NAME

growfs - non-destructively expand a UFS file system

SYNOPSIS

```
growfs [ -M mount-point ] [ newfs-options ]
       [ raw-device ]
```

AVAILABILITY

/usr/opt/SUNWmd/sbin

DESCRIPTION

growfs non-destructively expands a mounted or unmounted UNIX file system (UFS) to the size of the file system's slice(s).

Typically, disk space is expanded by first adding a slice to a metadevice, then running the growfs command. When adding space to a mirror, you expand each submirror before expanding the file system. On a trans metadevice, the master device is expanded, not the trans metadevice. Then the growfs command is run on the trans metadevice. (You can add space to a logging device, but you do not need to run the growfs command. The new space is automatically recognized.)

growfs will ``write-lock'' (see lockfs) 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. The argument for -s is the number of sectors, and must be a multiple of the cylinder size. Note: The file system cannot be grown if a cylinder size of less than 2 is specified. Refer to the newfs(1M) man page for information on the options available when growing a file system.

growfs displays the same information as mkfs during the expansion of the file system.

If growfs is aborted, recover any lost free space by unmounting the file system and running the fsck command, or run the growfs command again.

OPTIONS

Root privileges are required for all of the following options.

-M mount-point

The file system to be expanded is mounted on mount-

point. File system locking (lockfs) will be used.

newfs-options
The options are documented in the newfs man page.

raw-device
Specifies the name of a raw metadevice or raw special device, residing in /dev/md/rdsk, or /dev/rdsk, respectively, including the disk slice, where you want the file system to be grown.

EXAMPLES

The following example expands a nonmetadevice slice for the /export file system. In this example, the existing slice, /dev/dsk/c1t0d0s3, is converted to a metadevice so additional slices can be concatenated.

```
# metainit -f d8 2 1 c1t0d0s3 1 c2t0d0s3
# umount /export
```

(Edit the /etc/vfstab file to change the entry for /export to the newly defined metadevice, d8.)

```
# mount /export
# growfs -M /export /dev/md/rdsk/d8
```

This example starts by running the metainit command with the -f option to force the creation of a new concatenated metadevice d8 which consists of the existing slice /dev/dsk/c1t0d0s3 and a new slice /dev/dsk/c2t0d0s3. Next, the file system on /export must be unmounted. The /etc/vfstab file is edited to change the entry for /export to the newly defined metadevice name, rather than the slice name. After the file system is remounted, the growfs command is run to expand the file system.

The file system will span the entire metadevice when growfs completes. The -M option enables the growfs command to expand a mounted file system. During the expansion, write access for /export is suspended until growfs unlocks the file system. Read access is not affected, though access times are not kept when the lock is in effect.

The following example picks up from the previous one. Here, the /export file system mounted on metadevice d8 is dynamically expanded.

```
# metattach d8 c0t1d0s2
# growfs -M /export /dev/md/rdsk/d8
```

This example begins by using the metattach command to dynamically concatenate a new slice, /dev/dsk/c0t1d0s2, to the

end of an existing metadevice, d8. Next, the growfs command specifies that the mount-point is /export and that it is to be expanded onto the raw metadevice /dev/md/rdsk/d8. The file system will span the entire metadevice when growfs completes. During the expansion, write access for /export is suspended until growfs unlocks the file system. Read access is not affected, though access times are not kept when the lock is in effect.

The following example expands a mounted file system /files, to an existing mirror, d80, which contains two submirrors, d9 and d10.

```
# metattach d9 c0t2d0s5
# metattach d10 c0t3d0s5
# growfs -M /files /dev/md/rdsk/d80
```

In this example, the metattach command dynamically concatenates the new slices to each submirror. The metattach command must be run for each submirror. The mirror will automatically grow when the last submirror is dynamically concatenated. The mirror will grow to the size of the smallest submirror. The growfs command then expands the file system. The growfs command specifies that the mount-point is /files and that it is to be expanded onto the raw metadevice /dev/md/rdsk/d80. The file system will span the entire mirror when the growfs command completes. During the expansion, write access for the file system is suspended until growfs unlocks the file system. Read access is not affected, though access times are not kept when the lock is in effect.

SEE ALSO

fsck(1M), lockfs(1M), mkfs(1M), metattach(1M), newfs(1M),

Solstice DiskSuite User's Guide, Solstice DiskSuite Refer-

ence

LIMITATIONS

Only UFS file systems (either mounted or unmounted) can be expanded using the growfs command. Once a file system is expanded, it cannot be decreased in size.

The following conditions prevent you from expanding file systems:

- When acct is activated and the accounting file is on the target device.
- When C2 security is activated and the logging file is on the target file system.
- When there is a local swap file in the target file system.
- When the file system is root (/), /usr, or swap.

NAME

mdlogd, mdlogd.cf - Solstice DiskSuite SNMP support

SYNOPSIS

mdlogd
mdlogd.cf

AVAILABILITY

/usr/opt/SUNWmd/sbin/mdlogd
/etc/opt/SUNWmd/mdlogd.cf

DESCRIPTION

mdlogd implements a simple daemon that watches the system console looking for messages written by the DiskSuite device driver (md). When a DiskSuite message is detected the daemon will send a generic SNMP trap.

/etc/opt/SUNWmd/mdlogd.cf is used to control the daemon's behavior.

It is an ASCII file with the basic form:

ENTERPRISE = <enterprise-id>
OBJECTID = <object-id>

<reg-exp> <trap-destination> <generic trap #> <specific trap #>

[...]

<enterprise-id> and <object-id> are required and must be specified. They are used by all traps generated by the daemon.

<enterprise-id> is the SNMP identifier for the enterprise to which the system running the daemon belongs. For example, the Sun Microsystems enterprise ID is: 1.3.6.1.4.1.42.

<object-id> is the SNMP identifier of the system running the daemon. For example, the object-id of a host within the Sun enterprise might be: 1.3.6.1.4.1.42.10.

The remainder of the file consists of tuples which describe a regular expression and the specific SNMP trap to be generated when a matching message is written to the system console.

Each tuple has four fields:

<regular expression> specifies a regular expression to be matched.

<trap destination> specifies the destination for the SNMP trap, given in the following

form: "hostname:port-number:protocol"

<generic trap #> specifies the SNMP generic trap number, ranging from 0 to 6. These numbers have pre-defined meanings:

0 - cold-start
1 - warm-start
2 - link-down
3 - link-up
4 - authentication
5 - EGP Neighbor Lost
6 - enterprise specific trap

Traps of type 6 include an additional enterprise specific trap number.

```
        <specific trap #>     specifies an arbitrary number.
                              Interpretation of this number
                              is enterprise specific.

EXAMPLE:
     #
     #ident "@(#)mdlogd.cf 1.1 96/02/15"
     #
     #
     # DiskSuite SNMP Trap configuration file.
     #
     # This file specifies the SNMP trap data to be sent when a
     # notable condition related to the DiskSuite driver (md)
     # is detected.
     #
     # The conditions are based on the event logging which the
     # driver does using the cmn_err() interface.   The events
     # have different severity levels: NOTICE, WARNING and PANIC.
     # They appear on the console and look like this:
     #
     # unix: WARNING: md: d81: write error on /dev/md/dsk/d5
     # unix: WARNING: md: d81: /dev/md/dsk/d5 needs maintenance
     # unix: NOTICE: md: d81: hotspared device /dev/md/dsk/d5
     #                        with /dev/md/dsk/d2
     #
     # Using this configuration file, a different SNMP trap may
     # be associated with each level.  See mdlogd.cf for a
     # brief summary of the generic SNMP traps and their meanings.
     #
     # Generic trap variables:
     #
     # ENTERPRISE:  the enterprise to which the system belongs
     # OBJECTID:    the id of the system
     #



SunOS 5.7           Last change: 19 July 1996                   2



Maintenance Commands                                    MDLOGD(1M)



     ENTERPRISE = 1.3.6.1.4.1.42
     OBJECTID = 1.3.6.1.4.1.42.860
     #
     # 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

     Given this  configuration  file  and  an  error  written  to
```

/dev/console on the host which looks like:

```
    WARNING: md: d6: /dev/dsk/c3t3d0s7 needs maintenance
```

an SNMP trap will be dispatched.

If this trap were received by SunNetManager, it would look like:

```
    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
```

NOTES

The supported regular expressions (RE) are constructed as follows:

1.1 Any character that is not a special character (to be defined) matches itself.

1.2 A backslash (\) followed by a special character matches the literal character itself (i.e., this `escapes' the special character).

