show the slice being used to hot spare a failed slice.

Here is sample RAID5 metadevice output from `metastat`.

```
# metastat
d10: RAID
    State: Okay
    Interlace: 32 blocks
    Size: 10080 blocks
Original device:
    Size: 10496 blocks
        Device              Start Block  Dbase State        Hot Spare
        c0t0d0s1                 330     No    Okay
```

(continued)

```
        c1t2d0s1                330     No    Okay
        c2t3d0s1                330     No    Okay
```

Table 3–10 explains RAID5 metadevice states.

TABLE 3–10 RAID5 States (Command Line)

State	Meaning
Initializing	Slices are in the process of having all disk blocks zeroed. This is necessary due to the nature of RAID5 metadevices with respect to data and parity interlace striping. Once the state changes to the "Okay," the initialization process is complete and you are able to open the device. Up to this point, applications receive error messages.
Okay	The device is ready for use and is currently free from errors.
Maintenance	A single slice has been marked as errored due to I/O or open errors encountered during a read or write operation.

The slice state is perhaps the most important information when troubleshooting RAID5 metadevice errors. The RAID5 state only provides general status information, such as "Okay" or "Needs Maintenance." If the RAID5 reports a "Needs Maintenance" state, refer to the slice state. You take a different recovery action if the slice is in the "Maintenance" or "Last Erred" state. If you only have a slice in the "Maintenance" state, it can be repaired without loss of data. If you have a slice in the "Maintenance" state and a slice in the "Last Erred" state, data has probably been corrupted. You must fix the slice in the "Maintenance" state first then the "Last Erred" slice. Refer to "Overview of Replacing and Enabling Slices in Mirrors and RAID5 Metadevices" on page 105.

Table 3–11 explains the slice states for a RAID5 metadevice and possible actions to take.

TABLE 3–11 RAID5 Slice States (Command Line)

State	Meaning	Action
Initializing	Slices are in the process of having all disk blocks zeroed. This is necessary due to the nature of RAID5 metadevices with respect to data and parity interlace striping.	Normally none. If an I/O error occurs during this process, the device goes into the "Maintenance" state. If the initialization fails, the metadevice is in the "Initialization Failed" state and the slice is in the "Maintenance" state. If this happens, clear the metadevice and recreate it.
Okay	The device is ready for use and is currently free from errors.	None. Slices may be added or replaced, if necessary.
Resyncing	The slice is actively being resynced. An error has occurred and been corrected, a slice has been enabled, or a slice has been added.	If desired, monitor the RAID5 metadevice status until the resync is done.
Maintenance	A single slice has been marked as errored due to I/O or open errors encountered during a read or write operation.	Enable or replace the errored slice. See "How to Enable a Slice in a RAID5 Metadevice (Command Line)" on page 114, or "How to Replace a RAID5 Slice (Command Line)" on page 116. Note: The `metastat(1M)` command will show an `invoke` recovery message with the appropriate action to take with the `metareplace(1M)` command.
Maintenance/ Last Erred	Multiple slices have encountered errors. The state of the errored slices is either "Maintenance" or "Last Erred." In this state, no I/O is attempted on the slice that is in the "Maintenance" state, but I/O is attempted to the slice marked "Last Erred" with the outcome being the overall status of the I/O request.	Enable or replace the errored slices. See "How to Enable a Slice in a RAID5 Metadevice (Command Line)" on page 114, or "How to Replace a RAID5 Slice (Command Line)" on page 116. Note: The `metastat(1M)` command will show an `invoke` recovery message with the appropriate action to take with the `metareplace(1M)` command, which must be run with the `-f` flag. This indicates that data might be fabricated due to multiple errored slices.

Trans Metadevice Status (Command Line)

Running the `metastat(1M)` command on a trans metadevice shows the status of the metadevice.

Here is sample trans metadevice output from `metastat`:

```
# metastat
d20: Trans
    State: Okay
    Size: 102816 blocks
    Master Device: c0t3d0s4
    Logging Device: c0t2d0s3

        Master Device       Start Block  Dbase
        c0t3d0s4                   0     No

c0t2d0s3: Logging device for d0
    State: Okay
    Size: 5350 blocks

        Logging Device      Start Block  Dbase
        c0t2d0s3                 250     No
```

The `metastat` command also shows master and logging devices. For each device, the following information is displayed: the “Device” (device name of the slice or metadevice); “Start Block” on which the device begins; “Dbase” to show if the device contains a state database replica; and for the logging device, the “State.”

Table 3–12 explains trans metadevice states and possible actions to take.

TABLE 3–12 Trans Metadevice States (Command Line)

State	Meaning	Action
Okay	The device is functioning properly. If mounted, the file system is logging and will not be checked at boot.	None.
Attaching	The logging device will be attached to the trans metadevice when the trans is closed or unmounted. When this occurs, the device is transitioned to the Okay state.	Refer to the `metattach(1M)` man page.
Detached	The trans metadevice does not have a logging device. All benefits from UFS logging are disabled.	`fsck(1M)` automatically checks the device at boot time. Refer to the `metadetach(1M)` man page.

TABLE 3–12 Trans Metadevice States (Command Line) *(continued)*

State	Meaning	Action
Detaching	The logging device will be detached from the trans metadevice when the trans is closed or unmounted. When this occurs, the device transitions to the Detached state.	Refer to the `metadetach(1M)` man page.
Hard Error	A device error or file system panic has occurred while the device was in use. An I/O error is returned for every read or write until the device is closed or unmounted. The first open causes the device to transition to the Error state.	Fix the trans metadevice. See "How to Recover a Trans Metadevice With a File System Panic (Command Line)" on page 122, or "How to Recover a Trans Metadevice With Hard Errors (Command Line)" on page 122.
Error	The device can be read and written. The file system can be mounted read-only. However, an I/O error is returned for every read or write that actually gets a device error. The device does not transition back to the Hard Error state, even when a later device error of file system panic occurs.	Fix the trans metadevice. See "How to Recover a Trans Metadevice With a File System Panic (Command Line)" on page 122, or "How to Recover a Trans Metadevice With Hard Errors (Command Line)" on page 122. Successfully completing `fsck(1M)` or `newfs(1M)` transitions the device into the Okay state. When the device is in the Hard Error or Error state, `fsck` automatically checks and repairs the file system at boot time. `newfs` destroys whatever data may be on the device.

Hot Spare Pool and Hot Spare Status (Command Line)

Running the `metastat(1M)` command on a hot spare pool shows the status of the hot spare pool and its hot spares.

Here is sample hot spare pool output from `metastat`.

```
# metastat hsp001
hsp001: 1 hot spare
        c1t3d0s2                Available       16800 blocks
```

Table 3–13 explains hot spare pool states and possible actions to take.

TABLE 3–13 Hot Spare Pool States (Command Line)

State	Meaning	Action
Available	The hot spares are running and ready to accept data, but are not currently being written to or read from.	None.
In-use	Hot spares are currently being written to and read from.	Diagnose how the hot spares are being used. Then repair the slice in the metadevice for which the hot spare is being used.
Attention	There is a problem with a hot spare or hot spare pool, but there is no immediate danger of losing data. This status is also displayed if there are no hot spares in the Hot Spare Pool or all the hot spares are in use or any are broken.	Diagnose how the hot spares are being used or why they are broken. You can add more hot spares to the hot spare pool if desired.

▼ How to Check the Status of a Diskset (Command Line)

Make sure you have met the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79). Use the `metaset(1M)` command to view diskset status. Refer to the `metaset(1M)` man page for more information.

Note - Diskset ownership is only shown on the owning host.

Example — Checking Status of a Specified Diskset

```
red# metaset -s relo-red
Set name = relo-red, Set number = 1

Host                Owner
  red                Yes
  blue

Drive               Dbase
  c1t2d0             Yes
  c1t3d0             Yes
  c2t2d0             Yes
  c2t3d0             Yes
```

(continued)

(Continuation)

```
  c2t4d0             Yes
  c2t5d0             Yes
```

The `metaset(1M)` command with the –`s` option followed by the name of the `relo-red` diskset displays status information for that diskset. By issuing the `metaset` command from the owning host, `red`, it is determined that `red` is in fact the diskset owner. The `metaset` command also displays drives in the diskset.

Example — Checking Status of All Disksets

```
red# metaset
Set name = relo-red, Set number = 1

Host                Owner
  red                Yes
  blue

Drive               Dbase
  c1t2d0             Yes
  c1t3d0             Yes
  c2t2d0             Yes
  c2t3d0             Yes
  c2t4d0             Yes
  c2t5d0             Yes

Set name = relo-blue, Set number = 2

Host                Owner
  red
  blue

Drive               Dbase
  c3t2d0             Yes
  c3t3d0             Yes
  c3t4d0             Yes
  c3t5d0             Yes

Set name = rimtic, Set number = 3

Host                Owner
  red
  blue

Drive               Dbase
  c4t2d0             Yes
  c4t3d0             Yes
  c4t4d0             Yes
```

(continued)

(Continuation)

```
c4t5d0             Yes
```

The `metaset` command by itself displays the status of all disksets. In this example, three disksets named `relo-red`, `relo-blue`, and `rimtic` are configured. Because host `red` owns the `relo-red` diskset, `metaset` shows `red` as the owner. Host `blue` owns the other two disksets, `relo-blue` and `rimtic`. This could only be determined if `metaset` were run from host `blue`.

Replacing and Enabling Objects

This section contains the steps to replace and enable slices in DiskSuite objects, including:

- State database replicas
- Stripes and concatenations
- Mirrors (submirrors)
- RAID5 metadevices
- Hot spares

Note - To repair and replace physical disks, including those in a SPARCstorage Array, refer to Chapter 7.

Preliminary Information for Enabling State Database Replicas

When you enable (restore) a state database replica with DiskSuite Tool, two things happen. DiskSuite Tool first removes (deletes) the replica, then tries to add it back to the slice. If there is a problem, such as an errored slice, the delete still occurs, and you need to repair the slice before the state database replica can be restored.

▼ How to Enable a State Database Replica (DiskSuite Tool)

Use this task to bring a slice being used by the state database back online. You would use this task after physically replacing an errored slice (disk).

1. **Make sure you have met the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79) and have read the preliminary information (“Preliminary Information for Enabling State Database Replicas” on page 101).**

1. **Double-click the MetaDB object in the Objects list.**

   The MetaDB object appears on the canvas.

2. **Display the MetaDB object’s Info window.**

3. **Select the errored slice in the scrolling list then click Restore.**

   The Restore button is enabled only if a selected slice does not display the OK status.

4. **To verify that the restore occurred, display the Configuration Log.**

Preliminary Information for Recreating a Stripe or Concatenation

- Because a stripe or concatenation does not contain replicated data, when such a metadevice has a slice failure you must replace the slice, recreate the stripe or concatenation, and restore data from a backup.
- When recreating a stripe, use a replacement slice that has at least the same size as the errored slice.
- When recreating a concatenation, use a replacement slice that has at least the same capacity as the failed one.
- If the failed slice in the stripe or concatenation is a submirror, refer to “How to Enable a Slice in a Submirror (DiskSuite Tool)” on page 109 or “How to Replace a Slice in a Submirror (DiskSuite Tool)” on page 110.

▼ How to Recreate a Stripe or Concatenation After Slice Failure (DiskSuite Tool)

1. **Make sure you have met the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79) and have read the preliminary information (“Preliminary Information for Recreating a Stripe or Concatenation” on page 102).**

2. **Stop access to the metadevice.**

   For example, if the metadevice contains a mounted file system, unmount the file system.

3. **If possible, run the `ufsdump(1M)` command on the stripe or concatenation.**

```
# ufsdump [option...] [argument...] file-to-back-up...
```

   In this command,

option	Is a single string of one-letter option names.
argument	Identifies option arguments and may be multiple strings. The options and the arguments that go with them must be in the same order.
file-to-back-up	Identifies the file(s) to back up. These file names must always come last.

   Refer to the `ufsdump(1M)` man page for more information. If you cannot access the metadevice, you will have to rely on the most current backup.

4. **Double-click the stripe or concatenation to be deleted in the Objects list.**

   The metadevice object appears on the canvas.

5. **Choose Delete from the object's pop-up menu.**

6. **Click Really Delete on the dialog box that is displayed.**

7. **Recreate the metadevice.**

   Refer to "How to Create a Striped Metadevice (DiskSuite Tool)" on page 21 or "How to Create a Concatenation (DiskSuite Tool)" on page 25.

Note - If the metadevice is a stripe, the new slice must be the same size as the failed one. If the metadevice is a concatenation, the new slice must have at least the same capacity as the failed slice.

---

8. **If the metadevice was used for a file system, create a new file system on the metadevice.**

   Refer to "Creating File Systems on Metadevices" on page 73.

9. **Restore the data with the `ufsrestore(1M)` command.**

```
# ufsrestore [option...][argument...][filename...]
```

In this command,

option	Is a single string of one-letter option names. You must choose one and only one of these options: `i`, `r`, `R`, `t`, or `x`.
argument	Follows the option string with the arguments that match the options. The option names and the arguments that go with them must be in the same order.
filename	Specifies files to be restored as arguments to the –`x` or –`t` options, and must always come last.

Refer to the `ufsrestore(1M)` man page for more information.

10. Validate the data on the metadevice.

▼ How to Recreate a Stripe or Concatenation After Slice Failure (Command Line)

After checking the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79), and the preliminary information (“Preliminary Information for Recreating a Stripe or Concatenation” on page 102), use the `metaclear(1M)` and `metainit(1M)` commands to recreate a stripe or concatenation after a slice has failed. The `metastat` command does not show slice errors for concatenations and stripes. You will, however, see errors on the system console, such as:

```
WARNING: md d35: read error on /dev/dsk/c0t0d0s6
```

Refer to the `metaclear(1M)` and `metainit(1M)` man pages for more information.

Example — Recreating a Concatenation After Slice Failure

```
# umount /news
# init 0
ok boot -s
...

# ufsdump 0ucf /dev/rmt/0 /news

DUMP: Date of this level 0 dump: Fri Mar 1 15:17:45 1996
...
```

(continued)

```
DUMP: DUMP IS DONE
# metaclear d35
# metainit d35 2 1 c1t0d0s2 1 c1t0d1s2
# newfs /dev/md/rdsk/d35
# mount /dev/md/dsk/d35 /news
# cd /news
# ufsrestore rvf /dev/rmt 0
Verify volume and initialize maps
Media block size is 126
...
Check pointing the restore
# rm restoresymtable
# ls /news
```

Because `d35` contains a mounted file system, `/news`, it is unmounted, then the system is booted into single-user mode. The `ufsdump` command dumps data to tape, and the concatenation is cleared with the `metaclear` command. The `metainit` command recreates the concatenation using a new slice to replace the failed slice. Data is restored via the `ufsrestore` command, then validated, for example, by using the `ls` command.

Note - If the metadevice is a stripe, the new slice must be the same size as the failed one. If the metadevice is a concatenation, the new slice must have at least the same capacity as the failed slice.

Overview of Replacing and Enabling Slices in Mirrors and RAID5 Metadevices

DiskSuite has the capability to *replace* and *enable* slices within mirrors and RAID5 metadevices.

In DiskSuite terms, *replacing* a slice is a way to substitute an available slice on the system for a selected slice in a submirror or RAID5 metadevice. You can think of this as a “metareplace,” as opposed to physically replacing the slice. *Enabling* a slice means to “activate” or substitute a slice with itself (that is, the slice name is the same).

The following describes the two methods you can use and when you would use them.

Note - When recovering from disk errors, scan `/var/adm/messages` to see what kind of errors occurred. If the errors are of a transitory nature and the disks themselves do not have problems, try enabling the errored slices. You can also use the `format(1M)` command to test a disk.

Enabling a Slice

It is appropriate to enable a slice when:

- Attempting to recover from possible soft errors by enabling the currently errored slice
- Physically replacing a disk drive in place of a failed one, repartitioning the drive the same as the old one, then enabling the errored slice(s) within the metadevice.

You can enable a slice when:

1. DiskSuite cannot access the physical drive. This may have occurred, for example, due to a power loss, or a loose drive cable. In this case, DiskSuite puts the slices in the "Maintenance" state. You need to make sure the drive is accessible (restore power, recable, and so on) then enable the slices in the metadevices.
2. You suspect that a physical drive is having transitory problems that are not disk-related. You might be able to fix a slice in the "Maintenance" state by simply enabling it. If this does not fix the problem, then you need to either physically replace the disk drive and enable the slice, or "metareplace" the slice with another available slice on the system.

   When you physically replace a drive, be sure to partition it the same as old drive. Note that after the drive has been physically replaced and partitioned like the old one, the task to enable the errored slice(s) is the same as for the first condition described above.

Note - Always check for state database replicas and hot spares on the drive being replaced. Any state database replica shown to be in error should be deleted before replacing the disk and added back (making sure the size is the same) before enabling the slice. You should treat hot spares in the same manner.

Replacing a Slice with Another Available Slice

You use the DiskSuite "metareplace" slice feature when replacing or swapping an existing slice with a different slice that is available and not in use on the system.

You can use this method when:

1. A disk drive has problems, and you don't have a replacement drive but you do have available slices elsewhere on the system. (You might want to do this if a replacement is absolutely necessary but you don't want to shut down the system.)

2. You are seeing soft errors. Physical disks may report soft errors even though DiskSuite shows the mirror/submirror or RAID5 metadevice in the “Okay” state. Replacing the slice in question with another available slice enables you to perform preventative maintenance and potentially prevent hard errors from occurring.
3. You want to do performance tuning. For example, by using DiskSuite Tool’s performance monitor, you see that a particular slice in a RAID5 metadevice is experiencing a high load average, even though it’s in the “Okay” state. To balance the load on the metadevice, you can replace that slice with one from a disk that is less utilized. This type of replacement can be performed online without interrupting service to the metadevice.

Note - DiskSuite Tool enables you to replace an entire submirror if necessary. To do so, you create a new submirror (Concat/Stripe object) and drag it on top of the submirror to be replaced. This task is documented in “How to Replace a Submirror (DiskSuite Tool)” on page 112.

Maintenance vs. Last Erred States

When a slice in a mirror or RAID5 metadevice device experiences errors, DiskSuite puts the slice in the “Maintenance” state. No further reads or writes are performed to a slice in the “Maintenance” state. Subsequent errors on other slices in the same metadevice are handled differently, depending on the type of metadevice. A mirror may be able to tolerate many slices in the “Maintenance” state and still be read from and written to. A RAID5 metadevice, by definition, can only tolerate a single slice in the “Maintenance” state. When either a mirror or RAID5 metadevice has a slice in the “Last Erred” state, I/O is still attempted to the slice marked “Last Erred.” This is because a “Last Erred” slice contains the last good copy of data from DiskSuite’s point of view. With a slice in the “Last Erred” state, the metadevice behaves like a normal device (disk) and returns I/O errors to an application. Usually, at this point some data has been lost.

Always replace slices in the “Maintenance” state first, followed by those in the “Last Erred” state. After a slice is replaced and resynced, use the `metastat(1M)` command to verify its state, then validate the data to make sure it is good.

Mirrors: If slices are in the “Maintenance” state, no data has been lost. You can safely replace or enable the slices in any order. If a slice is in the “Last Erred” state, you cannot replace it until you first replace all the other mirrored slices in the “Maintenance” state. Replacing or enabling a slice in the “Last Erred” state usually means that some data has been lost. Be sure to validate the data on the mirror after repairing it.

RAID5 Metadevices: A RAID5 metadevice can tolerate a single slice failure. You can safely replace a single slice in the “Maintenance” state without losing data. If an error on another slice occurs, it is put into the “Last Erred” state. At this point, the RAID5 metadevice is a read-only device; you need to perform some type of error

recovery so that the state of the RAID5 metadevice is non-errored and the possibility of data loss is reduced. If a RAID5 metadevice reaches a “Last Erred” state, there is a good chance it has lost data. Be sure to validate the data on the RAID5 metadevice after repairing it.

Preliminary Information For Replacing and Enabling Slices in Mirrors and RAID5 Metadevices

When replacing slices in a mirror or a RAID5 metadevice, follow these guidelines:

- Always replace slices in the “Maintenance” state first, followed by those in the “Last Erred” state.
- After a slice is replaced and resynced, use the `metastat(1M)` command to verify the metadevice’s state, then validate the data to make sure it is good. Replacing or enabling a slice in the “Last Erred” state usually means that some data has been lost. Be sure to validate the data on the metadevice after repairing it. For a UFS, run the `fsck(1M)` command to validate the “metadata” (the structure of the file system) then check the actual user data. (Practically, users will have to examine their files.) A database or other application must have its own way of validating its internal data structure.
- Always check for state database replicas and hot spares when replacing slices. Any state database replica shown to be in error should be deleted before replacing the physical disk and added back before enabling the slice. The same applies to hot spares.
- RAID5 Metadevices – During slice replacement, data is recovered, either from a hot spare currently in use, or using the RAID level 5 parity, when no hot spare in use.
- Mirrors – When you replace a slice, DiskSuite automatically starts a resync of the new slice with the rest of the mirror. When the resync completes, the replaced slice becomes readable and writable. If the failed slice has been replaced with data from a hot spare, the hot spare is placed in the “Available” state and made available for other hot spare replacements.
- The new slice must be large enough to replace the old slice.
- As a precaution, back up all data before replacing “Last Erred” devices.

Note - A submirror or RAID5 metadevice may be using a hot spare in place of an errored slice. When that errored slice is enabled or replaced using the procedures in this section, the hot spare is marked “available” in the hot spare pool, and is ready for use.

▼ How to Enable a Slice in a Submirror (DiskSuite Tool)

Use this task to enable a slice in a submirror that is in the “Errored” state.

1. **Make sure you have met the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79), and have read the overview (“Overview of Replacing and Enabling Slices in Mirrors and RAID5 Metadevices” on page 105) and the preliminary information (“Preliminary Information For Replacing and Enabling Slices in Mirrors and RAID5 Metadevices” on page 108).**

2. **Double-click the errored Mirror object in the Objects list.**

   The object appears on the canvas. The submirror displays the error status.

3. **Click inside the stripe rectangle of the submirror with status of “Critical.” Then display the object’s pop-up window and choose Info.**

   The Stripe Information window appears.

4. **In the Stripe Info window, select the slice with the “Critical” status and click Enable. Then click Close.**

   The slice status changes from Critical to Enabled. The submirror status changes from Critical (Errored) to Critical (Uncommitted).

5. **Click inside the Mirror object. Then click Commit.**

   A mirror resync begins. The submirror status changes to “Component Resyncing.”

6. **When the resync is done, verify that the status of the mirror is OK.**

7. **Validate the data.**

Note - If DiskSuite still reports the slice in the “Errored” state after enabling the slice, refer to “How to Replace a Slice in a Submirror (DiskSuite Tool)” on page 110.

▼ How to Enable a Slice in a Submirror (Command Line)

After checking the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79), the overview (“Overview of Replacing and Enabling Slices in Mirrors and RAID5 Metadevices” on page 105), and the preliminary information (“Preliminary Information For Replacing and Enabling Slices in Mirrors and RAID5

Metadevices" on page 108), use the `metareplace(1M)` command to enable an errored slice in a submirror. `metareplace(1M)` automatically starts a resync to get the new slice in sync with the rest of the mirror.

Example — Enabling a Slice in a Submirror

```
# metareplace -e d11 c1t4d0s7
d11: device c1t4d0s7 is enabled
```

The mirror `d11` has a submirror that contains slice, `c1t4d0s7`, which had a soft error. The `metareplace` command with the –`e` option enables the errored slice.

Note - If a physical disk is defective, you can either replace it with another available disk (and its slices) on the system as documented in "How to Replace a Slice in a Submirror (Command Line)" on page 111, or repair/replace the disk, format it, and run `metareplace` with the –`e` option as shown in this example.

▼ How to Replace a Slice in a Submirror (DiskSuite Tool)

Use this procedure to replace a slice within a submirror with a new slice. The partitioning information for any disk used by DiskSuite should be saved in a safe place before any errors occur.

Note - Before using this procedure, make sure the replacement slice has been correctly partitioned.

---

1. **Make sure you have met the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79), and have read the overview ("Overview of Replacing and Enabling Slices in Mirrors and RAID5 Metadevices" on page 105) and the preliminary information ("Preliminary Information For Replacing and Enabling Slices in Mirrors and RAID5 Metadevices" on page 108).**

2. **Double-click the errored Mirror object in the Objects list.**

   The mirror appears on the canvas. The submirror displays the error status.

3. **Point inside the Submirror object (Concat/Stripe object). Display the submirror's pop-up menu and select Info. View the size and condition of the slice.**

4. **Click Slices to open the Slice Browser.**

Locate an available slice of the same or greater size than the slice that needs replacing.

5. **Drag the replacement slice from the Slice Browser to the stripe rectangle of the submirror object that contains the slice you are replacing.**

6. **Click the top rectangle of the Mirror object then click Commit.**
   DiskSuite starts a resync of the replaced submirror.

7. **To verify that the mirror was committed, display the Configuration Log.**

8. **Validate the data.**

Note - When dragging the replacement slice to the object, be sure to point the cursor inside the rectangle that contains the device number of the errored slice, instead of other rectangles inside the Concat/Stripe object.

▼ How to Replace a Slice in a Submirror (Command Line)

After checking the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79), the overview ("Overview of Replacing and Enabling Slices in Mirrors and RAID5 Metadevices" on page 105), and the preliminary information ("Preliminary Information For Replacing and Enabling Slices in Mirrors and RAID5 Metadevices" on page 108), use the `metareplace(1M)` command to replace a slice in a submirror.

Example — Replacing a Failed Slice in a Mirror

```
# metastat d6
d6: Mirror
    Submirror 0: d16
      State: Okay
    Submirror 1: d26
      State: Needs maintenance
...
d26: Submirror of d6
    State: Needs maintenance
    Invoke: metareplace d6 c0t2d0s2 <new device>
...
# metareplace d6 c0t2d0s2 c0t2d2s2
```

(continued)

(Continuation)

```
d6: device c0t2d0s2 is replaced with c0t2d2s2
```

The `metastat` command confirms that mirror `d6` has a submirror, `d26`, with a slice in the “Needs maintenance” state. The `metareplace` command replaces the slice as specified in the “Invoke” line of the `metastat` output with another available slice on the system. The system confirms that the slice is replaced, and starts a resync of the submirror.

▼ How to Replace a Submirror (DiskSuite Tool)

To replace an entire submirror, first construct a new stripe or concatenation that is equal to or greater than the size of the submirror that is being replaced. The slice(s) used for the replacement submirror should be on different controllers than the other submirror. Refer to Chapter 2 for instructions on how to create a stripe or a concatenation.

1. **Make sure you have met the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79), and have read the overview (“Overview of Replacing and Enabling Slices in Mirrors and RAID5 Metadevices” on page 105) and the preliminary information (“Preliminary Information For Replacing and Enabling Slices in Mirrors and RAID5 Metadevices” on page 108).**

2. **Double-click an errored Mirror object in the Objects list.**
   The mirror appears on the canvas.

3. **Double-click the Concat/Stripe object in the Objects list that will replace the submirror.**
   The object appears on the canvas.

4. **Drag the Concat/Stripe object to the top of the submirror that is being replaced.**
   The new Concat/Stripe object replaces the errored one.

5. **Click the top rectangle of the Mirror object then click Commit.**
   A resync of the new submirror is initiated.

6. **To verify that the mirror was committed, display the Configuration Log.**

7. **Validate the data.**

8. **[Optional] Delete the replaced Concat/Stripe object.**

▼ How to Replace a Submirror (Command Line)

After checking the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79), the overview (“Overview of Replacing and Enabling Slices in Mirrors and RAID5 Metadevices” on page 105), and the preliminary information (“Preliminary Information For Replacing and Enabling Slices in Mirrors and RAID5 Metadevices” on page 108), use the `metadetach(1M)`, `metaclear(1M)`, `metatinit(1M)`, and `metattach(1M)` commands to replace an entire submirror.

Example — Replacing a Submirror in a Mirror

```
# metastat d20
d20: Mirror
    Submirror 0: d21
      State: Okay
    Submirror 1: d22
      State: Needs maintenance
...
# metadetach -f d20 d22
d20: submirror d22 is detached
# metaclear d22
d22: Concat/Stripe is cleared
# metainit d22 2 1 c1t0d0s2 1 c1t0d1s2
d22: Concat/Stripe is setup
# metattach d20 d22
d20: components are attached
```

The `metastat` command confirms that the two-way mirror `d20` has a submirror, `d22`, in the “Needs maintenance” state. In this case, the entire submirror will be cleared and recreated. The `metadetach` command detaches the errored submirror from the mirror using the `–f` option (this forces the detach to occur). The `metaclear` command clears the submirror. The `metainit` command recreates submirror `d22`, with new slices. The `metattach` command attaches the rebuilt submirror, and a mirror resync begins automatically.

Note - You temporarily lose the capability for data redundancy while the mirror is a one-way mirror.

▼ How to Enable a Slice in a RAID5 Metadevice (DiskSuite Tool)

Use this task to enable a slice in a RAID5 metadevice that is in the “Maintenance” state.

1. **Make sure you have met the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79), and have read the overview (“Overview of**

Replacing and Enabling Slices in Mirrors and RAID5 Metadevices" on page 105) and the preliminary information ("Preliminary Information For Replacing and Enabling Slices in Mirrors and RAID5 Metadevices" on page 108).

2. **Double-click the errored RAID5 object in the Objects list.**
   The RAID5 object appears on the canvas.

3. **Display the RAID5 object's pop-up menu and select Info.**
   The RAID Information window appears.

4. **Select the slice with a status of "Maintenance" and click Enable. Then click Close.**
   The status of the slice changes to "Enabled" and the status of the RAID5 changes to "Urgent (Uncommitted)."

5. **Select the RAID5 object then click Commit.**
   The RAID status changes to "Urgent-Resyncing," and the slice status changes to "Resyncing."

6. **When the resync is done, verify that the status of the RAID5 object is OK.**

7. **Validate the data.**

Note - If DiskSuite still reports the slice in the "Maintenance" state after enabling the slice, refer to "How to Replace a RAID5 Slice (DiskSuite Tool)" on page 115.

▼ How to Enable a Slice in a RAID5 Metadevice (Command Line)

After checking the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79), the overview ("Overview of Replacing and Enabling Slices in Mirrors and RAID5 Metadevices" on page 105), and the preliminary information ("Preliminary Information For Replacing and Enabling Slices in Mirrors and RAID5 Metadevices" on page 108), use the `metareplace(1M)` command to enable an errored slice in a RAID5 metadevice. `metareplace(1M)` automatically starts a resync to get the new slice in sync with the rest of the RAID5 metadevice.

Example — Enabling a Slice in a RAID5 Metadevice

```
# metareplace -e d20 c2t0d0s2
```

The RAID5 metadevice `d20` has a slice, `c2t0d0s2`, which had a soft error. The `metareplace` command with the `–e` option enables the slice.

Note - If a disk drive is defective, you can either replace it with another available disk (and its slices) on the system as documented in “How to Replace a RAID5 Slice (Command Line)” on page 116, or repair/replace the disk, format it, and run `metareplace` with the `–e` option.

▼ How to Replace a RAID5 Slice (DiskSuite Tool)

Use this procedure to replace an errored slice in a RAID5 metadevice in which only one slice is errored.

Caution - Replacing an errored slice when multiple slices are in error may cause data to be fabricated. The integrity of the data in this instance is questionable.

---

1. **Make sure you have met the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79), and have read the overview (“Overview of Replacing and Enabling Slices in Mirrors and RAID5 Metadevices” on page 105) and the preliminary information (“Preliminary Information For Replacing and Enabling Slices in Mirrors and RAID5 Metadevices” on page 108).**

2. **Double-click the errored RAID5 metadevice in the Objects list.**

   The object appears on the canvas.

3. **Click Slices to open the Slice Browser.**

4. **Select an available slice to replace the errored slice then drag it to the slice in the RAID5 metadevice object rectangle.**

   The slice must be at least as large as the smallest slice in the device.

5. **Click the top rectangle of the RAID5 metadevice object then click Commit.**

   During the replacement, the state of the metadevice and the new slice will be “Resyncing.” You can continue to use the metadevice while it is in this state.

6. **When the resync is done, verify that the status of the RAID5 object is OK.**

   You might need to select Rescan Configuration from the File menu for the status to be updated.

7. **Validate the data.**

▼ How to Replace a RAID5 Slice (Command Line)

This task replaces an errored slice of a RAID5 metadevice in which only one slice is errored.

Caution - Replacing an errored slice when multiple slices are in error may cause data to be fabricated. The integrity of the data in this instance is questionable.

After checking the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79), the overview (“Overview of Replacing and Enabling Slices in Mirrors and RAID5 Metadevices” on page 105), and the preliminary information (“Preliminary Information For Replacing and Enabling Slices in Mirrors and RAID5 Metadevices” on page 108), follow this example to replace an errored slice in a RAID5 metadevice. Refer to the `metareplace(1M)` man page for more information.

Example — Replacing a RAID5 Slice

```
# metastat d1
d1: RAID
State: Needs Maintenance
    Invoke: metareplace d1 c0t14d0s6 <new device>
    Interlace: 32 blocks
    Size: 8087040 blocks
Original device:
    Size: 8087520 blocks
 Device              Start Block  Dbase State        Hot Spare
 c0t9d0s6                 330     No    Okay
 c0t13d0s6                330     No    Okay
 c0t10d0s6                330     No    Okay
 c0t11d0s6                330     No    Okay
 c0t12d0s6                330     No    Okay
 c0t14d0s6                330     No    Maintenance

# metareplace d1 c0t14d0s6 c0t4d0s6
d1: device c0t14d0s6 is replaced with c0t4d0s6
# metatstat d1
d1: RAID
    State: Resyncing
    Resync in progress: 98% done
    Interlace: 32 blocks
    Size: 8087040 blocks
Original device:
    Size: 8087520 blocks
 Device              Start Block  Dbase State        Hot Spare
 c0t9d0s6                 330     No    Okay
 c0t13d0s6                330     No    Okay
 c0t10d0s6                330     No    Okay
 c0t11d0s6                330     No    Okay
 c0t12d0s6                330     No    Okay
```

(continued)

(Continuation)

```
c0t4d0s6                330     No    Resyncing
```

The `metastat` command displays the action to take to recover from the errored slice in the `d1` RAID5 metadevice. After locating an available slice, the `metareplace` command is run, specifying the errored slice first, then the replacement slice. (If no other slices are available, run the `metareplace(1M)` command with the `-e` option to attempt to recover from possible soft errors by resyncing the errored device.) If multiple errors exist, the slice in the “Maintenance” state must first be replaced or enabled first. Then the slice in the “Last Erred” state can be repaired. After the `metareplace`, `metastat` monitors the progress of the resync. During the replacement, the state of the metadevice and the new slice will be “Resyncing.” You can continue to use the metadevice while it is in this state.

Note - You can use the `metareplace(1M)` command on non-errored devices to change a disk (slice). This can be useful for tuning performance of RAID5 metadevices.

Preliminary Information for Replacing Hot Spare Pools

- Hot spares can be replaced in any or all the hot spare pools to which they have been associated. However, hot spares that are in the “In Use” state cannot be replaced by other hot spares.
- The order of hot spares in the hot spare pools is not changed when a replacement occurs.
- Hot spares are placed in the “Broken” sate after an I/O error occurs. To fix this condition, first repair or replace the broken hot spare slice. Then bring the slice back to the “available” state by using DiskSuite Tool or the `metahs(1M)` command with the `-e` option.

Note - A submirror or RAID5 metadevice may be using a hot spare in place of an errored slice. When that errored slice is enabled or replaced, the hot spare is marked “available” in the hot spare pool, and is ready for use.

▼ How to Replace a Hot Spare in a Hot Spare Pool (DiskSuite Tool)

1. **Make sure you have met the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79), and have read the preliminary information (“Preliminary Information for Replacing Hot Spare Pools” on page 117).**

2. **Double-click a Hot Spare Pool object in the Objects list.**

   The object appears on the canvas.

3. **Click Slices to open the Slice Browser.**

4. **Locate a slice of the same size as the slice that needs replacing. Drag the slice to the rectangle of the Hot Spare Pool object that contains the slice you are replacing.**

5. **Click the top rectangle of the Hot Spare Pool object then click Commit.**

6. **To verify that the hot spare pool was committed, display the Configuration Log.**

▼ How to Replace a Hot Spare in a Hot Spare Pool (Command Line)

After checking the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79), and the preliminary information (“Preliminary Information for Replacing Hot Spare Pools” on page 117), use `metahs(1M)` to replace the hot spare. Refer to the `metahs(1M)` man page for more information.

Example — Replacing a Hot Spare in One Hot Spare Pool

```
# metastat hsp003
hsp003: 1 hot spare
        c0t2d0s2                Broken       5600 blocks
# metahs -r hsp003 c0t2d0s2 c3t1d0s2
hsp003: Hotspare c0t2d0s2 is replaced with c3t1d0s2
```

The `metastat` command makes sure that the hot spare is not in use. The `metahs –r` command replaces hot spare `/dev/dsk/c0t2d0s2` with `/dev/dsk/c3t1d0s2` in the hot spare pool `hsp003`.

Example — Replacing a Hot Spare in All Associated Hot Spare Pools

```
# metahs -r all c1t0d0s2 c3t1d0s2
hsp001: Hotspare c1t0d0s2 is replaced with c3t1d0s2
hsp002: Hotspare c1t0d0s2 is replaced with c3t1d0s2
hsp003: Hotspare c1t0d0s2 is replaced with c3t1d0s2
```

The keyword `all` replaces hot spare `/dev/dsk/c1t0d0s2` with `/dev/dsk/c3t1d0s2` in all its associated hot spare pools.

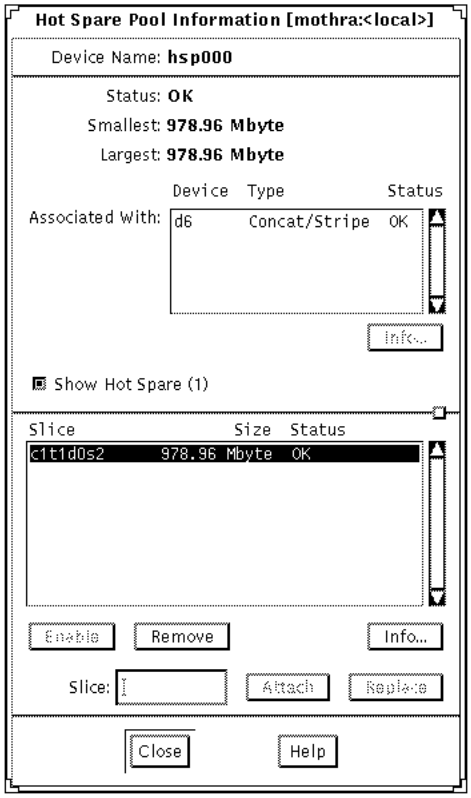
▼ How to Enable a Hot Spare (DiskSuite Tool)

Use this procedure to enable a hot spare (make it available) after it has been repaired.

1. **Make sure you have met the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79), and have read the preliminary information (“Preliminary Information for Replacing Hot Spare Pools” on page 117).**

2. **Double-click a Hot Spare Pool object in the Objects list.**
   The object appears on the canvas.

3. **Display the Hot Spare Pool object’s pop-up menu and select Info.**
   The Hot Spare Pool Information window appears.



4. **Select the slice in the Slice list that was repaired. Then click Enable.**

5. **Click Close to close the Hot Spare Information window.**

6. **Click the top rectangle of the Hot Spare Pool object then click Commit.**

7. **Verify that the status of the Hot Spare Pool object changes to "OK."**

▼ How to Enable a Hot Spare (Command Line)

After checking the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79), and the preliminary information ("Preliminary Information for Replacing Hot Spare Pools" on page 117), use the `metahs(1M)` command to bring a hot spare back to the "available" state. For more information, refer to the `metahs(1M)` man page.

Example — Enabling a Hot Spare

```
# metahs -e c0t0d0s2
```

This example places the hot spare `/dev/dsk/c0t0d0s2` in the available state after it has been repaired. You do not need to specify a hot spare pool.

Repairing Trans Metadevice Problems

Because a trans metadevice is a “layered” metadevice, consisting of a master device and logging device, and because the logging device can be shared among file systems, repairing an errored trans metadevice requires special recovery tasks.

Any device errors or file system panics must be dealt with using the command line utilities.

File System Panics

If a file system detects any internal inconsistencies while it is in use, it will panic the system. If the file system is setup for UFS logging, it notifies the trans metadevice that it needs to be checked at reboot. The trans metadevice transitions itself to the “Hard Error” state. All other trans metadevices sharing the same logging device also go into the “Hard Error” state.

At reboot, `fsck` checks and repairs the file system and transitions the file system back to the “Okay” state. `fsck` does this for all trans metadevices listed in the `/etc/vfstab` file for the affected logging device.

Trans Metadevice Errors

Device errors can cause data loss. Read errors occurring on a logging device can cause significant data loss. For this reason, it is strongly recommended that you mirror the logging device.

If a device error occurs on either the master device or the logging device while the trans metadevice is processing logged data, the device transitions from the “Okay” state to the “Hard Error” state. If the device is either in the “Hard Error” or “Error” state, either a device error has occurred, or a file system panic has occurred.

Note - Any devices sharing the errored logging device also go the “Error” state.

▼ How to Recover a Trans Metadevice With a File System Panic (Command Line)

For file systems that `fsck` cannot repair, run `fsck` on each trans metadevice whose file systems share the affected logging device.

Example — Recovering a Trans Metadevice

```
# fsck /dev/md/rdsk/trans
```

Only after all of the affected trans metadevices have been checked and successfully repaired will `fsck` reset the state of the errored trans metadevice to "Okay."

▼ How to Recover a Trans Metadevice With Hard Errors (Command Line)

Use this procedure to transition a trans metadevice to the "Okay" state.

Refer to "How to Check the Status of Metadevices and Hot Spare Pools (Command Line)" on page 91 to check the status of a trans metadevice.

If either the master or log devices encounter errors while processing logged data, the device transitions from the "Okay" state to the "Hard Error" state. If the device is in the "Hard Error" or "Error" state, either a device error or file system panic occurred. Recovery from both scenarios is the same.

Note - If a log (logging device) is shared, a failure in any of the slices in a trans metadevice will result in all slices or metadevices associated with the trans metadevice switching to an errored state.

The high-level steps in this procedure are:

- Unmounting the affected file system(s)
- Backing up any accessible data
- Fixing the device error
- Repairing the file system (`fsck(1M)` or `newfs(1M)`)

1. **After checking the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79) and the preliminary information ("Repairing Trans Metadevice Problems" on page 121), run the `lockfs(1M)` command to determine which file systems are locked.**

   ```
   # lockfs
   ```

Affected file systems will be listed with a lock type of `hard`. Every file system sharing the same logging device will be hard locked.

2. **Unmount the affected file system(s).**

   You can unmount locked file systems even if they were in use when the error occurred. If the affected processes try to access an opened file or directory on the hard locked or unmounted file system, an EIO error is returned.

3. **[Optional] Back up any accessible data.**

   Before attempting to fix the device error, you may want to recover as much data as possible. If your backup procedure requires a mounted file system (such as `tar` or `cpio`), you can mount the file system read-only. If your backup procedure does not require a mounted file system (such as `dump` or `volcopy`), you can access the trans metadevice directly.

4. **Fix the device error.**

   At this point, any attempt to open or mount the trans metadevice for read/write access starts rolling all accessible data on the logging device to the appropriate master device(s). Any data that cannot be read or written is discarded. However, if you open or mount the trans metadevice for read-only access, the log is simply rescanned and not rolled forward to the master device(s), and the error is not fixed. In other words, all of the data on the master and logging devices remains unchanged until the first read/write open or mount.

5. **Run `fsck(1M)` to repair the file system, or `newfs(1M)` if you need to restore data.**

   Run `fsck` on all of the trans metadevices sharing the same logging device. When all of these trans metadevices have been repaired by `fsck`, they then revert to the "Okay" state.

   The `newfs(1M)` command will also transition the file system back to the "Okay" state, but will destroy all of the data on the file system. `newfs(1M)` is generally used when you plan to restore file systems from backup.

   The `fsck(1M)` or `newfs(1M)` commands must be run on all of the trans metadevices sharing the same logging device before these devices revert back to the "Okay" state.

6. **Run the `metastat(1M)` command to verify that the state of the affected devices has reverted to "Okay."**

Example — Logging Device Error

```
# metastat d5
d5: Trans
    State: Hard Error
```

(Continuation)

```
    Size: 10080 blocks
    Master Device: d4
    Logging Device: c0t0d0s6

d4: Mirror
    State: Okay
...
c0t0d0s6: Logging device for d5
    State: Hard Error
    Size: 5350 blocks
...
# fsck /dev/md/rdsk/d5
** /dev/md/rdsk/d5
** Last Mounted on /fs1
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups
WARNING: md: logging device: /dev/dsk/c0t0d0s6 changed state to
Okay
4 files, 11 used, 4452 free (20 frags, 554 blocks, 0.4%
fragmentation)
# metastat d5
d5: Trans
    State: Okay
    Size: 10080 blocks
    Master Device: d4
    Logging Device: c0t0d0s6

d4: Mirror
    State: Okay
...

c0t0d0s6: Logging device for d5
    State: Okay
...
```

This example fixes a trans metadevice, `d5`, which has a logging device in the “Hard Error” state. You must run `fsck` on the trans device itself. This transitions the state of the trans metadevice to “Okay.” The `metastat` confirms that the state is “Okay.”

Expanding Slices and Metadevices

This section contains the tasks to expand, or add space to, a slice (non-metadevice) or a metadevice. For example, if a file system fills up a concatenated metadevice, you can add more slices, then “grow” the file system to the newly added space.

Preliminary Information for Expanding Slices and Metadevices

- A metadevice, regardless if it is used for a file system, application, or database, can be expanded by adding slices. This includes: striped metadevices, concatenations, and concatenated stripes; mirrors (submirrors); RAID5 devices; and trans metadevices.
- In most cases, you can concatenate a metadevice that contains an existing file system while the file system is in use. Then, as long as the file system is UFS, it can be expanded (with the `growfs(1M)` command) to fill the larger space without interrupting access to the data.
- Once a file system is expanded, it cannot be shrunk. This is a limitation of UFS.
- Applications and databases using the raw device must have their own method to "grow" the added space so that the application can recognize it. DiskSuite does not provide this capability.
- When a slice is added to a RAID5 metadevice, it becomes a concatenation to the device. The new slice does not contain parity information. However, data on the new slice is protected by the overall parity calculation that takes place for the device.
- Once a slice is attached to a metadevice, it cannot be removed.
- You can expand a logging device by adding additional slices. You do not need to run the `growfs(1M)` command, as DiskSuite automatically recognizes the additional space on reboot.

▼ How to Expand a Slice Containing Existing Data (DiskSuite Tool)

Use this task to create a concatenation from a single slice that has run out of space. You can use this task for a file system or an application, such as a database. The high-level steps in this procedure are:

- Putting the slice needing more space into a concatenation
- Unmounting then remounting the file system on the metadevice, or rebooting
- Adding another slice to the concatenation
- Growing the file system or application

1. **Make sure you have met the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79) and have read the preliminary information ("Preliminary Information for Expanding Slices and Metadevices" on page 125).**

2. **Click the Concat/Stripe template.**

   An unassigned and uncommitted Concat/Stripe object appears on the canvas. The metadevice name is automatically assigned.

3. **[Optional] To change the default metadevice name, display the object's pop-up menu and choose Info. Type the new metadevice name in the Device Name field and click Attach. Then click Close.**

4. **Click Slices to open the Slice Browser.**

5. **Select and drag the slice containing the data into the Concat/Stripe object.**

   If a dialog box appears that the slice is mounted, click Continue.

6. **Click the top rectangle of the Concat/Stripe object then click Commit.**

   Mounted file system only: A Commit Warning dialog box appears. Click Really Commit. (As long as the file system has an entry in the `/etc/vfstab` file, DiskSuite Tool updates the entry to use the metadevice name. If the file system is mounted by hand, you need to use the block metadevice name whenever the file system is mounted.)

7. **Mounted file system only: Unmount then remount the file system. If the file system is busy, you'll need to reboot.**

8. **In DiskSuite Tool, open the Concat object and Slice Browser. Drag another slice into the object.**

9. **Make sure the object is selected then click Commit.**

   Mounted file system only: A GrowFS dialog box appears. Click Grow Now to begin running the `growfs(1M)` command. A GrowFS Running message appears. If you click Terminate GrowFS, the command is aborted. Otherwise, when the `growfs(1M)` command finishes, you are returned to the DiskSuite Tool window.

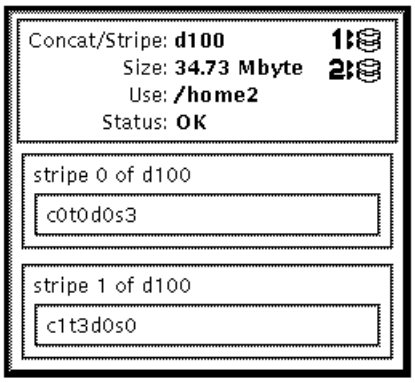
   Application using the raw device: Such applications must have their own method, outside of DiskSuite, to recognize the expanded space.

10. **The Configuration Log shows that the concatenation was committed.**

Note - During the expansion, the file system is locked and not available for write access. Write accesses are transparently suspended and are restarted when `growfs(1M)` unlocks the file system. Read accesses are not affected, though access times are not kept while the lock is in effect.

Example — Expanded Concatenation Object

This example shows a committed concatenation. The initial object consisted of a file system of one slice, `c0t0d0s3`. Slice `c1t3d0s0` was then concatenated, and the `growfs(1M)` command was run automatically to make the entire space available.



▼ How to Expand a Slice Containing Existing Data (Command Line)

After checking the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79), and the preliminary information (“Preliminary Information for Expanding Slices and Metadevices” on page 125), follow this example to expand a slice containing a mounted file system.

Example — Expanding a File System By Creating a Concatenation

```
# umount /docs
# metainit d25 2 1 c0t1d0s2 1 c0t2d0s2
d25: Concat/Stripe is setup
(Edit the /etc/vfstab file so that the file system references
the metadevice d25)
# mount /docs
```

This example creates a concatenation called `d25` out of two slices, `/dev/dsk/c0t1d0s2` (which contains a file system mounted on `/docs`) and `/dev/dsk/c0t2d0s2`. The file system must first be unmounted.

Caution - The first slice in the `metainit(1M)` command must be the slice containing the file system. If not, you will erase your data.

Next, the entry for the file system in the `/etc/vfstab` file is changed (or entered for the first time) to reference the metadevice. For example, the following line:

```
/dev/dsk/c0t1d0s2 /dev/rdsk/c0t1d0s2 /docs ufs 2 yes -
```

should be changed to:

```
/dev/md/dsk/d25 /dev/md/rdsk/d25 /docs ufs 2 yes -
```

Lastly, the file system is remounted.

Where to Go From Here

For a UFS, run the `growfs(1M)` command on the metadevice. Refer to "How to Grow a File System (Command Line)" on page 138.

An application, such as a database, that uses the raw metadevice must have its own way of recognizing the metadevice, or of growing the added space.

▼ How to Expand an Existing Concat/Stripe (DiskSuite Tool)

This task assumes that you are adding an additional stripe to an existing concatenation or stripe. If you need to recreate a concatenated stripe as part of disaster recovery, refer to "How to Recreate a Stripe or Concatenation After Slice Failure (DiskSuite Tool)" on page 102.

A concatenated stripe enables you to expand an existing stripe or concatenation. For example, if a stripe has run out of space, you can make it into a concatenated stripe, and expand it without having to back up and restore data.

Note - If you drag multiple slices into an existing Concat/Stripe object, you are given the optional of making the slices into a concatenation or a stripe.

---

1. **Make sure you have met the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79) and have read the preliminary information ("Preliminary Information for Expanding Slices and Metadevices" on page 125).**

2. **Double-click the Concat/Stripe object in the Objects list.**

   DiskSuite Tool displays the Concat/Stripe object on the canvas.

3. **Click Slices to open the Slice Browser window.**

4. **Select the slice(s) you want to concatenate as another stripe. Drag the slice(s) to the top rectangle of the metadevice object.**

   If you drag multiple slices, a dialog box prompts you to choose how you want to add the slices, either as a stripe or a concat. Click either Stripe or Concat.

   The additional striped metadevice or concatenated metadevice is added at the bottom of the Concat/Stripe object.

5. **Click the top rectangle of the Concat/Stripe object. Then click Commit.**

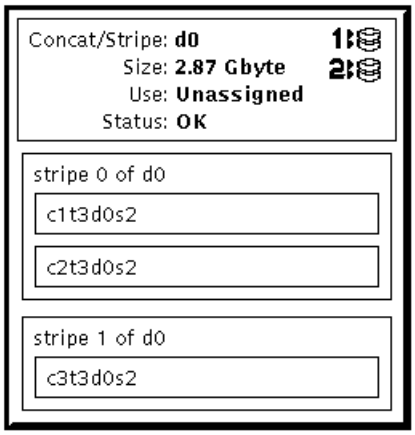
   Mounted file system only: A GrowFS dialog box appears. Click Grow Now to begin running the `growfs(1M)` command. A GrowFS Running message appears. If you click Terminate GrowFS, the command is aborted. Otherwise, when the `growfs(1M)` command finishes, you are returned to the DiskSuite Tool window.

   Application using the raw device: Such applications must have their own method, outside of DiskSuite, to recognize the expanded space.

6. **To verify that the concatenated stripe was committed, display the Configuration Log.**

Example — Concatenated Stripe Object

This example shows a striped metadevice consisting of two slices to which another slice has been added. The Concat/Stripe object displays the slices in two stripe rectangles labeled stripe 0 (the original stripe) and stripe 1 (the added stripe).



▼ How to Expand an Existing Stripe (Command Line)

This procedure assumes that you are adding an additional stripe to an existing stripe. If you need to recreate a concatenated stripe using the `metainit(1M)` command as part of disaster recovery, refer to “How to Recreate a Stripe or Concatenation After Slice Failure (Command Line)” on page 104.

A concatenated stripe enables you to expand an existing stripe. For example, if a stripe has run out of space, you can make it into a concatenated stripe, and expand it without having to back up and restore data.

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. When using the `metattach(1M)` command to add multiple slices to an existing striped metadevice, they must be added as a stripe.

After checking the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79), and the preliminary information ("Preliminary Information for Expanding Slices and Metadevices" on page 125), use the `metattach(1M)` command to create the concatenated stripe. Refer to the `metattach(1M)` man page for more information.

Example — Creating a Concatenated Stripe By Attaching a Single Slice

```
# metattach d2 c1t2d0s2
d2: components are attached
```

This example attaches a slice to an existing stripe, `d2`. The system verifies that the slice is attached.

Example — Creating a Concatenated Stripe By Adding the Same Number of Slices in the Existing Metadevice

```
# metattach d25 c1t2d0s2 c1t2d1s2 c1t2d3s2
d25: components are attached
```

This example takes an existing three-way striped metadevice, `d25`, and concatenates another three-way stripe. Because no interlace value is given for the attached slices, they inherit the interlace value configured for `d25`. The system verifies that the Concat/Stripe object has been set up.

Note - Depending on the type of application, by attaching the same number of slices, the metadevice might not experience a performance degradation.

Example — Creating a Concatenated Stripe From Scratch

```
# metainit d1 3 2 c0t0d0s2 c1t0d0s2 -i 16k \
  2 c1t2d0s2 c1t2d1s0 -i 32k \
  2 c2t0d0s2 c2t0d1s2
d1: Concat/Stripe is setup
```

Normally, you would not create a metadevice such as this one from scratch. The example illustrates that `d1` is a concatenation of three stripes (the first number `3`). The first stripe consists of two slices (the number `2` following the number `3`). The `–i 16k` specifies an interlace of 16 Kbytes. The second stripe (as indicated by the number `2` on the second line) consists of two slices, and uses an interlace of 32 Kbytes. The last stripe consists of a two slices. 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.

Where To Go From Here

For a UFS, run the `growfs(1M)` command on the metadevice. Refer to “How to Grow a File System (Command Line)” on page 138.

An application, such as a database, that uses the raw metadevice must have its own way of recognizing the metadevice, or of growing the added space.

To prepare a newly created concatenated stripe for a file system, refer to “How to Create a File System on a Metadevice (Command Line)” on page 74.

▼ How to Expand a Mirror (DiskSuite Tool)

Use this task to expand a mirror’s submirrors. You need to expand each submirror. You can expand a submirror while it is in use, without having to take it offline.

1. **Make sure you have met the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79) and have read the preliminary information (“Preliminary Information for Expanding Slices and Metadevices” on page 125).**

2. **Double-click an existing Mirror object in the Objects list.**

   The object appears on the canvas.

3. **To add a slice to the submirror, click Slices to open the Slice Browser. To add a Concat/Stripe object, select one from the Objects list.**

4. **Select and drag a slice from the Slice Browser, or a Concat/Stripe object from the Objects list, to the top of one of the submirror rectangles in the Mirror object.**

Select a slice or concat/stripe of the appropriate size. Use Control-click to select multiple slices.

5. **When you drag the slice or Concat/Stripe object to the Mirror object, a Warning dialog box appears because the submirror will be a different size after the addition. Click Continue.**

   The slice or concat/stripe object is added to the bottom of the submirror rectangle.

6. **Select and drag a slice or Concat/Stripe object to the top of one of the second submirror rectangles in the Mirror object.**

   A Validation dialog box appears. Click OK.

   If you have a third submirror, repeat this step.

7. **Click the top of the Mirror object then click Commit.**

   Mounted file system only: A GrowFS dialog box appears. Click Grow Now to begin running the `growfs(1M)` command. A GrowFS Running message appears. If you click Terminate GrowFS, the command is aborted. Otherwise, when the `growfs(1M)` command finishes, you are returned to the DiskSuite Tool window.

   Application using the raw device: Such applications must have their own method, outside of DiskSuite, to recognize the expanded space.

8. **To verify that the mirror was committed, display the Configuration Log.**

▼ How to Expand a Mirror (Command Line)

After checking the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79), and the preliminary information (“Preliminary Information for Expanding Slices and Metadevices” on page 125), use the `metattach(1M)` command to attach additional slices to each submirror. Each submirror in a mirror must be expanded. Refer to the `metattach(1M)` man page for more information.

Example — Expanding a Two-Way Mirror Containing a Mounted File System

```
# metastat
d8: Mirror
    Submirror 0: d9
      State: Okay
    Submirror 1: d10
      State: Okay
...
```

(continued)

```
# metattach d9 c0t2d0s5
d9: component is attached
# metattach d10 c0t3d0s5
d10: component is attached
```

This example shows how to expand a mirrored mounted file system by concatenating two disk drives to the mirror's two submirrors. The mirror is named `d8` and contains two submirrors named `d9` and `d10`.

Where to Go From Here

For a UFS, run the `growfs(1M)` command on the mirror metadevice. Refer to "How to Grow a File System (Command Line)" on page 138.

An application, such as a database, that uses the raw metadevice must have its own way of growing the added space.

▼ How to Expand a RAID5 Metadevice (DiskSuite Tool)

Use this procedure to expand an existing RAID5 metadevice by concatenating another slice. In general, this is a short-term solution to a RAID5 metadevice running out of space. For performance reasons, it is best to have a "pure" RAID5 metadevice.

1. **Make sure you have met the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79) and have read the preliminary information ("Preliminary Information for Expanding Slices and Metadevices" on page 125).**

2. **Double-click an existing RAID5 metadevice object in the Objects list.**

   The object appears on the canvas.

3. **Click Slices to open the Slice Browser. Then select the slice(s) to be concatenated to the RAID5 metadevice.**

4. **Drag the selected slice(s) to the top of the RAID5 object.**

   The slice must be at least as large as the smallest slice in the RAID5 metadevice. You can select multiple slices at the same time by using the Control-click technique.

   DiskSuite Tool displays the additional slice(s) at the bottom of the object.

5. **Click the top rectangle of the RAID5 object. Then click Commit.**

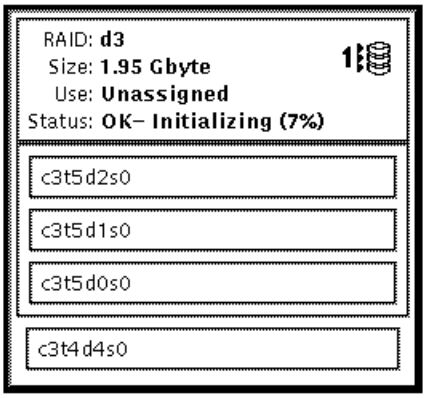
Mounted file system only: A GrowFS dialog box appears. Click Grow Now to begin running the `growfs(1M)` command. A GrowFS Running message appears. If you click Terminate GrowFS, the command is aborted. Otherwise, when the `growfs(1M)` command finishes, you are returned to the DiskSuite Tool window.

Application using the raw device: Such applications must have their own method, outside of DiskSuite, to recognize the expanded space.

6. **To verify that the RAID5 metadevice was committed, display the Configuration Log.**

Example — Expanded RAID5 Metadevice

This example shows a RAID5 metadevice, `d3`, to which slice `/dev/dsk/c3t4d4s0` has been added.



▼ How to Expand a RAID5 Metadevice (Command Line)

After checking the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79), and the preliminary information (“Preliminary Information for Expanding Slices and Metadevices” on page 125), use the `metattach(1M)` command to attach additional slices to a RAID5 metadevice. Refer to the `metattach(1M)` man page for more information.

In general, this is a short-term solution to a RAID5 metadevice running out of space. For performance reasons, it is best to have a “pure” RAID5 metadevice.

Example — Adding a Slice to a RAID5 Metadevice

```
# metattach d2 c2t1d0s2
d2: column is attached
```

This example shows the addition of slice `/dev/dsk/c2t1d0s2` to an existing RAID5 metadevice named `d2`.

Where to Go From Here

For a UFS, run the `growfs(1M)` command on the RAID5 metadevice. Refer to “How to Grow a File System (Command Line)” on page 138.

An application, such as a database, that uses the raw metadevice must have its own way of growing the added space.

▼ How to Expand a Trans Metadevice (DiskSuite Tool)

You can expand a master device within a trans metadevice as long as the master device is a metadevice. To expand a master that consists of a slice, you must tear down (clear) the trans, put the slice into a metadevice, then recreate the trans.

Use this procedure to expand a master device that makes up the trans metadevice.

Note - If the master device is a mirror, you need to expand each submirror.

---

1. **Make sure you have met the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79) and have read the preliminary information (“Preliminary Information for Expanding Slices and Metadevices” on page 125).**

2. **Double-click an existing Trans Metadevice object in the Objects list.**

   The object appears on the canvas.

3. **Expand the master device.**
   - Concat or Stripe: see “How to Expand an Existing Concat/Stripe (DiskSuite Tool)” on page 128
   - Mirror: see “How to Expand a Mirror (DiskSuite Tool)” on page 131
   - RAID5 metadevice: see “How to Expand a RAID5 Metadevice (DiskSuite Tool)” on page 133

4. **If the log is not mirrored, a dialog box appears. Click OK.**

5. **Click the top rectangle of the Trans Metadevice object then click Commit.**

   Mounted file system: A GrowFS dialog box appears. Click Grow Now to begin running the `growfs(1M)` command. A GrowFS Running message appears. If you click Terminate GrowFS, the command is aborted. Otherwise, when the `growfs(1M)` command finishes, you are returned to the DiskSuite Tool window.

   Application using the raw device: Such applications must have their own method, outside of DiskSuite, to recognize the expanded space.

6. **To verify that the trans metadevice was committed, display the Configuration Log.**

▼ How to Expand a Trans Metadevice (Command Line)

You can expand a master device within a trans device as long as the master is a metadevice. To expand a master that consists of a slice, you must tear down (clear) the trans, put the slice into a metadevice, then recreate the trans.

After checking the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79), and the preliminary information ("Preliminary Information for Expanding Slices and Metadevices" on page 125), use the `metattach(1M)` command to attach additional slices to a master device within the trans metadevice. Refer to the `metattach(1M)` man page for more information.

Note - If the master device is a mirror, you need to expand each submirror.

Example — Expanding a Mirrored Master Device Within a Trans Metadevice

```
# metastat d10
d10: Trans
    State: Okay
    Size: 102816 blocks
    Master Device: d0
    Logging Device: d1
d0: Mirror
    Submirror 0: d11
      State: Okay
...
    Submirror 1: d12
      State: Okay
...
# metattach d11 c0t2d0s5
```

(continued)

```
d11: component is attached
# metattach d12 c0t3d0s5
d12: component is attached
```

This example expands a trans device, `d10`, whose master device consists of a two-way mirror, `d0`, which contains two submirrors, `d11` and `d12`. The `metattach(1M)` command is run on each submirror. The system confirms that each slice was attached.

Where to Go From Here

For a UFS, run the `growfs(1M)` command on the trans metadevice (not the master device). Refer to “How to Grow a File System (Command Line)” on page 138.

An application, such as a database, that uses the raw metadevice must have its own way of growing the added space.

Growing a File System

This section describes how to grow a UFS that uses a metadevice that was expanded with additional slices.

Preliminary Information For Growing a File System

- After a metadevice containing a file system is expanded (more space is added), if that metadevice contains a UFS, you need to “grow” the file system to recognize the added space.
- If you use DiskSuite Tool to add additional space to a metadevice, you receive a `growfs` message automatically. If you use the command line interface, you must manually grow the file system with the `growfs(1M)` command.
- An application, such as a database, that uses the raw device must have its own method to grow added space. DiskSuite does not provide this functionality.
- The `growfs(1M)` command expands the file system, even while mounted.

▼ How to Grow a File System (Command Line)

After checking the prerequisites on "Prerequisites for Maintaining DiskSuite Objects" on page 79, and the preliminary information on "Preliminary Information For Growing a File System" on page 137, use the `growfs(1M)` command to grow a UFS. Refer to the `growfs(1M)` man page for more information.

Example — Growing a File System on a Concatenation

```
# df -k
Filesystem            kbytes    used   avail capacity  Mounted on
...
/dev/md/dsk/d10        69047   65426       0   100%    /home2
...
# growfs -M /home2 /dev/md/rdsk/d10
/dev/md/rdsk/d10:       295200 sectors in 240 cylinders of 15 tracks, 82 sectors
        144.1MB in 15 cyl groups (16 c/g, 9.61MB/g, 4608 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 19808, 39584, 59360, 79136, 98912, 118688, 138464, 158240, 178016, 197792,
 217568, 237344, 257120, 276896,
# df -k
Filesystem            kbytes    used   avail capacity  Mounted on
...
/dev/md/dsk/d10       138703   65426   59407    53%    /home2
...
```

A new slice was added to a concatenation, `d10`, which contains the mounted file system `/home2`. The `growfs` command specifies the mount point with the `–M` option to be `/home2`, which is expanded onto the raw device `/dev/md/rdsk/d10`. The file system will span the entire metadevice when the `growfs(1M)` command is done. Use the `df -k` command before and after to verify the total disk capacity.

The `growfs(1M)` command will "write-lock" (see `lockfs(1M)`) a mounted file system when expanding. The length of time the file system is write-locked can be shortened by expanding the file system in stages. For instance, to expand a 1 Gbyte file system to 2 Gbytes, the file system can be grown in 16 Mbyte stages using the `–s` option to specify the total size of the new file system at each stage.

During the expansion, the file system is not available for write access because of write-lock. Write accesses are transparently suspended and are restarted when `growfs(1M)` unlocks the file system. Read accesses are not affected, though access times are not kept while the lock is in effect.

Note - For mirror and trans metadevices, always run the `growfs(1M)` command on the top-level metadevice, not a submirror or master device, even though space is added to the submirror or master device.

Renaming Metadevices

This section describes DiskSuite's metadevice renaming capability.

Preliminary Information for Renaming Metadevices

- 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.
- Before renaming a metadevice, make sure that it is not currently 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 rename any metadevice except a metadevice that is being used as a logging device. To perform the equivalent of renaming a metadevice used as a logging device, detach the logging device then reattach a new logging device of the desired name. Or detach the logging device, rename it, then reattach it to the trans metadevice.
- You cannot rename a trans metadevice while it has an attached logging device. To rename a trans metadevice, first detach the logging device.
- Hot spare pools cannot be renamed once they are created.
- Metadevices within a diskset can be renamed. You cannot rename metadevices from one diskset to another.

Note - The `metarename` command with the `–x` option can "switch" metadevices that have a parent-child relationship. Refer to "Metadevice Name Switching" on page 278.

▼ How to Rename a Metadevice (DiskSuite Tool)

You cannot rename a metadevice that is mounted or open. You cannot rename a trans metadevice that has a logging device attached.

1. **Make sure you have met the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79), and have read the preliminary information ("Preliminary Information for Renaming Metadevices" on page 139).**

2. **Stop all access to the metadevice.**

   For example, if the metadevice contains a mounted file system, unmount it.

3. **Display the metadevice's Info window.**

4. **Type the new metadevice name in the Device Name field then click Attach.**

   If you do not see the Device Name field on the Info window, then the metadevice is still in use. Make sure you have stopped access to the metadevice.

5. **Click Close to close the Info window.**

   The metadevice object displays the new metadevice name.

   If the metadevice is used for a file system with an entry in the `/etc/vfstab` file, DiskSuite Tool changes the entry to reference the new metadevice name.

6. **Resume access to the metadevice.**

   For example, mount the file system.

7. **To verify that the rename was committed, display the Configuration Log.**

▼ How to Rename a Metadevice (Command Line)

After checking the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79), and the preliminary information ("Preliminary Information for Renaming Metadevices" on page 139), use the `metarename(1M)` command to rename a metadevice. Refer to the `metarename(1M)` man page for more information.

Example — Renaming a Metadevice Used For a File System

```
# umount /home
# metarename d10 d100
d10: has been renamed to d100
(Edit the /etc/vfstab file so that the file system references
the new metadevice)
# mount /home
```

The metadevice `d10` is renamed to metadevice `d100`. Because `d10` contains a mounted file system, the file system must be unmounted before the rename. If the metadevice is used for a file system with an entry in the `/etc/vfstab` file, the entry must be changed to reference the new metadevice name. For example, the following line:

```
/dev/md/dsk/d10 /dev/md/rdsk/d10 /docs ufs 2 yes -
```

should be changed to:

```
/dev/md/dsk/d100 /dev/md/rdsk/d100 /docs ufs 2 yes -
```

Lastly, the file system is remounted.

Working With Mirrors

This section describes maintenance tasks that you perform on mirrors, including unmirroring a file system, attaching and detaching submirrors, and offlining and onlining submirrors.

Preliminary Information for Mirrors

- **Unmirroring –** DiskSuite Tool does not support unmirroring root (/), /opt, /usr, or swap, or any other file system that cannot be unmounted while the system is running. Instead, use the command line procedure for these file systems.
- **Attaching –** You can attach a submirror to a mirror without interrupting service. You attach submirrors to mirrors to create two- and three-way mirrors.
- **Detach vs. Offline –** When you place a submirror offline, you prevent the mirror from reading from and writing to the submirror, but you preserve the submirror's logical association to the mirror. While the submirror is offline, DiskSuite keeps track of all writes to the mirror and they are written to the submirror when it is brought back online. By performing an optimized resync, DiskSuite only has to resync data that has changed, not the entire submirror. When you detach a submirror, you sever its logical association to the mirror. Typically you place a submirror offline to perform maintenance; you detach a submirror to remove it.

▼ How to Unmirror a File System (DiskSuite Tool)

Use this procedure to unmirror a file system that can be unmounted while the system is running. To unmirror root (/), /opt, /usr, or swap, or any other file system that cannot be unmounted while the system is running, refer to "How to Unmirror a File System That Cannot Be Unmounted (Command Line)" on page 144.

1. **Make sure you have met the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79), and have read the preliminary information ("Preliminary Information for Mirrors" on page 141).**

2. **Check that the status of at least one submirror is "OK."**

   A mirror with no submirrors in the "OK" state must be repaired first.

3. **Unmount the file system.**

4. **Double-click the Mirror object in the Objects list.**

   The object appears on the canvas.

5. **To delete the mirror, display the mirror's pull-down menu and choose the Delete option. Click the Really Delete button on the Confirmation dialog box that appears.**

   The mirror is split into its constituent submirrors, which appear on the canvas.

6. **Edit the `/etc/vfstab` file.**

   You must change the device name for the file system. To retain access to the data, change from the mirror to one of the submirrors. (The submirror is either a stripe or concatenated metadevice.) To remove access to the data, completely remove the entry for the file system.

7. **[Optional] If retain access to the data, mount the file system.**

   When the file system is remounted, it is on the stripe or concatenation.

8. **[Optional] To delete the submirror that is not in use, choose the Delete option from the submirror's pull-down menu.**

   Be sure to choose the submirror that you will not be using.

9. **[Optional] If the submirror is composed of a single slice, you can mount the file system on the slice. To do so, delete the stripe or concatenation. Then edit the `/etc/vfstab` file to mount the file system on the physical slice instead of the metadevice.**

10. **To verify that the mirror was deleted, display the Configuration Log.**

11. **[Optional] Select Rescan Configuration from the File menu to reflect the file system's new mount device.**

▼ How to Unmirror a File System (Command Line)

Use this procedure to unmirror a file system that can be unmounted while the system is running. To unmirror root (`/`), `/opt`, `/usr`, or `swap`, or any other file system that cannot be unmounted while the system is running. use the command line procedure on "How to Unmirror a File System That Cannot Be Unmounted (Command Line)" on page 144.

After checking the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79), and the preliminary information ("Preliminary Information for Mirrors" on page 141), use the `metadetach(1M)` and `metaclear(1M)` commands to

unmirror a file system. For more information refer to the `metadetach(1M)` and `metaclear(1M)` man pages.

The high-level steps to unmirror a mirror are:

- Running the `metastat(1M)` command to verify that at least one submirror is in the “Okay” state
- Unmounting the file system
- Running the `metadetach(1M)` command on the submirror that will continue to be used for the file system
- Running the `metaclear(1M)-r` command on the mirror
- Editing the `/etc/vfstab` file to use a nonmirror device, if the file system entry appears here
- Remounting the file system

Example — Unmirroring the `/var` File System

```
# metastat d4
d4: Mirror
    Submirror 0: d2
      State: Okay
    Submirror 1: d3
      State: Okay
...
# umount /var
# metadetach d4 d2
d4: submirror d2 is detached
# metaclear -r d4
d4: Mirror is cleared
d3: Concat/Stripe is cleared
(Edit the /etc/vfstab file so that the entry for /var is changed
from d4 to d2)
# mount /var
```

`/var` is made of a two-way mirror named `d4`; its submirrors are `d2` and `d3`, made of slices `/dev/dsk/c0t0d0s0` and `/dev/dsk/c1t0d0s0`, respectively. The `metastat(1M)` command verifies that at least one submirror is in the “Okay” state. (A mirror with no submirrors in the “Okay” state must be repaired first.) The file system is unmounted then submirror `d2` is detached. The `metaclear -r` command deletes the mirror and the other submirror, `d3`.

Next, the entry for `/var` in the `/etc/vfstab` file is changed to reference the submirror. For example, if `d4` were the mirror and `d2` the submirror, the following line:

```
/dev/md/dsk/d4  /dev/md/rdsk/d4  /var ufs  2  yes -
```

should be changed to:

```
/dev/md/dsk/d2  /dev/md/rdsk/d2  /var ufs  2  yes -
```

By using the submirror name, you can continue to have the file system mounted on a metadevice. Lastly, `/var` is remounted.

Note - By using `d2` instead of `d4` in the `/etc/vfstab` file, you have unmirrored the mirror. Because `d2` consists of a single slice, you can mount the file system on the slice name (`/dev/dsk/c0t0d0s0`) if you do not want the device to support a metadevice.

▼ How to Unmirror a File System That Cannot Be Unmounted (Command Line)

Use this task to unmirror file systems that cannot be unmounted during normal system operation, including root (`/`), `/usr`, `/opt`, and `swap`.

The high-level steps for this procedure are:

- Running the `metastat(1M)` command to verify that at least one submirror is in the “Okay” state
- Running the `metadetach(1M)` command on the mirror that contains root (`/`), `/usr`, `/opt`, or `swap` to make a one-way mirror
- For `/usr`, `/opt`, and `swap`: changing the file system entry in the `/etc/vfstab` file to use a non-DiskSuite device (slice)
- For root (`/`) only: running the `metaroot(1M)` command
- Rebooting the system
- Running the `metaclear(1M)` command to clear the mirror and submirrors

Example — Unmirroring root (/)

```
# metadetach d0 d20
d0: submirror d20 is detached
# metaroot /dev/dsk/c0t3d0s0
# reboot
...
# metaclear -r d0
d0: Mirror is cleared
d10: Concat/Stripe is cleared
# metaclear d20
d20: Concat/Stripe is cleared
```

In this example, root (/) is a two-way mirror named d0; its submirrors are d10 and d20, which are made of slices /dev/dsk/c0t3d0s0 and /dev/dsk/c1t3d0s0, respectively. The metastat command verifies that at least one submirror is in the "Okay" state. (A mirror with no submirrors in the "Okay" state must first be repaired.) Submirror d20 is detached to make d0 a one-way mirror. The metaroot command is then run, using the *rootslice* that the system is going to boot from. This edits the /etc/system and /etc/vfstab files to remove information specifying the mirroring of root (/). After a reboot, the metaclear -r command deletes the mirror and the other submirror, d10. The last metaclear command clears submirrror d20.

Example — Unmirroring swap

```
# metastat d1
d1: Mirror
    Submirror 0: d11
      State: Okay
    Submirror 1: d21
      State: Okay
...
# metadetach d1 d21
d1: submirror d21 is detached
(Edit the /etc/vfstab file to change the entry for swap from
metadevice to slice name)
# reboot
...
# metaclear -r d1
d1: Mirror is cleared
d11: Concat/Stripe is cleared
# metaclear d21
d21: Concat/stripe is cleared
```

In this example, swap is made of a two-way mirror named d1; its submirrors are d11 and d21, which are made of slices /dev/dsk/c0t3d0s1 and /dev/dsk/c1t3d0s1, respectively. The metastat command verifies that at least one submirror is in the "Okay" state. (A mirror with no submirrors in the "Okay" state must first be repaired.) Submirror d21 is detached to make d1 a one-way mirror. Next, the /etc/vfstab file must be edited to change the entry for swap to reference the slice that is in submirror d21. For example, if d1 was the mirror, and d21 the submirror containing slice /dev/dsk/c0t3d0s1, the following line:

```
/dev/md/dsk/d1 - - swap - no -
```

should be changed to:

```
/dev/dsk/c0t3d0s1 - - swap - no -
```

After a reboot, the metaclear -r command deletes the mirror and the other submirror, d11. The final metaclear command clears submirrror d21.

▼ How to Attach a Submirror (DiskSuite Tool)

Before starting, identify the concatenation or stripe to be used as the submirror. It must be the same size (or larger) as the existing submirror in the mirror. If you have not yet created either, refer "Creating Stripes and Concatenations" on page 20.

1. **Make sure you have met the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79), and have read the preliminary information ("Preliminary Information for Mirrors" on page 141).**

2. **Double-click an existing Mirror object in the Objects list and verify that its status is "OK."**

   The object appears on the canvas.

3. **Drag the Concat/Stripe object to be attached to the mirror to the top of the Mirror object.**

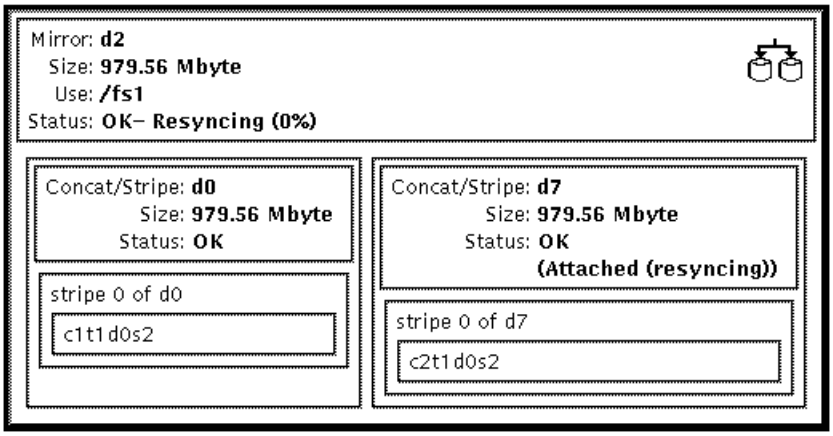
4. **Click the top rectangle of the Mirror object then click Commit.**

   A resync of the new submirror is initiated.

5. **To verify that the mirror was committed, display the Configuration Log.**

Example — Mirror Object With Attached Submirror

This example shows a a mirror, `d2`, to which a submirror, `d7` has been attached. The mirror automatically syncs the data on new submirror.



▼ How to Attach a Submirror (Command Line)

After checking the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79), and the preliminary information ("Preliminary Information for Mirrors"

on page 141), use the `metattach(1M)` command to attach a submirror to a mirror. Refer to the `metattach(1M)` man page for more information.

Before starting, identify the concatenation or stripe to be used as the submirror. It must be the same size (or larger) as the existing submirror in the mirror. If you have not yet created either, refer "Creating Stripes and Concatenations" on page 20.

Example — Attaching a Submirror

```
# metastat d30
d30: mirror
    Submirror 0: d60
      State: Okay
...
# metattach d30 d70
d30: submirror d70 is attached
# metastat d30
d30: mirror
    Submirror 0: d60
      State: Okay
    Submirror 1: d70
      State: Resyncing
    Resync in progress: 41 % done
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 2006130 blocks
...
```

This example shows the attaching of a submirror, `d70`, to a one-way mirror, `d30`, creating a two-way mirror. The mirror `d30` initially consists of submirror `d60`. `d70` is a concatenated metadevice. You verify that the status of the mirror is "Okay" with the `metastat(1M)` command, then attach the submirror. When the `metattach(1M)` command is run, the new submirror is resynced with the existing mirror. When you attach an additional submirror to the mirror, the system displays a message. To verify that the mirror is resyncing, use the `metastat(1M)` command.

▼ How to Detach a Submirror (DiskSuite Tool)

You might want to detach a submirror if you were going to reuse the underlying disks. You can detach that submirror without disrupting service from the system.

1. **Make sure you have met the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79), and have read the preliminary information ("Preliminary Information for Mirrors" on page 141).**

2. **Double-click the Mirror object in the Objects list.**

   The object appears on the canvas.

3. **Click inside the submirror to be detached.**

4. **Drag the submirror out of the Mirror object to the canvas.**
   If this is a two-way mirror, the mirror's status changes to "Urgent."

5. **Click the top rectangle of the Mirror object then click Commit.**

6. **To verify that the mirror was committed, display the Configuration Log.**

▼ How to Detach a Submirror (Command Line)

After checking the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79), and the preliminary information ("Preliminary Information for Mirrors" on page 141), use the `metadetach(1M)` command to detach a submirror from a mirror. Refer to the `metadetach(1M)` man page for more information.

Example — Detaching a Submirror

```
# metastat
d5: mirror
    Submirror 0: d50
...
# metadetach d5 d50
d5: submirror d50 is detached
```

In this example, mirror `d5` has a submirror, `d50`, which is detached with the `metadetach(1M)` command. The underlying slices from `d50` are going to be reused elsewhere. When you detach a submirror from a mirror, the system displays a confirmation message.

▼ How to Place a Submirror Offline and Online (DiskSuite Tool)

Placing a submirror offline and online is useful when repairing physical disks. For example, if a disk in a SCSI chain fails, all other metadevices in the chain could be taken offline while the broken disk is replaced. Metadevices are brought online after the replacement disk is installed.

When you take a submirror offline, DiskSuite keeps track of all I/O to the mirror. When you bring the submirror back online, DiskSuite performs an optimized resync of the data, and only has to resync changes, not the entire submirrror.

Note - A submirror that has been taken offline can only be mounted read-only.

The steps to place a submirror offline and online are essentially the same.