1.3 The `special' characters are: + * ? . [ ] ^ $

1.4 The period (.) matches any character except the newline. E.g., `.umpty' matches either `Humpty' or `Dumpty'.

1.5 A set of characters enclosed in brackets ([]) is a one-character RE that matches any of the characters in that set. E.g., `[akm]' matches either an `a', `k' or `m'. A range of characters can be indicated with a dash. E.g., `[a-z]' matches any lower-case letter. However, if the first character of the set is the

caret (^), the the RE matches any character except those in the set. It does not match the empty string. Example: [^akm] matches any character except `a', `k' or `m'. The caret loses its special meaning if it is not the first character of the set.

The following rules can be used to build a multicharacter RE:

2.1 A one-character RE followed by an asterisk (*) matches zero or more occurences of the RE. Hence, [a-z]* matches zero or more lower-case characters.

2.2 A one-character RE followed by a plus (+) matches one or more occurences of the RE. Hence, [a-z]+ matches one or more lower-case characters.

2.3 A question mark (?) is an optional element. The preceeding RE can occur zero or once in the string -- no more. E.g., xy?z matches either xyz or xz.

2.4 The concatenation of REs is an RE that matches the corresponding concatenation of strings. E.g., [A-Z][a-z]* matches any capitalized word.

Finally, the entire regular expression can be anchored to match only the beginning or end of a line:

3.1 If the caret (^) is at the beginning of the RE, then the matched string must be at the beginning of a line.

3.2 If the dollar sign ($) is at the end of the RE, then the matched string must be at the end of the line.

The following escape codes can be used to match control characters:

\b	backspace
\e	ESC (escape)
\f	formfeed
\n	newline
\r	carriage return

```
          \t     tab
          \xddd  the literal hex number 0xddd
          \^C    Control Code. E.g., \^D is `control-d'
```

SEE ALSO

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

NAME

metaclear - delete active metadevices and hot spare pools

SYNOPSIS

```
metaclear -h
metaclear [ -s setname ] -a [ -f ]
metaclear [ -s setname ] [ -f ] metadevice...
          hot_spare_pool...
metaclear [ -s setname ] -r [ -f ] metadevice...
          hot_spare_pool...
```

AVAILABILITY

/usr/opt/SUNWmd/sbin

DESCRIPTION

metaclear deletes all configured metadevice(s) and hot spare pool(s), or the specified metadevice and/or hot_spare_pool. Once a metadevice or hot spare pool is deleted, it must be recreated using metainit before it can be used again.

Any metadevice currently in use (open) cannot be deleted.

OPTIONS

Root privileges are required for all of the following options except -h.

-a Deletes all metadevices and configured hot spare pools.

-f Deletes a metadevice that contains a subcomponent in an errored state.

-h Displays usage message.

-r Recursively deletes specified metadevices and hot spare pools.

-s setname
Specifies the name of the diskset on which metaclear will work. Using the -s option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on local metadevices and/or hot spare pools.

metadevice ...
Specifies the name(s) of the metadevice(s) to be deleted.

hot_spare_pool ...

Specifies the name(s) of the hot spare pools to be deleted in the form hspnnn, where nnn is a number in the range 000-999.

EXAMPLES

This example deletes a metadevice named d10.

```
# metaclear /dev/md/dsk/d10
```

This example deletes all local metadevices and hot spare pools on the system.

```
# metaclear -a
```

This example deletes a mirror, d20, with an errored submirror.

```
# metaclear -f d20
```

This example deletes a hot spare pool, hsp001.

```
# metaclear hsp001
```

SEE ALSO

metadb(1M), metadetach(1M), metahs(1M), metainit(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metareplace(1M), metaroot(1M), metaset(1M), metastat(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

NAME

metadb - create and delete replicas of the metadevice state database

SYNOPSIS

```
metadb -h
metadb [ -s setname ]
metadb [ -s setname ] -a [ -f ] [ -k system-file ] mddbnnn
metadb [ -s setname ] -a [ -f ] [ -k system-file ]
       [ -c number ] [ -l length ] slice...
metadb [ -s setname ] -d [ -f ] [ -k system-file ] mddbnnn
metadb [ -s setname ] -d [ -f ] [ -k system-file ] slice...
metadb [ -s setname ] -i
metadb [ -s setname ] -p [ -k system-file ] [ mddb.cf-file ]
```

AVAILABILITY

/usr/opt/SUNWmd/sbin

DESCRIPTION

The metadb command creates and deletes replicas of the metadevice state database. State database replicas can be created on dedicated slices, or on slices that will later become part of a simple metadevice (concatenation or stripe), RAID5 metadevice, or trans metadevice.

The metadevice state database contains the configuration of all metadevices and hot spare pools in the system. Additionally, the metadevice state database keeps track of the current state of metadevices and hot spare pools, and their components. DiskSuite automatically updates the metadevice state database when a configuration or state change occurs. A submirror failure is an example of a state change. Creating a new metadevice is an example of a configuration change.

The metadevice state database is actually a collection of multiple, replicated database copies. Each copy, referred to as a replica, is subject to strict consistency checking to ensure correctness.

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

configuration data may be lost.

The majority consensus algorithm is conservative in the sense that it will fail if a majority consensus cannot be reached, even if one replica actually does contain the most up-to-date data. This approach guarantees that stale data will not be accidentally used, regardless of the failure

scenario. The majority consensus algorithm accounts for the following: the system will stay running with exactly half or more replicas; the system will panic when less than half the replicas are available; the system will not reboot without one more than half the total replicas.

When used with no options, the metadb command gives a short form of the status of the metadevice state database. Use metadb -i for an explanation of the flags field in the output.

The initial state database is created using the metadb command with both the -a and -f options, followed by the slice where the replica is to reside. The -a option specifies that a replica (in this case, the initial) state database should be created. The -f option forces the creation to occur, even though a state database does not exist. (The -a and -f options should be used together only when no state databases exist.)

Additional replicas beyond those initially created can be added to the system. They contain the same information as the existing replicas, and help to prevent the loss of the configuration information. Loss of the configuration makes operation of the metadevices impossible. To create additional replicas, use the metadb -a command, followed by the name of the new slice(s) where the replicas will reside. All replicas that are located on the same slice must be created at the same time.

To delete all replicas that are located on the same slice, the metadb -d command is used, followed by the slice name.

When used with the -i option, metadb displays the status of the metadevice state databases. The status can change if a hardware failure occurs or when state databases have been added or deleted.

To fix a replica in an errored state, delete the replica and add it back again.

OPTIONS

Root privileges are required for all of the following options except -h and -i.

The following options can be used with the metadb command. Not all the options are compatible on the same command line. Refer to the above synopsis line to see the supported use of the options.

-a Attach a new database device. The /etc/system file is automatically edited with the new information and the /etc/opt/SUNWmd/mddb.cf file is updated. An alternate way to create replicas is by defining them in the /etc/opt/SUNWmd/md.tab file and specifying the assigned name at the command line in the form, mddbnn, where nn is a two-digit number given to the replica definitions. Refer to the md.tab(4) man page for instructions on setting up replicas in that file.

-c number
Specifies the number of replicas to be placed on each device. The default number of replicas is 1.

-d Deletes all replicas that are located on the specified slice. The /etc/system file is automatically edited with the new information and the /etc/opt/SUNWmd/mddb.cf file is updated.

-f The -f option is used to create the initial state database. It is also used to force the deletion of replicas below the minimum of two. (The -a and -f options should be used together only when no state databases exist.)

-h Displays a usage message.

-i Inquire about the status of the replicas. The output of the -i option includes characters in front of the device name that represent the status of the state database. Explanations of the characters are printed following the replica status.

-k system-file
Specifies the name of the kernel file where the replica information should be patched. The default system-file is /etc/system. This option is for use with the local diskset only.

-l length
Specifies the size of each replica. The default length is 1034 blocks, which should be appropriate for most configurations.

-p Specifies patching the system file that is located in the current working directory (/system) with entries from the /etc/opt/SUNWmd/mddb.cf file. This option is normally used to patch a newly built system before it

is booted for the first time. If the system has been built on a system other than the one where it will run, the location of the mddb.cf on the local machine can be passed as an argument. The system file to be patched can be changed using the -k option. This option is for use with the local diskset only.

-s setname
Specifies the name of the diskset on which the metadb command will work. Using the -s option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on local database replicas.

slice
Specifies the logical name of the physical slice (partition), such as /dev/dsk/c0t0d0s2.

EXAMPLES
The following example creates the initial state database replicas on a new system.

```
# metadb -a -f c0t0d0s7 c0t1d0s2 c1t0d0s7 c1t1d0s2
```

The -a and -f options force the creation of the initial database and replicas. You could then create metadevices with these same slices, making efficient use of the system.