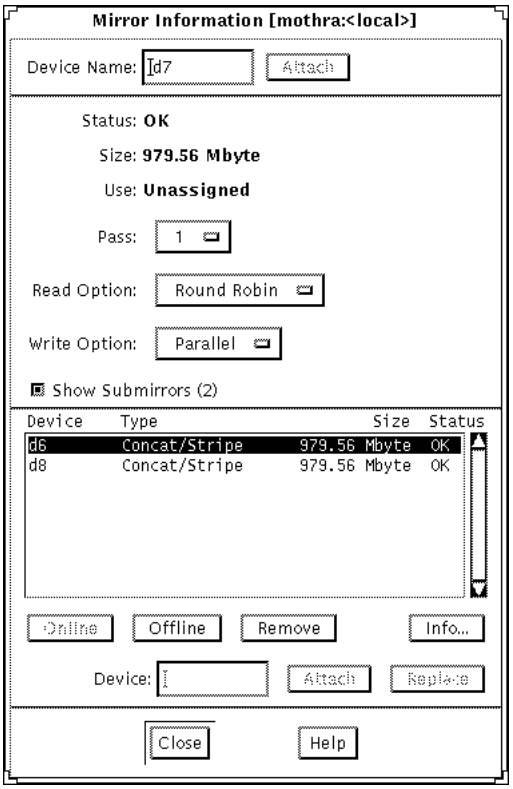
1. **Make sure you have met the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79), and have read the preliminary information (“Preliminary Information for Mirrors” on page 141).**

2. **Double-click an existing Mirror object in the Objects list.**

   The object appears on the canvas.

3. **Display the Mirror object’s pop-up window and select Info.**

   The Mirror Information window appears.



4. **Select the submirror in the device list.**

5. **If the submirror is offline, click Online. If the submirror is online, click Offline.**

The status of the submirror changes to “Offline (Scheduled),” or “Online (Scheduled).”

6. **Click Close.**

7. **Click the top rectangle of the Mirror object then click Commit.**

   If you are bringing a mirror offline, the mirror status changes to “Offline.” If you are bringing a mirror online, DiskSuite starts a resync operation.

▼ How to Place a Submirror Offline and Online (Command Line)

After checking the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79), and the preliminary information (“Preliminary Information for Mirrors” on page 141), use the `metaoffline(1M)` command to offline a submirror, or the `metaonline(1M)` command to online a submirror. Refer to the `metaoffline(1M)`or `metaonline(1M)`man pages for more information.

Example — Placing a Submirror Offline

```
# metaoffline d10 d11
d10: submirror d11 is offlined
```

This takes submirror `d11` offline from mirror `d10`. Reads will continue to be made from the other submirror. The mirror will be out of sync as soon as the first write is made. This inconsistency is corrected when the offlined submirror is brought back online.

Example — Placing a Submirror Online

```
# metaonline d10 d11
d10: submirror d11 is onlined
```

When ready (for example, after replacing a disk), the submirror `d11` is brought back online.

The `metaonline(1M)` command can only be used when a submirror was taken offline by the `metaoffline(1M)` command. After the `metaonline(1M)` command runs, DiskSuite automatically begins resyncing the submirror with the mirror.

Note - The `metaoffline(1M)` command's functionality is similar to that offered by `metadetach(1M)` however `metaoffline(1M)` does not sever the logical association between the submirror and the mirror.

Working With Disksets

This section describes maintenance tasks for disksets, including reserving and releasing disksets, and adding hosts and disks to a diskset.

Preliminary Information for Working With Disksets

- 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.)
- To work with a diskset, `root` must be a member of Group 14, or the `./rhosts` file must contain an entry for the other hostname (on each host).

▼ How to Reserve a Diskset (Command Line)

Disksets can be reserved *safely* or *forcibly*. When one host in a diskset reserves the diskset, the other host in the diskset cannot access data on drives in the diskset.

- **Safely –** Reserves the diskset for your host only if no other host has reserved the diskset.
- **Forcibly –** Reserves the diskset whether or not another host currently has the set reserved. Use this method when a host in the diskset is down or not communicating. If the other host had the diskset reserved at this point, it would panic due to reservation loss.

Note - If you are fairly certain that the hosts in the diskset are communicating, it is normally a good idea to perform a safe reservation.

Make sure you have met the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79) and have read the preliminary information ("Preliminary Information for Working With Disksets" on page 151). Use the `metaset(1M)` to reserve a diskset safely or forcibly. For more information, refer to the `metaset(1M)` man page.

Note - If another host has ownership of the diskset, it will panic due to a SCSI reservation conflict.

Example — Reserving a Diskset Safely

```
red# metaset
...
Set name = relo-red, Set number = 2

Host                Owner
  red
  blue
...
red# metaset -s relo-red -t
red# metaset
...
Set name = relo-red, Set number = 2

Host                Owner
  red                Yes
  blue
...
```

In this example, host `red` communicates with host `blue` and ensures that host `blue` has released any reservation of the diskset before host `red` attempts to reserve the set.

Note - In this example, if host `blue` owned the set `relo-red`, the "Owner" column in the above output would still have been blank. The `metaset(1M)` command only shows whether the issuing host owns the diskset, and not the other host.

Example — Reserving a Diskset Forcibly

```
# metaset -s relo-red -t -f
```

In this example, host `red` does not communicate with host `blue`. Instead, the drives in the diskset are reserved without warning. If host `blue` had the diskset reserved, it would now panic due to reservation loss.

▼ How to Release a Diskset (Command Line)

Releasing a diskset is 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 metadevices or hot spare pools defined in the set.

1. **Make sure you have met the prerequisites (“Prerequisites for Maintaining DiskSuite Objects” on page 79) and have read the preliminary information (“Preliminary Information for Working With Disksets” on page 151).**

2. **Release the diskset by using the** `metaset(1M)` **command**

```
# metaset -s diskset -r
```

In this command,

`–s` *diskset*	Specifies the name of a diskset on which `metaset` will work.
`–r`	Releases ownership of a diskset. The reservation of all the disks within the set is removed. The metadevices set up within the set are no longer accessible.

3. **Verify that the diskset has been released on this host by using the** `metaset(1M)` **command without any options.**

```
# metaset
```

Example — Releasing a Diskset

```
red# metaset -s relo-red -r
red# metaset -s relo-red

Set name = relo-red, Set number = 1

Host                Owner
  red
  blue

Drive               Dbase
  c1t0d1             Yes
  c1t2d0             No
  c1t3d0             No
  c1t4d1             No
  c2t2d0             Yes
  c3t0d1             Yes
  c3t2d0             No
```

(continued)

```
c3t3d0             No
c3t4d1             No
```

This example releases the diskset `relo-red`. Note that there is no owner of the diskset. Viewing status from host `red` could be misleading. A host can only determine if it does or does not own a diskset. For example, if host `blue` were to reserve the diskset, it would not appear so from host `red`; only host `blue` would be able to determine the reservation in this case.

▼ How to Add Additional Drives to a Diskset (Command Line)

You can add drives to a diskset after it has been defined.

1. **Make sure you have met the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79) and have read the preliminary information ("Preliminary Information for Working With Disksets" on page 151).**

2. **Add a drive to an existing diskset by using the `metaset(1M)` command.**

```
# metaset -s diskset -a drivename ...
```

In this command,

–s *diskset*	Specifies the name of a diskset on which `metaset` will work.
–a	Adds drives to the named diskset.
drivename...	Specifies the drives to add to the diskset. Drive names are in the form c*x*t*x*d*x*; no "s*x*" slice identifiers are at the end of the name. The *drivename* must have the same major and minor names on all hosts in the diskset.

When drives are added to a diskset, DiskSuite re-balances the metadevice state database replicas across the remaining drives. Refer to "Creating Disksets" on page 67 for more information.

Caution - You will lose data if you add drives that contain data.

3. **Verify that the host has been added to the diskset by using the** `metaset(1M)` **command without any options.**

```
# metaset
```

Example — Adding Additional Drives to a Diskset

```
red# metaset -s relo-red -a c2t5d0
red# metaset
Set name = relo-red, Set number = 1

Host                Owner
  red                Yes
  blue

Drive               Dbase
  c1t2d0             Yes
  c1t3d0             Yes
  c2t2d0             Yes
  c2t3d0             Yes
  c2t4d0             Yes
  c2t5d0             No
```

This example adds drive `c2t5d0` to diskset `relo-red`.

Note - If you add or delete drives to a diskset while DiskSuite Tool is running, a dialog box appears stating that the configuration has changed. Either reload the configuration by selecting Rescan Configuration from the File menu, or exit DiskSuite Tool then restart it.

▼ How to Add Another Host to a Diskset (Command Line)

DiskSuite supports a maximum of two hosts per diskset. You can add another host to an existing diskset that only has one host.

1. **Make sure you have met the prerequisites ("Prerequisites for Maintaining DiskSuite Objects" on page 79) and have read the preliminary information ("Preliminary Information for Working With Disksets" on page 151).**

2. **Add the host**:

```
# metaset -s diskset -a -h host ...
```

In this command,

–s *diskset*	Specifies the name of a diskset on which `metaset` will work.
–a	Adds hosts to the named diskset.
–h *host...*	Specifies one or more hostnames to be added to the diskset. Adding the first host creates the set. The hostname is the same name found in `/etc/nodename`.

3. Verify that the host has been added to the diskset by using the `metaset(1M)` command without any options.

```
# metaset
```

Example — Adding Another Host to a Diskset

```
red# metaset -s relo-red -a -h blue
red# metaset -s relo-red
Set name = relo-red, Set number = 1

Host                Owner
  red                Yes
  blue

Drive               Dbase
  c1t0d1             Yes
  c1t2d0             No
  c1t3d0             No
  c1t4d1             No
  c2t2d0             Yes
  c3t0d1             Yes
  c3t2d0             No
  c3t3d0             No
  c3t4d1             No
```

This example adds host `blue` to the diskset `relo-red`.

CHAPTER 4

Changing DiskSuite Objects

This chapter describes how to change DiskSuite devices, both with the DiskSuite Tool graphical interface and with the command line utilities.

Use the following to proceed directly to the section that provides step-by-step instructions using DiskSuite Tool.

Use the following to proceed directly to the section that provides step-by-step instructions using the command line interface.

Overview of Changing DiskSuite Objects

After you have created a DiskSuite object, the occasion may arise where you need to change it, modify its parameters, or reconfigure it. This section describes such tasks as:

- Saving and restoring the uncommitted changes to the DiskSuite configuration

- Replacing and restoring state database replicas
- Changing mirror's options

For general information on DiskSuite, see *Solstice DiskSuite 4.2 Reference Guide.*

Prerequisites for Changing DiskSuite Objects

Here are the prerequisites for the steps in this chapter:

- Make sure you have a current backup of all data.
- Make sure you have root privilege.
- If using the graphical user interface, start DiskSuite Tool.

  To work with "local" metadevices (metadevices not in a diskset configuration), type:

```
# metatool &
```

  To work with metadevices in a diskset, make sure you are the diskset owner and type:

```
# metatool -s diskset_name &
```

Working With the DiskSuite Configuration

This section describes how to save and restore the current DiskSuite configuration that you are working on.

Preliminary Information for the DiskSuite Configuration

- Saving a configuration saves uncommitted "state" changes on the Metadevice Editor's canvas. It does not save the current in-memory configuration. That is maintained by the state database replicas.
- The save and restore configuration options enable you to create, in effect, a template configuration. This could be useful in configuring like machines: you

create a configuration on one machine then copy to another, saving time in creating the second configuration.

- The save and restore configuration options are also useful if you need to interrupt work, yet ensure that what you have configured so far (but not yet committed) will not be lost.

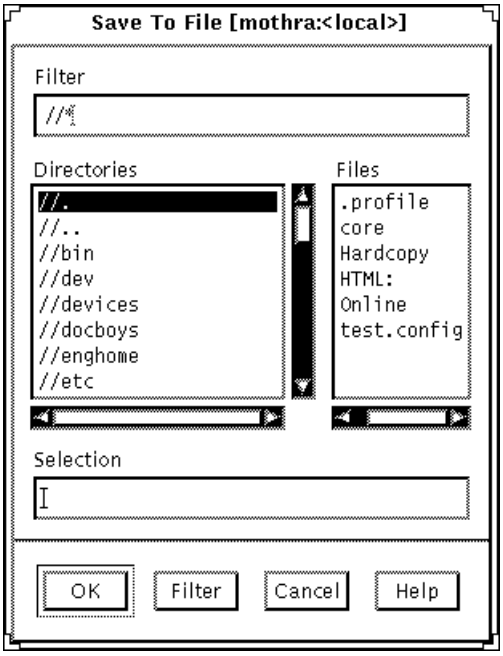
▼ How to Save a DiskSuite Configuration to Disk (DiskSuite Tool)

Use this task to save uncommitted changes that appear on the Metadevice Editor canvas. This does not save a “backup” of your configuration, only uncommitted changes.

1. **Make sure you have met the prerequisites (“Prerequisites for Changing DiskSuite Objects” on page 158) and have read the preliminary information (“Preliminary Information for the DiskSuite Configuration” on page 158).**

2. **Select Save to File from the File Menu.**

   The following dialog box appears:



3. **Type a file name in the Selection text field to save to and click OK.**

   You can also choose a file name from the Files list in the Save to File window. If the file exists, you can overwrite it, or cancel and use a different file name.

4. **When you save the DiskSuite configuration, a dialog box appears indicating the configuration was saved. Click OK.**

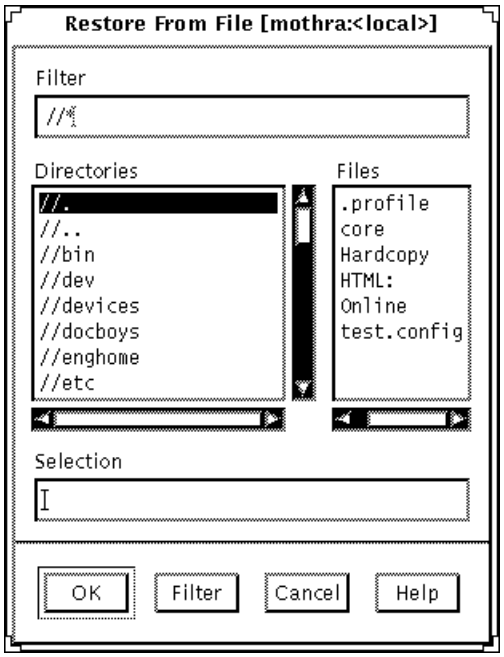
▼ How to Restore a DiskSuite Configuration From Disk (DiskSuite Tool)

Use this task to bring back to the Metadevice Editor canvas a configuration previously saved to disk. This does not restore a configuration in the sense of restoring from a backup. Instead, it opens a previously saved configuration of uncommitted changes on the canvas.

1. **Make sure you have met the prerequisites (“Prerequisites for Changing DiskSuite Objects” on page 158) and have read the preliminary information (“Preliminary Information for the DiskSuite Configuration” on page 158).**

2. **Select Restore From File from the File menu.**

   The following dialog box appears.



3. **Choose a file from the Files list in the Restore from File window.**

   You can also type a file name in the Selection text field.

4. **When you restore a configuration, a Restore From File dialog box appears, indicating the configuration was restored to the canvas. Click OK.**

Caution - Restoring from a file will undo any uncommitted changes you have made to the current configuration.

Modifying State Database Replicas

This section explains how to use the Metadevice State Database Information window to view and modify state database replicas.

Preliminary Information for Modifying State Database Replicas With the Metadevice State Database Information Window

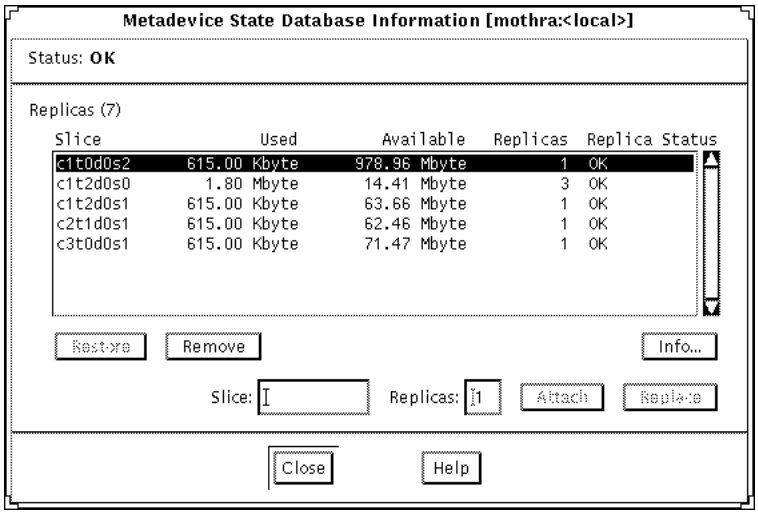
- You must have at least three state database replicas on the system. You should place them on different controllers to eliminate a single point-of-failure.
- The maximum number of state database replicas is 50. This also applies to replicas that are part of a diskset.

▼ How to Modify State Database Replicas (DiskSuite Tool)

1. **Make sure you have met the prerequisites (“Prerequisites for Changing DiskSuite Objects” on page 158) and have read the preliminary information (“Preliminary Information for Modifying State Database Replicas With the Metadevice State Database Information Window” on page 161).**

2. **View the Metadevice State Databse Information window.**

   Drag the MetaDB Object to the canvas then double-click it. The Metadevice State Database Information window appears:



3. **Make desired changes to the state database replicas by clicking or entering the required information. Click Close when done.**

4. **Once changes are made, you must commit the MetaDB Object before the changes take effect.**

5. **The Configuration Log shows that the MetaDB object was committed.**

   Table 4–1 lists the functionality associated with the regions of the Metadevice State Database Information window.

TABLE 4–1 Metadevice State Database Information Window Functionality

Field	Functions
Status	Description of the metadevice state database's status. See "How to Check the Status of State Database Replicas (DiskSuite Tool)" on page 81 for a description of these values.
Replica manipulation region	This region shows the following information and allows for manipulation. ■ **Replicas –** The number of replicas. ■ **Scrolling List –** A scrolling list of the slices that contains replicas. Includes the name of the slice, the amount of space used, space available, number of replicas on the slice and the replica status. ■ **Remove –** Removes the selected slices. ■ **Restore –** Restores the selected slices if they are in error. ■ **Info –** Displays the Slice Information window for the selected slice. ■ **Slice –** Specifies 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. The default is 1. ■ **Attach –** Adds the slice entered in the slice field to the Replica list. Available only when a slice name is entered.

Changing Mirror Options

This section explains how to use the Mirror Information window to view and modify mirror options.

Preliminary Information for Changing Mirror Options

- You can change a mirror's pass number, and read and write policies.
- Mirror options can be changed while the mirror is running.

Note - If you need to change the interlace value of stripe that is used as a submirror, refer to "How to Change the Interlace Value of Stripes in Mirrors (DiskSuite Tool)" on page 286.

▼ How to Change a Mirror's Options (DiskSuite Tool)

Table 4–2 describes mirror options that can be configured.

TABLE 4–2 Mirror Options

Option	Functions
Pass Number	A pass number in the range 0-9 can be assigned to a mirror. 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, the resync is skipped. A 0 should be used only for mirrors that are mounted as read-only. If different mirrors have the same pass number, they are resynced concurrently.
Read Policy	The three read options for mirrors are roundrobin, geometric, and first. The default is roundrobin, also called balanced load. roundrobin: All reads are made in a round robin order from all the submirrors in the mirrors. That is, the first read comes from the first submirror, the next read comes from the second submirror, and so forth. geometric: Provides faster performance on sequential reads or when you are using disks with track buffering. Geometric reads enable 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). first: Specifies reading from only the first submirror. Use only if you have a second submirror that has poor I/O characteristics.
Write Policy	A submirror can be set for parallel or serial writes. The default is parallel; writes are dispatched to all submirrors simultaneously. 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.

You can use the Mirror Information window to set options for mirrors, including read and write policy, the order in which mirrors are resynced during reboot, and if a submirror has a hot spare.

1. **Make sure you have met the prerequisites (“Prerequisites for Changing DiskSuite Objects” on page 158) and have read the preliminary information (“Preliminary Information for Changing Mirror Options” on page 163).**

1. **Double-click an existing Mirror object in the Objects list.**

   The Mirror object appears on the canvas.

2. **Display the Mirror object’s pop-up menu and choose Info.**

   The Mirror Information window appears.

3. **Change the desired options.**

   Refer to Table 4–2 for a description of mirror options. After making a change to a mirror option, click Close.

4. **Click the top rectangle of the Mirror object, then click Commit.**

5. **To verify that the change was committed, display the Configuration Log.**

▼ How to Change a Mirror’s Options (Command Line)

After checking the prerequisites listed (“Prerequisites for Changing DiskSuite Objects” on page 158), and the preliminary information (“Preliminary Information for Changing Mirror Options” on page 163), use the `metaparam(1M)` command to display and change a mirror’s options. Refer to Table 4–2 for a description of mirror options. For more information, refer to the `metaparam(1M)` man page.

Example — Changing a Mirror’s Read Policy

```
# metaparam -r geometric d30
# metaparam d30
d30: mirror current parameters are:
    Pass: 1
    Read option: geometric (-g)
    Write option: parallel (default)
```

The `–r` option changes a mirror’s read policy to `geometric`.

Example — Changing a Mirror's Write Policy

```
# metaparam -w serial d40
# metaparam d40
d40: mirror current parameters are:
    Pass: 1
    Read option: roundrobin (default)
    Write option: serial (-S)
```

The –w option changes a mirror's write policy to serial.

Example — Changing a Mirror's Pass Number

```
# metaparam -p 5 d50
# metaparam d50
d50: mirror current parameters are:
    Pass: 5
    Read option: roundrobin (default)
    Write option: parallel (default)
```

The –p option changes a mirror's pass number to five.

Sharing a Logging Device Among File Systems

This section explains how to configure trans metadevices to share logging devices.

Preliminary Information for Sharing a Logging Device

- Once you set up a trans metadevice, you can share the logging device among file systems.
- When sharing a logging device, it is a good idea to make the logging device a mirror.
- Consider sharing a logging device among file systems if your system does not have many available slices, or if the file systems sharing the logging device are read-mostly.

Caution - When one master device of a shared logging device goes into an errored state, the logging device is unable to roll its changes forward. This causes all master devices sharing the logging device to go into the hard error state.

▼ How to Share a Logging Device Among File Systems (DiskSuite Tool)

This procedure assumes you have already set up a trans metadevice with a logging device for another file system.

1. **Make sure you have met the prerequisites (“Prerequisites for Changing DiskSuite Objects” on page 158) and have read the preliminary information (“Preliminary Information for Sharing a Logging Device” on page 166).**

2. **Identify the slice or metadevice that contains the file system. You’ll need this when creating the master device.**

   If necessary, click Slices to display the Slice Browser for identifying the slice used by the file system.

3. **Click the trans metadevice template.**

   An unassigned and uncommitted Trans Metadevice object appears on the canvas. The metadevice name is automatically assigned.

4. **[Optional] To change the default metadevice name, display the object’s pop-up menu and select Info. Type the new metadevice name in the Device Name field and click Attach. Then click Close.**

5. **Double-click the existing Trans Metadevice object in the Objects list.**

   The objec is displayed on the canvas. The new trans metadevice will share logs with this one.

6. **Note the logging device name used by this trans metadevice.**

7. **Drag the slice from the Slice Browser, or the metadevice from the Objects List, containing the file system to be logged into the master rectangle of the new trans metadevice. A warning dialog box appears. Click Continue.**

   Note - If an entry exists in the `/etc/vfstab` file for the file system, and the file system is currently mounted, DiskSuite Tool automatically updates it to use the trans metadevice’s name.

8. **Display the Information window for the new trans metadevice and type the slice or metadevice name noted in the existing trans metadevice (Step 6 on page 167) into the Log text field.**

9. **Click Attach. You may see an evaluation warning dialog box. If so, click Accept. Then click Close on the Information window.**

10. **Click the top rectangle of the Trans Metadevice object then click Commit.**

    Click Really Commit on the warning dialog box that appears. This creates the new trans metadevice, sharing the logging device of the existing trans metadevice.

11. **To verify that the trans metadevice was committed, display the Configuration Log.**

12. **Logging becomes effective for the file system when you reboot the system.**

    Upon subsequent reboots, instead of checking the file systems, `fsck(1M)` displays message such as this for each logged file system:

```
/dev/md/rdsk/trans: is logging.
```

▼ How to Share a Logging Device Among File Systems (Command Line)

This procedure assumes you have already set up a trans metadevice with a log for another file system.

After checking the prerequisites (“Prerequisites for Changing DiskSuite Objects” on page 158), and the preliminary information (“Preliminary Information for Sharing a Logging Device” on page 166), use the `metainit(1M)` to share a log. Refer to the `metainit(1M)` man page for more information.

Example — Sharing a Logging Device

```
# umount /xyzfs
# metainit d64 -t c0t2d0s4 d10
d64: Trans is setup
(Edit the /etc/vfstab file so that the entry for /xyzfs references
the trans metadevice d64)
# mount /xyzfs
# metastat
...
```

(continued)

(Continuation)

```
d10: Logging device for d63 d64
...
```

This example shares a logging device (`d10`) defined as the log for a previous trans metadevice, with a new trans metadevice (`d64`). The file system to be set up as the master device is `/xyzfs` and is using slice `/dev/dsk/c0t2d0s4`. `metainit -t` specifies the configuration is a trans metadevice. The `/etc/vfstab` file must be edited to change (or enter for the first time) the entry for the file system to reference the trans metadevice. For example, the following line:

```
/dev/dsk/c0t2d0s4 /dev/rdsk/c0t2d0s4 /xyzfs ufs 2 yes -
```

should be changed to:

```
/dev/md/dsk/d64 /dev/md/rdsk/d64 /xyzfs ufs 2 yes -
```

The `metastat` command verifies that the log is being shared. Logging becomes effective for the file system when the system is rebooted.

Upon subsequent reboots, instead of checking the file system, `fsck(1M)` displays these messages for the two file systems:

```
/dev/md/rdsk/d63: is logging.
/dev/md/rdsk/d64: is logging.
```

CHAPTER 5

Removing DiskSuite Objects

This chapter describes how to remove DiskSuite objects, both with the DiskSuite Tool graphical interface and with the command line utilities.

Use the following to proceed directly to the section that provides step-by-step instructions for using DiskSuite Tool.

Use the following to proceed directly to the section that provides step-by-step instructions for using the command line interface.

Overview of Removing DiskSuite Objects

This chapter describes the steps to remove DiskSuite objects, including:

- State database replicas
- Metadevices
- Hot spares and hot spare pools
- Drives and hosts from disksets
- Disksets

You may need to remove objects when performing troubleshooting or reconfiguration, or to simply delete the object from the system. For general information on DiskSuite, see *Solstice DiskSuite 4.2 Reference Guide*.

Prerequisites for Removing DiskSuite Objects

Here are the prerequisites for the steps in this chapter:

- Make sure you have a current backup of all data.
- Make sure you have root privilege.
- If using the graphical user interface, start DiskSuite Tool.

  To work with "local" metadevices (metadevices not in a diskset configuration), type:

```
# metatool &
```

  To work with metadevices in a diskset, make sure you are the diskset owner and type:

```
# metatool -s diskset_name &
```

Removing State Database Replicas

This section describes how to remove state database replicas from the system.

Preliminary Information for Removing State Database Replicas

- It is possible to remove all state database replicas, but practically speaking, you should never do so while metadevices are still configured. Removing all state database replicas renders metadevices inoperable.
- Removing the entry for a state database replica from the `md.tab` file does not delete the replica from the system. You must use DiskSuite Tool, or run the `metadb -d` command.
- Before deleting the state database replica, make sure that it is no longer needed.

▼ How to Remove State Database Replicas (DiskSuite Tool)

1. **Make sure you have met the prerequisites (“Prerequisites for Removing DiskSuite Objects” on page 172) and have read the preliminary information (“Preliminary Information for Removing State Database Replicas” on page 173).**

2. **Drag the MetaDB object from the Objects list to the Metadevice Editor window’s canvas.**

3. **Drag the state database replica to be deleted from the MetaDB object.**

   Drag the selected slice to the Metadevice Editor’s canvas.

4. **Click inside the top rectangle of the MetaDB object and click Commit.**

   The status of the slice that contained the state database replica changes to Unassigned in the Slice Browser.

5. **Put the slice away.**

   Select the slice that was dragged out of the MetaDB object and click Put Away. The slice is returned to the Slice Browser.

6. **To verify that the MetaDB object was committed, display the Configuration Log.**

Note - A Warning message is displayed if you attempt to delete all state database replicas.

▼ How to Remove State Database Replicas (Command Line)

After checking the prerequisites (“Prerequisites for Removing DiskSuite Objects” on page 172), and the preliminary information (“Preliminary Information for Removing State Database Replicas” on page 173), use the `metadb(1M)` command with the `–d` option to delete state database replicas. Make sure that the state database replica is no longer needed. Refer to the `metadb(1M)` man page for more information.

Example — Deleting Two State Database Replicas

```
# metadb -d c0t2d0s0 c0t1d0s0
```

This example deletes two state database replicas, located on slices `/dev/dsk/c0t2d0s0` and `/dev/dsk/c0t1d0s0`.

Example — Forcing the Deletion of All State Database Replicas

```
# metadb -f -d c0t1d0s3 c4t1d0s3
```

This example deletes a system’s remaining two state database replicas. The `–f` option is needed.

Removing Stripes and Concatenations

This section describes how to remove stripes and concatenations from the system.

Caution - Any data that is on a metadevice is lost when it is removed from the system for good, and its underlying slices are reused. Data should be backed up if you need to save it.

Preliminary Information for Removing Stripes and Concatenations

- Before deleting a stripe or concatenation, make sure you no longer need it.
- If you delete a stripe or concatenation and reuse the slices that were part of the deleted metadevice, all data on the metadevice is gone from the system. Thus, make sure you back up the data on the metadevice before deleting it.

▼ How to Remove a Stripe, Concatenation, or Concatenated Stripe (DiskSuite Tool)

1. **Make sure you have met the prerequisites (“Prerequisites for Removing DiskSuite Objects” on page 172), and have read the preliminary information (“Preliminary Information for Removing Stripes and Concatenations” on page 175).**

2. **Make sure you have a current backup of the metadevice.**

3. **Stop access to the metadevice.**

   For example, a file system on the metadevice should be unmounted. For a non-UFS application, such as a database, perform the steps necessary to stop the application’s use of the metadevice.

4. **Double-click the stripe or concatenation to be deleted in the Objects list.**

   The metadevice object appears on the canvas.

5. **Display the object’s pop-up menu and choose Delete.**

6. **Click Really Delete on the dialog box that is displayed.**

7. **[Optional] If there is an entry in the `/etc/vfstab` file for this metadevice, delete that entry.**

   You do not want to confuse the system by asking it to mount a file system on a non-existent metadevice.

8. **To verify that the object was deleted, display the Configuration Log.**

Note - Hot spare pools assigned to the metadevice are not deleted.

▼ How to Remove a Stripe, Concatenation, or Concatenated Stripe (Command Line)

After checking the prerequisites (“Prerequisites for Removing DiskSuite Objects” on page 172), and the preliminary information (“Preliminary Information for Removing Stripes and Concatenations” on page 175), use the `metaclear(1M)` command to delete the metadevice. Refer to the `metaclear(1M)` man page for more information.

Example — Removing a Concatenation

```
# umount d8
# metaclear d8
d8: Concat/Stripe is cleared
(Edit the /etc/vfstab file)
```

This example clears the concatenation `d8` that also contains a mounted file system. The file system must be unmounted before the metadevice can be cleared. The system displays a confirmation message that the concatenation is cleared. If there is an entry in the `/etc/vfstab` file for this metadevice, delete that entry. You do not want to confuse the system by asking it to mount a file system on a non-existent metadevice.

Removing Mirrors

This section describes how to remove mirrors from the system.

Removing a mirror that is used by a file system that cannot be unmounted—such as root (`/`), `swap`, `/opt`, or `/usr`—essentially involves “unmirroring” the file system and mounting it on the underlying slice of one of the submirrors making up the mirror. Refer to “How to Unmirror a File System That Cannot Be Unmounted (Command Line)” on page 144.

Note - To remove a mirror and keep the same metadevice name as the mount device, refer to “Metadevice Name Switching” on page 278.

Preliminary Information for Removing Mirrors

- If you delete a mirror's submirrors and reuse the slices that were part of the deleted submirrors, all of that data is gone from the system. Thus, make sure you back up the data on the mirror before deleting it and its submirrors.
- When removing a mirror, you first "break apart" the mirror into its submirrors. You can choose to keep the data on one of the submirrors, or if a submirror consists of a single slice, you can keep the data on the slice.

▼ How to Remove a Mirror and Submirrors (DiskSuite Tool)

You can use this task for any file system or non-UFS application that uses a mirror, except for a file system that is root (`/`), `swap`, `/opt`, or `/usr`. To remove a mirror that is used by one of these file systems, refer to "How to Unmirror a File System That Cannot Be Unmounted (Command Line)" on page 144.

1. **Make sure you have met the prerequisites ("Prerequisites for Removing DiskSuite Objects" on page 172), and have read the preliminary information ("Preliminary Information for Removing Mirrors" on page 177).**

2. **Make sure you have a current backup of the metadevice.**

3. **Stop access to the metadevice.**

   For example, a mirrored file system should be unmounted. For a non-UFS application, such as a database, perform the steps necessary to stop the application's use of the metadevice.

4. **Double-click the Mirror object in the Objects list.**

   The mirror object appears on the canvas.

5. **Display the Mirror object's pop-up menu and choose Delete.**

6. **Click the Really Delete button.**

   The mirror is split into its underlying submirrors (Concat/Stripe metadevices).

7. **[Optional] If the mirror was used as a file system and there was an entry for the mirror in the `etc/vfstab` file, use one of the following to clean up the entry for the mirror.**

- You change the `/etc/vfstab` entry to mount the file system on one of the submirrors.
- If one of the submirrors consists of a one-way concatenation, you can change the `/etc/vfstab` entry to mount the file system on the underlying slice.
- If you do not plan on accessing the data anymore, you can remove the `/etc/vfstab` entry for the file system altogether.

8. Clean up the Concat/Stripe objects.

You should delete the submirror metadevice(s) that you no longer need.

9. To verify that the objects were deleted, display the Configuration Log.

▼ How to Remove a Mirror and Submirrors (Command Line)

You can use this task for any file system or non-UFS application that uses a mirror, except for a file system that is root (`/`), `swap`, `/opt`, or `/usr`. To remove a mirror that is used by one of these file systems, refer to "How to Unmirror a File System That Cannot Be Unmounted (Command Line)" on page 144.

The high-level steps for this procedure are:

- Unmounting the file system
- Running `metadetach(1M)` on the mirror and one of its submirrors
- Running `metaclear(1M)` on the mirror
- Optional: Editing the `/etc/vfstab` file to use a non-mirror device, if a file system entry appears here
- Optional: Remounting the file system

Example — Removing a Mirror, Keeping the Data on a Submirror

After checking the prerequisites ("Prerequisites for Removing DiskSuite Objects" on page 172), and the preliminary information ("Preliminary Information for Removing Mirrors" on page 177), use the `metadetach(1M)` and `metaclear(1M)` commands to delete the metadevice. Refer to the `metadetach(1M)` and `metaclear(1M)` man pages for more information.

```
# metastat d2
d2: Mirror
```

(continued)

```
    Submirror 0: d0
      State: Okay
    Submirror 1: d1
      State: Okay
...
# umount /news
# metadetach d2 d0
# metaclear d2
d2: Mirror is cleared
(Edit the /etc/vfstab file so that /news references submirror d0)
# mount /news
# metaclear d1
d1: Concat/Stripe is cleared
```

This example clears the mirror `d2` that also contains a mounted file system. The mirror consists of submirrors `d0` and `d1`. The `metastat` command reports that both submirrors are in the “Okay” state. The file system must be unmounted before the `metadetach` command detaches submirror `d0` from mirror `d2`. The mirror is then cleared.

To continue to access data on submirror `d0`, the entry for the file system in the `/etc/vfstab` file is changed from the mirror to the concatenation (submirror) `d0`.

After cleaning up the `/etc/vfstab` file to reference submirror (concatenation) `d0`, the file system is remounted. (It is remounted on `d0`.) The other submirror, `d1`, is cleared with the `metaclear` command.

Removing RAID5 Metadevices

This section describes how to remove RAID5 metadevices from the system.

Caution - Any data that is on the RAID5 metadevice is lost when it is removed from the system for good, and its underlying slices are reused. Data should be backed up if you need to save it.

Preliminary Information for Removing RAID5 Metadevices

- If you delete a RAID5 metadevice and reuse the slices that were part of the deleted device, all of that data is gone from the system. Thus, make sure you back up the data on the RAID5 metadevice before deleting it.

▼ How to Remove a RAID5 Metadevice (DiskSuite Tool)

1. **Make sure you have met the prerequisites (“Prerequisites for Removing DiskSuite Objects” on page 172) and have read the preliminary information (“Preliminary Information for Removing RAID5 Metadevices” on page 180).**

2. **Make sure you have a current backup of the metadevice.**

3. **Stop access to the RAID5 metadevice.**

   For example, a file system should be unmounted. For a non-UFS application, such as a database, perform the steps necessary to stop the application’s use of the metadevice.

4. **Double-click the RAID5 object to be deleted in the Objects list.**

   The RAID5 object appears on the canvas.

5. **Display the RAID5 object’s pop-up menu and select Delete.**

6. **Click Really Delete on the dialog box that is displayed.**

7. **[Optional] If there is an entry in the `/etc/vfstab` file for this metadevice, delete that entry.**

   You do not want to confuse the system by asking it to mount a file system on a non-existent device.

8. **To verify that the object was deleted, display the Configuration Log.**

Note - Hot spare pools assigned to the metadevice are not deleted.

▼ How to Remove a RAID5 Metadevice (Command Line)

After checking the prerequisites (“Prerequisites for Removing DiskSuite Objects” on page 172), and the preliminary information (“Preliminary Information for Removing RAID5 Metadevices” on page 180), use the `metaclear(1M)` command to delete the metadevice. Refer to the `metaclear(1M)` man page for more information.

Example — Removing a RAID5 Metadevice

```
# umount /nfs
# metaclear d80
d80: RAID is cleared
(Edit the /etc/vfstab file)
```

This example clears the RAID5 metadevice `d80` that also contains a mounted file system, `/nfs`. Access to `d80` is stopped by unmounting its file system. The system displays a confirmation message that the RAID5 metadevice is cleared. If there is an entry in the `/etc/vfstab` file for this metadevice, it should be deleted. You do not want to confuse the system by attempting to mount a file system on a non-existent metadevice.

Removing Trans Metadevices

This section describes how to remove trans metadevices (UFS logging).

Note - To remove a trans metadevice and keep the same metadevice name as the mount device, refer to “Metadevice Name Switching” on page 278.

Preliminary Information for Removing Trans Metadevices

- Removing a trans metadevice removes logging from the appropriate file systems. The underlying data (and possibly a metadevice, if the master consists of one) on the file systems is still preserved, unless you also remove that metadevice or remove the slice.

- Any information pertaining to the master device is rolled from the log device prior to removing the trans metadevice.

▼ How to Remove a Trans Metadevice (DiskSuite Tool)

Use this task to remove UFS logging from a file system that can be unmounted.

1. **Make sure you have met the prerequisites (“Prerequisites for Removing DiskSuite Objects” on page 172) and have read the preliminary information (“Preliminary Information for Removing Trans Metadevices” on page 181).**

2. **Make sure you have a current backup of the metadevice.**

3. **Unmount the file system.**

4. **Double-click the Trans Metadevice object in the Objects list.**
   The trans metadevice object appears on the canvas.

5. **Display the Trans Metadevice object’s pop-up menu and select Delete.**

6. **Click the Really Delete button.**
   The trans metadevice is split into its underlying master and logging devices.

7. **Edit the `/etc/vfstab` file, using one of the following:**
   - If the master device is a metadevice, you can change the `/etc/vfstab` entry to mount the file system on the metadevice that serves as the master device.
   - If the master device is metadevice consisting of a single slice, you can change the `/etc/vfstab` entry to mount the file system on the metadevice’s underlying slice.
   - If the master device is a slice, you can change the `/etc/vfstab` entry to mount the file system on the slice.
   - If you do not plan on accessing the data anymore, remove the `/etc/vfstab` entry for the file system altogether.

8. **To verify that the Trans Metadevice object was deleted, display the Configuration Log.**

9. **Run the `fsck(1M)` command.**
   Because the file system is no longer a logging device, you must run `fsck` before you can mount it. You run `fsck` either on the raw metadevice or raw device for the slice, depending on the configuration of the master device. Answer `y` to the following prompt:

```
# fsck raw_device
...
FILE SYSTEM STATE IN SUPERBLOCK IS WRONG; FIX? y
...
```

Note - If you are mounting the file system on a metadevice, run `fsck` on the raw device for that metadevice. Otherwise, run `fsck` on the raw device for slice upon which you will mount the file system.

10. Mount the file system.

The file system is no longer being logged.

11. [Optional] Clean up the master and logging devices.

If the master and logging devices were metadevices, you should delete the metadevices that you no longer need.

▼ How to Remove a Trans Metadevice (Command Line)

After checking the prerequisites ("Prerequisites for Removing DiskSuite Objects" on page 172) and the preliminary information ("Preliminary Information for Removing Trans Metadevices" on page 181), use the `metaclear(1M)` command to remove the trans metadevice. Refer to the `metaclear(1M)` man page for more information.

Example — Removing a Trans Metadevice

```
# umount /abcfs
# metaclear d64
(Edit the /etc/vfstab file)
# fsck /dev/rdsk/c0t2d0s6
...
FILE SYSTEM STATE IN SUPERBLOCK IS WRONG; FIX? y
...
# mount /abcfs
```

This example removes UFS logging from the `/abcfs` file system, which uses the trans metadevice `d64`. The underlying slice for the master device is

/dev/dsk/c0t2d0s6. When the trans metadevice is cleared, any information pertaining to the master device is rolled from the log prior to clearing the device. The entry for the file system in the /etc/vfstab file must be changed so that it references the block and raw devices containing the file system rather than the metadevice name for the trans metadevice. Because the file system is no longer a logging device, the fsck(1M) command is run before mounting it. The FIX? prompt is responded to with a y, then the file system is mounted on the underlying slice.

▼ How to Remove a Trans Metadevice From a File System That Cannot Be Unmounted (DiskSuite Tool)

Use this procedure to remove UFS logging from a file system that cannot be unmounted, such as root (/), /usr, and swap.

1. **Make sure you have met the prerequisites (“Prerequisites for Removing DiskSuite Objects” on page 172) and have read the preliminary information (“Preliminary Information for Removing Trans Metadevices” on page 181).**

2. **Make sure you have a current backup of the metadevice.**

3. **Double-click the Trans Metadevice object in the Objects list.**
   The object appears on the canvas.

4. **Drag the logging device out of the Trans Metadevice object to the canvas.**
   The status of the object changes to “Detach log (scheduled).”