This example shows how to add two replicas on two new disks that have been connected to a system currently running DiskSuite.

```
# metadb -a c0t2d0s2 c1t1d0s2
```

This example shows how to delete two replicas from the system. Assume that replicas have been set up on /dev/dsk/c0t2d0s2 and /dev/dsk/c1t1d0s2.

```
# metadb -d c0t2d0s2 c1t1d0s2
```

Note: Although you can delete all replicas, you should never do so while metadevices still exist. Removing all replicas causes existing metadevices to become inoperable.

FILES

/etc/opt/SUNWmd/mddb.cf
Contains the location of each copy of

the metadevice state database.

/etc/opt/SUNWmd/md.tab
Workspace file for metadevice database configuration.

/etc/system Kernel patch file.

SEE ALSO

metaclear(1M), metadetach(1M), metahs(1M), metainit(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metareplace(1M), metaroot(1M), metaset(1M), metastat(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

NAME

metattach, metadetach - attach or detach metadevice to or from a mirror or trans

SYNOPSIS

```
metattach [ -h ]
metattach [ -s setname ] mirror [ metadevice ]
metattach [ -s setname ] [ -i size ] concat/stripe
          component...
metattach [ -s setname ] RAID component...
metattach [ -s setname ] trans log
metadetach [ -s setname ] [ -f ] mirror submirror
metadetach [ -s setname ] [ -f ] trans
```

AVAILABILITY

/usr/opt/SUNWmd/sbin

DESCRIPTION

metattach is used to add submirrors to a mirror, add logging devices to trans devices, or grow metadevices. Growing metadevices can be done without interrupting service. To grow the size of a mirror or trans, the slices must be added to the submirrors or to the master devices.

DiskSuite supports one-to-three-way mirrors. Thus, you can only attach a metadevice to a mirror if there are two or fewer submirrors beneath the mirror. Once a new metadevice is attached to a mirror, metattach will automatically start a resync operation to the new submirror.

Attaching a new logging device to a busy trans metadevice is

allowed, although a trans metadevice will start using its new logging device only after the trans is idle (after it is unmounted, for example). The busy trans will be in an Attaching state (metastat) until the logging device is actually attached. Attaching a logging device in the Hard Error or Error state (metastat) is not allowed.

metadetach is used to detach submirrors from mirrors, or detach logging devices from trans metadevices.

When a submirror is detached from a mirror, it is no longer part of the mirror, thus reads and writes to and from that metadevice via the mirror are no longer performed through the mirror. Detaching the only existing submirror is not allowed. Detaching a submirror that has slices reported as needing maintenance (by metastat) is not allowed unless the -f (force) flag is used.

metadetach also detaches the logging device from a trans. Once detached, the logging device is no longer part of the trans, thus the trans is no longer logging and all benefits of logging are lost. Any information on the logging device that pertains to the master device is written to the master device before the logging device is detached.

Detaching the logging device from a busy trans device is not allowed unless the -f (force) flag is used. Even so, the logging device is not actually detached until the trans is idle. The trans is in the Detaching state (metastat) until the logging device is detached.

OPTIONS

Root privileges are required for all of the following options except -h.

-f Force the detaching of metadevices that have components that need maintenance or are busy.

-h Displays a usage message.

-i size
Specifies the interlace value for stripes, where size

is a specified value followed by either `k' for kilobytes, `m' for megabytes, or `b' for blocks. The units can be either upper case or lower case. If size is not specified, the size defaults to the interlace size of the last stripe of the metadevice. When an interlace size change is made on a stripe, it will be carried forward on all stripes that follow.

-s setname
Specifies the name of the diskset on which the metattach command or the metadetach command will work. Using the -s option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on local metadevices.

mirror
Specifies the mirror.

metadevice
Specifies the name of the metadevice to be attached to the mirror as a submirror. This metadevice must have been previously created by the metainit command.

concat/stripe
Specifies the metadevice name of the concatenation, stripe, or concatenation of stripes.

component...
The logical name for the physical slice (partition) on a disk drive, such as /dev/dsk/c0t0d0s2, being added to the concatenation, stripe, concatenation of stripes, or RAID5 metadevice.

RAID Specifies the metadevice name of the RAID5 metadevice.

trans
Specifies the metadevice name of the trans metadevice (not the master or logging device).

log Specifies the metadevice name of the logging device to be attached to the trans metadevice.

submirror
The metadevice name of the submirror to be detached

from the mirror.

EXAMPLES

This example concatenates a single new slice to an existing metadevice, d8. (Afterwards, you would use the growfs command to expand the file system.)

```
# metattach d8 /dev/dsk/c0t1d0s2
```

This example adds four slices to an existing metadevice, d9. (Afterwards, you would use the growfs command to expand the file system.)

```
# metattach d9 /dev/dsk/c0t1d0s2 /dev/dsk/c0t2d0s2 \
     /dev/dsk/c0t3d0s2 /dev/dsk/c0t4d0s2
```

This example detaches the logging device from a trans metadevice d9. Notice that you do not have to specify the logging device itself, as there can only be one.

```
# metadetach d9
```

This example expands a RAID5 metadevice, d45, by attaching another slice.

```
# metattach d45 /dev/dsk/c3t0d0s2
```

When you add additional slices to a RAID5 metadevice, the additional space is devoted to data. No new parity blocks are allocated. The data on the added slices is, however, included in the overall parity calculations, so it is protected against single device failure.

This example adds space to a two-way mirror by adding a

slice to each submirror. (Afterwards, you would use the growfs command to expand the file system.)

```
# metattach d9 /dev/dsk/c0t2d0s5
# metattach d10 /dev/dsk/c0t3d0s5
```

This example detaches a submirror, d2, from a mirror, d4.

```
# metadetach d4 d2
```

SEE ALSO

metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metareplace(1M), metaroot(1M), metaset(1M), metastat(1M), metasync(1M), md.tab(4), md.cf(4), mddb.cf(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

WARNING

When a submirror is detached from its mirror, the data on the metadevice may not be the same as the data that existed on the mirror prior to running metadetach. In particular, if the -f option was needed, the metadevice and mirror probably do not contain the same data.

NAME

metahs - manage hot spares and hot spare pools

SYNOPSIS

```
metahs [ -s setname ] -a all component
metahs [ -s setname ] -a hot_spare_pool [ component ]
metahs [ -s setname ] -d hot_spare_pool [ component ]
metahs [ -s setname ] -d all component
metahs [ -s setname ] -e component
metahs [ -s setname ] -r hot_spare_pool component-old
metahs [ -s setname ] -r all component-old component-new
metahs [ -s setname ] -i [ hot_spare_pool... ]
```

AVAILABILITY

/usr/opt/SUNWmd/sbin

DESCRIPTION

The metahs command manages existing hot spares and hot spare pools. It is used to add, delete, enable, and replace components (slices) in hot spare pools. Like the metainit command, the metahs command can also create an initial hot spare pool. The metahs command does not replace a component of a metadevice. This function is performed by the metareplace command.

Hot spares are always in one of three states: available, in-use, or broken. Available hot spares are running and ready to accept data, but are not currently being written to or read from. In-use hot spares are currently being written to and read from. Broken hot spares are out of service and should be repaired. The status of hot spares is displayed when metahs is invoked with the -i option.

OPTIONS

Root privileges are required for any of the following options except -i.

-a all component
: Adds component to all hot spare pools. all is not case sensitive.

-a hot_spare_pool [component]
: Adds the component to the specified hot_spare_pool. hot_spare_pool is created if it does not already exist.

-d all component
: Deletes component from all the hot spare pools. The component cannot be deleted if it is in the in-use state.

-d hot_spare_pool [component]
Deletes hot_spare_pool, if the hot_spare_pool is both empty and not referenced by a metadevice. If component is specified, it is deleted from the hot_spare_pool. Hot spares in the in-use state cannot be deleted.

-e component
Enables component to be available for use as a hot spare. The component can be enabled if it is in the broken state and has been repaired.

-i [hot_spare_pool ...]
Displays the status of the specified hot_spare_pool or for all hot spare pools if one is not specified.

-r all component-old component-new
Replace component-old with component-new in all hot spare pools which have the component associated. Components cannot be replaced from any hot spare pool if the old hot spare is in the in-use state.

-r hot_spare_pool component-old component-new
Replaces component-old with component-new in the specified hot_spare_pool. Components cannot be replaced from a hot spare pool if the old hot spare is in the in-use state.

-s setname
Specifies the name of the diskset on which metahs will work. Using the -s option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on local hot spare pools.

component
The logical name for the physical slice (partition) on a disk drive, such as /dev/dsk/c0t0d0s2.

hot_spare_pool
Hot spare pools must be of the form hspnnn, where nnn is a number in the range 000-999.

EXAMPLES
The following example adds a hot spare /dev/dsk/c0t0d0s7 to a hot spare pool hsp003.

```
# metahs -a hsp003 c0t0d0s7
```

When the hot spare is added to the pool, the existing order of the hot spares already in the pool is preserved. The new

hot spare is added at the end of the list of hot spares in

the hot spare pool specified.

This example adds a hot spare to the hot spare pools that are currently defined.

```
# metahs -a all c0t0d0s7
```

The keyword all in this example specifies adding the hot spare, /dev/dsk/c0t0d0s7, to all the hot spare pools.

This example deletes a hot spare, /dev/dsk/c0t0d0s7, from a hot spare pool, hsp003.

```
# metahs -d hsp003 c0t0d0s7
```

When you delete a hot spare, the position of the remaining hot spares in the pool changes to reflect the new order. For instance, if in this example /dev/dsk/c0t0d0s7 were the second of three hot spares, after deletion the third hot spare would move to the second position.

This example replaces a hot spare that was previously defined.

```
# metahs -r hsp001 c0t1d0s0 c0t3d0s0
```

In this example, the hot spare /dev/dsk/c0t1d0s0 is replaced by /dev/dsk/c0t3d0s0. The order of the hot spares does not change.

SEE ALSO

metaclear(1M), metadb(1M), metadetach(1M), metainit(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metareplace(1M), metaroot(1M), metaset(1M), metastat(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

NAME

metainit - configure metadevices

SYNOPSIS

```
metainit -h
metainit [ generic options ] concat/stripe
         numstripes width component... [ -i interlace ]
         [ width component... [ -i interlace ] ]
         [ -h hot_spare_pool ]
metainit [ generic options ] mirror -m submirror
         [ read_options ] [ write_options ] [ pass_num ]
metainit [ generic options ] RAID -r component...
         [ -i interlace ] [ -h hot_spare_pool ]
         [ -k ] [ -o original_column_count ]
metainit [ generic options ] trans -t master [ log ]
metainit [ generic options ] hot_spare_pool [ hotspare... ]
metainit [ generic options ] metadevice-name
metainit [ generic options ] -a
metainit -r
```

AVAILABILITY

/usr/opt/SUNWmd/sbin

DESCRIPTION

The metainit command configures metadevices and hot spares according to the information specified on the command line. Or, you can run metainit so that it uses configuration entries you specify in the /etc/opt/SUNWmd/md.tab file. All metadevices must be set up by the metainit command before they can be used.

If you edit the /etc/opt/SUNWmd/md.tab file to configure metadevices, specify one complete configuration entry per

line. You then run the metainit command with either the -a option, to activate all metadevices you entered in the /etc/opt/SUNWmd/md.tab file, or with the metadevice name corresponding to a specific configuration entry.

Note: DiskSuite never updates the /etc/opt/SUNWmd/md.tab file. Complete configuration information is stored in the metadevice state database, not md.tab. The only way information appears in md.tab is through editing it by hand.

GENERIC OPTIONS

Root privileges are required for all of the following options except -h.

-f Forces the metainit command to continue even if one of the slices contains a mounted file system or is being

used as swap. This option is useful when configuring mirrors on root (/), swap, and /usr.

-h Displays usage message.

-n Checks the syntax of your command line or /etc/opt/SUNWmd/md.tab entry without actually setting up the metadevice. If used with -a, all devices are checked but not initialized.

-r Only used in a shell script at boot time. Sets up all metadevices that were configured before the system crashed or was shut down. The information about previously configured metadevices is stored in the metadevice state database (see metadb).

-s setname
Specifies the name of the diskset on which metainit will work. Without the -s option, the metainit command operates on your local metadevices and/or hotspares.

CONCAT/STRIPE OPTIONS

concat/stripe
Specifies the metadevice name of the concatenation, stripe, or concatenation of stripes being defined.

numstripes

Specifies the number of individual stripes in the metadevice. For a simple stripe, numstripes is always 1. For a concatenation, numstripes is equal to the number of slices. For a concatenation of stripes, numstripes will vary according to the number of stripes.

width
Specifies the number of slices that make up a stripe. When width is greater than 1, the slices are striped.

component
The logical name for the physical slice (partition) on a disk drive, such as /dev/dsk/c0t0d0s2. For RAID5 metadevices, a minimum of three slices is necessary to enable striping of the parity information across slices.

-i interlace
Specifies the interlace size. This value tells DiskSuite how much data to place on a slice of a striped or RAID5 metadevice before moving on to the next slice. interlace is a specified value, followed by either `k' for kilobytes, `m' for megabytes, or `b' for blocks. The characters can be either uppercase or lowercase. The interlace specified cannot be less than 16 blocks,

or greater than 100 megabytes. If interlace is not specified, it defaults to 16 kilobytes.

-h hot_spare_pool
Specifies the hot_spare_pool to be associated with the metadevice. If you use the command line, the hot spare pool must have been previously created by the metainit command before it can be associated with a metadevice. The hot_spare_pool must be of the form hspnnn, where nnn is a number in the range 000-999. Use /-h hspnnn when the concat/stripe being created is to be used as a submirror.

MIRROR OPTIONS
mirror -m submirror
Specifies the metadevice name of the mirror. The -m specifies that the configuration is a mirror. submirror is a metadevice (stripe or concatentation) that makes up the initial one-way mirror. DiskSuite supports a

maximum of three-way mirroring. When defining mirrors, first create the mirror with the metainit command as a one-way mirror. Then attach subsequent submirrors using the metattach command. This method ensures that DiskSuite properly syncs the mirrors. (The second and third submirrors are first created via the metainit command.)

read_options
The following read options for mirrors are available:

-g Enables the geometric read option, which results in faster performance on sequential reads.

-r Directs all reads to the first submirror. This should only be used when the devices comprising the first submirror are substantially faster than those of the second mirror. This flag cannot be used with the -g flag.

If neither the -g nor -r flags are specified, reads are made in a round-robin order from all submirrors in the mirror. This enables load balancing across the submirrors.

write_options
The following write options for mirrors are available:

-S Performs serial writes to mirrors. The first submirror write completes before the second is started. This may be useful if hardware is susceptible to partial sector failures. If -S is not specified, writes are replicated and dispatched to all mirrors simultaneously.

pass_num
A number in the range 0-9 at the end of an entry defining a mirror that determines the order in which that mirror is resynced during a reboot. The default is 1. Smaller pass numbers are resynced first. Equal pass numbers are run concurrently. If 0 is used, the resync is skipped. 0 should be used only for mirrors mounted as read-only, or as swap.

RAID5 OPTIONS
RAID -r

Specifies the name of the RAID5 metadevice. The -r specifies that the configuration is RAID5.

-k For RAID5 metadevices, informs the driver that it is not to initialize (zero the disk blocks) due to existing data. Only use this option to recreate a previously created RAID5 device.

-o original_column_count
For RAID5 metadevices, used with the -k option to define the number of original slices in the event the originally defined metadevice was grown. This is necessary since the parity segments are not striped across concatenated devices.

WARNING For -k and -o
Use extreme caution when using the -k and -o options. When used, these options set the disk blocks to the OK state. If any errors exist on disk blocks within the metadevice, DiskSuite may begin fabricating data. Instead of using these options, you may want to initialize the device and restore data from tape.

TRANS OPTIONS

trans -t master [log]
trans specifies the name of the trans metadevice, which consists of master and log devices, or just a master device. The -t specifies that the configuration is a trans metadevice. If log is not specified when you create the trans metadevice, no logging can take place until a logging device is provided by using the metattach command. master and log can be simple, mirror, or RAID5 metadevices. They cannot be trans metadevices. master should be a UFS file system. You can configure an existing file system for logging by creating a trans metadevice as follows: make the existing file system into the master trans device, then create the log device on a separate, unused slice. The minimum log size is 1 Mbyte of disk space. Under heavy sustained loads, small logs will detract from performance because old data must be copied from the log to the file system

before new data can be logged. The maximum log size is 1 Gbyte. Large logs might increase performance. However, logs larger than 64 Mbytes can have negligible

performance benefits.

HOT SPARE POOL OPTIONS

hot_spare_pool [hotspare...]

When used as arguments to the metainit command, hot_spare_pool defines the name for a hot spare pool, and hotspare... is the logical name for the physical slice(s) for availability in that pool. hot_spare_pool is a number of the form hspnnn, where nnn is a number in the range 000-999.

md.tab FILE OPTIONS

metadevice-name

When the metainit command is run with a metadevice-name as its only argument, it searches the /etc/opt/SUNWmd/md.tab file to find that name and its corresponding entry. The order in which entries appear in the md.tab file is unimportant. For example, consider the following md.tab entry:

d0 2 1 c1t0d0s0 1 c2t1d0s0

When you run the command metainit d0, it configures metadevice d0 based on the configuration information found in the md.tab file.