5. **Click inside the top rectangle of the Trans Metadevice object then click Commit.**
   A dialog box warns that the logging device will be detached after the trans metadevice is unmounted or after the next reboot. Click Really Commit.
   The status of the trans metadevice changes to “Detach log (in-progress).”

   Note - If an entry exists for the file system in the /etc/vfstab file, and the file system is currently mounted, DiskSuite Tool automatically updates it to use the slice name instead of the trans metadevice name.

6. **Reboot. You may see a message indicating that the file system is being checked.**

```
...
The /usr file system (/dev/md/rdsk/d0) is being checked.
```

(continued)

```
/dev/md/rdsk/d0: 11576 files, 198318 used, 42081 free
/dev/md/rdsk/d0: (737 frags, 5168 blocks, 0.3% fragmentation)
...
```

7. **Edit the `/etc/vfstab` file to remove the trans metadevice.**

   Change the entry for the file system so that it references the block and raw devices containing the file system rather than the trans metadevice.

8. **Reboot.**

   This reboot enables the system to recognize that the file system is no longer mounted on the trans metadevice but on its underlying slice.

9. **To delete the trans metadevice, double-click the Trans Metadevice object in the Objects list. The object appears on the canvas. Point inside the top rectangle of the object and display the pull-down menu. Choose the Delete option.**

10. **Click the Really Delete button on the Confirmation dialog box that appears.**

11. **To verify that the Trans Metadevice object was deleted, display the Configuration Log.**

▼ How to Remove a Trans Metadevice From a File System That Cannot Be Unmounted (Command Line)

After checking the prerequisites (“Prerequisites for Removing DiskSuite Objects” on page 172) and the preliminary information (“Preliminary Information for Removing Trans Metadevices” on page 181), use the `metadetach(1M)` and `metaclear(1M)` commands to remove the trans metadevice. Refer to the `metadetach(1M)` and `metaclear(1M)` man pages for more information.

Use this procedure to remove a trans metadevice from a file system, such as `/usr`, that cannot be unmounted during normal system operation.

Example — Removing a Trans Metadevice from /usr

```
# metadetach -f d20
d20: logging device c0t0d0s1 will be detached at unmount or reboot
# reboot
...
The /usr file system (/dev/rdsk/c0t3d0s3) is being checked.
...
(Edit the /etc/vfstab file)
# reboot
...
# metaclear d20
d20: Trans is cleared
```

In this example, d20 is a trans metadevice providing UFS logging for the /usr file system. To remove logging, the metadetach command is run with the -f option to force a detach of the logging device, and the system is rebooted. Next, the /etc/vfstab file is edited to change the entry for the file system so that it references the slice containing the file system rather than the trans metadevice. Another reboot places the /usr file system on its new mount device. The metaclear command clears the trans metadevice d20 from the system.

Removing Hot Spares and Hot Spare Pools

This section describes how to remove hot spares and hot spare pools from the system.

Preliminary Information for Removing Hot Spares and Hot Spare Pools

- Hot spares in the in use state cannot be deleted.
- Before a hot spare pool can be deleted, first delete all of the hot spare pool's associations to submirrors and RAID5 metadevices.

▼ How to Remove a Hot Spare From a Hot Spare Pool (DiskSuite Tool)

1. **Make sure you have met the prerequisites ("Prerequisites for Removing DiskSuite Objects" on page 172) and have read the preliminary information**

(“Preliminary Information for Removing Hot Spares and Hot Spare Pools” on page 186).

2. **Double-click the Hot Spare Pool object in the Objects list.**
   The object appears on the canvas.

3. **Drag the hot spare slice that you want to remove from the Hot Spare Pool object to the canvas.**

4. **Click inside the top rectangle of the Hot Spare Pool object then click Commit.**

5. **To verify that the hot spare pool was committed, display the Configuration Log.**

▼ How to Remove a Hot Spare From a Hot Spare Pool (Command Line)

After checking the prerequisites (“Prerequisites for Removing DiskSuite Objects” on page 172) and the preliminary information (“Preliminary Information for Removing Hot Spares and Hot Spare Pools” on page 186), use the `metahs(1M)` command to remove the hot spare. Refer to the `metahs(1M)` man page for more information.

Example — Removing a Hot Spare From a Hot Spare Pool

```
# metahs -d hsp003 /dev/dsk/c2t1d0s2
hsp003: Hotspare is deleted
```

This example removes the hot spare `/dev/dsk/c2t1d0s2` from the hot spare pool `hsp003`.

Example — Removing a Hot Spare From All Hot Spare Pools

```
# metahs -d all /dev/dsk/c2t1d0s2
hsp003: Hotspare is deleted
hsp004: Hotspare is deleted
# metahs -i
...
hsp003: 2 hot spares
        c1t3d0s6                Available       912800 blocks
        c0t0d0s4                Available       5600 blocks
```

(continued)

```
hsp004: 2 hot spares
        c1t3d0s6                Available       912800 blocks
        c0t0d0s4                Available       5600 blocks
```

This example removes the hot spare `/dev/dsk/c2t1d0s2` from all its associated hot spare pools. The `metahs` command with the `-i` option shows that the hot spare slice is no longer part of the hot spare pools from which it was deleted.

▼ How to Remove a Hot Spare Pool (DiskSuite Tool)

Before you can remove a hot spare pool you must remove all associations to submirrors and RAID5 metadevices.

1. **Make sure you have met the prerequisites (“Prerequisites for Removing DiskSuite Objects” on page 172) and have read the preliminary information (“Preliminary Information for Removing Hot Spares and Hot Spare Pools” on page 186).**

2. **Select Hot Spare Pools from the Browse menu to display the Hot Spare Pool Browser window.**

3. **Double-click the hot spare pool in the scrolling list that you want to delete.**

   The Hot Spare Pool Information window for that hot spare pool appears. An “Associated With” listing displays all the metadevices using the hot spare pool.

4. **Remove the hot spare pool association for each metadevice (submirror or RAID5 metadevice) in the Associated With list.**

   a. **One by one, double-click each metadevice in the Associated With list.**

      The appropriate Information window appears.

   b. **Click Remove. The Hot Spare Pool field becomes blank. Then click Close.**

   c. **Repeat 4a and 4b for each metadevice in the Associated With list.**

   d. **When finished removing the hot spare pool association for each metadevice, click Close on the Hot Spare Pool Information window.**

5. **Commit each mirror or RAID5 device that had its hot spare pool association removed.**

   The “Use” status of the hot spare pool changes to “None.”

6. **Double-click the Hot Spare Pool object in the Objects list to display it on the canvas.**

7. **Delete the hot spare pool.**

   Display the object's pop-up window and choose Delete.

8. **To verify that the hot spare pool was deleted, display the Configuration Log.**

▼ How to Remove a Hot Spare Pool (Command Line)

After checking the prerequisites ("Prerequisites for Removing DiskSuite Objects" on page 172) and the preliminary information ("Preliminary Information for Removing Hot Spares and Hot Spare Pools" on page 186), use the `metaparam(1M)` and `metahs(1M)` commands to remove a hot spare pool. For more information, refer to the `metaparam(1M)` and `metahs(1M)` man pages.

Example — Removing a Hot Spare Pool

```
# metastat
...
d30: Mirror
    State: Okay
...
d31: Submirror of d30
    Hot spare pool: hsp001
...
d32: Submirror of d30
    Hot spare pool: hsp001
...
# metaparam -h none d30
# metaparam -h none d31
# metahs -d hsp001
hsp001: Hotspare pool is cleared
```

This example shows how the hot spare pool `hsp001` is removed. To find out the hot spare pool associations, use the `metastat` command. The `metastat` output shows two submirrors, `d31` and `d32`, using hot spare pool `hsp001`. The `metaparam` command with the `-h` and `none` options removes the association to the hot spare pool, first for submirror `d31`, then `d32`. When the hot spare pool has no more associations, it is removed with the `metahs -d` command.

Removing Disksets

This section describes how to remove disksets from the system.

Preliminary Information for Removing Hosts and Disks From Disksets

- You can remove hosts from a diskset. However, the last host can be removed from a diskset only after all drives in the diskset have been removed.
- Removing the last host from a diskset destroys the diskset.
- When drives are removed from a diskset, DiskSuite re-balances the metadevice state database replicas across the remaining drives.

▼ How to Remove a Host From a Diskset (Command Line)

1. **Make sure you have met the prerequisites ("Prerequisites for Removing DiskSuite Objects" on page 172) and have read the preliminary information ("Preliminary Information for Removing Hosts and Disks From Disksets" on page 190).**

2. **Remove a host from a diskset by using the `metaset(1M)` command.**

```
# metaset -s diskset -d -h host
```

In this command,

–s *diskset*	Specifies the name of a diskset on which `metaset` will work.
–d	Deletes the specified host.
–h *host*	Specifies one or more hostnames to be deleted from the diskset. The hostname is the same name found in `/etc/nodename`.

3. **To verify that a host has been removed from the diskset, use the `metaset(1M)` command without any options.**

```
# metaset
```

Example — Removing a Host from a Diskset

```
red# metaset -s relo-red -d -h blue
red# metaset

Set name = relo-red, Set number = 1
Host                Owner
  red                Yes
...
```

This example removes the host `blue` from the diskset `relo-red`.

▼ How to Remove a Drive From a Diskset (Command Line)

1. **Make sure you have met the prerequisites ("Prerequisites for Removing DiskSuite Objects" on page 172) and have read the preliminary information ("Preliminary Information for Removing Hosts and Disks From Disksets" on page 190).**

2. **Remove the drive from the diskset by using the `metaset(1M)` command.**

```
# metaset -s diskset -d [-f] drive...
```

In this command,

–s *diskset*	Specifies the name of a diskset on which `metaset` will work.
–d	Deletes the specified drivename(s).
–f	Forces the deletion of the last drive in the diskset, because this drive would implicitly contain the last state database replica.
drive	Specifies the drive(s) to be deleted from the diskset. They must be in the form c*x*t*x*d*x* with no slice specified.

Note - Use the –f option to delete the last drive in the diskset.

3. **To verify that a drive has been removed from the diskset, use the `metaset(1M)` command.**

Example — Removing a Drive from a Diskset

```
red# metaset -s relo-red -d c2t5d0
red# metaset
...
Host                Owner
  red                Yes
  blue

Drive               Dbase
  c1t2d0             Yes
  c1t3d0             Yes
  c2t2d0             Yes
  c2t3d0             Yes
  c2t4d0             Yes
```

This example removes drive `c2t5d0` from diskset `relo-red`. The `metaset` command confirms that the deleted drive is no longer part of the diskset.

▼ How to Remove a Diskset (Command Line)

Removing a diskset completely means:

- Removing all metadevices and hot spare pools in the diskset
- Removing all but one host in the diskset
- Removing all drives in the diskset
- Removing the final host

Example — Removing a Diskset

After checking the prerequisites (“Prerequisites for Removing DiskSuite Objects” on page 172), and the preliminary information (“Preliminary Information for Removing Hosts and Disks From Disksets” on page 190), use the `metaclear(1M)` and `metaset(1M)` commands to completely remove a diskset. Refer to the `metaclear(1M)` and `metaset(1M)` man pages for more information.

```
red# metaclear -s relo-red -a
red# metaset -s relo-red -d -h blue
red# metaset -s relo-red -f -d c1t2d0 c1t3d0 c2t2d0 c2t3d0 c2t4d0 c2t5d0
red# metaset -s relo-red -d -h red
```

The `metaclear -a` command clears (removes) all metadevices and hot spare pools from the `relo-red` diskset. Next, host `blue` is deleted from the diskset, followed by the shared disk drives. Finally, the last host in the set, in this case, `red`, is deleted from the diskset. This last command removes the diskset from the system.

CHAPTER 6

Managing the System

This chapter describes how to perform system management tasks related to DiskSuite.

Use the following to proceed directly to the section that provides step-by-step instructions for using DiskSuite Tool.

- "How to Add Devices to the Statistics Graph Window (DiskSuite Tool)" on page 212
- "How to Remove Devices From the Statistics Graph Window (DiskSuite Tool)" on page 213
- "How to Enable SunNet Manager to Launch DiskSuite Tool (SunNet Manager)" on page 213
- "How to Launch DiskSuite Tool From SunNet Manager (SunNet Manager)" on page 214
- "How to Launch File System Manager and Disk Manager (DiskSuite Tool)" on page 218

Use the following to proceed directly to the section that provides step-by-step instructions for using the command line interface.

- "How to Configure DiskSuite SNMP Support (Command Line)" on page 215
- "How to Enable DiskSuite to Launch Storage Manager (Command Line)" on page 217

Overview of Managing the System

The tasks associated with managing systems running DiskSuite include:

- Working with and administering the SPARCstorage Array from the DiskSuite Tool graphical user interface
- Using DiskSuite Tool to graphically monitor metadevice performance
- Integrating with Solstice SunNet Manager™
- Integrating with SNMP alerts
- Integrating with Solstice AdminSuite™ Storage Manager

For general information on DiskSuite, see *Solstice DiskSuite 4.2 Reference Guide.*

Prerequisites for Managing the System

Here are the prerequisites for the steps in this chapter:

- Make sure you have a current backup of all data.
- Make sure you have root privilege.
- If using the graphical user interface, start DiskSuite Tool.

  To work with "local" metadevices (metadevices not in a diskset configuration), type:

```
# metatool &
```

To work with metadevices in a diskset, make sure you are the diskset owner and type:

```
# metatool -s diskset_name &
```

Working With the Graphical View of the SPARCstorage Array

This section explains how to:

- Select objects (disks, trays, or controllers) in the Disk View window
- Check the status of SPARCstorage Array disks
- Check the fan and battery status of SPARCstorage Array controllers
- Display a SPARCstorage Array's World Wide Number

Preliminary Information for Working Graphically with the SPARCstorage Array

DiskSuite Tool's Disk View window presents a graphical view of the SPARCstorage Array. When you display the Disk View window, the icon for a SPARCstorage Array controller resembles a fibre channel icon, both on the Disk View canvas and in the Controllers window.

Each tray within the SPARCstorage Array is represented by an "etched-in" frame, with a title showing the tray number. Differences between the SPARCstorage Array 100 and 200 are:

- SPARCstorage Array 100 – Each SCSI bus within the tray (up to two) is shown by a second etched-in frame within the frame showing the tray number. The disks attached to each SCSI bus within the SSA100 tray appear sequentially within the SCSI frame, with no lines connecting them.

  Figure 6–1 shows how the SPARCstorage Array 100 appears in the Disk View window.

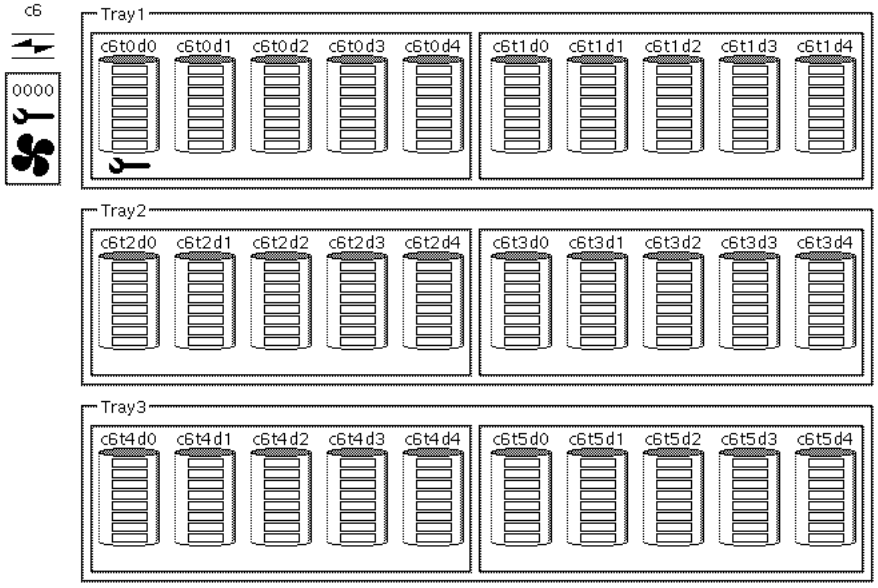


Figure 6–1 SPARCstorage Array 100

If one or both of the busses on the tray has no disks, the bus is “stubbed-out” by showing an empty frame. If one or more of the trays have no disks, DiskSuite Tool does not stub-out the tray(s).

- **SPARCstorage Array 200 –** Just the disks that are within the tray frame are shown, as there is no SCSI bus within the tray. The disks appear sequentially with no lines connecting them.

  If one or more of the trays have no disks, DiskSuite Tool does not stub-out the tray(s).

  Figure 6–2 shows how the SPARCstorage Array 200 appears in the Disk View window.

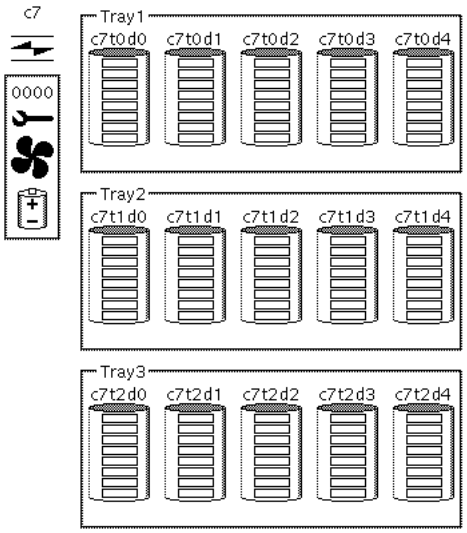


Figure 6–2 SPARCstorage Array 200

You can also use the Disk View window to display the SPARCstorage Array's unique 12-digit identification number, known as the World Wide Name. The last four digits appear just below the controller icon on the canvas (and on the LCD panel on the SPARCstorage Array itself). You can see the entire number by displaying a controller's Information window.

The Disk View window enables you to check at a glance the status of SPARCstorage Array components, including the battery and fan, and to see if a disk is spun up or down.

▼ How to Select Objects in the Disk View Window (DiskSuite Tool)

This task explains how to select objects in the Disk View window. The selection model is hierarchical: selecting an object selects not only the object, but all of its child objects as well.

1. **Make sure you have met the prerequisites ("Prerequisites for Managing the System" on page 196) and have read the preliminary information ("Preliminary Information for Working Graphically with the SPARCstorage Array" on page 197).**

2. **Click Disk View to display the Disk View window.**

3. **To select an object, click it.**

   For SPARCstorage Arrays:

- Selecting all slices in a disk selects the disk. Selecting a disk selects all slices in the disk.
- Selecting all disks in a tray selects the tray. Selecting a tray selects all disks in the tray.
- Selecting all trays on a controller selects the controller. Selecting a controller selects all trays on the controller.

For non-SPARCstorage Array configurations:

- Selecting all slices in a disk selects the disk. Selecting a disk selects all slices in the disk.
- Selecting all disks on a controller selects the controller. Selecting a controller selects all disks on the controller.

▼ How to Check the Status of SPARCstorage Array Disks (DiskSuite Tool)

Use this task to view the status of SPARCstorage Array disks (spun up or down).

1. **Make sure you have met the prerequisites ("Prerequisites for Managing the System" on page 196) and have read the preliminary information ("Preliminary Information for Working Graphically with the SPARCstorage Array" on page 197),**

2. **Click Disk View to display the Disk View window.**

3. **Look at the canvas. A disk that is spun down appears with a down arrow icon beneath the disk.**

   If necessary, display the desired controllers that contain the disks by selecting them from the Controllers window.

Note - You can use DiskSuite Tool to spin down non-SPARCstorage Array disks. However, the Disk View window shows a spun up or spun down status only for SPARCstorage Array disks. Also, any I/O performed on a non-SPARCstorage Array disk that is spun down automatically spins up the disk.

Where to Go From Here

To spin up SPARCstorage Array disks that have been spun down, refer to "How to Start a Disk (DiskSuite Tool)" on page 209.

▼ How to Check the Status of a SPARCstorage Array Controller's Fan and Battery (DiskSuite Tool)

1. **Make sure you have met the prerequisites ("Prerequisites for Managing the System" on page 196) and have read the preliminary information ("Preliminary Information for Working Graphically with the SPARCstorage Array" on page 197).**

2. **Click Disk View to display the Disk View window.**

3. **Look at the icons that appear beside each controller. (If necessary, display the desired controllers by selecting them in the Controllers window.)**

   DiskSuite Tool indicates the status for a SPARCstorage Array controller's fans and battery with the same icons that appear on the SPARCstorage Array LCD screen:

   - Wrench icon – the object needs maintenance.
   - Wrench and Fan icons – a fan has failed, or excessive internal temperatures have occurred. In most cases, the SSA will be taken off-line automatically.
   - Wrench and battery icons – the NVRAM battery is low, or not installed.

Note - The Disk View window also shows problems on non-SPARCstorage Array disks. If a wrench icon appears beneath one of these disks, there is a disk problem.

Figure 6–3 shows a controller that has the wrench and battery icons displayed.

Figure 6–3 SPARCstorage Array Wrench and Battery Icons

Note - One or more failures can occur simultaneously. In this case, multiple icons can appear. The Controller Information window will also contain information on the failures, as will the Problem Log.

Where to Go From Here

For information on resolving problems with SPARCstorage Arrays, refer to Chapter 7.

▼ How to Display a SPARCstorage Array Controller's World Wide Name (DiskSuite Tool)

1. **Make sure you have met the prerequisites ("Prerequisites for Managing the System" on page 196) and have read the preliminary information ("Preliminary Information for Working Graphically with the SPARCstorage Array" on page 197).**

2. **Click Disk View to display the Disk View window.**

3. **Look at the controllers on the canvas. (If necessary, display the desired controllers by selecting them in the Controllers window.)**

   The last four digits of the World Wide Name appear below each SPARCstorage Array controller.

4. **To view the entire 12-digit number, select the controller, display its pop-up menu and select Info.**

   The entire World Wide Name appears in the Info window as the Serial Number.

Administering the SPARCstorage Array

This section explains how to use DiskSuite Tool to:

- Enable and disable NVRAM
- Flush and purge NVRAM
- Reserve and release disks
- Start and stop disks

DiskSuite Tool provides an integrated solution for working with SPARCstorage Arrays, enabling you to view the physical layout of the SPARCstorage Array hardware, and to perform various administrative tasks. You must use the DiskSuite Tool Disk View window to perform these tasks; DiskSuite does not provide equivalent command line utilities. To administer a SPARCstorage Array from the command line, use the `ssaadm(1M)` command.

Caution - Use this functionality with care. It provides a powerful way to manage the SPARCstorage Array. As a minimum precaution, you should have a current backup of your data before using DiskSuite Tool in this way.

Preliminary Information For Enabling and Disabling NVRAM

- Fast writes can be configured: at the controller level, in which case fast writes are set for all drives in the SPARCstorage Array; at the tray level, affecting all drives in a single tray; or at the drive level, setting fast write for that individual drive.
- When fast write is enabled, it is saved as part of the SPARCstorage Array's configuration—across power cycles.
- If the NVRAM battery is low, missing, or has failed, the menus will be disabled, because fast write is disabled on the controller under these conditions.
- Before enabling fast write, quiesce all I/O to the controller, tray, or disk. Be sure that no I/O is active on metadevices that exist on disks within a SPARCstorage Array. In particular, insure that ownership of any diskset metadevices has been released since an implicit I/O stream exists while ownership of a diskset is maintained. Refer to "Working With Disksets" on page 151 for details of taking and releasing ownership of a diskset.

▼ How to Enable NVRAM on a Controller, Tray, or Disk (DiskSuite Tool)

1. **Make sure you have met the prerequisites ("Prerequisites for Managing the System" on page 196) and have read the preliminary information ("Preliminary Information For Enabling and Disabling NVRAM" on page 203).**

2. **Refer to "How to Stop a Disk (DiskSuite Tool)" on page 208 to quiesce all I/O.**

   Be sure that no I/O is active on metadevices defined that exist on disks within a SPARCstorage Array. In particular, ensure that ownership of any diskset metadevices has been released since an implicit I/O stream exists while ownership of a diskset is maintained.

3. **Click Disk View.**

   The Disk View window appears.

4. **In the Disk View window, select a controller, tray, or individual disk or disks.**

5. **Choose Fast Write from the Object menu, then choose Enable.**

6. **A Fast Write dialog box appears, reminding you that all I/O to the object should be stopped. Click Continue.**

7. **A confirmation appears, indicating that fast write has been enabled. Click OK.**

8. **Refer to "How to Start a Disk (DiskSuite Tool)" on page 209 to restart the disks.**

▼ How to Enable NVRAM For Synchronous Writes on a Controller, Tray, or Disk (DiskSuite Tool)

This procedure enables fast writes for synchronous writes only. To enable fast writes for "all" writes, use the previous procedure.

1. **Make sure you have met the prerequisites ("Prerequisites for Managing the System" on page 196) and have read the preliminary information ("Preliminary Information For Enabling and Disabling NVRAM" on page 203).**

2. **Refer to "How to Stop a Disk (DiskSuite Tool)" on page 208 to quiesce all I/O.**

   Be sure that no I/O is active on metadevices that exist on disks within a SPARCstorage Array. In particular, ensure that ownership of any diskset metadevices has been released since an implicit I/O stream exists while ownership of a diskset is maintained.

3. **Click Disk View.**

   The Disk View window appears.

4. **In the Disk View window, select a controller, tray, or individual disk or disks.**

5. **Choose Fast Write from the Object menu, then choose Synchronous.**

6. **A Fast Write dialog box appears, reminding you that all I/O to the object should be stopped. Click Continue.**

7. **A confirmation appears, indicating that fast write for synchronous writes has been enabled. Click OK.**

8. **Refer to "How to Start a Disk (DiskSuite Tool)" on page 209 to restart the disks.**

▼ How to Disable NVRAM on a Controller, Tray, or Disk (DiskSuite Tool)

1. **Make sure you have met the prerequisites (“Prerequisites for Managing the System” on page 196) and have read the preliminary information (“Preliminary Information For Enabling and Disabling NVRAM” on page 203).**

2. **Click Disk View.**

   The Disk View window appears.

3. **In the Disk View window, select a controller, tray, or individual disk or disks.**

4. **Choose Fast Write from the Object menu, then choose Disable.**

5. **A confirmation appears, indicating that fast write has been disabled. Click OK.**

Preliminary Information for Purging and Flushing NVRAM Data

- The flush option flushes any outstanding writes from NVRAM to the disk drive.
- If you get an error while flushing data, you must purge the data. Purging data “throws away” any outstanding writes in NVRAM.
- Purging fast write data should be performed with caution, and only when a drive has failed, as it could result in the loss of data.
- If the NVRAM battery is low, missing, or has failed, the menus will be disabled, because the NVRAM is non-functional and data there would be lost.

▼ How to Flush Outstanding Writes From NVRAM (DiskSuite Tool)

This task flushes out all outstanding writes for the selected controller (and all disks), tray (and all disks in that tray), or individual disk(s), from the NVRAM to disk.

1. **Make sure you have met the prerequisites (“Prerequisites for Managing the System” on page 196) and have read the preliminary information (“Preliminary Information for Purging and Flushing NVRAM Data” on page 205).**

2. **Click Disk View.**

   The Disk View window appears.

3. **In the Disk View window, select a controller, tray, or individual disk or disks.**

4. **Choose Sync NVRAM from the Object menu.**

5. **A confirmation dialog box appears, indicating the NVRAM will be synced. Click OK.**

▼ How to Purge Fast Write Data NVRAM (DiskSuite Tool)

Use this task only if you can no longer access the SPARCstorage Array, such as when a drive has failed. If you can access drives in the SPARCstorage Array, refer to "How to Flush Outstanding Writes From NVRAM (DiskSuite Tool)" on page 205. Purging fast write data gets rid of any outstanding writes.

1. **Make sure you have met the prerequisites ("Prerequisites for Managing the System" on page 196) and have read the preliminary information ("Preliminary Information for Purging and Flushing NVRAM Data" on page 205).**

2. **Click Disk View.**
   The Disk View window appears.

3. **In the Disk View window, select a controller, tray, or disk and display the object's pop-up menu.**

4. **Select Purge NVRAM. A confirmation dialog box appears, indicating that the NVRAM will be purged. Click Purge.**

Preliminary Information for Reserving and Releasing Disks

- In a diskset configuration, with a SPARCstorage Array connected to more than one host, it might be necessary for a single host to reserve individual drives, or all the drives in the array. When the issuing host reserves a drive or drives, no other host can use those drives until the issuing host releases them.

- If DiskSuite Tool is run *on a host other than the host issuing the reserve command*, a status of "Reserved" is shown for those disks in the Information window, and a "lock" icon appears under the disks in the Disk View canvas. However, the host issuing the reservation will not see a lock icon.

Note - This functionality is currently available only for SPARC systems using SPARCstorage Array disks.

▼ How to Reserve a Disk for Host Exclusive Use (DiskSuite Tool)

1. **Make sure you have met the prerequisites (“Prerequisites for Managing the System” on page 196) and have read the preliminary information (“Preliminary Information for Reserving and Releasing Disks” on page 206).**

2. **Click Disk View.**

   The Disk View window appears.

3. **Select a controller, tray, or single disk and choose Reserve Disks from the Object menu.**

4. **A confirmation dialog box appears, indicating the disk reservation. Click OK.**

   There is no visual indication that the disk has been reserved. However, if DiskSuite Tool is run on a host other than the host issuing the reserve command, a status of “Reserved” is shown for those disks in the Information window, and a “lock” icon appears in the Disk View canvas.

▼ How to Release a Disk Reserved by Host (DiskSuite Tool)

1. **Make sure you have met the prerequisites (“Prerequisites for Managing the System” on page 196) and have read the preliminary information (“Preliminary Information for Reserving and Releasing Disks” on page 206).**

2. **Click Disk View.**

   The Disk View window appears.

3. **Select a controller, tray, or single disk and choose Release Disks from the Object menu.**

4. **A confirmation dialog box appears, indicating the disk release. Click OK.**

Preliminary Information for Stopping and Starting Disks

- DiskSuite supports starting and stopping not only SPARCstorage Array disks, but non-SPARCstorage Array (SCSI) disks as well.
- If a SPARCstorage Array disk is spun down, a “down arrow” appears beneath the disk on the Disk View canvas. The Disk Information window also shows the disk’s

status. (Currently, the Disk View window does not display a “down arrow” for a non-SPARCstorage Array (SCSI) disk. Any I/O performed on a non-SPARCstorage Array disk automatically spins up that disk.)

- Stopping a disk is usually performed as part of a sequence of steps when performing hardware maintenance. Normally you would not stop a disk on its own.
- You can spin down a single SPARCstorage Array tray to replace a disk without having to power down the entire SPARCstorage Array.
- Currently, DiskSuite does not generate an event notification when stopping or starting a disk. This means that if you are running multiple instances of DiskSuite Tool on the same host, stopping or starting a disk in one DiskSuite Tool will not update the other instance. To avoid this potential problem, do not run multiple instances of DiskSuite Tool on the same host when stopping and starting disks.

Caution - The `ssaadm(1M)` command does not generate an event notification for DiskSuite when used to start and stop a disk. For this reason, do not use DiskSuite Tool and `ssaadm(1M)` together when stopping and starting disks. Doing so could cause DiskSuite Tool to display a disk‘s status incorrectly. Always use one or the other to both stop and start a disk.

▼ How to Stop a Disk (DiskSuite Tool)

Prerequisites for Stopping a Disk (SPARCstorage Array)

Because SPARCstorage Array disks are in trays, to stop one disk you should stop all disks in the tray to ensure proper shutdown. This involves:

- Unmounting file systems on the disks in the tray
- Stopping all database processes accessing the disks in the tray
- Stopping all other processes accessing the disks in the tray
- Stopping the disks

Note - Stopping the disk also causes a `sync_cache` to flush outstanding writes in NVRAM.

1. **Make sure you have met the prerequisites (“Prerequisites for Managing the System” on page 196) and have read the preliminary information (“Preliminary Information for Stopping and Starting Disks” on page 207).**

2. **Check for other DiskSuite objects on the affected controller, disk, or tray.**

   In the Disk View window, display the controller. Display the controller’s Info window and select Device Mappings to view which objects (metadevices, hot spares, state database replicas) might be using the physical device(s).

3. **Stop all access to the disk.**

   For example, unmount any file systems associated with the disk, or stop all I/O to a database using the disk.

4. **In the Disk View window, select a controller, tray, or single disk and choose Stop Disks from the Object menu.**

5. **A Stop Disk dialog box appears, reminding you that all I/O to the disk should be stopped. Click Continue.**

6. **The system notifies you when the disk is stopped. Click OK.**

   For a SPARCstorage Array, a down arrow appears beneath each disk that is stopped, indicating that the disk is spun down. If you stop a SCSI disk, no down arrow appears.

▼ How to Start a Disk (DiskSuite Tool)

1. **Make sure you have met the prerequisites ("Prerequisites for Managing the System" on page 196) and have read the preliminary information ("Preliminary Information for Stopping and Starting Disks" on page 207).**

2. **Click Disk View.**

   The Disk View window appears. SPARCstorage Array disks that are spun down appear on the canvas with a down arrow beneath the disk.

3. **Select a controller, to start all disks connected to it, or select individual disks using Control-click.**

   You can also select multiple controllers using Control-click.

4. **Choose Start Disks from the Object menu.**

5. **The system notifies you when the disk is started. Click OK.**

   For a SPARCstorage Array disk, the down arrow disappears from beneath the disk, indicating it has spun up. There is no indication for a SCSI disk.

Monitoring and Graphing Performance

This section explains how to:

- View metadevice statistics
- Graph metadevice statistics
- Set grapher options

Performance Monitoring vs. Performance Analysis

DiskSuite Tool gives you the capability for simple *performance monitoring* of metadevices and physical disks. The goal of performance monitoring is to maximize system performance by making small changes while maintaining data availability. Performance monitoring assumes that you are using a given configuration that you cannot easily or often reconfigure.

Performance analysis, on the other hand, aims to maximize system throughput or minimize latency by trying out various configurations until the “best performance” is achieved. Performance analysis assumes that you can set up multiple configurations and collect a different set of data from each configuration. This requires a different methodology and set of tools to cover such tasks as device configuration, file system management, data collection, test load generation, and the like.

The performance monitoring capability within DiskSuite Tool is best suited for the day-to-day operation of DiskSuite in spotting performance problems associated with hot-spots and bottlenecks. This enables you to monitor general performance trends, look for abrupt changes, and compare data collected for different parts of a configuration. For example, while monitoring a RAID5 metadevice, you might notice one column (slice) that shows much more I/O than the other columns. By moving that column to another disk, you are able to better balance the I/O for the entire metadevice.

Preliminary Information for Performance Monitoring and Graphing

- DiskSuite Tool displays a subset of the raw statistical data provided by the `kstat(3K)` interface. (Refer to the `kstat(3K)` and `iostat(1M)` man pages for more information.)
- Because DiskSuite Tool uses the `kstat` interface, it can only present disk-level statistics, not slice-level.

- You can drag and drop objects from any of DiskSuite Tool's windows to the Statistics Graphs window. This includes the Metadevice Editor canvas, the Objects list, the Slice Browser window, and the Disk View window.
- When you drag a slice to the Statistics Graphs window, you effectively drag the entire disk. You cannot get slice-level statistics.
- State database replicas, hot spare pools, and individual slices cannot show statistics.
- Tip: A good way to start using statistics is by first looking at the metadevice's Load value. Next, examine whether reads or writes seem responsible. You can then add the disks themselves that make up the metadevice and view each disk's load.
- In general, the raw throughput data and the Load value seem to be the most useful values to watch over time.

▼ How to View Device Statistics (DiskSuite Tool)

Use this task to display a device statistic sheet that shows a "snap-shot" for the current values of the statistical variables.

1. **Select an object from either the Metadevice Editor window or the Disk View window.**

   The MetaDB object, hot spare pools, and slices are not able to show statistics.

2. **Select Statistics from the Object menu in either window to display the Device Statistics window for the object.**

   For reference information on the Device Statistics window, see the online help.

3. **To view a new snap-shot of statistical information, click Update.**

Note - DiskSuite Tool displays a separate Device Statistics window for each disk selected in the Disk View window. Thus, if you select a controller or tray with many disks, you'll see as many separate Device Statistics windows as disks selected.

▼ How to Graph Device Statistics (DiskSuite Tool)

Use this task to display instantaneous statistical information in the form of a graph for a device.

1. **Select Statistics Graphs from the Browse menu in the Metadevice Editor window to display the Statistics Graphs window.**

2. **Drag an object from either the Metadevice Editor window or the Disk View window to the Statistics Graphs window.**

   The Statistics Graphs window will display statistics and a blank area for the graph of the object. The MetaDB object, hot spare pools, and slices are not able to graph statistics.

3. **Choose the statistical information to be graphed.**

   The Statistics Graphs window contains two "y-axis" (two vertical scales) of statistics. Thus, you can graph any two of the types of statistics available:

   Ops/sec – The number of disk I/O operations per second, measured in terms of reads, writes, or as a total.

   Kbytes/sec – The number of kilobytes of disk I/O, measured in terms of reads, writes, or as a total.

   Load – The disk load measured either as the percentage of time busy, or as a wait queue, which represents the average number of transactions waiting for service.

4. **When done using the Statistics Graphs window, select Put Away from the All Graphs menu.**

   If you do not put away a device, the Statistics Graphs window automatically includes it the next time it is opened.

▼ How to Add Devices to the Statistics Graph Window (DiskSuite Tool)

1. **Select Statistics Graphs from the Browse menu in the Metadevice Editor window to display the Statistics Graphs window.**

2. **Drag an object from either the Metadevice Editor window or the Disk View window to the Statistics Graphs window.**

   The Statistics Graphs window will display statistics and a blank area for the graph of the object. The MetaDB object, hot spare pools, and slices are not able to graph statistics.

3. **Refer to Step 3 on page 212 in the "How to Graph Device Statistics (DiskSuite Tool)" on page 211 task ("How to Graph Device Statistics (DiskSuite Tool)" on page 211).**

▼ How to Remove Devices From the Statistics Graph Window (DiskSuite Tool)

1. **Select Statistics Graphs from the Browse menu in the Metadevice Editor window to display the Statistics Graphs window.**

2. **To remove a single device, select it and click the Put Away button.**
   The Statistics Graphs window clears the device from the list.

3. **To remove all devices, select Put Away from the All Graphs menu in the Statistics Graphs window.**

Integrating SunNet Manager With DiskSuite

DiskSuite and SunNet Manager can be configured for cooperative ease of use. DiskSuite has the capability to forward its driver messages using asynchronous SNMP traps via a daemon that monitors the system console log. Messages, such as failures and errors, can then be detected by a SunNet Manager console. When you receive DiskSuite messages within SunNet Manager, you can launch DiskSuite Tool.

For more information on SunNet Manager, see the *Site/SunNet/Domain Manager Administration Guide.*

▼ How to Enable SunNet Manager to Launch DiskSuite Tool (SunNet Manager)

SunNet Manager automatically receives DiskSuite console messages. To enable SunNet Manager to launch DiskSuite Tool, follow these steps:

1. **With SunNet Manager running, select Save To File from the File menu, then Exit from the File menu.**

2. **Edit the** `/opt/SUNWconn/snm/struct/elements.schema` **file to change the** `elementCommand` **structure for your server. The entry should look like this:**

```
instance elementCommand (
```

(continued)

(Continuation)

```
...

(component.sun-server ‘‘DiskSuite Tool...’’
‘‘/usr/opt/SUNWmd/sbin/metatool’’)

...
```

Add this entry to the “component” category.

3. **Restart SunNet Manager with the `-i` option.**

   The `-i` option reads the modified `elements.schema` file and reinitializes the SunNet Manager database.

▼ How to Launch DiskSuite Tool From SunNet Manager (SunNet Manager)

Use this task to launch DiskSuite Tool from SunNet Manger. This task assumes you have followed the steps in the above task, “How to Enable SunNet Manager to Launch DiskSuite Tool (SunNet Manager)” on page 213.

1. **Select a host object (that has DiskSuite installed) within the SunNet Manger canvas.**

2. **Display the host’s pop-up menu and choose Tools.**

3. **From the Tools menu, choose DiskSuite Tool.**

   DiskSuite Tool is displayed for the selected host.

Integrating SNMP Alerts With DiskSuite

DiskSuite can send SNMP trap data (alerts) to any network management console capable of receiving SNMP messages. This is accomplished through a separate DiskSuite daemon, `mdlogd`, that you choose to install. Through a configuration file, you specify the trap variables as well as the kinds of messages to look for.

To use the DiskSuite SNMP daemon, you must install the optional SUNWmdn package. See *Solstice DiskSuite 4.2 Installation and Product Notes.*

▼ How to Configure DiskSuite SNMP Support (Command Line)

This task assumes you have installed the DiskSuite SNMP package, SUNWmdn.

The steps to configure DiskSuite for SNMP support are:

- Configure the mdlogd SNMP daemon
- Edit the /etc/opt/SUNWmd/mdlogd.cf SNMP trap configuration file

For more information, refer to the mdlogd(4) man page.

When you reboot the system after installing the SUNWmdn package, the following message appears:

```
Starting mdlogd ...

        /etc/opt/SUNWmd/mdlogd.cf: no configuration information
```

DiskSuite displays this message because you have not yet configured the mdlogd.cf file.

1. **Make sure that the** mdlogd **daemon is configured to load automatically at boot.**
   (A postinstall script should have taken care of this when you added the package.)

2. **Edit and save the SNMP trap configuration file,** mdlogd.cf**.**
   a. **Change the line** ENTERPRISE = **to the SNMP identifier for the enterprise to which the host running the daemon belongs.**

   b. **Change the line** OBJECTID = **to the SNMP identifier of the host running the daemon.**

   c. **Configure the** SubStrings **the daemon will look for. These are messages generated by DiskSuite.**

   d. **Change the** trap destination **tuple to the name of the host to receive the SNMP trap; the port; and the protocol.**
      Most likely you will always use port 162 and protocol udp, as these are the defaults for SNMP traps.

   e. **Leave the** Generic SNMP Trap # **set to** 6**.**

   f. **You can use your own coding scheme for Specific Trap #.**

For example, use number `1` for a low priority message, number `2` for medium priority, and `3` for high priority.

Example — `mdlogd.cf` File

```
##ident "@(#)mdlogd.cf 1.1 96/02/15"
# DiskSuite SNMP Trap configuration file.
...
ENTERPRISE = 1.3.6.1.4.1.42
OBJECTID = 1.3.6.1.4.1.860
#
# SubString Trap Destination      SNMP Trap #   Specific Trap #
#              (host:port:protocol)   0 < n <= 6       0 < n

"NOTICE: md:"  "spin:162:udp"            6             1
"WARNING: md:" "spin:162:udp"            6             2
```

This example dispatches SNMP traps for DiskSuite errors written to `/dev/console` to a host named `spin`.

Here's what the trap generated by the error message "`WARNING: md: d6: /dev/dsk/c3t3d0s7 needs maintenance`" would look like when received by a SunNet Manager console:

```
Wed Feb 21 15:40:41 1996 [ spin ] : Trap:

    sequence=2
    receive-time=Wed Feb 21 15:40:41 1996
    version=0
    community=public
    enterprise=Sun Microsystems
    source-time=00:00:00.00
    trap-type=enterprise specific trap: 2

    1.3.6.1.4.1.860 = Feb 21 15:40:41 1996 spin WARNING:
            md: d6: /dev/dsk/c3t3d0s7 needs maintenance
```

Integrating Storage Manager With DiskSuite

DiskSuite and Storage Manager can be configured for cooperative ease of use. The Storage Manager application contains two tools, Disk Manager and File System Manager, that enable you to manage a server's disk configurations and file systems. For example, you could create a metadevice in DiskSuite Tool, then drag it to File System Manager to create a UFS file system on the metadevice.

You can configure DiskSuite so that it can launch Storage Manager from the DiskSuite Tool Tools menu.

For more information on Storage Manager, refer to Appendix A.

▼ How to Enable DiskSuite to Launch Storage Manager (Command Line)

This task assumes you have installed the Storage Manager packages. Refer to *Solstice DiskSuite 4.2 Installation and Product Notes* for more information.

Enabling DiskSuite to launch Storage Manager's tools, involves configuring the `/usr/opt/SUNWmd/lib/metatool-toolsmenu` file. For more information, refer to the `metatool-toolsmenu(4)` man page.

The supplied `metatool-toolsmenu` file has two lines already configured to use File System Manager and Disk Manager. Uncomment (remove the leading # sign) the two lines and save the file.

Example — `metatool-toolsmenu` File

Here is a sample `metatool-toolsmenu` file, with two entries that enable File System Manager and Disk Manager to be launched from DiskSuite Tool.

```
#
# metatool 'Tools' menu registry file...
#
# Entries are of the form:
#
# :0:<name>:<command>:
#
# 0 is a format specifier.  '0' is the only valid specifier for metatool.
#
# <name> becomes a menu item1.5 in the Metadevice Editor and DiskView Tools menu
# <command> is passed to system() when item is chosen from the menu.
#
```

(continued)

(Continuation)

```
# ':' is a field delimiter, one of: '+' '|' ':' '^'
#
# sample entries:
 +0+File System Manager...+/opt/SUNWadm/2.2/bin/stomgr -F+
 :0:Disk Manager...:/opt/SUNWadm/2.2/bin/stomgr -D:
#
#
```

▼ How to Launch File System Manager and Disk Manager (DiskSuite Tool)

This task assumes you have configured the `metatool-toolsmenu` file, as explained in the previous task.

1. **Start DiskSuite Tool.**

```
# metatool &
```

DiskSuite Tool's menu options now includes a "Tools" item, with File System Manager and Disk Manager selections.

2. **Select the appropriate item from the Tools menu to start either File System Manager or Disk Manager.**

CHAPTER 7

Troubleshooting the System

This chapter describes how to troubleshoot DiskSuite.

Use the following to proceed directly to the section that provides step-by-step instructions for the particular task.

Overview of Troubleshooting the System

This chapter describes some DiskSuite problems and their appropriate solution. It is not intended to be all-inclusive but rather to present common scenarios and recovery procedures.

Prerequisites for Troubleshooting the System

Here are the prerequisites for the steps in this section:

- Have root privilege.
- Have a current backup of all data.

General Guidelines for Troubleshooting DiskSuite

Have the following information on hand when troubleshooting a DiskSuite problem:

- Contents of the `/etc/vfstab` file
- Status of state database replicas, metadevices, and hot spares, either from DiskSuite Tool or from the output of the `metadb(1M)` and `metastat(1M)` commands
- Disk partitioning information, either from the `prtvtoc(1M)` command or Storage Manager (Solaris systems), or the `fdisk` command (x86 systems)
- Solaris version
- Solaris patches
- DiskSuite patches

Recovering the DiskSuite Configuration

The `/etc/opt/SUNWmd/md.cf` file is a backup file of the DiskSuite configuration for a “local” diskset. 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 created or changed in the meantime, you can use the `md.cf` file to recover your DiskSuite configuration.

Note - The `md.cf` file does not maintain information on active hot spares. Thus, if hot spares were in use when the DiskSuite configuration was lost, those metadevices that were hot-spared will likely be corrupted.

▼ How to Use the `md.cf` File to Recover a DiskSuite Configuration

Caution - Only use this procedure if you have experienced a complete loss of your DiskSuite configuration.

---

1. **Recreate the state database replicas.**

   Refer to Chapter 1 for information on creating state database replicas.

2. **Make a backup copy of the `/etc/opt/SUNWmd/md.tab` file.**

3. **Copy the information from the `md.cf` file to the `md.tab` file.**

4. **Edit the “new” `md.tab` file so that:**
   - All mirrors are one-way mirrors. If a mirror’s submirrors are not the same size, be sure to use the smallest submirror for this one-way mirror. Otherwise data could be lost.
   - RAID5 metadevices are recreated with the `-k` option, to prevent reinitialization of the device. (Refer to the `metainit(1M)` man page for more information on this option.)

5. **Run the `metainit(1M)` command to check the syntax of the `md.tab` file entries.**

```
# metainit -n -a
```

6. **After verifying that the syntax of the `md.tab` file entries is correct, run the `metainit(1M)` command to recreate the metadevices and hot spare pools from the `md.tab` file.**

```
# metainit -a
```

7. **Run the `metattach(1M)` command to make the one-way mirrors into multi-way mirrors.**

8. **Validate the data on the metadevices.**

Changing DiskSuite Defaults

By default, the DiskSuite configuration defaults to 128 metadevices and state database replicas that are sized to 1034 blocks. The default number of disksets is four. All of these values can be changed if necessary, and the tasks in this section tell you how.

Preliminary Information for Metadevices

- The default number of metadevices for a system is 128. If you need to configure more than the default, you can increase this value up to 1024.
- When you add large numbers of metadevices, you may begin to see some system performance degradation only while administering metadevices (using DiskSuite Tool or the command line utilities). A large number of metadevices should not have an impact on normal system operation.
- If you increase the number of metadevices to gain a larger namespace for partitioning the types of devices within certain numeric ranges, but you create fewer than 128 devices, you should not see any performance degradation. In this case, you should not have to add larger state database replicas.

▼ How to Increase the Number of Default Metadevices (Command Line)

This task describes how to increase the number of metadevices from the default value of 128.

Caution - If you lower this number, any metadevice existing between the old number and the new number may not be available, potentially resulting in data loss. If you see a message such as "`md: d20: not configurable, check /kernel/drv/md.conf`" you will need to edit the `md.conf` file as explained in this task.

1. **After checking the prerequisites ("Prerequisites for Troubleshooting the System" on page 220) and the preliminary information ("Preliminary Information for Metadevices" on page 222), edit the `/kernel/drv/md.conf` file.**

2. **Change the value of the `nmd` field. Values are supported up to 1024.**

3. **Save your changes.**

4. **Perform a reconfiguration reboot to build the metadevice names.**

```
# boot -r
```

Example — `md.conf` File

Here is a sample `md.conf` file configured for 256 metadevices.

```
#
#ident "@(#)md.conf   1.7     94/04/04 SMI"
#
# Copyright (c) 1992, 1993, 1994 by Sun Microsystems, Inc.
#
name="md" parent="pseudo" nmd=256 md_nsets=4;
```

Preliminary Information for Disksets

The default number of disksets for a system is 4. If you need to configure more than the default, you can increase this value up to 32. The number of shared disksets is always one less than the `md_nsets` value, because the local set is included in `md_nsets`.

▼ How to Increase the Number of Default Disksets (Command Line)

This task shows you how to increase the number of disksets from the default value of 4.

Caution - If you lower this number, any diskset existing between the old number and the new number may not be persistent.

1. **After checking the prerequisites (“Prerequisites for Troubleshooting the System” on page 220), edit the** `/kernel/drv/md.conf` **file.**

2. **Change the value of the** `md_nsets` **field. Values are supported up to 32.**

3. **Save your changes.**

4. **Perform a reconfiguration reboot to build the metadevice names.**

```
# boot -r
```

Example — `md.conf` File

Here is a sample `md.conf` file configured for five disksets. The value of `md_nsets` is six, which results in five disksets and one local diskset.

```
#
#ident "@(#)md.conf   1.7     94/04/04 SMI"
#
# Copyright (c) 1992, 1993, 1994 by Sun Microsystems, Inc.
#
name="md" parent="pseudo" nmd=255 md_nsets=6;
```

Preliminary Information for State Database Replicas

- If you create large numbers of metadevices, the state database replicas may eventually be too small to contain all the necessary information. If this happens, try adding larger state database replicas by using the `-l` option to the `metadb(1M)` command then removing the smaller state database replicas.
- As a general rule, if you’ve doubled the number of default metadevices, double the size of the state database replicas.

▼ How to Add Larger State Database Replicas (Command Line)

After checking the prerequisites (“Prerequisites for Troubleshooting the System” on page 220), and reading the preliminary information (“Preliminary Information for

State Database Replicas" on page 224), use the `metadb` command to add larger state database replicas, then to delete the old, smaller state database replicas. Refer to the `metadb(1M)` man page for more information.

Example — Adding Larger State Database Replicas

```
# metadb -a -l 2068 c1t0d0s3 c1t1d0s3 c2t0d0s3 c2t1d0s3
# metadb -d c1t0d0s7 c1t1d0s7 c2t0d0s7 c2t1d0s7
```

The first `metadb` command adds state database replicas whose size is specified by the `-l 2068` option (2068 blocks). This is double the default replica size of 1034 blocks. The second `metadb` command removes those smaller state database replicas from the system.

Checking For Errors

When DiskSuite encounters a problem, such as being unable to write to a metadevice due to physical errors at the slice level, it changes the status of the metadevice, for example, to "Maintenance." However, unless you are constantly looking at DiskSuite Tool or running `metastat(1M)`, you may never see these status changes in a timely fashion.

There are two ways that you can automatically check for DiskSuite errors:

- Using SNMP traps
- Using a script to constantly check for errors

The first method is described in "Integrating SNMP Alerts With DiskSuite" on page 214.

The following section describes the kind of script you can use to check for DiskSuite errors.

▼ How to Automate Checking for Slice Errors in Metadevices (Command Line)

One way to continually and automatically check for a bad slice in a metadevice is to write a script that is invoked by `cron`. Here is an example:

```
#
#ident "@(#)metacheck.sh   1.3     96/06/21 SMI"
#
# Copyright (c) 1992, 1993, 1994, 1995, 1996 by Sun Microsystems, Inc.
#

#
# DiskSuite Commands
#
MDBIN=/usr/opt/SUNWmd/sbin
METADB=${MDBIN}/metadb
METAHS=${MDBIN}/metahs
METASTAT=${MDBIN}/metastat

#
# System Commands
#
AWK=/usr/bin/awk
DATE=/usr/bin/date
MAILX=/usr/bin/mailx
RM=/usr/bin/rm

#
# Initialization
#
eval=0
date=‘${DATE} ’+%a %b %e %Y’‘
SDSTMP=/tmp/sdscheck.${$}
${RM} -f ${SDSTMP}

MAILTO=${*:-"root"}   # default to root, or use arg list

#
# Check replicas for problems, capital letters in the flags indicate an error.
#
dbtrouble=‘${METADB} | tail +2 | \
    ${AWK} ’{ fl = substr($0,1,20); if (fl ~ /[A-Z]/) print $0 }’‘
if [ "${dbtrouble}" ]; then
        echo ""   >>${SDSTMP}
        echo "SDS replica problem report for ${date}" >>${SDSTMP}
        echo ""   >>${SDSTMP}
        echo "Database replicas are not active:"     >>${SDSTMP}
        echo ""   >>${SDSTMP}
        ${METADB} -i >>${SDSTMP}
        eval=1
fi

#
# Check the metadevice state, if the state is not Okay, something is up.
#
mdtrouble=‘${METASTAT} | \
    ${AWK} ’/State:/ { if ( $2 != "Okay" ) print $0 }’‘
if [ "${mdtrouble}" ]; then
        echo ""  >>${SDSTMP}
        echo "SDS metadevice problem report for ${date}"  >>${SDSTMP}
        echo ""  >>${SDSTMP}
        echo "Metadevices are not Okay:"  >>${SDSTMP}
        echo ""  >>${SDSTMP}
```

(continued)

(Continuation)

```
        ${METASTAT} >>${SDSTMP}
        eval=1
fi

#
# Check the hotspares to see if any have been used.
#
hstrouble=‘${METAHS} -i | \
    ${AWK} ’ /blocks/ { if ( $2 != "Available" ) print $0 }’‘
if [ "${hstrouble}" ]; then
        echo ""  >>${SDSTMP}
        echo "SDS Hot spares in use  ${date}"  >>${SDSTMP}
        echo ""  >>${SDSTMP}
        echo "Hot spares in usage:"  >>${SDSTMP}
        echo ""  >>${SDSTMP}
        ${METAHS} -i >>${SDSTMP}
        eval=1
fi
#
# If any errors occurred, then mail the report to root, or whoever was called
# out in the command line.
#
if [ ${eval} -ne 0 ]; then
        ${MAILX} -s "SDS problems ${date}" ${MAILTO} <${SDSTMP}
        ${RM} -f ${SDSTMP}
fi

exit ${eval}
```

For information on invoking scripts in this way, refer to the `cron(1M)` man page.

Note - This script serves as a starting point for automating DiskSuite error checking. You may need to modify it for your own configuration.

Boot Problems

Because DiskSuite enables you to mirror root (`/`), `swap`, and `/usr`, special problems can arise when you boot the system, either through hardware or operator error. The tasks in this section are solutions to such potential problems.

Table 7–1 describes these problems and points you to the appropriate solution.

TABLE 7–1 Common DiskSuite Boot Problems

The System Does Not Boot Because ...	Refer To ...
The `/etc/vfstab` file contains incorrect information.	"How to Recover From Improper `/etc/vfstab` Entries (Command Line)" on page 228
There are not enough state database replicas.	"How to Recover From Insufficient State Database Replicas (Command Line)" on page 231
A boot device (disk) has failed.	"How to Recover From a Boot Device Failure (Command Line)" on page 234
The boot mirror has failed.	"SPARC: How to Boot From the Alternate Device (Command Line)" on page 239 or "x86: How to Boot From the Alternate Device (Command Line)" on page 240

Preliminary Information for Boot Problems

- If the metadevice driver takes a metadevice offline due to errors, unmount all file systems on the disk where the failure occurred. Because each disk slice is independent, multiple file systems may be mounted on a single disk. If the metadisk driver has encountered a failure, other slices on the same disk will likely experience failures soon. File systems mounted directly on disk slices do not have the protection of metadisk driver error handling, and leaving such file systems mounted can leave you vulnerable to crashing the system and losing data.
- Minimize the amount of time you run with submirrors disabled or offline. During resyncing and online backup intervals, the full protection of mirroring is gone.

▼ How to Recover From Improper `/etc/vfstab` Entries (Command Line)

If you have made an incorrect entry in the `/etc/vfstab` file, for example, when mirroring root (`/`), the system will appear at first to be booting properly then fail. To remedy this situation, you need to edit `/etc/vfstab` while in single-user mode.

The high-level steps to recover from improper `/etc/vfstab` file entries are:

- Booting the system to single-user mode
- Running `fsck(1M)` on the mirror metadevice
- Remounting file system read-write
- Optional: running the `metaroot(1M)` command for a root (`/`) mirror
- Verifying that the `/etc/vfstab` file correctly references the metadevice for the file system entry
- Rebooting

Example — Recovering the root (/) Mirror

In the following example, root (`/`) is mirrored with a two-way mirror, `d0`. The root (`/`) entry in `/etc/vfstab` has somehow reverted back to the original slice of the file system, but the information in `/etc/system` still shows booting to be from the mirror `d0`. The most likely reason is that the `metaroot(1M)` command was not used to maintain `/etc/system` and `/etc/vfstab`, or an old copy of `/etc/vfstab` was copied back.

The incorrect `/etc/vfstab` file would look something like the following:

```
#device        device          mount           FS      fsck   mount    mount
#to mount      to fsck         point           type    pass   at boot  options
#
/dev/dsk/c0t3d0s0 /dev/rdsk/c0t3d0s0  /       ufs      1     no       -
/dev/dsk/c0t3d0s1 -                    -       swap     -     no       -
/dev/dsk/c0t3d0s6 /dev/rdsk/c0t3d0s6  /usr    ufs      2     no       -
#
/proc             -                  /proc    proc     -     no       -
fd                -                  /dev/fd  fd       -     no       -
swap              -                  /tmp     tmpfs    -     yes      -
```

Because of the errors, you automatically go into single-user mode when the machine is booted:

```
ok boot
...
SunOS Release 5.5 Version Generic [UNIX(R) System V Release 4.0]
Copyright (c) 1983-1995, Sun Microsystems, Inc.
configuring network interfaces: le0.
Hostname: antero
mount: /dev/dsk/c0t3d0s0 is not this fstype.
setmnt: Cannot open /etc/mnttab for writing

INIT: Cannot create /var/adm/utmp or /var/adm/utmpx

INIT: failed write of utmpx entry:"  "
```

(continued)

(Continuation)

```
INIT: failed write of utmpx entry:"  "

INIT: SINGLE USER MODE

Type Ctrl-d to proceed with normal startup,
(or give root password for system maintenance): <root-password>
```

At this point, root (/) and /usr are mounted read-only. Follow these steps:

1. Run `fsck(1M)` on the root (/) mirror.

Note - Be careful to use the correct metadevice for root.

```
# fsck /dev/md/rdsk/d0
** /dev/md/rdsk/d0
** Currently Mounted on /
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups
2274 files, 11815 used, 10302 free (158 frags, 1268 blocks,
0.7% fragmentation)
```

2. Remount root (/) read/write so you can edit the `/etc/vfstab` file.

```
# mount -o rw,remount /dev/md/dsk/d0 /
mount: warning: cannot lock temp file </etc/.mnt.lock>
```

3. Run the `metaroot(1M)` command.

```
# /usr/opt/SUNWmd/metaroot d0
```

This edits the /etc/system and /etc/vfstab files to specify that the root (/) file system is now on metadevice d0.

4. Verify that the `/etc/vfstab` file contains the correct metadevice entries.

The root (/) entry in the /etc/vfstab file should appear as follows so that the entry for the file system correctly references the mirror:

```
#device           device              mount    FS      fsck   mount   mount
#to mount         to fsck             point    type    pass   at boot options
#
/dev/md/dsk/d0    /dev/md/rdsk/d0     /        ufs     1      no      -
/dev/dsk/c0t3d0s1 -                   -        swap    -      no      -
/dev/dsk/c0t3d0s6 /dev/rdsk/c0t3d0s6  /usr     ufs     2      no      -
#
/proc             -                  /proc     proc    -      no      -
fd                -                  /dev/fd   fd      -      no      -
swap              -                  /tmp      tmpfs   -      yes     -
```

5. **Reboot.**

   The system returns to normal operation.

▼ How to Recover From Insufficient State Database Replicas (Command Line)

If for some reason the state database replica quorum is not met, for example, due to a drive failure, the system cannot be rebooted. In DiskSuite terms, the state database has gone “stale.” This task explains how to recover.

The high-level steps in this task are:

- Deleting the stale state database replicas and rebooting
- Repairing the problem disk
- Adding back the state database replica(s)

Example — Recovering From Stale State Database Replicas

In the following example, a disk containing two replicas has gone bad. This leaves the system with only two good replicas, and the system cannot reboot.

1. **Boot the machine to determine which state database replicas are down.**

```
ok boot
...
Hostname: demo
metainit: demo: stale databases
```

(Continuation)

```
Insufficient metadevice database replicas located.

Use metadb to delete databases which are broken.
Ignore any "Read-only file system" error messages.
Reboot the system when finished to reload the metadevice
database.
After reboot, repair any broken database replicas which were
deleted.

Type Ctrl-d to proceed with normal startup,
(or give root password for system maintenance): <root-password>
Entering System Maintenance Mode

SunOS Release 5.5 Version Generic [UNIX(R) System V Release 4.0]
```

2. Use the `metadb(1M)` command to look at the metadevice state database and see which state database replicas are not available.

```
# /usr/opt/SUNWmd/metadb -i
   flags      first blk      block count
    a m  p  lu    16               1034                 /dev/dsk/c0t3d0s3
    a   p  l      1050             1034                 /dev/dsk/c0t3d0s3
    M  p        unknown      unknown                    /dev/dsk/c1t2d0s3
    M  p        unknown      unknown                    /dev/dsk/c1t2d0s3
...
```

The system can no longer detect state database replicas on slice `/dev/dsk/c1t2d0s3`, which is part of the failed disk. The `metadb` command flags the replicas on this slice as having a problem with the master blocks.

3. Delete the state database replicas on the bad disk using the `-d` option to the `metadb(1M)` command.

At this point, the root (`/`) file system is read-only. You can ignore the `mddb.cf` error messages:

```
# /usr/opt/SUNWmd/metadb -d -f c1t2d0s3
metadb: demo: /etc/opt/SUNWmd/mddb.cf.new: Read-only file
system
```

4. **Verify that the replicas were deleted.**

```
# /usr/opt/SUNWmd/metadb -i
    flags        first blk       block count
     a m  p  lu         16              1034            /dev/dsk/c0t3d0s3
     a    p  l          1050            1034            /dev/dsk/c0t3d0s3
```

5. **Reboot.**

6. **Once you have a replacement disk, halt the system, replace the failed disk, and once again, reboot the system. Use the** `format(1M)` **command or the** `fmthard(1M)` **command to partition the disk as it was before the failure.**

```
# halt
...
ok boot
...
# format /dev/rdsk/c1t2d0s0
...
```

7. **Use the** `metadb(1M)` **command to add back the state database replicas and to determine that the state database replicas are correct.**

```
# /usr/opt/SUNWmd/metadb -a -c 2 c1t2d0s3
# /usr/opt/SUNWmd/metadb
   flags        first blk  block count
  a m  p  luo      16           1034         dev/dsk/c0t3d0s3
  a    p  luo      1050         1034         dev/dsk/c0t3d0s3
  a       u        16           1034         dev/dsk/c1t2d0s3
  a       u        1050         1034         dev/dsk/c1t2d0s3
```

The `metadb` command with the `-c 2` option adds two state database replicas to the same slice.

▼ How to Recover From a Boot Device Failure (Command Line)

If you have a root (/) mirror and your boot device fails, you'll need to set up an alternate boot device.

The high-level steps in this task are:

- Booting from the alternate root (/) submirror
- Determining the errored state database replicas and metadevices
- Repairing the problem disk
- Restoring metadevice state database and metadevices to their original state

Example — Recovering From a Boot Device Failure

In the following example, the boot device containing two of the six state database replicas and the root (/), `swap`, and `/usr` submirrors fails.

Initially, when the boot device fails, you'll see a message similar to the following. This message may differ among various architectures.

```
Rebooting with command:
Boot device: /iommu/sbus/dma@f,81000/esp@f,80000/sd@3,0   File and args: kadb
kadb: kernel/unix
The selected SCSI device is not responding
Can't open boot device
...
```

When you see this message, note the device. Then, follow these steps:

1. **Boot from another root (/) submirror.**

   Since only two of the six state database replicas in this example are in error, you can still boot. If this were not the case, you would need to delete the stale state database replicas in single-user mode. This procedure is described in "How to Recover From Insufficient State Database Replicas (Command Line)" on page 231.

   When you created the mirror for the root (/) file system, you should have recorded the alternate boot device as part of that procedure. In this example, `disk2` is that alternate boot device.

```
ok boot disk2
...
SunOS Release 5.5 Version Generic [UNIX(R) System V Release 4.0]
Copyright (c) 1983-1995, Sun Microsystems, Inc.
```

(continued)

(Continuation)

```
Hostname: demo
...
demo console login: root
Password: <root-password>
Last login: Wed Dec 16 13:15:42 on console
SunOS Release 5.1 Version Generic [UNIX(R) System V Release 4.0]
...
```

2. Use the `metadb(1M)` command to determine that two state database replicas have failed.

```
# /usr/opt/SUNWmd/metadb
       flags         first blk    block count
    M     p          unknown      unknown      /dev/dsk/c0t3d0s3
    M     p          unknown      unknown      /dev/dsk/c0t3d0s3
    a m  p  luo      16           1034         /dev/dsk/c0t2d0s3
    a    p  luo      1050         1034         /dev/dsk/c0t2d0s3
    a    p  luo      16           1034         /dev/dsk/c0t1d0s3
    a    p  luo      1050         1034         /dev/dsk/c0t1d0s3
```

The system can no longer detect state database replicas on slice `/dev/dsk/c0t3d0s3`, which is part of the failed disk.

3. Use the `metastat(1M)` command to determine that half of the root (`/`), `swap`, and `/usr` mirrors have failed.

```
# /usr/opt/SUNWmd/metastat
d0: Mirror
    Submirror 0: d10
      State: Needs maintenance
    Submirror 1: d20
      State: Okay
...

d10: Submirror of d0
    State: Needs maintenance
    Invoke: "metareplace d0 /dev/dsk/c0t3d0s0 <new device>"
    Size: 47628 blocks
    Stripe 0:
 Device              Start Block  Dbase State        Hot Spare
 /dev/dsk/c0t3d0s0          0     No    Maintenance
```

(continued)

(Continuation)

```
d20: Submirror of d0
    State: Okay
    Size: 47628 blocks
    Stripe 0:
 Device              Start Block  Dbase State        Hot Spare
 /dev/dsk/c0t2d0s0          0     No    Okay

d1: Mirror
    Submirror 0: d11
      State: Needs maintenance
    Submirror 1: d21
      State: Okay
...

d11: Submirror of d1
    State: Needs maintenance
    Invoke: "metareplace d1 /dev/dsk/c0t3d0s1 <new device>"
    Size: 69660 blocks
    Stripe 0:
 Device              Start Block  Dbase State        Hot Spare
 /dev/dsk/c0t3d0s1          0     No    Maintenance

d21: Submirror of d1
    State: Okay
    Size: 69660 blocks
    Stripe 0:
 Device              Start Block  Dbase State        Hot Spare
 /dev/dsk/c0t2d0s1          0     No    Okay

d2: Mirror
    Submirror 0: d12
      State: Needs maintenance
    Submirror 1: d22
      State: Okay
...

d2: Mirror
    Submirror 0: d12
      State: Needs maintenance
    Submirror 1: d22
      State: Okay
...

d12: Submirror of d2
    State: Needs maintenance
    Invoke: "metareplace d2 /dev/dsk/c0t3d0s6 <new device>"
    Size: 286740 blocks
    Stripe 0:
 Device              Start Block  Dbase State        Hot Spare
 /dev/dsk/c0t3d0s6          0     No    Maintenance


d22: Submirror of d2
```

(continued)

```
   State: Okay
   Size: 286740 blocks
   Stripe 0:
Device              Start Block  Dbase State        Hot Spare
/dev/dsk/c0t2d0s6          0     No    Okay
```

In this example, the `metastat` shows that following submirrors need maintenance:

- Submirror `d10`, device `c0t3d0s0`
- Submirror `d11`, device `c0t3d0s1`
- Submirror `d12`, device `c0t3d0s6`

4. Halt the system, repair the disk, and use the `format(1M)` command or the `fmthard(1M)` command, to partition the disk as it was before the failure.

```
# halt
...
Halted
...
ok boot
...
# format /dev/rdsk/c0t3d0s0
```

5. Reboot.

Note that you must reboot from the other half of the root (`/`) mirror. You should have recorded the alternate boot device when you created the mirror.

```
# halt
...
ok boot disk2
```

6. To delete the failed state database replicas and then add them back, use the `metadb(1M)` command.

```
# /usr/opt/SUNWmd/metadb
       flags         first blk    block count
    M     p          unknown      unknown      /dev/dsk/c0t3d0s3
    M     p          unknown      unknown      /dev/dsk/c0t3d0s3
    a m  p  luo      16           1034         /dev/dsk/c0t2d0s3
    a    p  luo      1050         1034         /dev/dsk/c0t2d0s3
    a    p  luo      16           1034         /dev/dsk/c0t1d0s3
    a    p  luo      1050         1034         /dev/dsk/c0t1d0s3
# /usr/opt/SUNWmd/metadb -d c0t3d0s3
# /usr/opt/SUNWmd/metadb -c 2 -a c0t3d0s3
# /usr/opt/SUNWmd/metadb
       flags         first blk    block count
     a m  p  luo     16           1034         /dev/dsk/c0t2d0s3
     a    p  luo     1050         1034         /dev/dsk/c0t2d0s3
     a    p  luo     16           1034         /dev/dsk/c0t1d0s3
     a    p  luo     1050         1034         /dev/dsk/c0t1d0s3
     a        u      16           1034         /dev/dsk/c0t3d0s3
     a        u      1050         1034         /dev/dsk/c0t3d0s3
```

7. Use the `metareplace(1M)` **command to re-enable the submirrors.**

```
# /usr/opt/SUNWmd/metareplace -e d0 c0t3d0s0
Device /dev/dsk/c0t3d0s0 is enabled

# /usr/opt/SUNWmd/metareplace -e d1 c0t3d0s1
Device /dev/dsk/c0t3d0s1 is enabled

# /usr/opt/SUNWmd/metareplace -e d2 c0t3d0s6
Device /dev/dsk/c0t3d0s6 is enabled
```

After some time, the resyncs will complete. You can now return to booting from the original device.

▼ How to Record the Path to the Alternate Boot Device (Command Line)

When mirroring root (/), you might need the path to the alternate boot device later if the primary device fails.

Example — SPARC: Recording the Alternate Boot Device Path

In this example, you would determine the path to the alternate root device by using the `ls -l` command on the slice that is being attached as the second submirror to the root (`/`) mirror.

```
# ls -l /dev/rdsk/c1t3d0s0
lrwxrwxrwx 1  root root  55 Mar 5 12:54  /dev/rdsk/c1t3d0s0 -> ../.
./devices/sbus@1,f8000000/esp@1,200000/sd@3,0:a
```

Here you would record the string that follows the `/devices` directory: `/sbus@1,f8000000/esp@1,200000/sd@3,0:a`

DiskSuite users who are using a system with Open Boot Prom can use the OpenBoot `nvalias` command to define a “backup root” devalias for the secondary root mirror. For example:

```
ok  nvalias backup_root /sbus@1,f8000000/esp@1,200000/sd@3,0:a
```

In the event of primary root disk failure, you then would only enter:

```
ok  boot backup_root
```

Example — x86: Recording the Alternate Boot Device Path

In this example, you would determine the path to the alternate boot device by using the `ls -l` command on the slice that is being attached as the second submirror to the root (`/`) mirror.

```
# ls -l /dev/rdsk/c1t0d0s0
lrwxrwxrwx 1  root root  55 Mar 5 12:54  /dev/rdsk/c1t0d0s0 -> ../.
./devices/eisa/eha@1000,0/cmdk@1,0:a
```

Here you would record the string that follows the `/devices` directory: `/eisa/eha@1000,0/cmdk@1,0:a`

▼ SPARC: How to Boot From the Alternate Device (Command Line)

To boot a SPARC system from the alternate boot device, type:

```
# boot alternate-boot-device
```

The procedure, “How to Record the Path to the Alternate Boot Device (Command Line)” on page 238, describes how to determine the alternate boot device.

▼ x86: How to Boot From the Alternate Device (Command Line)

Use this task to boot an x86 system from the alternate boot device.

1. **Boot your system from the Multiple Device Boot (MDB) diskette.**

   After a moment, a screen similar to the following is displayed:

```
Solaris/x86 Multiple Device Boot Menu
Code    Device    Vendor     Model/Desc          Rev
============================================================

10      DISK      COMPAQ     C2244               0BC4
11      DISK      SEAGATE    ST11200N SUN1.05    8808
12      DISK      MAXTOR     LXT-213S SUN0207    4.24
13      CD        SONY       CD-ROM CDU-8812     3.0a
14      NET       SMC/WD     I/O=300 IRQ=5
80      DISK      First IDE drive (Drive C:)
81      DISK      Second IDE drive (Drive D:)

Enter the boot device code:
```

2. **Enter your alternate disk code from the choices listed on the screen. The following is displayed:**

```
Solaris 2.4 for x86               Secondary Boot Subsystem,vsn 2.11

                  <<<Current Boot Parameters>>>
Boot path:/eisa/eha@1000,0/cmdk@0,0:a
Boot args:/kernel/unix

Type b[file-name] [boot-flags] <ENTER>     to boot with options
or   i<ENTER>                             to enter boot interpreter
or   <ENTER>                               to boot with defaults

                    <<<timeout in 5 seconds>>>
```

3. **Type `i` to select the interpreter.**

4. **Type the following commands:**

```
>setprop boot-path /eisa/eha@1000,0/cmdk@1,0:a
>^D
```

The Control-D character sequence quits the interpreter.

Replacing SCSI Disks

This section describes how to replace SCSI disks that are not part of a SPARCstorage Array in a DiskSuite environment.

▼ How to Replace a Failed SCSI Disk (Command Line)

The high-level steps to replace a SCSI disk that is not part of a SPARCstorage Array are:

- Identifying the disk that needs replacing
- Deleting any metadevice state database replicas that are on the problem disk
- Deleting hot spares marked “Available” that are on the problem disk
- Locating and detaching submirrors that use slices on the problem disk
- Halting the system and booting to singler-user mode
- Physically replacing the disk
- Repartitioning the new disk
- Adding metadevice state database replicas that were deleted
- Performing one of the following, depending on how the slice that failed was used:

  For a simple slice: use normal recovery procedures For a stripe or concatenation: `newfs` entire metadevice, restore from backup For a mirror: reattach detached submirrors For a RAID5 metadevice: resync (enable) affected slices For a trans metadevice: run `fsck(1M)`
- Adding hot spares that were deleted to hot spare pools

1. **Identify the disk to be replaced by examining `/var/adm/messages` and `metastat` output.**

2. **Locate any local metadevice state database replicas that may have been placed on the problem disk. Use the `metadb` command to find the replicas.**

   Errors may be reported for the replicas located on the failed disk. In this example, `c0t1d0` is the problem device.

```
# metadb
   flags       first blk        block count
  a m     u         16               1034            /dev/dsk/c0t0d0s4
  a       u         1050             1034            /dev/dsk/c0t0d0s4
  a       u         2084             1034            /dev/dsk/c0t0d0s4
  W   pc luo        16               1034            /dev/dsk/c0t1d0s4
  W   pc luo        1050             1034            /dev/dsk/c0t1d0s4
  W   pc luo        2084             1034            /dev/dsk/c0t1d0s4
```

   The output above shows three state database replicas on Slice 4 of each of the local disks, `c0t0d0` and `c0t1d0`. The `W` in the flags field of the `c0t1d0s4` slice indicates that the device has write errors. Three replicas on the `c0t0d0s4` slice are still good.

Caution - If, after deleting the bad state database replicas, you are left with three or less, add more state database replicas before continuing. This will ensure that your system reboots correctly.

3. **Record the slice name where the replicas reside and the number of replicas, then delete the state database replicas.**

   The number of replicas is obtained by counting the number of appearances of a slice in `metadb` output in Step 2 on page 242. In this example, the three state database replicas that exist on `c0t1d0s4` are deleted.

```
# metadb -d c0t1d0s4
```

4. **Locate any submirrors using slices on the problem disk and detach them.**

   The `metastat` command can show the affected mirrors. In this example, one submirror, `d10`, is also using `c0t1d0s4`. The mirror is `d20`.

```
# metadetach d20 d10
d20: submirror d10 is detached
```

5. Delete hot spares on the problem disk.

```
# metahs -d hsp000 c0t1d0s6
hsp000: Hotspare is deleted
```

6. Halt the system and boot to single-user mode.

```
# halt
...
ok boot -s
...
```

7. Physically replace the problem disk.

8. Repartition the new disk.

Use the `format(1M)` command or the `fmthard(1M)` command to partition the disk with the same slice information as the failed disk.

9. If you deleted replicas in Step 3 on page 242, add the same number back to the appropriate slice.

In this example, `/dev/dsk/c0t1d0s4` is used.

```
# metadb -a c 3 c0t1d0s4
```

10. Depending on how the disk was used, you may have a variety of things to do. Use the following table to decide what to do next.

TABLE 7–2 SCSI Disk Replacement Decision Table

Type of Device	Do the Following ...
Slice	Use normal data recovery procedures.
Unmirrored Stripe or Concatenation	If the stripe/concat is used for a file system, run `newfs(1M)`, mount the file system then restore data from backup. If the stripe/concat is used as an application that uses the raw device, that application must have its own recovery procedures.
Mirror (Submirror)	Run `metattach(1M)` to reattach a detached submirror.
RAID5 metadevice	Run `metareplace(1M)` to re-enable the slice. This causes the resyncs to start.
Trans metadevice	Run `fsck(1M)` to repair the trans metadevice.

11. Replace hot spares that were deleted, and add them to the appropriate hot spare pool(s).

```
# metahs -a hsp000 c0t0d0s6
hsp000: Hotspare is added
```

12. Validate the data.

Check the user/application data on all metadevices. You may have to run an application-level consistency checker or use some other method to check the data.

Working With SPARCstorage Arrays

This section describes how to troubleshoot SPARCstorage Arrays using DiskSuite. The tasks in this section include:

- Replacing a failed disk in a mirror
- Replacing a failed disk in a RAID5 metadevice

- Removing a tray
- Replacing a tray
- Replacing a controller
- Recovering from power loss

Installation

The SPARCstorage Array should be installed according to the *SPARCstorage Array Software* instructions found with the SPARCstorage Array CD. The SPARCstorage Array Volume Manager need not be installed if you are only using DiskSuite.

Device Naming

DiskSuite accesses SPARCstorage Array disks exactly like any other disks, with one important exception: the disk names differ from non-SPARCstorage Array disks.

The SPARCstorage Array 100 disk naming convention is:

`c[0-n]t[0-5]d[0-4]s[0-7]`

In this name:

- `c` indicates the controller attached to an SSA unit
- `t` indicates one of the 6 SCSI strings within an SSA
- `d` indicates one of the 5 disks on an internal SCSI string
- `s` indicates the disk slice number
- Strings `t0` and `t1` are contained in tray 1, `t2` and `t3` in tray 2, and `t4` and `t5` are in tray 3

The SPARCstorage Array 200 disk naming convention is:

`c[0-n]t[0-5]d[0-6]s[0-7]`

In this name:

- `c` indicates the controller attached to an SSA unit
- `t` indicates one of the 6 targets (trays) within the SSA unit
- `d` indicates one of the 7 disks on an internal SCSI string
- `s` indicates the disk slice number

Note - Older trays hold up to six disks; newer trays can hold up to seven.

The main difference between the SSA100 and SSA200 is that the SSA100 arranges pairs of targets into a tray, whereas the SSA200 has a separate tray for each target.

Preliminary Information for Replacing SPARCstorage Array Components

The SPARCstorage Array components that can be replaced include the disks, fan tray, battery, tray, power supply, backplane, controller, optical module, and fibre channel cable.

Some of the SPARCstorage Array components can be replaced without powering down the SPARCstorage Array. Other components require the SPARCstorage Array to be powered off. Consult the SPARCstorage Array documentation for details.

To replace SPARCstorage Array components that require power off without interrupting services, you perform the steps necessary for tray removal for all trays in the SPARCstorage Array before turning off the power. This includes taking submirrors offline, deleting hot spares from hot spare pools, deleting state database replicas from drives, and spinning down the trays.

After these preparations, the SPARCstorage Array can be powered down and the components replaced.

Note - Because the SPARCstorage Array controller contains a unique World Wide Name, which identifies it to Solaris, special procedures apply for SPARCstorage Array controller replacement. Contact your service provider for assistance.

▼ How to Replace a Failed SPARCstorage Array Disk in a Mirror (DiskSuite Tool)

The steps to replace a SPARCstorage Array disk in a DiskSuite environment depend a great deal on how the slices on the disk are being used, and how the disks are cabled to the system. They also depend on whether the disk slices are being used as is, or by DiskSuite, or both.

Note - This procedure applies to a SPARCstorage Array 100. The steps to replace a disk in a SPARCstorage Array 200 are similar.

The high-level steps in this task are:

- Identifying the disk that needs replacing and determining its location
- Deleting hot spares marked “Available” that are in the tray that must be pulled
- Deleting state database replicas that are on the disks in the tray that must be pulled
- Locating submirrors using disks in the tray that must be pulled
- Detaching submirrors with slices on the disk that is being replaced
- Offlining other submirrors using disks in the tray
- Spinning down all disks in the tray

- Pulling the tray and replacing the disk
- Making sure that all disks in the tray spin back up
- Repartitioning the new disk
- Bringing submirrors in the tray back online
- Attaching detached submirrors in the tray
- Replacing hot spares that were deleted
- Adding hot spares that were deleted to hot spare pool
- Adding metadevice state database replicas that were deleted

---

Note - You can use this procedure if a submirror is in the "Maintenance" state, replaced by a hot spare, or is generating intermittent errors.

To locate and replace the disk, perform the following steps:

1. **Identify the disk to be replaced, either by using DiskSuite Tool to look at the Status fields of objects, or by examining** `metastat` **and** `/var/adm/messages` **output.**

```
# metastat
...
 d50:Submirror of d40
      State: Needs Maintenance
...
# tail -f /var/adm/messages
...
Jun 1 16:15:26 host1 unix: WARNING: /io-
unit@f,e1200000/sbi@0.0/SUNW,pln@a0000000,741022/ssd@3,4(ssd49):
Jun 1 16:15:26 host1 unix: Error for command 'write(I))' Err
Jun 1 16:15:27 host1 unix: or Level: Fatal
Jun 1 16:15:27 host1 unix: Requested Block 144004, Error Block: 715559
Jun 1 16:15:27 host1 unix: Sense Key: Media Error
Jun 1 16:15:27 host1 unix: Vendor 'CONNER':
Jun 1 16:15:27 host1 unix: ASC=0x10(ID CRC or ECC error),ASCQ=0x0,FRU=0x15
...
```

The `metastat` command shows that a submirror is in the "Needs Maintenance" state. The `/var/adm/messages` file reports a disk drive that has an error. To locate the disk drive, use the `ls` command as follows, matching the symbolic link name to that from the `/var/adm/messages` output.

```
# ls -l /dev/rdsk/*
...
lrwxrwxrwx   1 root     root          90 Mar  4 13:26 /dev/rdsk/c3t3d4s0 -
> ../../devices/io-
unit@f,e1200000/sbi@0.0/SUNW,pln@a0000000,741022/ssd@3,4(ssd49)
```

(Continuation)

```
...
```

Based on the above information and `metastat` output, it is determined that drive `c3t3d4` must be replaced.