-a Activates all metadevices defined in the /etc/opt/SUNWmd/md.tab file.

EXAMPLES

Examples listed here include creation of a concatenation, a stripe, a concatenation of stripes, a mirror, a trans metadevice, a RAID5 metadevice, and a hot spare pool. All drives in the following examples have the same size of 525 Mbytes.

Concatenation

This example shows a metadevice, /dev/md/dsk/d7, consisting of a concatenation of four slices.

metainit d7 4 1 c0t1d0s0 1 c0t2d0s0 1 c0t3d0s0 1 /dev/dsk/c0t4d0s0

The number 4 indicates there are four individual stripes in the concatenation. Each stripe is made of one slice, hence the number 1 appears in front of each slice.

Note: The first disk sector in all of the above devices

contains a disk label. To preserve the labels on devices /dev/dsk/c0t2d0s0, /dev/dsk/c0t3d0s0, and /dev/dsk/c0t4d0s0, the metadisk driver must skip at least the first sector of those disks when mapping accesses across the concatenation boundaries. Because skipping only the first sector would create an irregular disk geometry, the entire first cylinder of these disks will be skipped. This allows higher level file system software to optimize block allocations correctly.

Stripe

This example shows a metadevice, /dev/md/dsk/d15, consisting of two slices.

```
# metainit d15 1 2 c0t1d0s2 c0t2d0s2 -i 32k
```

The number 1 indicates that one stripe is being created. Because the stripe is made of two slices, the number 2 follows next. The optional -i followed by 32k specifies the interlace size will be 32 Kbytes. If the interlace size were not specified, the stripe would use the default value of 16 Kbytes.

Concatentation of Stripes

This example shows a metadevice, /dev/md/dsk/d75, consisting of a concatenation of two stripes of three disks.

```
# metainit d75 2 3 c0t1d0s2 c0t2d0s2 \
        c0t3d0s2 -i 16k \
      3 c1t1d0s2 c1t2d0s2 c1t3d0s2 -i 32k
```

On the first line, the -i followed by 16k specifies that the stripe interlace size is 16 Kbytes. The second set specifies the stripe interlace size will be 32 Kbytes. If the second set did not specify 32 Kbytes, the set would use the default interlace value of 16 Kbytes. The blocks of each set of three disks are interlaced across three disks.

Mirroring

This example shows a two-way mirror, /dev/md/dsk/d50, consisting of two submirrors. This mirror does not contain any existing data.

```
# metainit d51 1 1 c0t1d0s2
# metainit d52 1 1 c0t2d0s2
# metainit d50 -m d51
# metattach d50 d52
```

In this example, two submirrors, d51 and d52, are created with the metainit command. These two submirrors are simple

concatenations. Next, a one-way mirror, d50,
is created using the -m option wtih d51. The second submirror is attached later using the metattach command. When creating a mirror, any combination of stripes and concatenations can be used. The default read and write options in this example are a round-robin read algorithm and parallel writes to all submirrors.

Logging (trans)

This example shows trans metadevice, /dev/md/dsk/d1, with mirrors for the master and logging devices. This trans does not contain any existing data.

```
# metainit d11 1 1 c0t1d0s2
# metainit d12 1 1 c0t2d0s2
# metainit d21 1 1 c1t1d0s3
# metainit d22 1 1 c1t2d0s3
# metainit d10 -m d11
# metattach d10 d12
# metainit d20 -m d21
# metattach d20 d22
# metainit d1 -t d10 d20
```

This example begins by defining four concatenations, d11, d12, d21, and d22. Next, mirror d10 is defined, followed by mirror d20. The mirrors are initially defined as one-way mirrors, then the second submirrors are attached later with the metattach command. Finally, the trans metadevice d1 is defined, with d10 as the master device and d20 as the logging device by using the -t option.

RAID5

This example shows a RAID5 device, d80, consisting of three slices:

```
# metainit d80 -r c1t0d0s2 c1t1d0s2 c1t3d0s2 -i 20k
```

In this example, a RAID5 metadevice is defined using the -r option with an interlace size of 20 Kbytes. The data and parity segments will be striped across the slices, c1t0d0s2, c1t2d0s2, and c1t3d0s2.

Hot Spare

This example shows a two-way mirror, /dev/md/dsk/d10, and a hot spare pool with three hot spare components. The mirror does not contain any existing data.

```
# metainit hsp001 c2t2d0s2 c3t2d0s2 c1t2d0s2
# metainit d41 1 1 c1t0d0s2 -h hsp001
# metainit d42 1 1 c3t0d0s2 -h hsp001
# metainit d40 -m d41
# metattach d40 d42
```

In the above example, a hot spare pool, hsp001, is created with three disks used as hot spares. Next, two submirrors are created, d41 and d42. These are simple concatenations. The metainit command uses the -h option to associate the hot spare pool hsp001 with each submirror. A one-way mirror is then defined using the -m option. The second submirror is attached using the metattach command.

FILES

/etc/opt/SUNWmd/md.tab
: Contains list of metadevice and hot spare configurations for batch-like creation.

WARNING

Do not use the metainit command to create a multi-way mirror. Rather, create a one-way mirror with metainit then attach additional submirrors with metattach. When the metattach command is not used, no resync operations occur and data could become corrupted.

If you use metainit to create a mirror with multiple submirrors, the following message is displayed:

WARNING: This form of metainit is not recommended.
The submirrors may not have the same data.
Please see ERRORS in metainit(1M) for additional information.

SEE ALSO

metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metareplace(1M), metaroot(1M), metaset(1M), metastat(1M), metasync(1M), metattach(1M), md.cf(4), md.tab(4), mddb.cf(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

LIMITATIONS

Recursive mirroring is not allowed; that is, a mirror cannot appear in the definition of another mirror.

Recursive logging is not allowed; that is, a trans metadevice cannot appear in the definition of another metadevice.

Stripes and RAID5 metadevices must consist of slices only.

Mirroring of RAID5 metadevices is not allowed.

NAME

metaoffline, metaonline - place submirrors offline and online

SYNOPSIS

```
metaoffline -h
metaoffline [ -s setname ] [ -f ] mirror submirror
metaonline -h
metaonline [ -s setname ] mirror submirror
```

AVAILABILITY

/usr/opt/SUNWmd/sbin

DESCRIPTION

The metaoffline command prevents DiskSuite from reading and writing to the submirror that has been taken offline. While the submirror is offline, all writes to the mirror will be kept track of (by region) and will be written when the sub-

mirror is brought back online. The metaoffline command can also be used to perform online backups: one submirror is taken offline and backed up while the mirror remains accessible. (However, if this is a two-way mirror, data redundancy is lost while one submirror is offline.) The metaoffline command differs from the metadetach command because it does not sever the logical association between the submirror and the mirror. To completely remove a submirror from a mirror, use the metadetach command.

A submirror that has been taken offline will only remain offline until the metaonline command is invoked or the system is rebooted.

When the metaonline command is used, reading from and writing to the submirror resumes. A resync is automatically invoked to resync the regions written while the submirror was offline. Writes are directed to the submirror during resync. Reads, however, will come from a different submirror. Once the resync operation completes, reads and writes are performed on that submirror. The metaonline command is only effective on a submirror of a mirror that has been taken offline.

Note: A submirror that has been taken offline with the metaoffline command can only be mounted as read-only.

OPTIONS

Root privileges are required for all of the following options except -h.

-f Forces offlining of submirrors that have slices requiring maintenance.

-h Displays usage message.

-s setname
Specifies the name of the diskset on which metaoffline and metaonline will work. Using the -s option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on local metadevices.

mirror
Specifies the metadevice name of the mirror from which the submirror will be either taken offline or put online.

submirror
Specifies the metadevice name of the submirror to be either taken offline or put online.

EXAMPLES

This example takes one submirror, d9, offline from mirror d10.

```
# metaoffline d10 d9
```

SEE ALSO

metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1M), metaparam(1M), metareplace(1M), metaroot(1M), metaset(1M), metastat(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

NAME

metaoffline, metaonline - place submirrors offline and online

SYNOPSIS

```
metaoffline -h
metaoffline [ -s setname ] [ -f ] mirror submirror
metaonline -h
metaonline [ -s setname ] mirror submirror
```

AVAILABILITY

/usr/opt/SUNWmd/sbin

DESCRIPTION

The metaoffline command prevents DiskSuite from reading and writing to the submirror that has been taken offline. While the submirror is offline, all writes to the mirror will be kept track of (by region) and will be written when the submirror is brought back online. The metaoffline command can also be used to perform online backups: one submirror is taken offline and backed up while the mirror remains accessible. (However, if this is a two-way mirror, data redundancy is lost while one submirror is offline.) The metaoffline command differs from the metadetach command because it does not sever the logical association between the submirror and the mirror. To completely remove a submirror from a mirror, use the metadetach command.