2. Determine the affected tray by using DiskSuite Tool.

To find the SPARCstorage Array tray where the problem disk resides, use the Disk View window.

a. Click Disk View to display the Disk View window.

b. Drag the problem metadevice (in this example, a mirror) from the Objects list to the Disk View window.

The Disk View window shows the logical to physical device mappings by coloring the physical slices that make up the metadevice. You can see at a glance which tray contains the problem disk.

c. An alternate way to find the SPARCstorage Array tray where the problem disk resides is to use the `ssaadm(1M)` command.

```
host1# ssaadm display c3
        SPARCstorage Array Configuration
Controller path: /devices/io-
unit@f,e1200000/sbi@0.0/SUNW,soc@0,0/SUNW,pln@a0000000,741022:ctlr
        DEVICE STATUS
        TRAY1          TRAY2          TRAY3
Slot
1       Drive:0,0      Drive:2,0      Drive:4,0
2       Drive:0,1      Drive:2,1      Drive:4,1
3       Drive:0,2      Drive:2,2      Drive:4,2
4       Drive:0,3      Drive:2,3      Drive:4,3
5       Drive:0,4      Drive:2,4      Drive:4,4
6       Drive:1,0      Drive:3,0      Drive:5,0
7       Drive:1,1      Drive:3,1      Drive:5,1
8       Drive:1,2      Drive:3,2      Drive:5,2
9       Drive:1,3      Drive:3,3      Drive:5,3
10      Drive:1,4      Drive:3,4      Drive:5,4

        CONTROLLER STATUS
Vendor:    SUNW
Product ID:  SSA100
Product Rev: 1.0
Firmware Rev: 2.3
Serial Num: 000000741022
```

(continued)

```
Accumulate performance Statistics: Enabled
```

The `ssaadm` output for controller (`c3`) shows that `Drive 3,4` (`c3t3d4`) is the closest to you when you pull out the middle tray.

3. [Optional] If you have a diskset, locate the diskset that contains the affected drive.

The following commands locate drive `c3t3d4`. Note that no output was displayed when the command was run with `logicalhost2`, but `logicalhost1` reported that the name was present. In the reported output, the `yes` field indicates that the disk contains a state database replica.

```
host1# metaset -s logicalhost2 | grep c3t3d4
host1# metaset -s logicalhost1 | grep c3t3d4
c3t3d4 yes
```

Note - If you are using Solstice HA servers, you'll need to switch ownership of both logical hosts to one Solstice HA server. Refer to the Solstice HA documentation.

---

4. Determine other DiskSuite objects on the affected tray.

Because you must pull the tray to replace the disk, determine what other objects will be affected in the process.

a. In DiskSuite Tool, display the Disk View window. Select the tray. From the Object menu, choose Device Mappings. The Physical to Logical Device Mapping window appears.

b. Note all affected objects, including state database replicas, metadevices, and hot spares that appear in the window.

5. Prepare for disk replacement by preparing other DiskSuite objects in the affected tray.

a. **Delete all hot spares that have a status of "Available" and that are in the same tray as the problem disk.**

   Record all the information about the hot spares so they can be added back to the hot spare pools following the replacement procedure.

b. **Delete any state database replicas that are on disks in the tray that must be pulled. You must keep track of this information because you must replace these replicas in Step 14 on page 251.**

   There may be multiple replicas on the same disk. Make sure you record the number of replicas deleted from each slice.

c. **Locate the submirrors that are using slices that reside in the tray.**

d. **Detach all submirrors with slices on the disk that is being replaced.**

e. **Take all other submirrors that have slices in the tray offline.**

   This forces DiskSuite to stop using the submirror slices in the tray so that the drives can be spun down.

   To remove objects, refer to Chapter 5. To detach and offline submirrors, refer to "Working With Mirrors" on page 141.

6. **Spin down all disks in SPARCstorage Array tray.**

   Refer to "How to Stop a Disk (DiskSuite Tool)" on page 208.

Note - The SPARCstorage Array tray should not be removed as long as the LED on the tray is illuminated. Also, you should not run any DiskSuite commands while the tray is spun down as this may have the side effect of spinning up some or all of the drives in the tray.

---

7. **Pull the tray and replace the bad disk.**

   Instructions for the hardware procedure are found in the SPARCstorage Array Model 100 Series Service Manual and the *SPARCcluster High Availability Server Service Manual*.

8. **Make sure all disks in the tray of the SPARCstorage Array spin up.**

   The disks in the SPARCstorage Array tray should automatically spin up following the hardware replacement procedure. If the tray fails to spin up automatically within two minutes, force the action by using the following command.

```
# ssaadm start -t 2 c3
```

9. **Use `format(1M)`, `fmthard(1M)`, or Storage Manager to repartition the new disk. Make sure you partition the new disk exactly as the disk that was replaced.**

   Saving the disk format information before problems occur is always a good idea.

10. **Bring all submirrors that were taken offline back online.**

    Refer to “Working With Mirrors” on page 141.

    When the submirrors are brought back online, DiskSuite automatically resyncs all the submirrors, bringing the data up-to-date.

11. **Attach submirrors that were detached.**

    Refer to “Working With Mirrors” on page 141.

12. **Replace any hot spares in use in the submirrors attached in Step 11 on page 251.**

    If a submirror had a hot spare replacement in use before you detached the submirror, this hot spare replacement will be in effect after the submirror is reattached. This step returns the hot spare to the “Available” status.

13. **Add all hot spares that were deleted.**

14. **Add all state database replicas that were deleted from disks on the tray.**

    Use the information saved previously to replace the state database replicas.

15. **[Optional] If using Solstice HA servers, switch each logical host back to its default master.**

    Refer to the Solstice HA documentation.

16. **Validate the data.**

    Check the user/application data on all metadevices. You may have to run an application-level consistency checker or use some other method to check the data.

▼ How to Replace a Failed SPARCstorage Array Disk in a RAID5 Metadevice (DiskSuite Tool)

When setting up RAID5 metadevices for online repair, you will have to use a minimum RAID5 width of three slices. While this is not an optimal configuration for RAID5, it is still slightly less expensive than mirroring, in terms of the overhead of the redundant data. You should place each of the three slices of each RAID5 metadevice within a separate tray. If all disks in a SPARCstorage Array are configured this way (or in combination with mirrors as described above), the tray containing the failed disk may be removed without losing access to any of the data.

Caution - Any applications using non-replicated disks in the tray containing the failed drive should first be suspended or terminated.

1. **Refer to through Step 9 on page 251 in the previous procedure, "How to Replace a Failed SPARCstorage Array Disk in a Mirror (DiskSuite Tool)" on page 246.**

   You are going to locate the problem disk and tray, locate other affected DiskSuite objects, prepare the disk to be replaced, replace, then repartition the drive.

2. **Use the** `metareplace -e` **command to enable the new drive in the tray.**

3. **Refer to Step 12 on page 251 through Step 16 on page 251 in the previous procedure, "How to Replace a Failed SPARCstorage Array Disk in a Mirror (DiskSuite Tool)" on page 246.**

▼ How to Remove a SPARCstorage Array Tray (Command Line)

Before removing a SPARCstorage Array tray, halt all I/O and spin down all drives in the tray. The drives automatically spin up if I/O requests are made. Thus, it is necessary to stop all I/O before the drives are spun down.

1. **Stop DiskSuite I/O activity.**

   Refer to the `metaoffline(1M)` command, which takes the submirror offline. When the submirrors on a tray are taken offline, the corresponding mirrors will only provide one-way mirroring (that is, there will be no data redundancy), unless the mirror uses three-way mirroring. When the submirror is brought back online, an automatic resync occurs.

   Note - If you are replacing a drive that contains a submirror, use the `metadetach(1M)` command to detach the submirror.

2. **Use the** `metastat(1M)` **command to identify all submirrors containing slices on the tray to be removed. Also, use the** `metadb(1M)` **command to identify any replicas on the tray. Any available hot spare devices must also be identified and the associated submirror identified using the** `metahs(1M)` **command.**

   With all affected submirrors offline, I/O to the tray will be stopped.

3. **Refer to "How to Stop a Disk (DiskSuite Tool)" on page 208.**

Either using DiskSuite Tool or the `ssaadm` command, spin down the tray. When the tray lock light is out the tray may be removed and the required task performed.

▼ How to Replace a SPARCstorage Array Tray

When you have completed work on a SPARCstorage Array tray, replace the tray in the chassis. The disks will automatically spin up.

However if the disks fail to spin up, you can use DiskSuite Tool (or the `ssaadm` command) to manually spin up the entire tray. There is a short delay (several seconds) between starting drives in the SPARCstorage Array.

After the disks have spun up, you must place online all the submirrors that were taken offline. When you bring a submirror online, an optimized resync operation automatically brings the submirrors up-to-date. The optimized resync copies only the regions of the disk that were modified while the submirror was offline. This is typically a very small fraction of the submirror capacity. You must also replace all state database replicas and add back hot spares.

Note - If you used `metadetach(1M)` to detach the submirror rather than `metaoffline`, the entire submirror must be resynced. This typically takes about 10 minutes per Gbyte of data.

▼ How to Recover From SPARCstorage Array Power Loss (Command Line)

When power is lost to one SPARCstorage Array, the following occurs:

- I/O operations to the DiskSuite objects will generate errors.
- Errors are reported at the slice level rather than the drive level.
- Errors are not reported until I/O operations are made to the disk.
- Hot spare activity may be initiated if affected devices have assigned hot spares.

You must monitor the configuration for these events using the `metastat(1M)` command as explained in “Checking Status of DiskSuite Objects” on page 80.

You may need to perform the following after power is restored:

- Identify errored devices with `metastat`
- Enable errored submirrors or RAID5 metadevices
- Delete/recreate affected state database replicas

1. **After power is restored, use the `metastat` command to identify the errored devices.**

```
# metastat
...
d10: Trans
    State: Okay
    Size: 11423440 blocks
    Master Device: d20
    Logging Device: d15

d20: Mirror
    Submirror 0: d30
      State: Needs maintenance
    Submirror 1: d40
      State: Okay
...
d30: Submirror of d20
    State: Needs maintenance
...
```

2. **Return errored devices to service using the** `metareplace` **command**:

```
# metareplace -e metadevice slice
```

The `-e` option transitions the state of the slice to the "Available" state and resyncs the failed slice.

Note - Slices that have been replaced by a hot spare should be the last devices replaced using the `metareplace` command. If the hot spare is replaced first, it could replace another errored slice in a submirror as soon as it becomes available.

A resync can be performed on only one slice of a submirror (metadevice) at a time. If all slices of a submirror were affected by the power outage, each slice must be replaced separately. It takes approximately 10 minutes for a resync to be performed on a 1.05-Gbyte disk.

Depending on the number of submirrors and the number of slices in these submirrors, the resync actions can require a considerable amount of time. A single submirror that is made up of 30 1.05-Gbyte drives might take about five hours to complete. A more realistic configuration made up of five-slice submirrors might take only 50 minutes to complete.

3. **After the loss of power, all state database replicas on the affected SPARCstorage Array chassis will enter an errored state. While these will be reclaimed at the next reboot, you may want to manually return them to service by first deleting and then adding them back.**

```
# metadb -d slice
# metadb -a slice
```

Note - Make sure you add back the same number of state database replicas that were deleted on each slice. Multiple state database replicas can be deleted with a single `metadb` command. It may require multiple invocations of `metadb -a` to add back the replicas deleted by a single `metadb -d`. This is because if you need multiple copies of replicas on one slice these must be added in one invocation of `metadb` using the `-c` flag. Refer to the `metadb(1M)` man page for more information.

Because state database replica recovery is not automatic, it is safest to manually perform the recovery immediately after the SPARCstorage Array returns to service. Otherwise, a new failure may cause a majority of state database replicas to be out of service and cause a kernel panic. This is the expected behavior of DiskSuite when too few state database replicas are available.

▼ How to Move SPARCstorage Array Disks Between Hosts (Command Line)

This procedure explains how to move disks containing DiskSuite objects from one SPARCstorage Array to another.

1. **Repair any devices that are in an errored state or that have been replaced by hot spares on the disks that are to be moved.**

2. **Identify the state database replicas, metadevices, and hot spares on the disks that are to be moved, by using the output from the `metadb` and `metastat -p` commands.**

3. **Physically move the disks to the new host, being careful to connect them in a similar fashion so that the device names are the same.**

4. **Recreate the state database replicas.**

```
# metadb -a [-f] slice ...
```

Be sure to use the same slice names that contained the state database replicas as identified in Step 2 on page 255. You might need to use the `-f` option to force the creation of the state database replicas.

5. **Copy the output from the `metastat -p` command in Step 2 on page 255 to the `md.tab` file.**

6. **Edit the `md.tab` file, making the following changes:**
   - Delete metadevices which you did not move.
   - Change the old metadevice names to new names.
   - Make any mirrors into one-way mirrors for the time being, selecting the smallest submirror (if appropriate).

7. **Check the syntax of the `md.tab` file.**

```
# metainit -a -n
```

8. **Recreate the moved metadevices and hot spare pools.**

```
# metainit -a
```

9. **Make the one-way mirrors into multi-way mirrors using the `metattach(1M)` command as necessary.**

10. **Edit the `/etc/vfstab` file for file systems that are to be automatically mounted at boot. Then remount file systems on the new metadevices as necessary.**

Using the SPARCstorage Array as a System Disk

This section contains information on making a SPARCstorage Array function as a system disk (boot device).

Making a SPARCstorage Array Bootable

The minimum boot requirements for the SPARCstorage Array are:

- Solaris 2.4 hardware 3/95 or later
- Fcode 1.33 or later on the SOC card in the host
- Firmware 1.9 or above in the SPARCstorage Array

To update or check the Fcode revision, use the `fc_update` program, which is supplied on the SPARCstorage Array CD, in its own subdirectory.

Consult the SPARCstorage Array documentation for more details.

▼ How to Make SPARCstorage Array Disks Available Early in the Boot Process

Add the following `forceload` entries to `/etc/system` file to ensure that the SPARCstorage Array disks are made available early in the boot process. This is necessary to make the SPARCstorage Array function as a system disk (boot device).

```
*ident "@(#)system 1.15 92/11/14 SMI" /* SVR4 1.5 */
*
* SYSTEM SPECIFICATION FILE
*
...
* forceload:
*
* Cause these modules to be loaded at boot time, (just before mounting
* the root filesystem) rather than at first reference. Note that
*  forceload expects a filename which includes the directory. Also
* note that loading a module does not necessarily imply that it will
* be installed.
*
forceload: drv/ssd
forceload: drv/pln
forceload: drv/soc
...
```

Note - When creating a root (/) mirror on a SPARCstorage Array disk, running the `metaroot(1M)` command puts the above entries in the `/etc/system` file automatically.

CHAPTER 8

Tips and Tricks

This chapter describes some hints and tips for using DiskSuite.

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

State Database Replicas and Trans Metadevices

Creating a trans metadevice (UFS logging) is an easy way to increase availability of UFS. Here's a tip that makes efficient use of slices when using trans metadevices:

- On new systems, create two to three small slices (8-10MB each) on each disk. Use these slices to hold not only state database replicas but also logging devices. As a general rule of thumb, as you add new disks to a system, use this method for configuring state database replicas and trans metadevices. That way, you can take care of two DiskSuite objects with one slice.

For more information on adding a state database replica or creating a trans metadevice, refer to Chapter 2.

DiskSuite and Prestoserve

Prestoserve™ is a hardware/software product that speeds response time in disk write-bound applications. The product accelerates performance by selectively caching disk block device write operations in non-volatile memory, reducing disk I/O bottlenecks.

Prestoserve improves NFS™ server performance, many disk I/O bound applications, and many file systems.

DiskSuite is fully compatible with Prestoserve, with the following restrictions.

DiskSuite Objects Compatible With Prestoserve

- Stripes/concatenations
- Top-level metadevices (mirror is discouraged; see "Why is Using Prestoserve With Mirrors Discouraged?" on page 260)

DiskSuite Objects Incompatible With Prestoserve

- Underlying component (i.e., submirror)
- State database replicas
- Trans Metadevices (and the underlying master devices and logging devices)

Why is Using Prestoserve With Mirrors Discouraged?

The simple reason is that using Prestoserve together with mirors introduces a single point of failure into the I/O subsystem, which is exactly what mirrors are designed

to avoid. The use of Prestoserve lowers the MTBF of a mirror to approximately the same as a single disk.

Why is Using Prestoserve With Trans Metadevices Discouraged?

Prestoserve cannot be used on trans metadevices. Using Prestoserve on a logging UFS can cause system hangs or panics. Prestoserve operates by redirecting I/O from a device to NVRAM. This redirection interferes with the communication protocol between a logging UFS and a metadevice.

▼ How to Configure Prestoserve With DiskSuite (Command Line)

The following steps describe how to load and enable Prestoserve for use with DiskSuite. Basically, you edit the `/etc/system` file to load Prestoserve after the DiskSuite driver.

1. **Add the following line to the `/etc/system` file.**

```
exclude: drv/pr
```

2. **Edit the `/etc/init.d/SUNWmd.init` file and add the following lines to the end of the "start" clause.**

```
'start')
 rm -f /tmp/.mdlock
 if [ -x "$METAINIT" -a -c "$METADEV" ]; then
  #echo "$METAINIT -r"
  $METAINIT -r
  error=$?
  #echo "$error"
  case "$error" in
  0|1) ;;

  66)
echo "Insufficient metadevice database replicas located."
echo ""
echo "Use metadb to delete databases which are broken."
echo "Ignore any \"Read-only file system\" error messages."
echo "Reboot the system when finished to reload the metadevice
database."
echo "After reboot, repair any broken database replicas which
were deleted."
   /sbin/sulogin < /dev/console
echo "Resuming system initialization. Metadevice database will
remain stale."
```

(continued)

(Continuation)

```
  ;;

 *) echo "Unknown $METAINIT -r failure $error."
  ;;
 esac
   modload /kernel/drv/pr
   presto -p >/dev/null
fi
;;
```

3. Edit the `/etc/init.d/prestoserve` file.

Replace the following line:

```
presto -u
```

with the line:

```
presto -u /filesystem...
```

In this command, *filesystem*`...` is a list of every file system to be accelerated with Prestoserve. Do not include any of the following:

- root (`/`)
- `/usr`
- `/usr/kvm`
- `/var`
- `/var/adm`

DiskSuite Configuration Guidelines

A poorly designed DiskSuite configuration can degrade performance. This section offers tips for getting good performance from DiskSuite.

General Guidelines

- **Disk and controllers –** Place drives in a metadevice on separate drive paths. For SCSI drives, this means separate host adaptors. For IPI drives, this means separate controllers. Spreading the I/O load over several controllers improves metadevice performance and availability.

  For SPARCstorage Arrays, you should use drives in a mirror on different chassis, if possible. This type of configuration ensures that all mirror data would survive a SPARCstorage Array chassis failure. If you cannot spread drives across different chassis, then use drives in different trays. This enables you to offline submirrors and the tray to be spun down or removed for maintenance while the mirror stays online.

  For example, consider a two-way mirror with each submirror composed of a concatenation of three SPARCstorage Array disks. One submirror would consist of three disks in tray 1, and the other submirror would consist of drives in tray 2. Using the command line interface to initialize this configuration would look like this:

```
# metainit d1 3 1 c0t0d0s2 1 c0t0d1s2 c0t0d2s2
d1: Concat/Stripe is setup
# metainit d2 3 1 c0t2d0s2 1 c0t2d1s2 c0t2d2s2
d2: Concat/Stripe is setup
# metainit d0 -m d1
d0: Mirror is setup
# metattach d0 d2
d0: Component d2 is attached
```

Strings `t0` and `t1` are contained in tray 1, `t2` and `t3` in tray 2, and `t4` and `t5` are in tray 3. Hence, in the above commands, to create submirrors in different trays, we use string `t0` for one submirror, and `t2` for the second submirror.

- **System Files –** Never edit or remove the `/etc/opt/SUNWmd/mddb.cf` or `/etc/opt/SUNWmd/md.cf` files.

  Make sure these files are backed up on a regular basis.

- After a slice is defined as a metadevice and activated, do not use it for any other purpose.

- Have a hardcopy of output from the `prtvtoc(1M)` command in case you need to reformat a bad disk.

State Database Replica Guidelines

- When a state database replica is placed on a slice that becomes part of a metadevice, the capacity of the metadevice is reduced by the space occupied by the replica(s). The space used by a replica is rounded up to the next cylinder

boundary and this space is skipped by the metadevice. However, the default size of a state database replicas is only 1034 blocks, and combining replicas and metadevices in this way is actually a very efficient use of DiskSuite.

Striping Guidelines

- Stripes are created only from slices, not other metadevices.
- Do not use slices from disks with different disk geometry.
- Use slices on the same controller but on different disks. Using stripes that are each on different controllers can increase the number of simultaneous reads and writes that can be performed.
- Set up a stripe's interlace value to better match the I/O requests made by the system or applications.
- Do not stripe partitions that are on a single disk. This practice eliminates simultaneous access and causes performance problems.
- Use the same size disk components. Striping different size disk components results in unused disk space.
- If you use slices of different sizes when striping, then disk capacity is limited to a multiple of the smallest.

Concatenation Guidelines

- Concatenations are created only from slices, not other metadevices.
- Avoid using slices with different disk geometries.
- When possible, distribute the components of a concatenated metadevice across different controllers and buses.
- Concatenation uses less CPU cycles than striping. It performs well for small random I/O and for even I/O distribution.

Concatenated Stripe Guidelines

Read the guidelines for stripes and concatenations above.

Mirror Guidelines

- **Disks and controllers –** Keep the slices of different submirrors on different disks and controllers. Data protection is diminished considerably if slices of two or more

submirrors of the same mirror are on the same disk. Likewise, organize submirrors across separate controllers, because controllers and associated cables tends to fail more often than disks. This practice also improves mirror performance.

- **Same disk –** Do not define mirrors on the same disk. Writes to the same drive contend for the same resources, and the failure of the one drive would mean the loss of all data.
- **Read/write performance –** Mirroring may improve read performance, but write performance is always degraded. Mirroring improves read performance only in threaded or asynchronous I/O situations. No performance gain results if there is only a single thread reading from the metadevice.
- **Same-size submirrors –** Use the same size submirrors. Different size submirrors result in unused disk space.
- **Same-type disks and controllers –** Use the same type of disks and controllers in a single mirror. Particularly in old SCSI or SMD storage devices, different models or brands of disk or controller can have widely varying performance. Mixing the different performance levels in a single mirror can cause performance to degrade significantly.
- **Setting read and write policies for submirrors –** Experimenting with the mirror read policies can improve performance. For example, the default read mode is to alternate reads in a round-robin fashion among the disks. This is the default because it tends to work best for UFS multi-user, multi-process activity.

  In some cases, the `geometric` read option improves performance by minimizing head motion and access time. This option is most effective when there is only one slice per disk, when only one process at a time is using the slice/file system, and when I/O patterns are highly sequential or when all accesses are read.

  To change mirror options, refer to “How to Change a Mirror’s Options (DiskSuite Tool)” on page 164.
- **Mounting mirrors –** Only mount the mirror device directly. Do not try and mount a submirror directly, unless it is offline and mounted read-only. Do not mount a slice that is part of a submirror; this could destroy data and crash the system.
- **Mirroring** `swap` **–** Use `swap -l` to check for all `swap` devices. Slices specified as `swap` must be mirrored separately.

RAID5 Metadevice Guidelines

- **Follow the 20-percent write rule –** Because of the complexity of parity calculations, metadevices with greater than about 20 percent writes should probably not be RAID5 metadevice devices. If data redundancy is needed, consider mirroring.
- **Drawbacks to a “slice-heavy” RAID5 metadevice –** The more slices a RAID5 metadevice contains, the longer read and write operations will take when a component fails.

- RAID5 metadevices cannot be mirrored.
- **RAID5 metadevices and striping guidelines –** Striping guidelines also apply to RAID5 metadevice configurations. Refer to “Striping Guidelines” on page 264.
- **Use different controllers –** When creating RAID5 devices, use slices across separate controllers, because controllers and associated cables tends to fail more often than disks. This practice also improves mirror performance.
- **Use same-size slices –** Use the same size disk slices. Creating a RAID5 metadevice of different size slices results in unused disk space.
- **Interlace value –** It is configurable at the time the metadevice is created; thereafter, the value cannot be modified. The default interlace value is 16 Kbytes. This is reasonable for most applications. If the different slices in the RAID5 metadevice reside on different controllers and the accesses to the metadevice are primarily large sequential accesses, then an interlace value of 32 Kbytes might have better performance.
- **Concatenating to a RAID5 metadevice –** Concatenating a new slice to an existing RAID5 will have an impact on the overall performance of the metadevice because the data on concatenations is sequential; data is not striped across all components. The original slices of the metadevice have data and parity striped across all slices. This striping is lost for the concatenated slice, although the data is still recoverable from errors because the parity is used during the component I/O.

  Concatenated slices also differ in the sense that they do not have parity striped on any of the regions. Thus, the entire contents of the slice are available for data.

  Any performance enhancements for large or sequential writes are lost when slice are concatenated.

UFS Logging Guidelines

- **For logging and master devices –** Place logging devices and master device that belong to the same trans metadevice on separate drives and controllers.
- **For trans metadevices and shared logging devices –** Trans metadevices can share metadevice logging devices. But file systems with the heaviest loads should have separate logs.
- **For small file systems –** Small file systems with mostly read operations probably do not need to be logged.
- **For mirroring logging devices –** Mirror all logging devices whenever possible. Losing the data in a log because of device errors can leave a file system in an inconsistent state which `fsck(1M)` may not be able to repair without user intervention.
- Larger logging devices result in greater concurrency.

Hot Spare Guidelines

- **For hot spares as temporary fixes –** Hot spares are not designed to remain a permanent part of your configuration. They need to be replaced with repaired or new slices.
- **For hot spares and state database replicas –** Hot spares cannot contain state database replicas.
- **For cross-controller assigning –** Ideally, slices added to the hot spare pool should be attached to different controllers. This ensures availability of data due to controller error or failure.
- **For wrong-size hot spares –** Do not associate hot spares of the wrong size with submirrors or RAID5 metadevices.
- **For hot spares marked In Use –** Make sure that all hot spares within a hot spare pool are not marked *In-Use*.
- **For one-way mirrors and hot spares –** Do not assign a hot spare pool to a submirror in a one-way mirror.
- **Hot spares are used on a first-fit basis –** When adding hot spares of different sizes to a hot spare pool, add the smaller slices first.

File System Guidelines

- Do not mount file systems on a metadevice's underlying slice. If a slice is used for a metadevice of any kind, you must not mount that slice as a file system. If possible, unmount any physical device you intend to use as a metadevice before you activate it. For example, if you create a trans metadevice for a UFS, in the `/etc/vfstab` file, you would specify the trans metadevice name as the device to mount and `fsck`.

Labeled Partitions

- All physical devices must have a disk label, normally created by programs such as `install`, `format`, or `fmthard`. The label can appear on more than one of the logical partitions that are defined in the label. Physical partitions that contain a label should not allow a user to write to the block that contains the label. Normally, this is block 0. UNIX device drivers allow a user to overwrite this label.

Security Considerations

- DiskSuite does not provide an audit trail for any reconfiguration of metadevices that may be performed on the system. This means that DiskSuite does not support C2 security.

Compatibility

- Systems running Solstice DiskSuite 4.2 must be running Solaris 2.4, Solaris 2.5, Solaris 2.5.1, Solaris 2.6, or Solaris 7.
- UFS logging and disksets require you to run Solaris 2.4, Solaris 2.5, Solaris 2.5.1, Solaris 2.6, or Solaris 7.
- DiskSuite is compatible with the Solstice Backup™ 4.2 product.
- Disksets are not supported on x86 systems.

Working With Disk Drives

If you need to repartition a disk drive, for example, after a disk replacement, you can create a script using the `fmthard(1M)` command to quickly recreate the VTOC (Volume Table of Contents) information on the disk.

▼ How to Use `fmthard(1M)`

1. **Use the `prtvtoc(1M)` command to get a listing of partitioning information for a disk.**

```
# prtvtoc /dev/rdsk/c2t0d0s0 > /tmp/vtoc
```

   In this example, the information for disk `c2t0d0` is redirected to a file on disk.

2. **Create and run a script similar to the following, making use of the `fmthard(1M)` command.**

```
for i in 1 2 3 5
do
fmthard -s /tmp/vtoc /dev/rdsk/c2t${i}d0s2
done
```

Trans Metadevices (UFS Logging) and Disk Quotas

You can set up quotas to limit the amount of disk space and number of inodes (roughly equivalent to the number of files) available to users. (This is a feature of Solaris, not of DiskSuite.) These quotas are activated automatically each time a file system is mounted.

- File systems set up for quotas are faster to check by using `quotacheck` if they are also setup for logging. Such a setup can decrease the amount of time `quotacheck` needs to run.

To create a trans metadevice, refer to Chapter 2. For more information on quotas, refer to *System Administration Guide, Volume II*.

Using DiskSuite Tool

This section describes some advanced uses (and limitations) of DiskSuite Tool.

Limitations

- **Color Mappings –** DiskSuite Tool cannot save color mappings in the Disk View window when exiting the application. The color mapping is in effect during that particular usage of DiskSuite Tool. It cannot be saved.
- **Logical Names for Metadevices –** DiskSuite Tool currently does not have a mechanism to assign logical names, such as table1, log1, and so on, to metadevices.
- **Slice Browser "Use" Column –** The "Use" column in the Slice Browser does not change from "unassigned" when a slice is used as a raw device. All raw devices, not just metadevices used as the raw device, currently share this problem. There is no way to register devices for use other than as file systems or as `swap`. DiskSuite Tool does not have a way of its own for this purpose.

Using the Metadevice Editor

Here are three tips to help manage screen real estate on the Metadevice Editor's canvas:

- Selecting Collapse from an object's pop-up menu enables you to fit more objects onto the canvas.
- Selecting Clean Up Canvas from the Edit menu is useful when you have lots of objects on the canvas, and you are putting some of them away, or you are repositioning objects with the mouse. The Clean Up Canvas option rearranges objects on the canvas in a grid, making viewing easier.
- Use the sash to resize the canvas. You can widen the canvas area by clicking the sash at the bottom of the Metadevice Editor window and dragging to the right.

Using the Slice View, Disk View, and Filters

Setting filters within DiskSuite Tool on the Slice View and Disk View windows can help you quickly locate suitable slices for the task at hand.

▼ How to Filter for Slice Size (DiskSuite Tool)

If you have a system with many disks (and slices), searching for available slices of a certain size can be a chore. Using the Slice Filter window can save you time in this activity.

This task describes how to create a filter in the Slice View window for available slices larger than 200 MBytes, then drag and drop these slices to the Disk View window to see where they are located.

1. **Click Slices to display the Slice View window.**

   The Slice View window appears.

2. **Select Set Filters from the Filters menu in the Slice View window.**

   The Slice Filters window appears.

3. **To search for available slices, make sure the "Available for use as" radio button as is checked, and that "Anything" is selected in the pull-down.**

4. **To filter for slices greater than 200 Mbytes, check the "Size" radio button, select "greater than" in the first pull-down, type 200 in the text box, and select Mbytes in the second pull-down.**

5. **Click Apply and view the results in the Slice Browser window.**

   If necessary, change values in the Slice Filters window and click Apply to change the filtering scheme.

6. **After adjusting the filtering scheme to your satisfaction, close the Slice Filters window by clicking OK.**

7. **Click Disk View to display the Disk View window.**

   The Disk View window appears.

8. **In the Slice Browser window, click Select All. Then drag the selected slices to a color drop site in the Disk View window.**

9. **View the results in the Disk View window.**

   DiskSuite Tool uses the selected drop site's color for all slices dragged to the Disk View window. You can now make your slice selection (for example to create a submirror) following the considerations outlined in "General Guidelines" on page 263.

▼ How to Filter for Slice Replacement (DiskSuite Tool)

This task shows how to use DiskSuite Tool to find a suitably sized replacement slice for an errored slice in a submirror.

Note - This approach is not limited to mirrors. You can use this task to find replacement slices for any type of metadevice.

---

1. **Click Disk View to display the Disk View window.**

   The Disk View window appears.

2. **Drag the errored Mirror object from the Objects list to the canvas.**

3. **Select one submirror (a Concat/Stripe object) within the mirror and drag it to the Disk View window. Then do the same for the second submirror (and third submirror if this is a three-way mirror).**

   The Disk View window colors the slices with a different color corresponding to the submirrors in the Mirror object. This helps you see where the slices are located, for example, across controllers.

4. **Click Slices to display the Slice View window.**

   The Slice View window appears.

5. **Click Set Filters in the Slice View window.**

   The Slice Filters window appears.

6. **To search for available slices, make sure the "Available for use as" radio button is checked, and that "Metadevice Component" is selected in the pull-down.**

7. **Filter for slices to replace the errored slice.**

   One way to do this is to set up a filter that finds slices greater than a size that is slightly smaller than the errored slice. This will display a larger range of slices than if you set up a filter that searches for slices equal to the errored slice size.

8. **Check the "Size" radio button, select "greater than" in the first pull-down, type the size of the slice (in Mbytes, and slightly smaller than the errored slice's size) in the text box, and select Mbytes in the second pull-down.**

9. **Click Apply and view the results in the Slice Browser window.**

   If necessary, change values in the Slice Filters window and click Apply to change the filtering scheme.

10. **In the Slice Browser window, click Select All. Then drag the selected slices to a color drop site in the Disk View window.**

11. **View the results in the Disk View window.**

    DiskSuite Tool uses the color for all slices dragged to the Disk View window.

12. **Select a replacement slice.**

    You can now make your slice selection for a DiskSuite object following the guidelines outlined in "General Guidelines" on page 263. Pick a replacement that is large enough and follows mirror guidelines (on different controller, or at least a different disk.)

13. **Drag the replacement slice from the Disk View window to the rectangle of the Concat/Stripe object with the errored slice.**

14. **Commit the mirror.**

    Click inside the top of the Mirror object then click Commit. A mirror resync begins.

Changing DiskSuite Tool's Colors and Fonts

By default, DiskSuite Tool uses colors and fonts that are compatible with the OpenWindows™ desktop applications. This section describes how to change these colors and fonts.

DiskSuite Tool and Colors

DiskSuite Tool uses a variety of colors:

- **Standard foreground color –** Determines the main colors used to display almost all of the application's elements. The standard background provides the default color for windows, buttons, and other controls.
- **Standard background color –** Provides the color value used to display information presented in windows, buttons, and other controls.
- **Canvas background color –** Provides the background color for data areas. For example, the display areas for the Editor, Disk View window, and scrolling lists all use the canvas background color.
- **Mapping colors –** Display the mappings from logical devices to their slices in the Disk View window. There are eight mapping colors, one for each of the DiskView mappings.
- **Status colors –** Highlight status information for objects needing attention. There are three status conditions requiring unique colors: Attention, Urgent and Critical.

The X Window System RGB (Red, Green, Blue) color specification mechanism enables you to specify a nearly infinite variety of colors. Of course, many of these colors will appear similar, varying only slightly in shade or intensity.

To aid in selecting and specifying colors, the X Window System provides a standard default set of colors that you can specify by name instead of RGB values. This "database" of color names can be examined using the standard X utility `showrgb`. It shows the RGB values and a corresponding descriptive alias. For example:

```
# showrgb
  199  21 133 medium violet red
  176 196 222 light steel blue
  102 139 139 paleturquoise4
  159 121 238 mediumpurple2
  141 182 205 lightskyblue3
    0 238 118 springgreen2
  255 160 122 light salmon
  154 205  50 yellowgreen
  178  58 238 darkorchid2
   69 139 116 aquamarine4
...
   107 107 107 gray42
    71  71  71 gray28
    61  61  61 gray24
   255 255 255 white
     0 205 205 cyan3
     0   0   0 black
```

You can also examine the default color name database by looking at the `/usr/openwin/lib/X11/rgb.txt` file.

Unfortunately, there are no standard applications for browsing colors. If you don't have access to a public domain color browser, experiment by trial and error.

DiskSuite Tool's Default Colors

DiskSuite Tool's default colors are shown in Table 8–1.

TABLE 8–1 DiskSuite Tool's Default Colors

Color Type	Color
Standard Foreground	black
Standard Background	gray
Canvas Background	gray66
Mapping Colors:	
mappingColor1	blue
mappingColor2	green
mappingColor3	magenta
mappingColor4	cyan
mappingColor5	purple
mappingColor6	mediumseagreen
mappingColor7	firebrick
mappingColor8	tan
mappingColor9	white
Status Colors:	
Critical	red
Urgent	orange
Attention	yellow

DiskSuite Tool and Fonts

DiskSuite Tool uses four different fonts:

- **Standard font –** Displays almost all text in the tool, for example, in button labels, menus, and dialog boxes.
- **Mono-spaced (fixed-width) font –** Enables consistent columnar alignment, for example, in the various browsers and scrolling lists. This needs to be specified several times.
- **Bold font –** Distinguishes attribute names and labels from the actual attribute values. The names/labels in Information windows appear in the standard font and the corresponding values appear in the bold font. This font is used sparingly.
- **Small font –** Shows the physical devices at the 50 percent scaling level in the Disk View window.

Available Fonts in DiskSuite

The available fonts depend on which X Window System server you use to display the application. The standard X utility, `xlsfonts`, displays the available fonts on a server. For example:

```
# xlsfonts
--courier-bold-o-normal--0-0-0-0-m-0-iso8859-1
--courier-bold-r-normal--0-0-0-0-m-0-iso8859-1
--courier-medium-o-normal--0-0-0-0-m-0-iso8859-1
--courier-medium-r-normal--0-0-0-0-m-0-iso8859-1
--symbol-medium-r-normal--0-0-0-0-p-0--symbol
-symbol-medium-r-normal--0-0-0-0-p-0-sun-fontspecific
-adobe-courier-bold-i-normal--0-0-0-0-m-0-iso8859-1
...
utopia-bolditalic
utopia-italic
utopia-regular
variable
vshd
vtbold
vtsingle
zapfchancery-mediumitalic
zapfdingbats
```

Another helpful utility for displaying available fonts is `xftonsel`. Refer to the man pages for these utilities for more information.

DiskSuite Tool's Default Fonts

DiskSuite Tool's default fonts all come from the Lucida font family:

TABLE 8–2 DiskSuite Tool's Default Fonts

Font Type	Font
Standard Font	lucidasans12
Mono-spaced Font	lucidasans-typewriter12
Bold Font	lucidasans-bold12
Small Font	lucidasans8

DiskSuite Tool uses the X Window System's resource database mechanism to determine which fonts to use. The default resource specifications are:

TABLE 8–3 DiskSuite Tool's Default Font Resource Specifications

Resource	Font
Metatool*fontList:	lucidasans12
Metatool*smallFontList:	lucidasans8
Metatool*boldFontList:	lucidasans-bold12
Metatool*fixedFontList:	lucidasans-typewriter12
Metatool*XmList.fontList:	lucidasans-typewriter12
Metatool*Help*helpsubjs.fontlist:	lucidasans-typewriter12
Metatool*Help*helptext.fontlist:	lucidasans-typewriter12

▼ How to Change DiskSuite Tool's Default Colors and Fonts

You can change DiskSuite Tool's default colors and fonts by using one of the following four methods.

- For one invocation of DiskSuite Tool, use the `xrm` utility to specify the alternate font or color resources.

```
# metatool -xrm 'resource'
```

- For all of your invocations of DiskSuite Tool, edit your own `.Xdefaults` file and specify the alternate color or font resources. The `.Xdefaults` file is typically loaded when you start your desktop session. After editing this file, the next time you start your desktop session, the new or changed resources will be used.
- For the current session, without having to restart, use the `xrdb` utility.

```
# xrdb -merge path_to_.Xdefaults
```

- For all users of DiskSuite Tool, edit the `/usr/opt/SUNWmd/lib/x11/app-defaults/Metatool` file. Changes made to this file are recognized the next time DiskSuite Tool is started.

Example — Changing Fonts

This example changes the standard font to lucidasans16 for a single invocation of DiskSuite Tool.

```
# metatool -xrm 'Metatool*fontList: lucidasans16'
```

Metadevice Naming Conventions

Using a naming convention for your metadevices can help with your DiskSuite administration, and enable you at a glance to easily identify the metadevice type. Here are a few suggestions:

- Use ranges for each particular type of metadevice. For example, assign numbers 0-20 for mirrors, 21-40 for stripes and concatenations, and so on.
- Use a naming relationship for mirrors. For example, name mirrors with a number ending in zero (0), and submirrors ending in one (1) and two (2). For example: mirror-`d10`, submirrors `d11` and `d12`; mirror-`d20`, submirrors `d21` and `d22`, and so on.
- Use a naming method that maps the slice number and disk number to metadevice numbers.

Note - The `metarename` command enables you to reorganize your metadevice names. Refer to the `metarename(1M)` man page for more information.

Metadevice Name Switching

In addition to renaming metadevices, DiskSuite's `metarename` command also provides the ability to switch "layered" metadevices. When used with the `-x` option, `metarename` switches (exchanges) the names of an existing layered metadevice and one of its subdevices. This includes a mirror and one of its submirrors, or a trans metadevice and its master device.

Note - You must use the command line to exchange metadevices. This functionality is currently unavailable in DiskSuite Tool, although you can rename a metadevice with either the command line or DiskSuite Tool.