A submirror that has been taken offline will only remain offline until the metaonline command is invoked or the system is rebooted.

When the metaonline command is used, reading from and writing to the submirror resumes. A resync is automatically invoked to resync the regions written while the submirror was offline. Writes are directed to the submirror during resync. Reads, however, will come from a different submirror. Once the resync operation completes, reads and writes are performed on that submirror. The metaonline command is only effective on a submirror of a mirror that has been taken offline.

Note: A submirror that has been taken offline with the metaoffline command can only be mounted as read-only.

OPTIONS

Root privileges are required for all of the following options except -h.

-f Forces offlining of submirrors that have slices requiring maintenance.

-h Displays usage message.

-s setname
Specifies the name of the diskset on which metaoffline and metaonline will work. Using the -s option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on local metadevices.

mirror
Specifies the metadevice name of the mirror from which the submirror will be either taken offline or put online.

submirror
Specifies the metadevice name of the submirror to be either taken offline or put online.

EXAMPLES

This example takes one submirror, d9, offline from mirror d10.

```
# metaoffline d10 d9
```

SEE ALSO

metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1M), metaparam(1M), metareplace(1M), metaroot(1M), metaset(1M), metastat(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

NAME

metaparam - modify parameters of metadevices

SYNOPSIS

```
metaparam -h
metaparam [ -s setname ] [ concat/stripe or RAID5 options ]
          concat/stripe | RAID
metaparam [ -s setname ] [ mirror options ] mirror
```

AVAILABILITY

/usr/opt/SUNWmd/sbin

DESCRIPTION

The metaparam command is used to display or modify current parameters of metadevices. The current parameters can be displayed by the metastat command.

If just the metadevice is specified as an argument to the metaparam command, the current settings are displayed.

The metaparam command enables metadevice parameters to be changed with the exception of the interlace value, which initially must be set using the metainit command.

OPTIONS

Root privileges are required for all of the following options except -h.

-h Displays usage message.

-s setname
Specifies the name of the diskset on which metaparam will work. Using the -s option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on local metadevices.

CONCAT/STRIPE OR RAID5 OPTIONS

-h hot_spare_pool | none
Specifies the hot spare pool to be used by a metadevice. If none is specified, the metadevice is disassociated with the hot spare pool assigned to it. If the metadevice is currently using a hot spare, then meta-

param cannot replace the hot spare pool.

concat/stripe | RAID
Specifies the metadevice name of the concatenation, stripe, or concatenation of stripes, or of the RAID5 metadevice.

MIRROR OPTIONS

-r roundrobin | geometric | first
Modifies th read option for a mirror. The -r option must be followed by either roundrobin, geometric, or first. roundrobin, which is the default action under the metainit command, specifies reading the disks in a round-robin (load balancing) method. geometric allows for faster performance on sequential reads. first specifies reading only from the first submirror.

-w parallel | serial
Modifies the write option for a mirror. The -w option must be followed by either parallel or serial. parallel, the default action under the metainit command, specifies that all writes are parallel. serial specifies that all writes are serial.

-p pass_number
A number from 0-to-9 that specifies the order in which a mirror is resynced during reboot. The default is 1. Smaller pass numbers are resynced first. Equal pass numbers are run concurrently. If 0 is used, the mirror resync is skipped. 0 should only be used for mirrors mounted as read-only, or as swap.

mirror
Specifies the metadevice name of the mirror.

EXAMPLES

This example associates a hot spare pool, hsp005, with a RAID5 metadevice, d80.

```
# metaparam -h hsp005 d80
```

This example changes the read option on a mirror d50 from the default of roundrobin to geometric.

```
     # metaparam -r geometric d50
```

SEE ALSO

metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1M), metaoffline(1M), metaonline(1M), metareplace(1M), metaroot(1M), metaset(1M), metastat(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

NAME

metarename - rename metadevice or switch layered metadevice names

SYNOPSIS

```
metarename [ -s setname ] metadevice1 metadevice2
metarename [ -s setname ] [ -f ] -x metadevice1 metadevice2
metarename -h
```

AVAILABILITY

/usr/opt/SUNWmd/sbin

DESCRIPTION

There are two ways to use metarename. The first renames an existing metadevice to a new name. This makes managing the metadevice namespace easier. The metadevice being renamed cannot be mounted or open, nor can the new name already exist. For example, to rename a metadevice that contains a mounted file system, you would first need to unmount the file system.

Secondly, when used with the -x option, metarename switches (exchanges) the names of an existing layered metadevice and one of its subdevices. (In DiskSuite terms, a layered metadevice can be either a mirror or a trans metadevice.) The -x option enables you to switch the metadevice names of a mirror and one of its submirrors, or a trans metadevice and its master device.

metarename -x makes it easier to mirror or unmirror an

existing stripe or concatenation, and to create or remove a trans of an existing metadevice.

When used to mirror an existing stripe or concatenatation, you must stop access to the device. For example, if the device contains a mounted file system, you must first unmount the file system before doing the rename.

metarename -x can also be used to create a trans metadevice from an existing metadevice, or to untrans the device. This applies only to the master device. A logging device cannot be created or removed via metarename. Before you can rename a trans device, you must detach the logging device. Then you must stop access to the trans metadevice itself.

You cannot rename or switch metadevices that are in an errored state or that have subcomponents in an errored state, or metadevices actively using a hot spare replacement.

You can only switch metadevices that have a direct child/parent 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 flag when switching members of a trans metadevice.

Only metadevices can be switched, not slices.

OPTIONS

Force the switching of trans metadevice members.

-h Display a help message.

-s setname
Specifies the name of the diskset on which metarename will work. Using the -s option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on the local metadevices.

-x Exchange the metadevice names metadevice1 and metadev-

ice2.

metadevice1
Specifies the metadevice to be renamed or switched.

metadevice2
Specifies the target metadevice name for the rename or switch operation.

EXAMPLES

This example renames a metadevice named d10 to d100. Note that d100 must not exist for the rename to succeed.

```
# metarename d10 d100
```

This example creates a two-way mirror from an existing stripe named d1 with a mounted file system, /home2.

```
# metainit d2 1 1 c13d0s1
# metainit -f d20 -m d1
# umount /home2
# metarename -x d20 d1
# metattach d1 d2
# mount /home2
```

First, a second concatenation d2, is created. (d1 already exists.) The metainit command creates a one-way mirror, d20, from d1. Next, you umount the file system and switch d1 for d20, making d1 the top-level device (mirror). You attach

the second submirror, d2, to create a two-way mirror. Lastly, you remount the file system.

This example takes an existing mirror named d1 with a mounted file system, and ends up with the file system mounted on a stripe d1.

```
# umount /fs2
# metarename -x d1 d20
# metadetach d20 d1
# metaclear -r d20
# mount /fs2
```

First, you unmount the file system, then switch the mirror

d1 and its submirror d20. This makes the mirror into d20. Next, you detach d1 from d20, then delete the mirror d20 and its other submirror. You then remount the file system.

This example creates a trans metadevice from an existing RAID5 metadevice named d1 which contains the file system /myhome.

```
# umount /myhome
# metainit d21 -t d1
# metarename -f -x d21 d1
# metattach d1 d0
# mount /myhome
```

You umount the file system before using the metainit command to create the trans metadevice d21, with d1 as the master device. You then switch d21 and d1, making d1 the top-level metadevice (trans metadevice). A logging device d0 is attached with the metattach command. You then remount the file system.

This example deletes a trans metadevice named d10 while its mount point is /myhome. The master device, which is a stripe, is named d2. The logging device, also a stripe, is named d5.

```
# umount /myhome
# metadetach d10
# metarename -f -x d10 d2
# metaclear d2
# metaclear d5
# fsck /dev/md/dsk/d10
# mount /myhome
```

You umount the file system first, then detach the trans metadevice's logging device. The trans metadevice is switched with the master device, making the trans metadevice d2 and the underlying stripe d10. You clear the trans

metadevice d2 and the logging device d5. d10 must be fsck'd, and then the file system is remounted.