- **When to use metadevice name switching –** The `metarename -x` command can make it easier to mirror or unmirror an existing stripe or concatenation, and to create or remove a trans metadevice of an existing metadevice.
- **Advantages of using metadevice name switching –** Switching metadevice names is an administrative convenience for management of metadevice names. For example, you could arrange all file system mount points in a desired numeric range.
- **Combinations of metadevices that can be switched –** The `metarename -x` command can be used to switch:

  - Mirror and submirror (concatenation or stripe)
  - Trans metadevice and master device, where the master device is a concatenation, stripe, mirror, or RAID5 metadevice

  The metadevice name switch can take place in both directions.
- **Trans metadevice that has a mirrored master device –** If the master device is a mirror, you cannot directly switch one of the mirror's submirrors for the trans metadevice. You can switch the mirror and the trans metadevice names, or the mirror and one of its submirrors. The relationship for the switch is always parent-child. In essence, you could achieve the same outcome of a submirror/trans metadevice exchange by using a two-step process: first, switch the submirror for the mirror; then switch the mirror for the trans metadevice.
- **"Rename busy" message when trying to switch components of a trans metadevice –** This message could mean one or more of the following: you did not first detach the logging device; you did not unmount the file system using the trans metadevice; you did not use the `-f` (force) flag option of the `metarename` command.

Prerequisites for Using Metadevice Name Switching

- You cannot switch (or rename) a metadevice that is currently in use. This includes metadevices used as mounted file systems, as `swap`, or as active storage for applications or databases. Thus, before using the `metarename` command, stop all access to the metadevice being renamed. For example, unmount a mounted file system using a metadevice. An application or database must have its own internal method for stopping access.
- You cannot switch metadevices in an errored state, or metadevices using a hot spare replacement.
- A switch can only take place between metadevices with a direct parent/child relationship. You could not, for example, directly exchange a stripe in a mirror that is a master device with the trans metadevice.
- You must use the `-f` (force) flag when switching members of a trans device.
- You cannot switch (or rename) a logging device. The workaround is to either detach the logging device, rename it, then reattach it to the trans device; or detach the logging device and attach another logging device of the desired name.
- Only metadevices can be switched. You cannot switch slices or hot spares.

Creating a Metadevice Using Name Switching

If you have an existing stripe, you can use the `metarename -x` command to create a compound metadevice. This includes creating a mirror from a concat/stripe, or a trans device with a metadevice as the master device.

▼ How to Create a Mirror From an Existing Concat/Stripe (Command Line)

This example begins with a concatenation, `d1`, with a mounted file system, and ends up with the file system mounted on a two-way mirror named `d1`.

```
# metastat d1
d1: Concat/Stripe
    Size: 5600 blocks
    Stripe 0:
        Device              Start Block  Dbase
        c0t0d0s1                  0     No
# metainit d2 1 1 c1t3d0s1
d2: Concat/Stripe is setup
# metainit -f d20 -m d1
d20: Mirror is setup
# umount /fs2
```

(continued)

(Continuation)

```
# metarename -x d20 d1
d20 and d1 have exchanged identities
# metastat d1
d1: Mirror
    Submirror 0: d20
      State: Okay
...

d20: Submirror of d1
    State: Okay
...
# metattach d1 d2
d1: submirror d2 is attached
# metastat d1
d1: Mirror
    Submirror 0: d20
      State: Okay
    Submirror 1: d2
      State: Okay
...
# mount /fs2
```

The `metastat` command confirms that the concatenation `d1` is in the “Okay” state. You use the `metainit` command to create a second concatenation (`d2`), and then to force (`-f`) the creation of mirror `d20` from `d1`. You must unmount the file system before using `metarename -x` to switch `d20` for `d1`; `d1` becomes the top-level device (the mirror), which `metastat` confirms. You attach `d2` as the second submirror, verify the state of the mirror with `metastat`, then remount the file system. Note that because the mount device for `/fs2` did not change, you do not have to edit the `/etc/vfstab` file.

▼ How to Create a Trans Metadevice From an Existing Metadevice (Command Line)

This example begins with a mirror, `d1`, with a mounted file system, and ends up with the file system mounted on a trans device named `d1`.

```
# metastat d1
d1: Mirror
    Submirror 0: d20
      State: Okay
    Submirror 1: d2
      State: Okay
...
# umount /fs2
```

(continued)

```
# metainit d21 -t d1
d21: Trans is setup
# metarename -f -x d21 d1
d21 and d1 have exchanged identities
# metastat d1
d1: Trans
    State: Detached
    Size: 5600 blocks
    Master Device: d21
...
# metattach d1 d0
d1: logging device d0 is attached
# mount /fs2
```

The `metastat` command confirms that the mirror `d1` is in the “Okay” state. You must unmount the file system before using the `metainit` command to create the trans device `d21`, with `d1` as the master. The `metarename -f -x` command forces the switch of `d21` and `d1`; `d1` is now the top-level trans metadevice, as confirmed by the `metastat` command. A logging device `d0` is attached with the `metattach` command. You then remount `/fs2`. Note that because the mount device for `/fs2` has not changed (it is still `d1`), you do not have to edit the `/etc/vfstab` file.

Removing a Metadevice Using Name Switching

If you have an existing mirror or trans metadevice, you can use the `metarename -x` command to remove the mirror or trans metadevice and keep data on an underlying metadevice. For a trans metadevice, as long as the master device is a metadevice (stripe/concat, mirror, or RAID5 metadevice), you keep data on that metadevice.

When you use `metarename -x` as part of this process, the mount point of the file system remains the same.

▼ How to Unmirror a File System and Retain the Mount Device (Command Line)

This example begins with a mirror, `d1`, containing a mounted file system, and ends up with the file system mounted on a stripe named `d1`.

```
# metastat d1
d1: Mirror
    Submirror 0: d20
      State: Okay
    Submirror 1: d2
```

(Continuation)

```
      State: Okay
    Pass: 1
...
# umount /fs2
# metarename -x d1 d20
d1 and d20 have exchanged identities
# metastat d20
d20: Mirror
    Submirror 0: d1
      State: Okay
    Submirror 1: d2
      State: Okay
...

# metadetach d20 d1
d20: submirror d1 is detached
# metaclear -r d20
d20: Mirror is cleared
d2: Concat/Stripe is cleared
# mount /fs2
```

The `metastat` command confirms that mirror `d1` is in the “Okay” state. This file system is unmounted before exchanging the mirror `d1` and its submirror `d20`. This makes the mirror `d20`, as confirmed by `metastat`. Next, `d1` is detached from `d20`, then mirror `d20` and the other submirror, `d2` are deleted. Finally, `/fs2` is remounted. Note that because the mount device for `/fs2` did not change, the `/etc/vfstab` file does not require editing.

▼ How to Remove a Trans Metadevice and Retain the Mount Device (Command Line)

This example begins with a trans metadevice, `d1`, containing a mounted file system, and ends up with the file system mounted on the trans metadevice‘s underlying master device, which will be `d1`.

```
# metastat d1
d1: Trans
    State: Okay
    Size: 5600 blocks
    Master Device: d21
    Logging Device: d0

d21: Mirror
    Submirror 0: d20
      State: Okay
```

(continued)

```
    Submirror 1: d2
      State: Okay
...

d0: Logging device for d1
    State: Okay
    Size: 5350 blocks
# umount /fs2
# metadetach d1
d1: logging device d0 is detached
# metarename -f -x d1 d21
d1 and d21 have exchanged identities
# metastat d21
d21: Trans
    State: Detached
    Size: 5600 blocks
    Master Device: d1

d1: Mirror
    Submirror 0: d20
      State: Okay
    Submirror 1: d2
      State: Okay
# metaclear 21
# fsck /dev/md/dsk/d1
** /dev/md/dsk/d1
** Last Mounted on /fs2
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups

FILE SYSTEM STATE IN SUPERBLOCK IS WRONG; FIX? y

3 files, 10 used, 2493 free (13 frags, 310 blocks, 0.5%
fragmentation)
# mount /fs2
```

The `metastat` command confirms that the trans metadevice, `d1`, is in the “Okay” state. The file system is unmounted before detaching the trans metadevice’s logging device. The trans metadevice and its mirrored master device are exchanged using the `–f` (force) flag. Running `metastat` again confirms that the exchange occurred. The trans metadevice and the logging device (if desired) are cleared, in this case, `d21` and `d0`, respectively. Next, the the `fsck` command is run on the mirror, `d1`, and the prompt is answered with a `y`. After the `fsck` command is done, the file system is remounted. Note that because the mount device for `/fs2` did not change, the `/etc/vfstab` file does not require editing.

Working With Stripes

This section describes a technique for regaining access to a metadevice that is defined on a failing controller, causing sporadic system panics. If there is another available controller on the system, the metadevice can in effect be “moved” to the new controller by moving the disks to the controller and redefining the metadevice. This technique does away with the need to back up and restore data to the metadevice.

▼ How to Move a Stripe to a Different Controller (Command Line)

This example consists of a disk that has two slices that are each part of two separate striped metadevices, `d100` and `d101`, containing file systems `/user6` and `/maplib1`, respectively. The affected controller was `c5`; the disks will be moved to a free controller (`c4`). This example also uses the `md.tab` file.

1. **Stop access to the affected stripes.**

   For example, unmount any file systems associated with the striped metadevice.

```
# umount /user6
# umount /maplib1
```

2. **Use `metaclear` to clear the striped metadevice.**

```
# metaclear d100
d100: Concat/Stripe is cleared
# metaclear d101
d101: Concat/Stripe is cleared
```

3. **Shut down the server and move the disks to the new controller.**

4. **Edit the `md.tab` file to indicate the new controller in the metadevice names. This example uses “c4” not “c5” for the disk, because the disk was moved to controller 4.**

```
Lines from the md.tab file before change:
# Stripe /user6
/dev/md/dsk/d100 1 2 /dev/dsk/c5t0d0s3 /dev/dsk/c2t2d0s3
# Stripe /maplib1
/dev/md/dsk/d101 1 2 /dev/dsk/c5t0d0s0 /dev/dsk/c2t2d0s0

after change:
# Stripe /user6
/dev/md/dsk/d100 1 2 /dev/dsk/c4t0d0s3 /dev/dsk/c2t2d0s3
# Stripe /maplib1
/dev/md/dsk/d101 1 2 /dev/dsk/c4t0d0s0 /dev/dsk/c2t2d0s0
```

5. **Use** `metainit` **to initialize that striped metadevices and mount it, without reinitializing the file system with** `newfs`**.**

```
# metainit d100
d100: Concat/Stripe is setup
# metainit d101
d101: Concat/Stripe is setup
```

6. **Run** `mountall` **to remount the file systems.**

7. **Run** `metastat` **to verify that the metadevices are online.**

Caution - Don't run the `newfs` command on the metadevice or its associated file system. It will result in a massive data loss, and the need to restore from tape.

Working With Mirrors

This section describes some tips regarding mirrors and their operation.

Advanced Mirror Techniques

The following two tasks show how to change the interlace value of submirrors without destroying a mirror, and how to use a mirror for an online backup.

▼ How to Change the Interlace Value of Stripes in Mirrors (DiskSuite Tool)

Use this task to change the interlace value of a mirror's underlying submirrors which are composed of striped metadevices. Using this method does away with the need to recreate the mirror and submirrors and restore data.

Note - To use the command line to perform this task, refer to the `metadetach(1M)`, `metainit(1M)`, and `metattach(1M)`man pages.

The high-level overview of the steps in this task are:

- Detaching submirror1
- Clearing submirror1
- Creating a new stripe, with the new interlace value, to be used as submirror1
- Attaching submirror1 to the mirror
- Waiting for the mirror resync to finish
- Repeating the above steps for submirror2

1. **Make sure DiskSuite Tool is started.**
2. **Double-click the Mirror object in the Objects list.**
   The object appears on the canvas.
3. **Click inside the submirror to be detached.**
4. **Drag the submirror out of the Mirror object to the canvas.**
   If this is a two-way mirror, the mirror's status changes to "Urgent."
5. **Click the top rectangle of the Mirror object then click Commit.**
6. **Create a new submirror with the desired interlace value.**
   Refer to "How to Create a Striped Metadevice (DiskSuite Tool)" on page 21.
7. **Drag the new Submirror object to the Mirror object. Then click Commit to commit the mirror.**
   A mirror resync begins.

8. **The Configuration Log shows that the mirror was committed.**

9. **Repeat Step 3 on page 286 through Step 7 on page 286 for the second (and possibly third) submirror in the mirror.**

▼ How to Use a Mirror to Make an Online Backup (Command Line)

Although DiskSuite is not meant to be a "backup product," it does provide a means for backing up mirrored data without unmounting the mirror or taking the entire mirror offline, and without halting the system or denying users access to data. This happens as follows: one of the submirrors is taken offline—temporarily losing the mirroring—and backed up; that submirror is then placed online and resynced as soon as the backup is complete.

You can use this procedure on any file system except root (/). Be aware that this type of backup creates a "snapshot" of an active file system. Depending on how the file system is being used when it is write-locked, some files and file content on the backup may not correspond to the actual files on disk.

Limitations

- If you use this procedure on a two-way mirror, be aware that data redundancy is lost while one submirror is offline for backup. A three-way mirror does not have this problem.
- There is some overhead on the system when the offlined submirror is brought back online after the backup is complete.

Note - If you use these procedures regularly, put them into a script for ease of use.

The high-level steps in this procedure are:

- Write locking the file system (UFS only). Do not lock root (/).
- Using the `metaoffline(1M)` command to take one submirror offline from the mirror
- Unlocking the file system
- Backing up the data on the offlined submirror
- Using the `metaonline(1M)` command to place the offlined submirror back online

1. **Before beginning, run the `metastat(1M)` command to make sure the mirror is in the "Okay" state.**

   A mirror that is in the "Maintenance" state should be repaired first.

2. **For all file systems except root (/), lock the file system from writes.**

```
# /usr/sbin/lockfs -w mount point
```

Only a UFS needs to be write-locked. If the metadevice is set up as a raw device for database management software or some other specific application, running `lockfs(1M)` is not necessary. (You may, however, want to run the appropriate vendor-supplied utility to flush any buffers and lock access.)

Caution - Write-locking root (/) causes the system to hang, so it should never be performed.

3. **Take one submirror offline from the mirror.**

```
# metaoffline mirror submirror
```

In this command,

mirror	Is the metadevice name of the mirror.
submirror	Is the metadevice name of the submirror (metadevice) being taken offline.

Reads will continue to be made from the other submirror. The mirror will be out of sync as soon as the first write is made. This inconsistency is corrected when the offlined submirror is brought back online in Step 6 on page 289.

There is no need to run `fsck(1M)` on the offlined file system.

4. **Unlock the file system and allow writes to continue.**

```
# /usr/sbin/lockfs -u mount point
```

You may need to perform necessary unlocking procedures based on vendor-dependent utilities used in Step 2 on page 288 above.

5. **Perform a backup of the offlined submirror. Use `ufsdump(1M)` or your usual backup utility.**

Note - To ensure a proper backup, use the *raw* metadevice, for example, `/dev/md/rdsk/d4`. Using "`rdsk`" allows greater than 2 Gbyte access.

6. Place the mirror back online.

```
# metaonline mirror submirror
```

DiskSuite automatically begins resyncing the submirror with the mirror.

Example — Using a Mirror to Make an Online Backup

This example uses a mirror named `d1`, consisting of submirrors `d2` and `d3`. `d3` is taken offline and backed up while `d2` stays online. The file system on the mirror is `/home1`.

```
# /usr/sbin/lockfs -w /home1
# metaoffline d1 d3
d1: submirror d3 is offlined
# /usr/sbin/lockfs -u /home1
(Perform backup using /dev/md/rdsk/d3)
# metaonline d1 d3
d1: submirror d3 is onlined
```

How Booting Into Single-User Mode Affects Mirrors

If a system with mirrors for root (`/`), `/usr`, and `swap`—the so-called “boot” file systems—is booted into single-user mode (`boot -s`), these mirrors and possibly all mirrors on the system will appear in the “Needing Maintenance” state when viewed with the `metastat` command. Furthermore, if writes occur to these slices, `metastat` shows an increase in dirty regions on the mirrors.

Though this appears potentially dangerous, there is no need for concern. The `metasync -r` command, which normally occurs during boot to resync mirrors, is interrupted when the system is booted into single-user mode. Once the system is rebooted, `metasync -r` will run and resync all mirrors.

If this is a concern, run `metasync -r` manually.

Hot Spares

A hot spare pool can contain 0 to *n* hot spares. A hot spare pool can be associated with multiple submirrors and RAID5 metadevices. You can define one hot spare pool with a variety of different size slices, and associate it with all the submirrors or RAID

5 metadevices. DiskSuite knows how to use the correctly sized hot spare when necessary.

Place hot spares in the same hot spare pool across controllers, to preserve lines of failure. In this respect, follow the same guidelines as you would for creating submirrors.

Working With Disksets

This section provides tips for configuring disksets.

Note - Currently, DiskSuite only supports disksets on SPARCstorage Array disks.

▼ How to Configure Disk Drive Device Names for a Diskset (Command Line)

Configuring the hardware for use in diskset configuration can be problematic. The disk drives must be symmetric; that is, the shared drives must have the same device number, which implies the same device name and number (controller / target / drive). This task explains how to configure this setup.

Note - On a set of new machines, where the hardware was pre-configured, the desired symmetry occurs by default. You do not need to perform this task.

You must configure device names done before creating any metadevices in the diskset. Any other drives that are on non built-in controllers will also be affected.

1. **Make sure that the disk controllers are located in slots that will be found in the same order.**

   This is best achieved by having the controllers for a given SPARCstorage Array in the same slots on identical processor models. If this is not possible, then you must make sure that the order of the slots will probe out identically on both processors. Because the probing of the Sbus is conducted in an orderly fashion, this can be achieved, but not easily. It is also recommended that slots be used in order from lowest to highest numbered slot leaving all the unused slots at the high end.

   Note - The configuration system numbers controllers of the same type in sequence. In this case, “disk drive” is the type, so all controllers for disk drives will affect the order that devices are found. To this end, all the devices that are to be shared should probably be placed before any other disk controllers in the system to make sure that they will be found and accounted for in the correct order.

Once this has been done, you can do one of two things: a complete install on both host machines, or continue with this task. The latter is considerably faster.

2. One at a time, become root on each host and perform the following:

```
# rm /etc/path_to_inst*
# reboot -- '-rav'
reboot: rebooted by root
syncing file systems... [1] done
rebooting...
Resetting ...

Rebooting with command: -rav
Boot device: /iommu/sbus/espdma@f,400000/esp@f,800000/sd@3,0
File and args: -rav
Enter filename [kernel/unix]:
Size: 253976+126566+39566 Bytes
Enter default directory for modules [/kernel /usr/kernel]:
SunOS Release 5.4 Generic [UNIX(R) System V Release 4.0]
Copyright (c) 1983-1995, Sun Microsystems, Inc.
Name of system file [etc/system]:
The /etc/path_to_inst on your system does not exist or is empty.
Do you want to rebuild this file [n]? y
Using default device instance data
root filesystem type [ufs]:
Enter physical name of root device
[/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000
/sd@3,0:a]:
...
The system is ready.

console login: root
Password:<root-password>

# /usr/bin/rm -r /dev/*dsk/*
# /usr/sbin/disks
# ^D
```

Given that the hardware is set up correctly, this will ensure that the software reflects that setup. Because the `/etc/path_to_inst` file is used to keep things from sliding, which will occur generally when controllers are moved around, it is removed to make sure that controllers slide to the correct location. The '–`rav`' option with `reboot` makes sure that the kernel will interact with the user during boot and do a reconfigure reboot. The removal of `/dev/*dsk/*` is used to make sure that the sym-links get created correctly when the `/usr/sbin/disks` program is run.

Note - Because the SPARCstorage Array controller contains a unique World Wide Name, which identifies it to Solaris, special procedures apply for SPARCstorage Array controller replacement. Contact your service provider for assistance.

▼ How to Change State Database Replica Size in a Diskset (Command Line)

If you want to change the size of your state database replicas in a diskset, the basic steps are adding two disks to the diskset, deleting one of the new disk's state database replicas, then deleting the other disk from the diskset. You then add the deleted disk back to the diskset, along with any other disks you want added to the diskset. The state database replicas will automatically resize themselves to the new size.

Example — Changing State Database Replica Size in a Diskset

```
# metadb -s rimtic -d c1t0d0s7
# metadb -s rimtic -a -l 2068 c1t0d0s7
# metaset -s rimtic -d c1t1d0
# metaset -s rimtic -a c1t1d0
# metadb -s rimtic
```

This example assumes you have already added two disks to the diskset, `rimtic`, and that there is no data on the rest of the disk to which the replica will be added. The new size of the state database replica is 2068 blocks, as specified by the `–l 2068` option. The `metadb` command confirms the new size of the state database replicas.

APPENDIX **A**

Using Storage Manager

The Storage Manager application contains two tools, Disk Manager and File System Manager, that enable you to manage disk configurations and file systems on servers that are on your network.

This is a list of the step-by-step instructions in this appendix.

Storage Manager's Load Context Property Book

Storage Manager introduces the concept of a *property book*. A property book is the mechanism by which you identify, view, and modify the properties of editable *objects*, such as disks or file systems. A property book, using a book metaphor, contains a list of chapters that represent properties for the object. You can expand each chapter to view or modify the properties.

The purpose of the Load Context Property Book is for you to set the *context* in which Storage Manager will operate. Generally, a context is the way to manage the properties of an object. In the case of File System Manager, the context includes which host to manage, what name service to modify or how to view and modify the file systems on a server. In the case of Disk Manager, the context includes what *diskset* (a logical grouping of disks) to modify if you have DiskSuite software installed.

The Load Context Property Book is displayed when you start Storage Manager from the Solstice Launcher. If you start either Disk Manager or File System Manager from the DiskSuite Tool Tools menu, you can also display this window by choosing Load from the File menu, or by clicking on the Load Context icon in either tool bar. Figure A–1 shows the Storage Manager's Load Context Property Book.

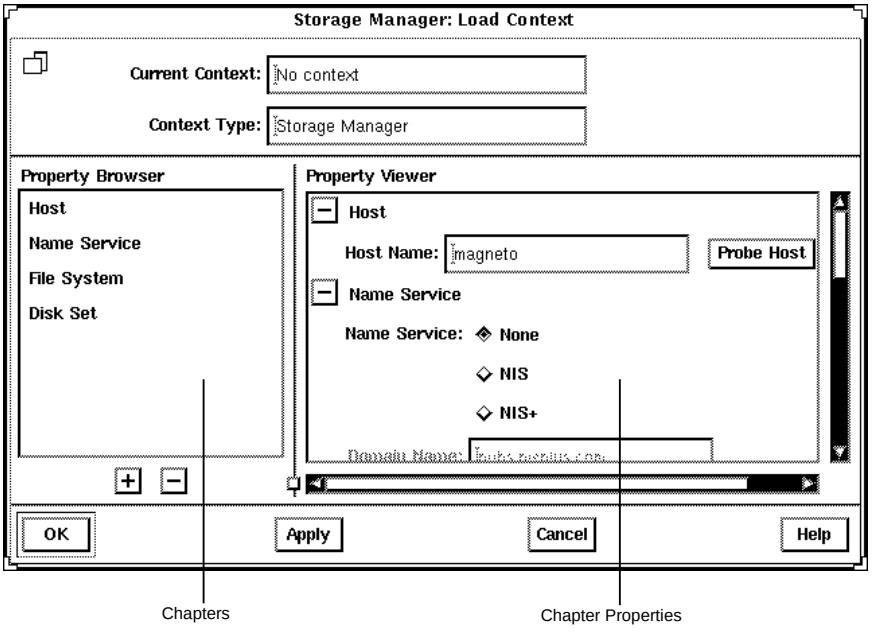


Figure A–1 Storage Manager's Load Context Property Book

Here are some brief descriptions of the areas within Storage Manager's Load Context Property Book:

- **Current Context –** Is a read-only field that displays the host name of the current context. Storage Manager initially has no context set.
- **Context Type –** Is a read-only field that indicates the name of the tool with which the current context is associated. In Figure A–1, the tool is Storage Manager.
- **Property Browser –** Is like a table of contents in a book. It lists all the *chapters* and subchapters (if any) in the book, which represent properties of an object. The Property Browser enables you to navigate quickly to the desired chapter and to expand or collapse its contents as desired.

  Clicking on a chapter in the Property Browser causes the corresponding chapter in the Property Viewer to become visible, if it is not already visible. Double-clicking on a chapter in the Property Browser first makes the chapter visible in the Property Viewer, and then expands or collapses the chapter in the Property Viewer.
- **Chapters –** Contain a common set of properties for the object that you can view or modify.
- **The + and - Buttons –** Appear below the Property Browser. These buttons perform an expand all (+) or collapse all (-) function on the chapters in the Property Viewer. This is a quick way of viewing all chapter properties or only the chapter names.

  The buttons that appear next to chapter names in property books also perform an expand all or collapse all function, but only on the selected chapter.
- **Property Viewer –** Displays the properties you can set when changing your current context. By default, the chapters in this Property Viewer appear expanded, so that you can easily specify the context on which you want to operate. At a minimum, you must probe a server in the Host chapter in order to specify a context. For more information, see "How to Load a Different Context" on page 296.
- **Chapter Properties –** Enables you to set the context you want to display and modify.

For more reference information on the chapters that are available from the Load Context Property Book (Host, Name Service, File System, Disk Set), refer to online help.

▼ How to Load an Initial Context

This procedure assumes that the Load Context window is displayed as a result of clicking on the Storage Manager icon in the Solstice Launcher, or by choosing Load from the File menu in either File System Manager or Disk Manager.

1. **If you want to view or modify the context of the system where you started the Solstice Launcher and Storage Manager, make needed changes to the Name Service, File System, and Disk Set chapters, if any. Skip to Step 5 on page 296.**

2. **If you want to view or modify the context of a system other than the one where you started the Solstice Launcher and Storage Manager, delete the existing name in the Host Name field and type the name of the host whose context you want to view or modify.**

3. **Click Probe Host.**

   A System Discovery window is displayed, indicating that information is being updated from the specified host.

4. **If desired, make changes to the Name Service, File System, and Disk Set chapters.**

5. **Click OK.**

   A System Discovery window is displayed, indicating that Storage Manager is validating context parameters as well as discovering devices, directories, and mount points on the specified host.

▼ How to Load a Different Context

This procedure assumes that Storage Manager has an active current context (that is, the File System Manager main window or Disk Manager main window is open).

1. **Choose Load from the File menu, or click the Load Context icon in the tool bar.**

   The Load Context window is displayed, with the current context's host name displayed in the Host chapter.

2. **Delete the existing name in the Host Name field and type the name of the host whose context you want to view or modify.**

3. **Click Probe Host.**

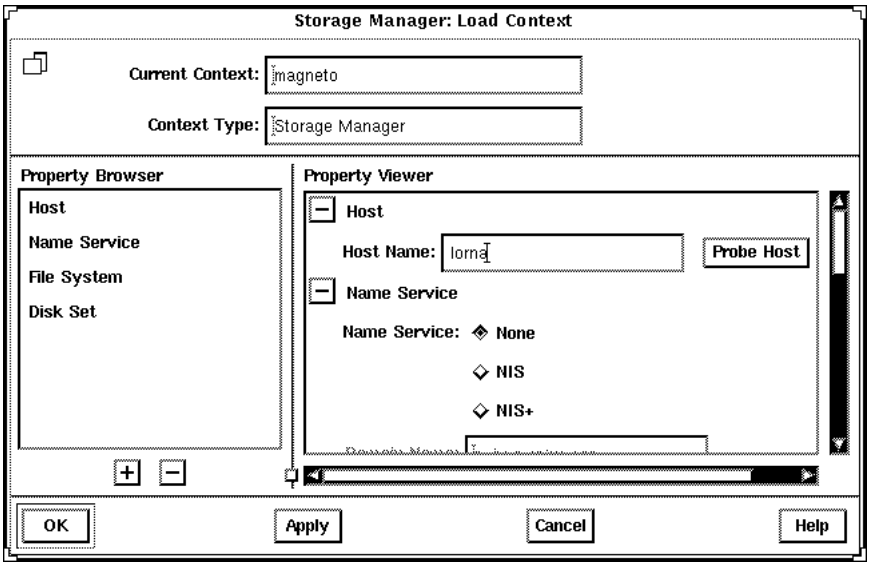
   A System Discovery window is displayed, indicating that information is being updated from the specified host.

4. **If desired, make changes to the Name Service, File System, and Disk Set chapters.**

5. **Click OK.**

   A System Discovery window is displayed, indicating that Storage Manager is validating context parameters as well as discovering devices, directories, and mount points on the specified host.

Example — Loading a Different Context



File System Manager Overview

File System Manager is a tool that enables you to create and modify file systems, mount points, and directories using two types of windows, the main window and a Property Book. The main window displays a hierarchical view of directories and file systems, as well as the mount points and shared resources for the current context. The Property Book displays the chapters and their properties for a selected directory or file system that you can view or modify.

Specifically, File System Manager is a tool that enables you to complete the following tasks:

- Create new file systems
- Modify file system options in the `/etc/vfstab` file
- Manage `/etc/vfstab` files on a single or group of diskless clients or AutoClient™ systems.
- Mount or unmount file systems
- Share or unshare file systems
- Include a file system in existing automounter maps
- Convert a directory into a mount point

For step-by-step instructions on how to complete these tasks, refer to Table A–1. Also, these instructions are included in the online help provided with the File System Manager tool.

File System Manager's Main Window

Figure A–2 shows the important areas of the File System Manager's main window.

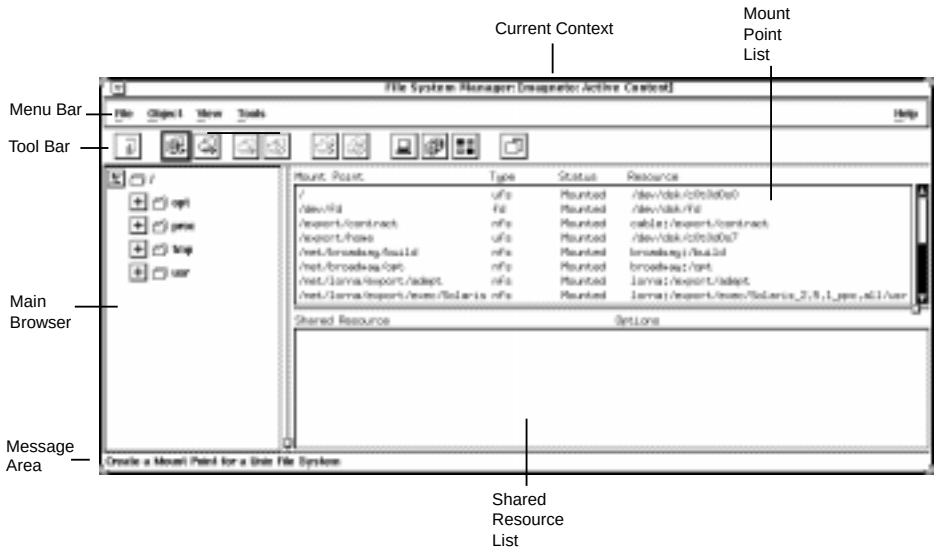


Figure A–2 File System Manager's Main Window

Here are some brief descriptions of the areas within File System Manager's main window:

- **Current Context –** Specifies the server or clients on the server where you are managing file systems. You can change the current context by choosing Load from the File menu or by clicking on the Load Context icon in the tool bar.
- **Menu Bar –** Displays the menus that enable you to perform operations in File System Manager. For detailed descriptions of the menus, see the online help.
- **Tool Bar –** Displays icons that provide an easier way to select commonly-used operations provided by the main menu, such as mount and unmount, and the ability to launch other tools. When you move the mouse pointer over an active icon, the message area describes what operation the icon will perform. You can choose Toolbar from the View menu to turn the tool bar off or on (by default, it is turned on).

- **Main Browser –** Displays the directory hierarchy for the current context. You can use the navigation buttons to expand or collapse the view of the directories and mount points. The browser initially displays the top-level of the file system hierarchy.
- **The + and - Buttons –** Are three-state buttons used to expand and collapse the hierarchical structure you are viewing. The three states are:
  - Collapsed all (the + is displayed)
  - Collapsed managed objects (the +/- is displayed)
  - Expanded all (the - is displayed)

  The +/- state means that the corresponding entry is only partially expanded or collapsed. Clicking on a button in this state will further expand the entry.
- **Message Area –** Provides information about the main window or icon where the mouse pointer is located.
- **Mount Point List –** Displays an entry for each mount point defined in the current context. Each entry contains the full path name of the mount point, the type of file system mounted on that mount point, whether the associated resource is currently mounted on the mount point, and the name of that resource.
- **Shared Resource List –** Displays an entry for each shared resource (directory or mount point) defined in the current context. Each entry contains the full path name of the resource, followed by the options controlling its export.

File System Manager Property Book

There are three ways to open the File System Manager Property Book.

- Select a mount point or directory in the main window and choose Properties from the Object menu.
- Select a mount point or directory in the main window and click the Property Book icon in the tool bar.
- Double-click a mount point or directory in the main window.

Figure A–3 shows the important features of the File System Manager Property Book.

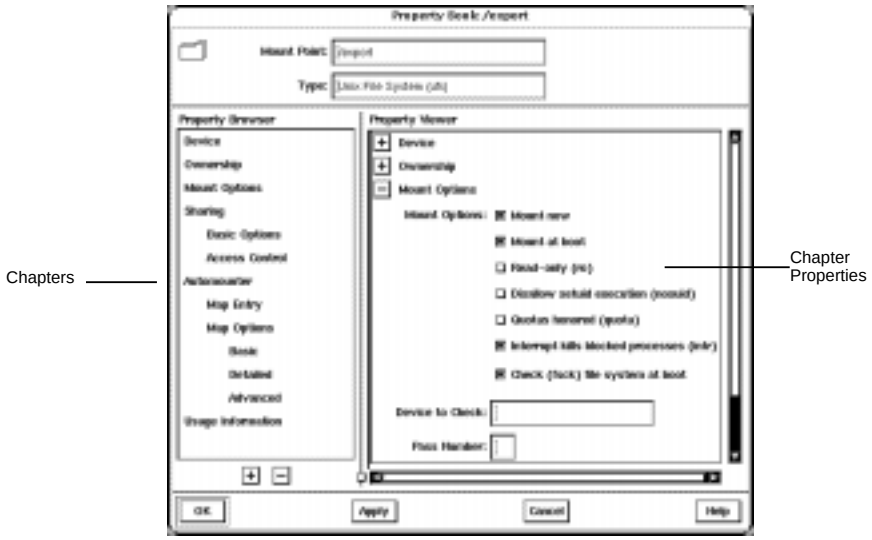


Figure A–3 File System Manager Property Book

- **Mount Point –** Specifies the name of the mount point that you are managing in the property book. In Figure A–3 the object is the `/export` mount point.
- **Type –** Specifies the type of the mount point.
- **Property Browser –** Functions much the same way as the Storage Manager Load Context Property Browser. It lists all the chapters and subchapters (if any) for the selected directory, mount point, or file system.

  Double-clicking on a chapter expands the chapter in the Property Viewer and displays the subchapters (if any) or the object properties. Double-clicking on a subchapter expands that subchapter in the Property Viewer.
- **Chapter –** Contains a common set of properties for the object that you can view or modify.
- **The + and - Buttons –** The buttons that appear below the Property Browser perform an expand all (+) or collapse all (-) function on the chapters in the Property Viewer. This is a quick way of viewing all chapter properties or only the chapter names.

  The buttons that appear next to chapter names in property books also perform an expand all or collapse all function, but only on the selected chapter.
- **Property Viewer –** Enables you to expand and collapse the contents of chapters. You can expand a chapter down to its object properties, enabling you to view or modify the property for the object.
- **Chapter Properties –** Specifies the properties that you can view or modify for the object. There can be one or more properties in a chapter.

For more reference information on the chapters that are available from the File System Manager Property Book, refer to the online help.

Managing File Systems, Mount Points, and Directories With File System Manager

TABLE A–1 Task Map: Managing Files With File System Manager

Task	Description	For Instructions, Go To
Create a UFS File System	Create a new file system on a specified device.	"How to Create a UFS File System" on page 302
Create a Mount Point	Create a local (UFS) or remote (NFS) mount point.	"How to Create a Mount Point" on page 303
Modify the Properties of a Mount Point or Directory	Mount or unmount a file system, share or unshare a directory, or modify an automounter map.	"How to Modify the Properties of a Mount Point or Directory" on page 305
Mount or Unmount a File System	Mount or unmount a file system.	"How to Mount or Unmount a File System" on page 307
Share or Unshare a Directory	Share or unshare a directory.	"How to Share or Unshare a Directory" on page 308

TABLE A–1 Task Map: Managing Files With File System Manager *(continued)*

Task	Description	For Instructions, Go To
View Static Client File Systems	View the file systems that are mounted and directories that are shared at boot time on the server's diskless clients and AutoClient systems.	"How to View Static Client File Systems" on page 310
View Active Server File Systems	View the file systems that are currently mounted and the directories that are shared on the server.	"How to View Active Server File Systems" on page 311
View Static Server File Systems	View the server's file systems that are mounted and directories that are shared at boot time on the server.	"How to View Static Server File Systems" on page 312
Remove a Mount Point From `/etc/vfstab`	Remove a mount point from the `/etc/vfstab` file.	"How to Remove a Mount Point From the `/etc/vfstab` File" on page 313

▼ How to Create a UFS File System

1. **Choose Create File System from the Object Menu.**
   The New File System property book is displayed.

2. **Open the Device chapter.**

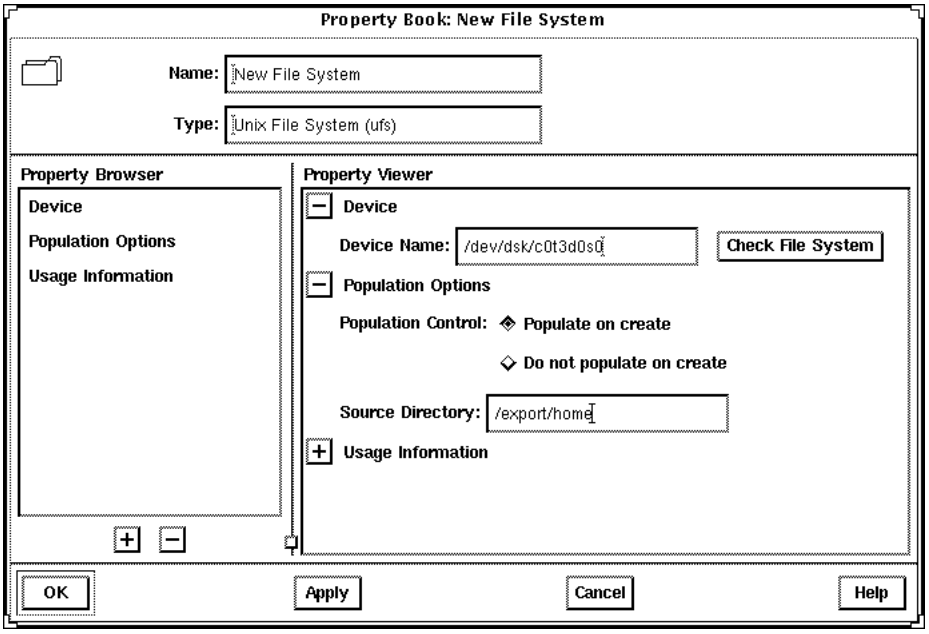
3. **Enter the device name of an unused slice or metadevice on which to create the UFS file system.**
   You can either type the name of a device, or drag and drop a slice from Disk Manager or a metadevice from DiskSuite Tool.

4. **If you want to verify that the specified device is currently available, click the Check File System button.**

5. **If you want to copy the contents of an existing directory into the new file system, open the Population Options chapter and select Populate on Create. Enter the Source Directory from which to copy the contents into the new file system.**

6. **Click OK.**

Example — Creating a UFS File System



▼ How to Create a Mount Point

1. **Click the appropriate icon in the tool bar to create a UFS or NFS mount point, or choose the appropriate option from the Object menu.**
   The Mount Point Name window appears.

2. **Type a name and click OK.**
   The property book for the mount point is displayed.

3. **If you are creating a UFS mount point, skip to Step 6 on page 304. If you are creating an NFS mount point, open the Server Path chapter.**

4. **Type the name of a server and click Probe Server.**
   A list of exported file systems is displayed in the Exported File Systems list.

5. **Click the desired file system name or type a name in the Path Name field. Skip to Step 9 on page 304.**

6. **Open the Device chapter.**

7. **Enter the device name of an unused slice or metadevice on which to create the UFS mount point.**

   You can either type the name of a device or drag and drop a slice from Disk Manager.

8. **If you want to verify that the specified device is currently available, click the Check File System button.**

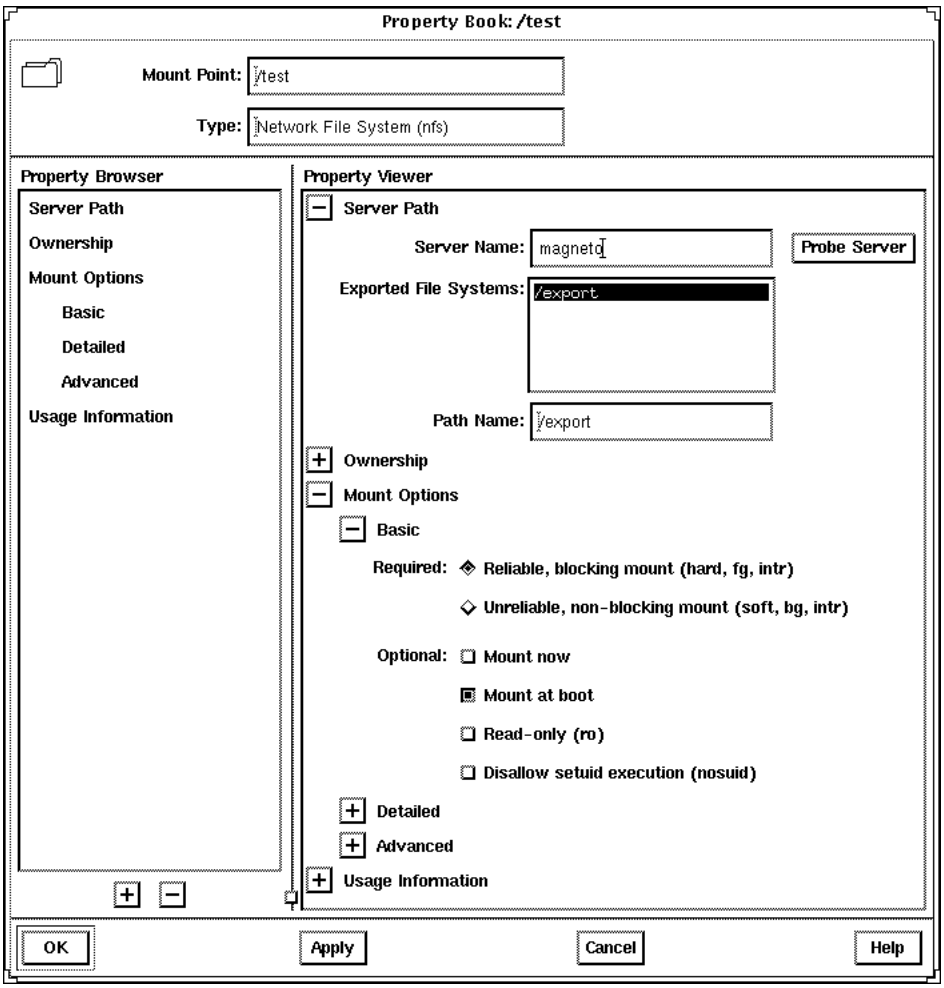
9. **Make modifications to the other chapters, if needed.**

   For example, in the Mount Options chapter you can choose to mount now and/or mount at boot time.