SEE ALSO

metaclear(1M), metainit(1M), metastat(1M)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

LIMITATIONS

Renaming and exchanging metadevice names can only be used for metadevices. A physical slice cannot be renamed to a metadevice, nor can a metadevice be exchanged with a physical slice name.

Metadevice names are (still) limited to strings of the pattern d<xyz> where xyz is a value between 0 and 1024 . You cannot use logical names for metadevices.

NAME

metareplace - enable or replace components of submirrors or RAID5 metadevices

SYNOPSIS

```
metareplace -h
metareplace [ -s setname ] -e mirror component
metareplace [ -s setname ] mirror
            component-old component-new
metareplace [ -s setname ] -e RAID component
metareplace [ -s setname ] [ -f ] RAID
            component-old component-new
```

AVAILABILITY

/usr/opt/SUNWmd/sbin

DESCRIPTION

The metareplace command is used to enable or replace components (slices) within a submirror or a RAID5 metadevice.

When you replace a component, the metareplace command automatically starts resyncing the new component with the rest of the metadevice. When the resync completes, the replaced component becomes readable and writeable. If the failed component has been hot spare replaced, the hot spare is placed in the available state and made available for other hot spare replacements.

Note that the new component must be large enough to replace the old component.

A component may be in one of several states. The Last Erred and the Maintenance states require action. Always replace components in the Maintenance state first, followed by a resync and validation of data. After components requiring maintenance are fixed, validated, and resynced, components in the Last Erred state should be replaced. To avoid data loss, it is always best to back up all data before replacing Last Erred devices.

OPTIONS

Root privileges are required for all of the following options except -h.

-e Transitions the state of component to the available state and resyncs the failed component. If the failed component has been hot spare replaced, the hot spare is placed in the available state and made available for other hot spare replacements. This command is useful

when a component fails due to human error (for example, accidentally turning off a disk), or because the component was physically replaced. In this case, the replacement component must be partitioned to match the disk being replaced before running the metareplace command.

-f Forces the replacement of an errored component of a metadevice in which multiple components are in error. The component determined by the metastat display to be in the ``Maintenance'' state must be replaced first. This option may cause data to be fabricated since multiple components are in error.

-h Display help message.

-s setname
Specifies the name of the diskset on which metareplace will work. Using the -s option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on local metadevices.

mirror
The metadevice name of the mirror.

component
The logical name for the physical slice (partition) on a disk drive, such as /dev/dsk/c0t0d0s2.

component-old
The physical slice that is being replaced.

component-new
The physical slice that is replacing component-old.

RAID The metadevice name of the RAID5 device.

EXAMPLES
This example shows how to recover when a single component in a RAID5 metadevice is errored.

```
# metareplace d10 c3t0d0s2 c5t0d0s2
```

In this example, a RAID5 metadevice d10 has an errored component, c3t0d0s2, replaced by a new component, c5t0d0s2.

This example shows the use of the -e option after a physical

disk in a submirror has been replaced.

```
# metareplace -e d11 c1t4d0s2
```


Note: The replacement disk must be partitioned to match the disk it is replacing before running the metareplace command.

SEE ALSO

metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metaroot(1M), metaset(1M), metastat(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

NAME

metaroot - setup system files for root (/) metadevice

SYNOPSIS

```
metaroot -h
metaroot [ -n ] [ -k system-name ] [ -v vfstab-name ]
         [ -c mddb.cf-name ] device
```

AVAILABILITY

/usr/opt/SUNWmd/sbin

DESCRIPTION

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.

If necessary, the metaroot command can reset a system that has been configured to boot the root file system (/) on a metadevice so that it uses a physical slice.

OPTIONS

Root privileges are required for all of the following options except -h.

-c mddb.cf-name

Uses mddb.cf-name instead of the default /etc/opt/SUNWmd/mddb.cf file as a source of metadevice database locations.

-h Displays a usage message.

-k system-name
Edits a user-supplied system-name instead of the default /etc/system system configuration information file.

-n Print what would be done without actually doing it.

-v vfstab-name
Edits vfstab-name instead of the default /etc/vfstab table of file system defaults.

device
Specifies either the metadevice or the conventional disk device (slice) used for the root file system (/).

EXAMPLES

The following command edits /etc/system and /etc/vfstab to specify that the root filesystem is now on metadevice d0.

```
# metaroot d0
```

The following command edits /etc/system and /etc/vfstab to specify that the root filesystem is now on the SCSI disk device /dev/dsk/c0t3d0s0.

```
# metaroot /dev/dsk/c0t3d0s0
```

FILES

/etc/system Kernel patch file.

/etc/vfstab File system defaults.

/etc/opt/SUNWmd/mddb.cf
Metadevice state database locations.

NOTES

WARNING: forceload of misc/md_hotspares failed may appear during boot if root is on a metadevice and no hot spares are specified. This can be eliminated by defining an empty hot

spare pool.

WARNING: forceload of misc/md_trans failed may appear if no trans devices have been configured.

WARNING: forceload of misc/md_raid failed may appear if no RAID5 devices have been configured.

You can safely ignore these messages. This is an artifact of the way drivers are loaded during the boot process.

SEE ALSO

metadb(1M), metainit(1M), metastat(1M), mddb.cf(4), system(4), vfstab(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

NAME

metaset - configure shared disksets

SYNOPSIS

```
metaset -s setname -a -h hostname...
metaset -s setname -a [ -l length ] drivename...
metaset -s setname -d [ -f ] -h hostname...
metaset -s setname -d [ -f ] drivename...
metaset -s setname -r
metaset -s setname -t [ -f ]
metaset -s setname -b
metaset -s setname -o [ -h hostname ]
metaset [ -s setname ]
```

AVAILABILITY

/usr/opt/SUNWmd/sbin

DESCRIPTION

In a diskset configuration, two hosts are physically connected to the same set of disks. When one host fails, the other host has exclusive access to the disks. The metaset command administers sets of disks shared for exclusive (but not concurrent) access between such hosts. While disksets enable a high-availability configuration, DiskSuite itself does not actually provide a high-availability environment.

Shared metadevices/hot spare pools can be created only from drives which are in the diskset created by metaset. To create a set, one or more hosts must be added to the set. To create metadevices within the set, one or more devices must be added to the set. The drivename specified must be in the form cxtxdx with no slice specified.

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 after repartitioning. After adding a drive to a diskset, you can repartition the drive as necessary. However, Slice 7 should not be moved, removed, or overlapped with any other partition.

After a diskset is created and metadevices are set up within the set, the metadevice name will be in the following form:

/dev/md/setname/{dsk,rdsk}/dnumber

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

Hot spare pools within local disksets use standard DiskSuite naming conventions. Hot spare pools with shared disksets use the following convention:

setname/hspnumber

where setname is the name of the diskset, and number is the number of the hot spare pool (0-999).

OPTIONS

-a Adds drives or hosts to the named set. For a drive to be accepted into a set, the drive must not be in use within another metadevice or diskset, mounted on, or swapped on. When the drive is accepted into the set, it is repartitioned and the metadevice state database replica (for the set) may be placed on it. However, if a Slice 7 starts at cylinder 0, and is large enough to hold a state database replica, then the disk is not repartioned. Also, a drive is not accepted if it cannot be found on all hosts specified as part of the set. This means that if a host within the specified set is unreachable due to network problems, or is administratively down, the add will fail.

-b Insures that the replicas are distributed according to the replica layout algorithm. This can be invoked at any time, and will do nothing if the replicas are correctly distributed. In cases where the user has used the metadb command to manually remove or add replicas, this command can be used to insure that the distribution of replicas matches the replica layout algorithm.

-d Deletes drives or hosts from the named diskset. For a drive to be deleted, it must not be in use within the set. The last host cannot be deleted unless all of the drives within the set are deleted.

-f Forces one of three actions to occur: takes ownership of a diskset when used with -t; deletes the last disk drive from the diskset; or deletes the last host from the diskset. (Deleting the last drive or host from a diskset requires the -d option.) When used to forcibly take ownership of the diskset, this causes the diskset to be grabbed whether or not another host owns the set. All of the disks within the set are taken over (reserved) and fail fast is enabled, causing the other host to panic if it had diskset ownership. The metadevice state database will be read in by the host performing the take, and the shared metadevices contained in the set will be accessible. The -f option is also used to delete the last drive in the diskset, because

this drive would implicitly contain the last state database replica. The -f option is also used for deleting hosts from a set. When specified with a partial list of hosts, it can be used for one-host administration. One-host administration could be useful when a host is known to be non-functional, thus avoiding timeouts and failed commands. When specified with a complete list of hosts, the set is completely deleted. It is generally specified with a complete list of hosts to clean up after one-host administration has been performed.