10. **Click OK.**

    The mount point appears in the Mount Point list.

Example — Creating a Mount Point



▼ How to Modify the Properties of a Mount Point or Directory

1. **Select a directory or mount point from the main window.**

   You can select a directory or mount point from the main browser, the Mount Point list, or the Shared Resource list.

Note - Once you select a mount point or directory, the Object menu may allow you to automatically mount a file system, unmount a file system, share a directory, or unshare a directory. If you want to complete one of these tasks, it is faster to use the Object menu rather than make the change using the Property Book.

2. **Choose Properties from the Object menu.**

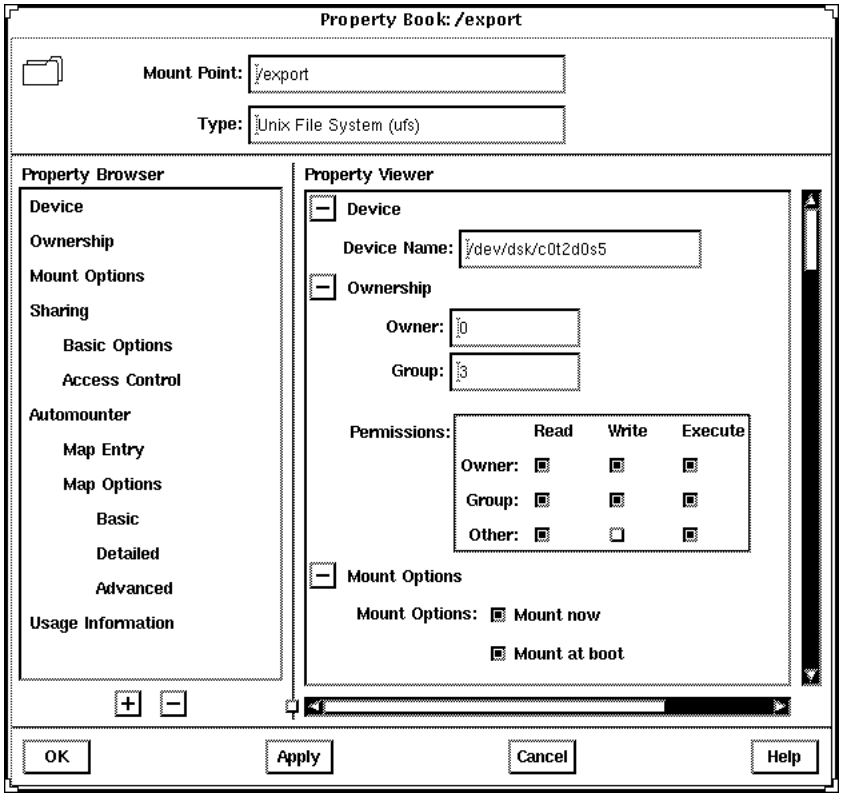
   The property book for the file system or directory is displayed.

3. **Open the available chapters to modify the properties for the mount point or directory.**

   Click Help in the property book to see detailed information about each chapter.

4. **Click OK.**

Example — File System Manager Property Book



▼ How to Mount or Unmount a File System

1. **Select a mount point from the main browser, Mount Point list, or Shared Resource list.**

Note - Once you select a mount point, the Object menu may allow you to automatically mount or unmount a file system. This method is faster than making the change using the Property Book.

---

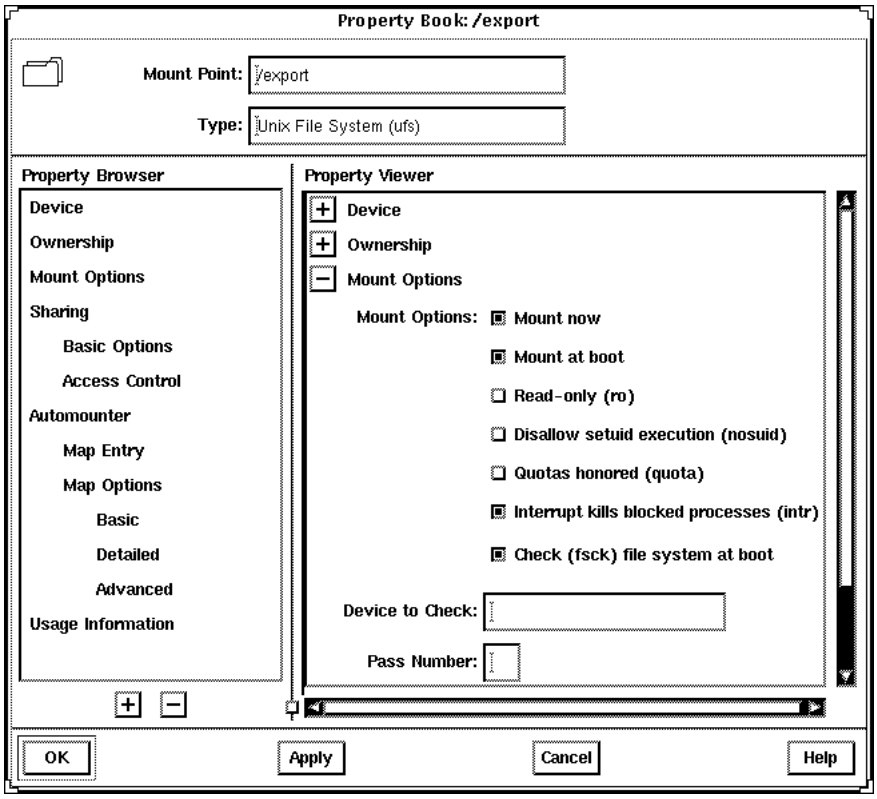
2. **Choose Properties from the Object menu.**

   The property book for the file system is displayed.

3. **Open the Mount Options chapter to modify the mount options (for example, mount or unmount the file system).**

4. **Click OK.**

Example — Mounting a File System



▼ How to Share or Unshare a Directory

1. **Select a directory or UFS mount point from the main browser, Mount Point list, or Shared Resource list.**

   Note - Once you select a directory or mount point, the Object menu may allow you to automatically share or unshare it. This method is faster than making the change using the Property Book.

   ---

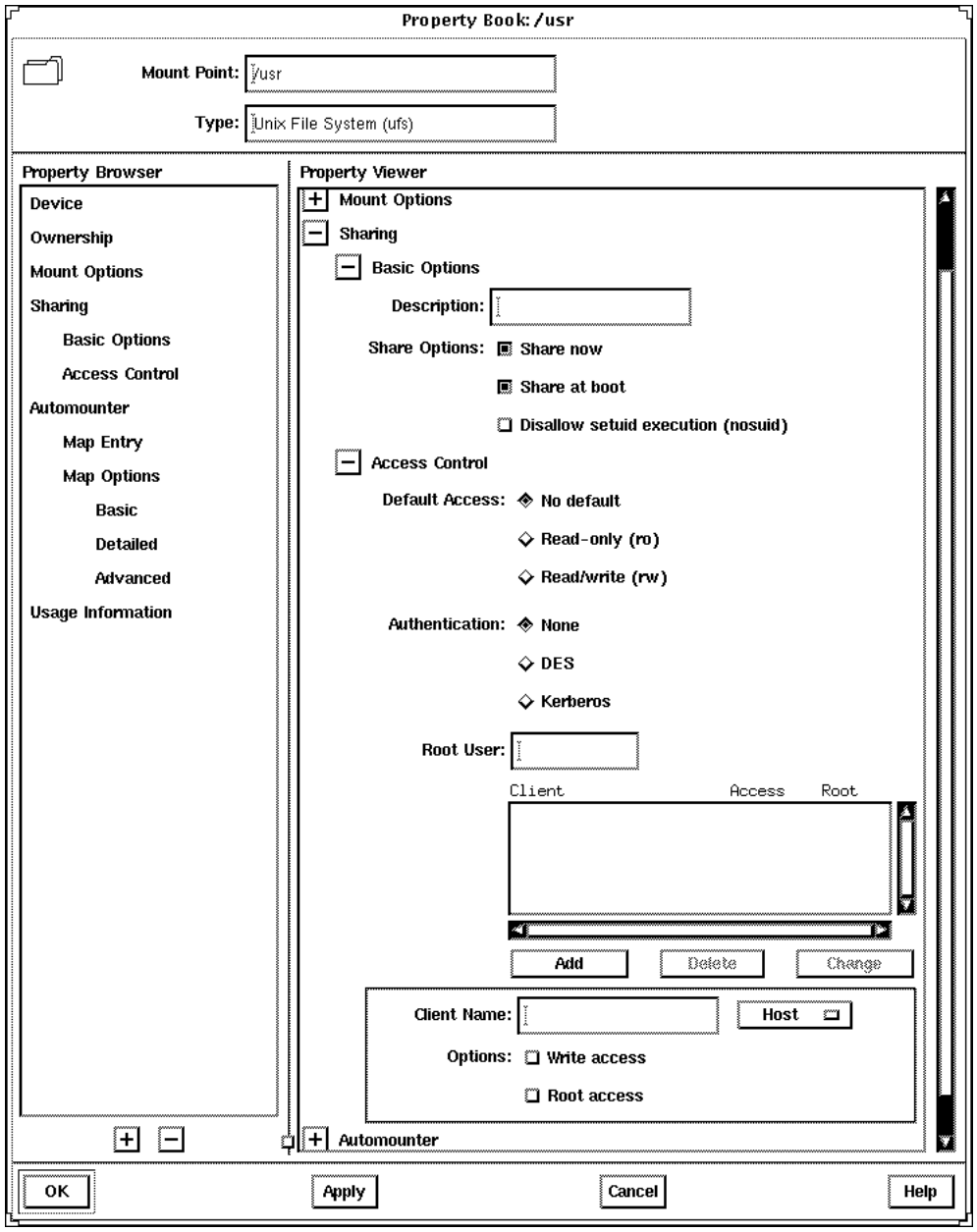
2. **Choose Properties from the Object menu.**

   The property book for the file system is displayed.

3. **Open the Sharing chapter to modify the share options (for example, share or unshare the file system).**

4. **Click OK.**

Example — Sharing a Directory



▼ How to View Static Client File Systems

Static client file systems are those file systems that will be mounted on a server's AutoClient or diskless clients when they boot.

1. **Click the Load Context icon or choose Load from the File menu.**

   The Load Context property book is displayed.

2. **Open the File System chapter, if not done already.**

3. **Click the Static Client File System button.**

   Note - This button is only active if there are AutoClient systems or diskless clients configured on the system.

   The Client Context field is activated.

4. **Select either Client Group or Individual Client.**

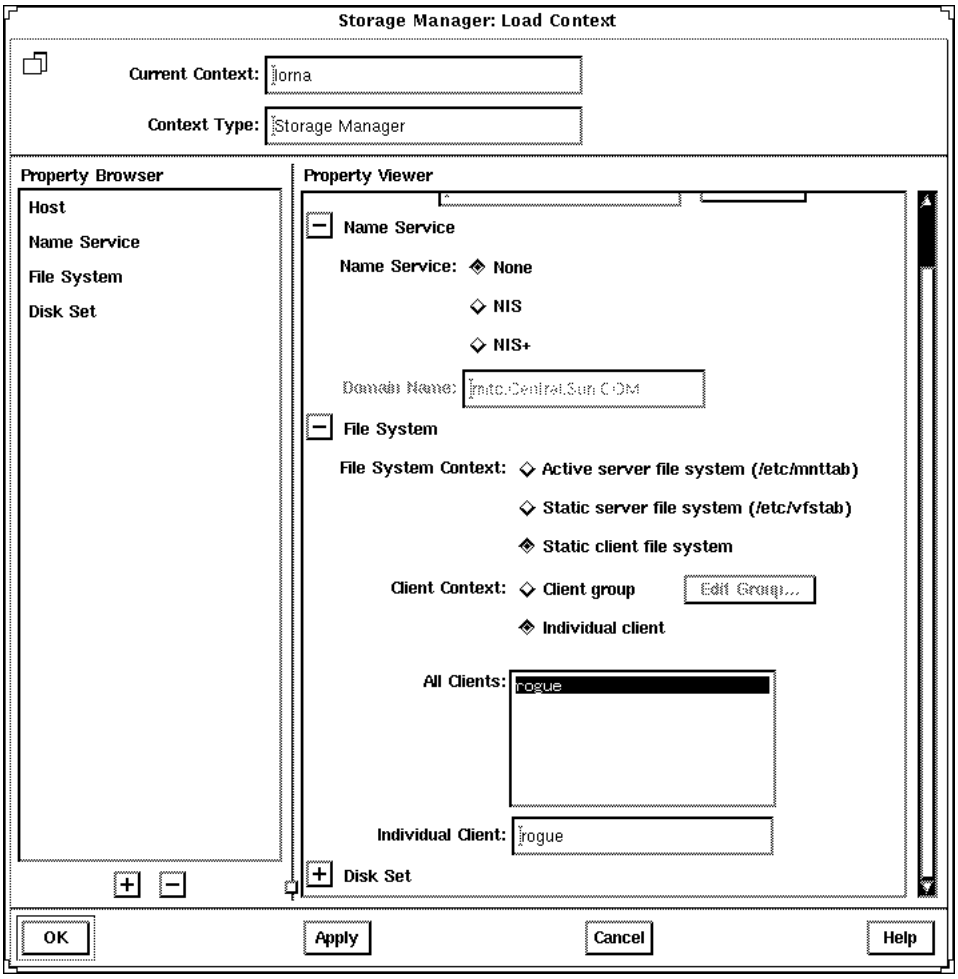
5. **Select a system from the Client Group or All Clients list.**

6. **Click OK.**

   The file systems that will be mounted at boot time for the clients are displayed in the Mount Point list on the main window.

   The directories that will be shared at boot time for the clients are displayed in the Shared Resources list on the main window.

Example — Viewing Static Client File Systems



▼ How to View Active Server File Systems

Active server file systems are the file systems on a server that are currently mounted or shared. This is the same information as contained in the `/etc/mnttab` file.

1. **Click the Load Context icon or choose Load from the File menu.**
   The Load Context property book is displayed.

2. **Open the File System chapter, if not done already.**

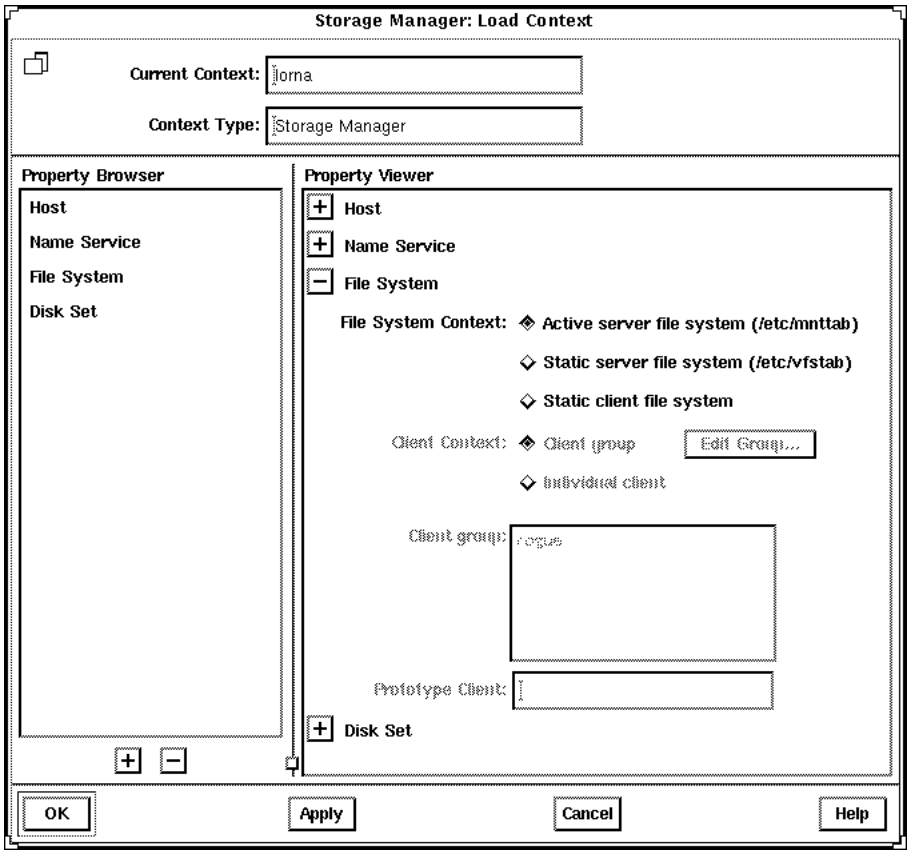
3. **Click the Active Server File System button.**

4. **Click OK.**

   The file systems that are currently mounted on the server are displayed in the Mount Point list on the main window.

   The directories that are currently shared on the server are displayed in the Shared Resources list on the main window.

Example — Viewing Active Server File Systems



▼ How to View Static Server File Systems

Static server file systems are the file systems on a server that will be mounted or shared at boot time. This is the same information as contained in the `/etc/vfstab` file.

1. **Click the Load Context icon or choose Load from the File menu.**

   The Load Context property book is displayed.

2. **Open the File System chapter, if not done already.**

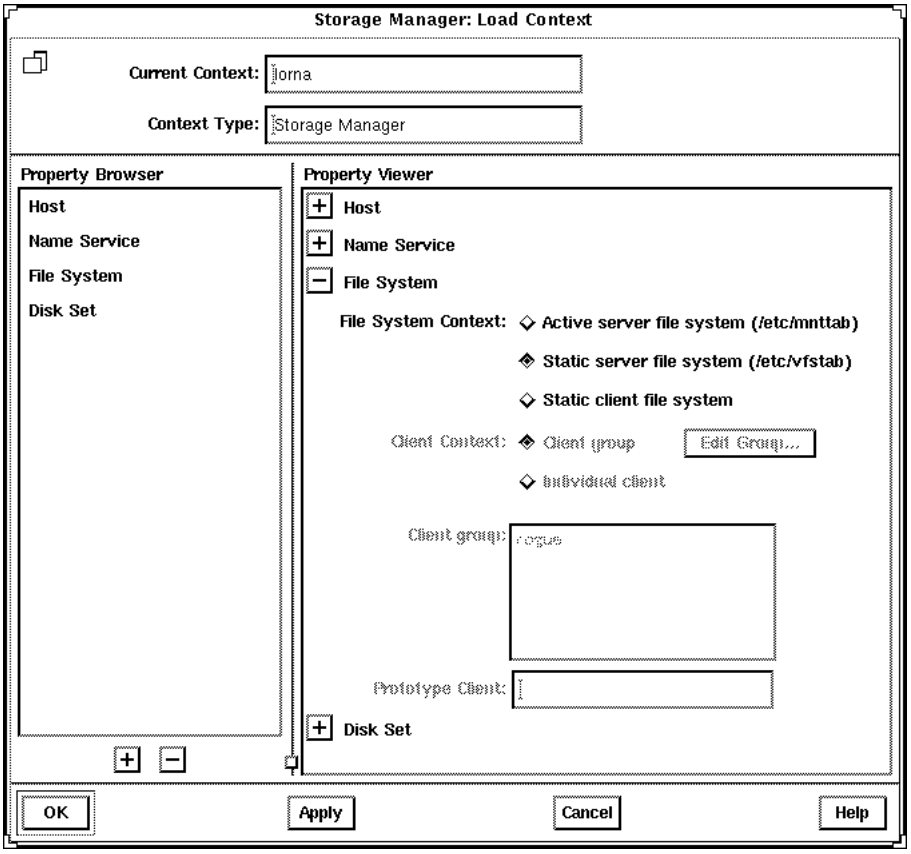
3. **Click the Static Server File System Button.**

4. **Click OK.**

   The file systems that will be mounted at boot time on the server are displayed in the Mount Point list on the main window.

   The directories that will be shared at boot time on the server are displayed in the Shared Resources list on the main window.

Example — Viewing Static Server File Systems



▼ How to Remove a Mount Point From the `/etc/vfstab` File

1. **Perform the procedures in “How to View Static Server File Systems” on page 312.**

2. **Select a mount point from the main browser, Mount Point list, or Shared Resource list.**

3. **Choose Properties from the Object menu.**
   The property book for the file system is displayed.

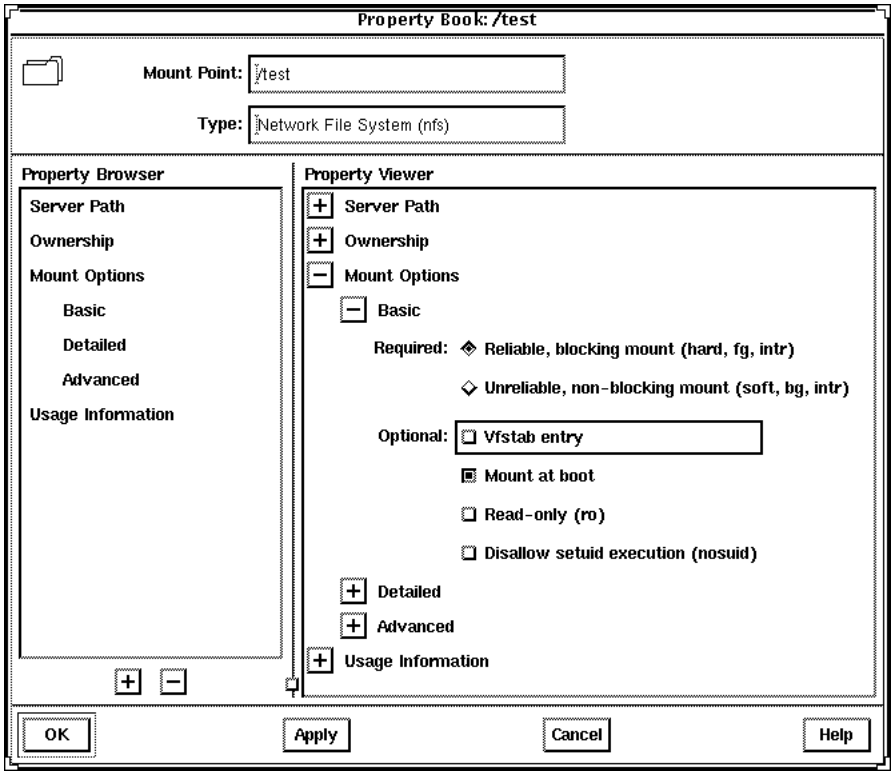
4. **Open the Mount Options chapter.**

5. **Open the Basic subchapter.**

6. **Click Vfstab Entry.**
   You are toggling (deselecting) this property.

7. **Click OK.**
   The mount point no longer appears in the Mount Point list.

Example — Removing a Mount Point Entry From the `/etc/vfstab` File



Disk Manager Overview

Disk Manager is a tool that enables you to view and edit `fdisk` partitions and slices using two types of windows, the main window and a Property Book. The main window displays the controllers, targets, disks, and slices for the current context. The Property Book displays the chapters and their properties for the selected disk(s), and it is at this level that you can view and edit disk properties.

Specifically, you can complete the following tasks with Disk Manager.

- Assign a volume name to a disk.
- View and modify `fdisk` partitions on x86 platforms.
- Show and set the active `fdisk` partition on x86 platforms.
- View and modify slice geometry on SPARC and x86 platforms.
- Copy a disk's characteristics to one or more disks of the same type.

Note - Before modifying `fdisk` partitions and slices, you might want to back up critical data.

For step-by-step instructions on how to complete these tasks, refer to Table A–2. Also, these instructions are included in the online help provided with the Disk Manager tool.

Disk Manager's Main Window

Figure A–4 shows the important areas of the Disk Manager's main window.

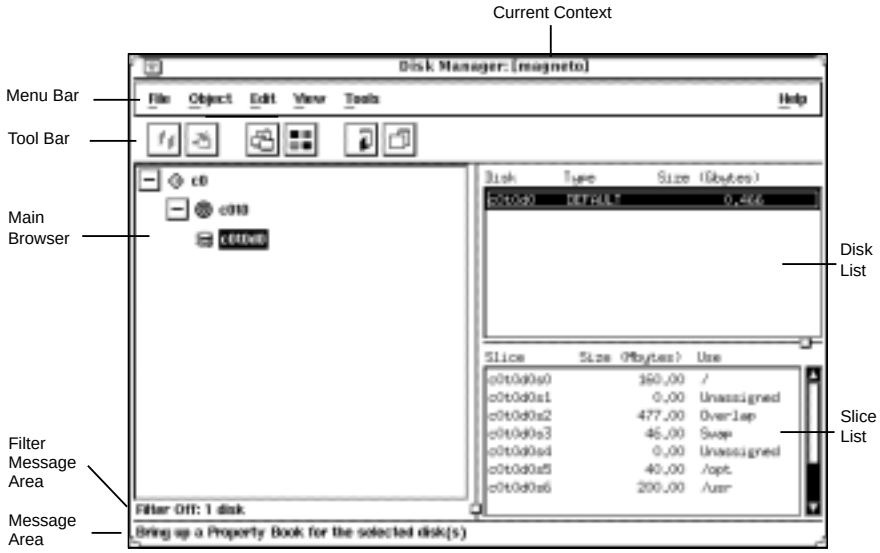


Figure A–4 Disk Manager's Main Window

Here are some brief descriptions of the areas within Disk Manager's main window:

- **Current Context –** Specifies the server where you are managing disks. You can change the current context by choosing Load from the File menu or by clicking on the Load Context icon in the tool bar.
- **Menu Bar –** Displays the menus that enable you to perform operations in Disk Manager. For detailed descriptions of the menus, see online help.
- **Tool Bar –** Displays icons that provide an easier way to select commonly-used operations provided by the main menu, such as cutting, pasting, and launching other tools. When you move the mouse pointer over an active icon, the message area describes what operation the icon will perform. You can choose Toolbar from the View menu to turn the tool bar off or on (by default, it is turned on).
- **Main Browser –** Displays a hierarchical view of the disk controllers, SCSI targets, and disks for the current system context. You can use the expand/collapse buttons to expand or collapse the view of the SCSI targets and disks under each disk controller. The main browser initially displays the disk controllers for all the available disks in the current context, in their collapsed state.
- **The + and - Buttons –** Are three-state buttons used to expand and collapse the hierarchical structure you are viewing. The three states are:

  - Collapsed all (the + is displayed)
  - Collapsed managed objects (the +/- is displayed)
  - Expanded all (the - is displayed)

  The +/- state means that the corresponding entry is only partially expanded or collapsed. Clicking on a button in this state will further expand the entry.

- **Filter Message Area –** Displays whether a filter is on or off, and if a filter is on, how many disks are filtered.
- **Message Area –** Provides information about the main window or icon where the mouse pointer is located.
- **Disk List –** Displays the type and size information for the disk(s) currently selected in the main browser.
- **Slice List –** Displays the slices and their sizes for the disk(s) currently selected in the disk list.

Selecting Multiple Disks

If you have multiple disks that are the same vendor type and have the same physical geometry, you can perform an operation on them simultaneously. This is called *batch editing* or *batching*. To select more than one disk in the main browser or disk list, click SELECT (by default, the left mouse button) on the first disk. Then select each subsequent disk by pressing the Shift key and clicking SELECT.

Disk Manager Property Book

There are three ways to open the Disk Manager Property Book.

- Select one or more disks in the main browser and choose Properties from the Object menu.
- Select one or more disks in the main browser and click the Property Book icon in the tool bar.
- Double-click a disk in the main browser.

Figure A–5 shows the important features of the Disk Manager Property Book.

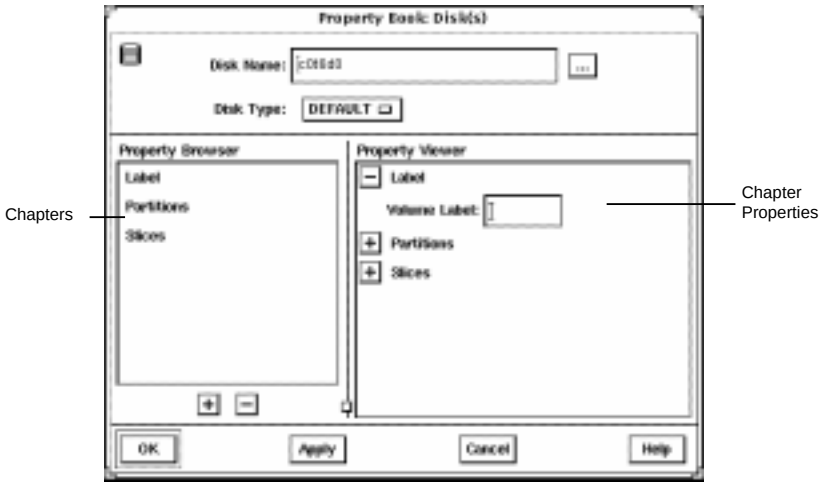


Figure A–5 Disk Manager Property Book

Here are some brief descriptions of the areas within Disk Manager's main window:

- **Disk Name –** Specifies the object or group of objects that you can manage in the property book. In Figure A–5, the object is a disk whose name is `c0t6d0`.
- **The ... Button –** Indicates that there is at least one other disk of the same type as the disk being edited on the system. Clicking on the ... button displays the Choose Disks to Edit window. This allows you to select more than one disk (of the same type) for simultaneous edit operations.
- **Disk Type –** Specifies the type of the object. In Figure A–5, the object is a disk that has a disk type of DEFAULT.
- **Property Browser –** Functions much the same way as the Storage Manager Load Context Property Browser. It lists all the chapters or subchapters (if any) for the selected disk(s).

  Double-clicking on a chapter expands the chapter in the Property Viewer and displays the subchapters (if any) or the object properties.
- **Chapters –** Contains a common set of properties for the object that you can view or modify.
- **The + and - Buttons –** The buttons that appear below the Property Browser perform an expand all (+) or collapse all (-) function on the chapters in the Property Viewer. This is a quick way of viewing all chapter properties or only the chapter names.

  The buttons that appear next to chapter names in property books also perform an expand all or collapse all function, but only on the selected chapter.
- **Property Viewer –** Enables you to expand and collapse the contents of chapters. You can expand a chapter down to its object properties, enabling you to view or modify the property for the object.

- **Chapter Properties –** Specifies the properties that you can view or modify for the object. There can be one or more properties in a chapter. In Figure A–5, the Label chapter has a Volume Label field that provides a way to assign a name to a disk.

  For more reference information on the chapters that are available from the Disk Manager Property Book, see online help.

Managing Disks With Disk Manager

TABLE A–2 Task Map: Managing Disks With Disk Manager

Task	Description	For Instructions, Go To
Specify a Viewing Filter	Specify the attributes of the disk(s) that you want to view in the Disk Manager main browser.	"How to Specify a Viewing Filter" on page 320
Specify a Volume Label	Assign a name to a disk.	"How to Specify a Volume Label" on page 321
Modify `fdisk` Partitions	Select an active `fdisk` partition, modify `fdisk` partition sizes, or modify the type of `fdisk` partitions.	"How to Modify `fdisk` Partitions" on page 322
Modify Slice Geometry	Modify slice sizes.	"How to Modify Slice Geometry" on page 323
Clone a Disk	Copy a disk's characteristics onto other disks of the same type.	"How to Clone a Disk" on page 324

▼ How to Specify a Viewing Filter

1. **Choose Filter from the View menu.**

   The Filter Disks and Slices window appears with a list of the available disk attributes in the Available Attributes list.

2. **Specify which disks, with specific disk attributes, you want to display in the main window.**

   a. **Click a disk attribute in the Available Attributes list.**

   b. **Click the >> button to move the attribute to the Show Disks list.**

   c. **Repeat Step 2 on page 320 and Step 2 on page 320 until the Show Disks list contains all the disk attributes that disks displayed in the main window will have.**

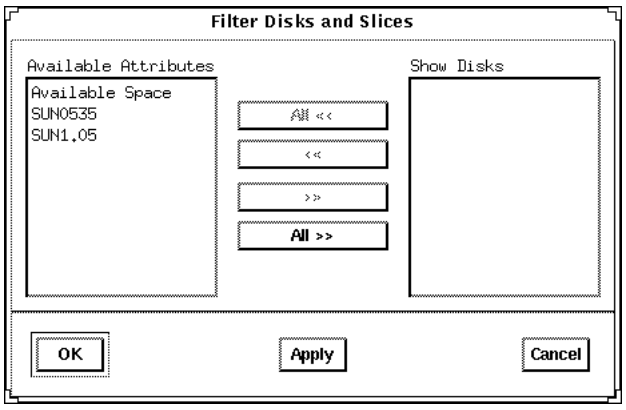
   Note - Clicking on the All >> button moves the entire list of attributes in the Available Attributes list to the Show Disks list. Clicking on the All << button, moves the entire list of attributes in the Show Disks list to the Available Attributes list.

   ---

3. **Click OK.**

   The main window refreshes, displaying only the disks that match the criteria specified in the Show Disks list. The message area below the main browser displays the number of filtered disks.

Example — Filter Disks and Slices Window



The Available Attributes section contains an entry for each disk type present in the current context, and an entry called “Available Space” that corresponds to all disks and slices with free space. The Show Disks section lists the attributes of the disks displayed in the main window. By default, this section is empty because filtering is turned off.

▼ How to Specify a Volume Label

1. **Select the disk that you want to modify in the Disk Manager main browser.**

2. **Open the Property Book for the selected disk.**

   For more information see “Disk Manager Property Book” on page 317.

   The Property Book window appears.

3. **Open the Label chapter.**

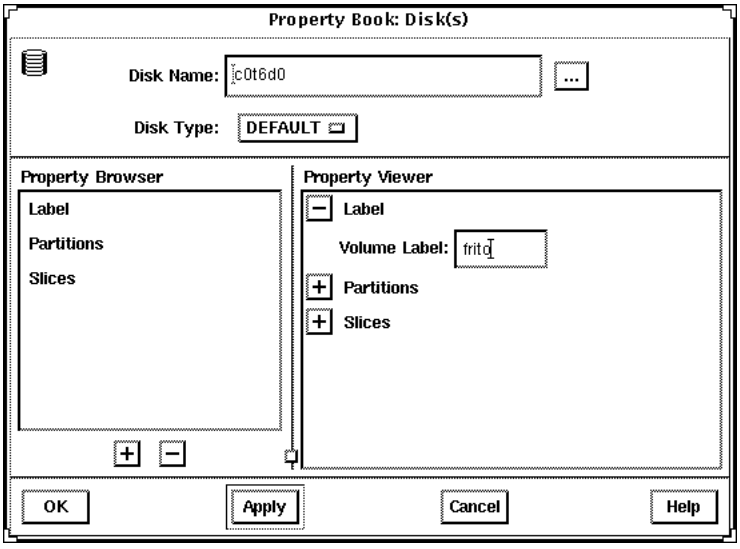
   For more information see “Disk Manager Property Book” on page 317.

4. **Delete the existing name in the Volume Label field, if applicable.**

5. **Enter the name of the volume label, which must be an alphanumeric string of 8 or fewer characters.**

6. **Click OK.**

Example — Specifying a Volume Label



▼ How to Modify `fdisk` Partitions

1. **Select the disk that you want to modify in the Disk Manager main browser.**

2. **Open the Property Book for the selected disk.**
   For more information see “Disk Manager Property Book” on page 317.
   The Property Book window appears.

3. **Open the Partitions chapter.**
   For more information see “Disk Manager Property Book” on page 317.
   Size the Property Book window so that the entire partition layout is visible.

4. **Click the select box in the Editable column that corresponds to the `fdisk` partition you want to edit.**

5. **Modify the size of an `fdisk` partition(s) by clicking on the arrows in the Slider portion of the window, or click the appropriate Start or End field in the Partition Data portion of the window, type in a value, and press Return.**
   For reference information, see online help.

   Note - For x86 platforms, `fdisk` Solaris partitions must start at cylinder 1 or higher and they may not overlap.

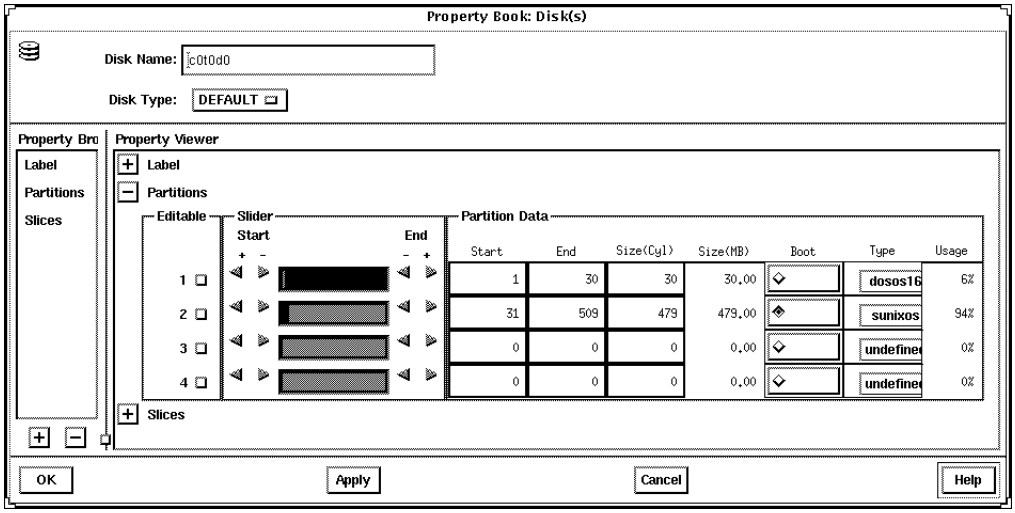
   ---

6. **If desired, select the button in the Boot column to make the `fdisk` partition active (the one whose operating system will be used at system start-up).**

7. **Choose the type of the `fdisk` partition.**
   Choose the appropriate type using the menus in the Type column.

8. **Click OK.**

Example — Modifying `fdisk` Partitions



▼ How to Modify Slice Geometry

1. **Select the disk that you want to modify in the Disk Manager main browser.**

2. **Open the Property Book for the selected disk.**

   For more information see “Disk Manager Property Book” on page 317.

   The Property Book window appears.

3. **Open the Slices chapter.**

   For more information see “Disk Manager Property Book” on page 317.

   Size the Property Book window so that the entire slice layout is visible.

4. **Click the select box in the Editable column that corresponds to the slice you want to edit.**

5. **Modify the size of a slice(s) by clicking on the arrows in the Slider portion of the window, dragging the bar indicators, or typing values in the Start and End fields.**

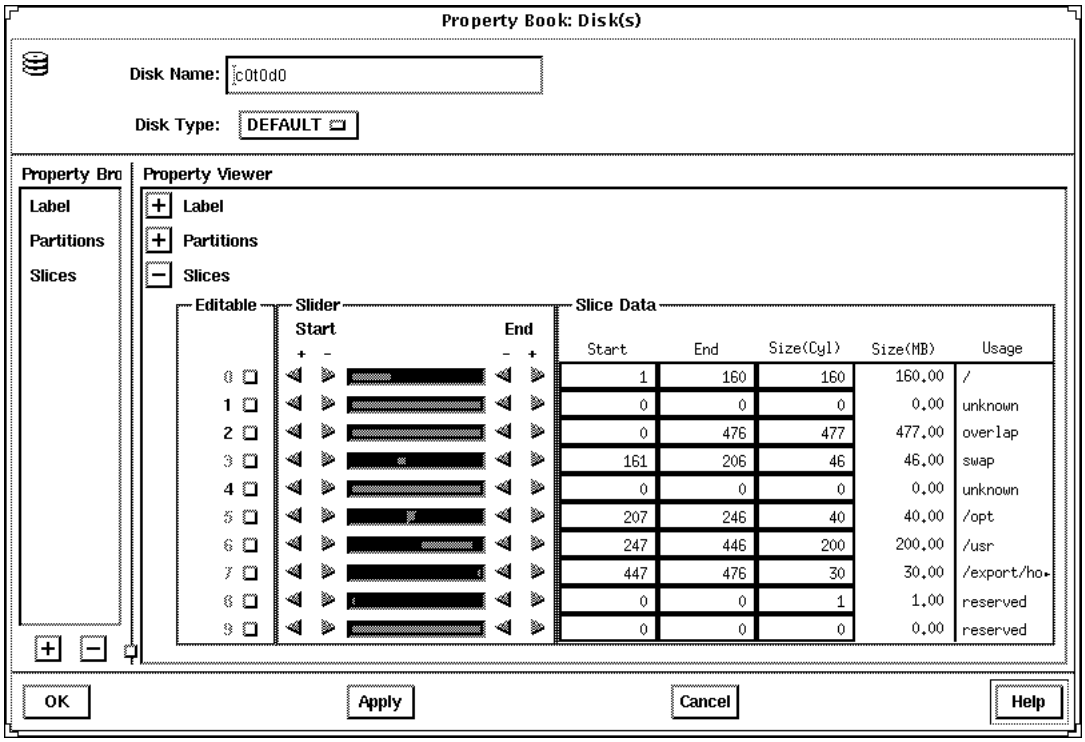
   For reference information, see online help.

   Note - For x86 platforms, slices must start at cylinder 1 or higher and may not overlap.

   ---

6. **Click OK.**

Example — Modifying Slice Geometry



▼ How to Clone a Disk

1. **Select the disk you want to clone in the Disk Manager main browser.**

2. **Click the copy icon in the tool bar or choose Copy from the Edit menu.**

3. **Select a unused disk of the same type, or select a controller containing one or more unused disks of the same type, in the Disk Manager main browser.**

4. **Click the paste icon in the tool bar, or choose Paste from the Edit menu.**

Note - An alternate method to performing the tasks described in Step 2 on page 324 to Step 4 on page 324 is to press ADJUST (by default the middle mouse key) over the disk you want to copy, drag the cursor onto the disk of the same type, and release the ADJUST button.

Index

E

F

G

H

N

O

P

R

S

T

U

V

X