-h hostname...
Specifies one or more host names to be added to or deleted from a diskset. Adding the first host creates the set. The last host cannot be deleted unless all of the drives within the set have been deleted. The host name is not accepted if all of the drives within the set cannot be found on the specified host. The host name is the same name found in /etc/nodename.

-o Returns an exit status of 0 if the local host or the host specified with the -h option is the owner of the diskset.

-r Releases ownership of a diskset. All of the disks within the set are released. The metadevices set up within the set are no longer accessible.

-s setname
Specifies the name of a diskset on which metaset will work. If no setname is specified, all disksets are returned.

-t Takes ownership of a diskset safely. If metaset finds that another host owns the set, this host will not be allowed to take ownership of the set. If the set is not owned by any other host, all the disks within the set will be owned by the host on which metaset was executed. The metadevice state database is read in, and the shared metadevices contained in the set become accessible. The -t option will take a diskset that has stale databases. When the databases are stale, metaset will exit code 66, and a message will be printed. At that point, the only operations permitted are the addition and deletion of replicas. Once the addition or deletion of the replicas has been completed, the diskset should be released and retaken to gain full access to the data.

EXAMPLES

This example defines a diskset.

```
# metaset -s relo-red -a -h red blue
```

The name of the diskset is relo-red. The names of the first and second hosts added to the set are red and blue, respectively. (The hostname is found in /etc/nodename.) Adding the first host creates the diskset. A diskset can be created with just one host, with the second added later. The last host cannot be deleted until all of the drives within the set have been deleted.

This example adds drives to a diskset.

```
# metaset -s relo-red -a c2t0d0 c2t1d0 c2t2d0 c2t3d0 c2t4d0
c2t5d0
```

The name of the previously created diskset is relo-red. The names of the drives are c2t0d0, c2t1d0, c2t2d0, c2t3d0, c2t4d0, and c2t5d0. Note that there is no slice identifier ("sx") at the end of the drive names.

FILES

/etc/opt/SUNWmd/md.tab
Contains list of metadevice configurations.

NOTES

Diskset administration, including the addition and deletion of hosts and drives, requires all hosts in the set to be accessible from the network.

SEE ALSO

metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metareplace(1M), metaroot(1M), metastat(1M), metasync(1M), metattach(1M), md.cf(4), md.tab(4), mddb.cf(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

NAME

metastat - display status for metadevice or hot spare pool

SYNOPSIS

```
metastat -h
metastat [ -s setname ] [ -p ] [ -t ] [ metadevice... ]
         [ hot_spare_pool... ]
```

AVAILABILITY

/usr/opt/SUNWmd/sbin

DESCRIPTION

The metastat command displays the current status for each metadevice (including stripes, concatenations, concatenations of stripes, mirrors, RAID5, and trans devices) or hot spare pool, or of specified metadevices or hot spare pools.

It is helpful to run the metastat command after using the metattach command to view the status of the metadevice.

OPTIONS

-h Displays usage message.

-p Displays the list of active metadevices and hot spare pools in a format like md.tab.

-s setname
Specifies the name of the diskset on which metastat will work. Using the -s option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on metadevices and/or hot spare pools in the local diskset.

-t Prints the current status and timestamp for the specified metadevices and hot spare pools. The timestamp provides the date and time of the last state change.

metadevice...
Displays the status of the specified metadevice(s). If a trans metadevice is specified, the status of the master and log devices is also displayed.

hot_spare_pool...
Displays the status of the specified hot spare pool(s).

EXAMPLES

The following example shows the partial output of the metastat command after creating a mirror, d0, consisting of two submirrors, d70 and d80.

```
# metastat d0
d0: Mirror
    Submirror 0: d80
      State: Okay
    Submirror 1: d70
      State: Resyncing
    Resync in progress: 15 % done
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 2006130 blocks
    .
    .
    .
```

WARNING

metastat prints states as of the time the command is entered. It is unwise to use the output of the metastat -p command to create a md.tab(4) file for a number of reasons:

- The output of metastat -p may show hot spares being used.
- It may show mirrors with multiple submirrors. See metainit for instructions for creating multi-way mirrors using metainit and metattach.
- A slice may go into an error state after metastat -p is issued.

SEE ALSO

metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metareplace(1M), metaroot(1M), metaset(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

NAME

metasync - handle metadevice resync during reboot

SYNOPSIS

```
metasync -h
metasync [ -s setname ] [ buffer_size ] metadevice
metasync [ -s setname ] -r [ buffer_size ]
```

AVAILABILITY

/usr/opt/SUNWmd/sbin

DESCRIPTION

The metasync command starts a resync operation on the specified metadevice. All components that need to be resynced are resynced. If the system crashes during a RAID5 initialization, or during a RAID5 resync, either an initialization or resync restarts when the system reboots.

Applications are free to access a metadevice at the same time that it is being resynced by metasync. Also, metasync performs the copy operations from inside the kernel, which makes the utility more efficient.

Use the -r option in boot scripts to resync all possible submirrors.

OPTIONS

-h Displays usage message.

-r Specifies that the metasync command handle special resync requirements during a system reboot. metasync -r should only be invoked from /etc/init.d/SUNWmd.sync. The metasync command only resyncs those metadevices that need to be resynced. metasync schedules all the mirror resyncs according to their pass numbers.

-s setname
Specifies the name of the diskset on which metasync will work. Using the -s option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on local metadevices.

buffer_size
Specifies the size (number of 512-byte disk blocks) of the internal copy buffer for the mirror resync. The size defaults to 63 512-byte disk blocks (31.5 Kbytes). It can be no more than 126 blocks.

SEE ALSO

metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metareplace(1M), metaroot(1M), metaset(1M), metastat(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

NAME

metatool - create and administer DiskSuite configurations

SYNOPSIS

metatool [-s diskset] [-r registry-file]

AVAILABILITY

/usr/opt/SUNWmd/sbin

DESCRIPTION

The metatool command runs DiskSuite Tool, Solstice DiskSuite's graphical user interface. metatool displays a graphical representation of all metadevices, hot spare pools, and the metadevice state database, and a drag-and-drop interface for manipulation of these objects.

All functionality available using the DiskSuite command line interface is available from metatool, with the exceptions of the creation of disksets and the unmirroring of file systems you cannot umount. Disksets must be created using the Disk-Suite command-line utilities. Metadevices and hot spare pools within disksets can then be displayed and administered by metatool using the -s option.

When you run metatool on a system with an existing DiskSuite configuration, you see all existing metadevices, hot spare pools, and the metadevice state database in the Objects list of the Metadevice Editor window for that diskset, either local or shared.

When you run metatool on a system that has no DiskSuite con-figuration, you see an empty MetaDB (metadevice state data-base) object in the Objects list of the Metadevice Editor window. This object must be populated with a minimum of three state database replicas before metadevices can be created.

OPTIONS

The following options can be used with metatool:

-s diskset

Display metadevices configured in the specified diskset.

-r registry-file

Load the Tools pulldown menu using entries from registry-file instead of the default registry /usr/opt/SUNWmd/lib/metatool-toolsmenu.

ENVIRONMENT

metatool supports the standard X11 environment variables. See environ(5) for descriptions of the following environment variables that affect the execution of the metatool command: LC_MESSAGES, LANG, NLSPATH.

RESOURCES

The file, /usr/opt/SUNWmd/lib/X11/app-defaults/Metatool, contains a list of all the X resources used by metatool.

SEE ALSO

growfs(1M), metaclear(1M), metadb(1M), metahs(1M), metainit(1M), metaoffline(1M), metaparam(1M), metareplace(1M), metaroot(1M), metaset(1M), metastat(1M), metasync(1M), metattach(1M), metatool-toolsmenu(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

NAME

metattach, metadetach - attach or detach metadevice to or from a mirror or trans

SYNOPSIS

```
metattach [ -h ]
metattach [ -s setname ] mirror [ metadevice ]
metattach [ -s setname ] [ -i size ] concat/stripe
          component...
metattach [ -s setname ] RAID component...
metattach [ -s setname ] trans log
metadetach [ -s setname ] [ -f ] mirror submirror
metadetach [ -s setname ] [ -f ] trans
```

AVAILABILITY

/usr/opt/SUNWmd/sbin

DESCRIPTION

metattach is used to add submirrors to a mirror, add logging devices to trans devices, or grow metadevices. Growing metadevices can be done without interrupting service. To grow the size of a mirror or trans, the slices must be added to the submirrors or to the master devices.

DiskSuite supports one-to-three-way mirrors. Thus, you can only attach a metadevice to a mirror if there are two or fewer submirrors beneath the mirror. Once a new metadevice is attached to a mirror, metattach will automatically start a resync operation to the new submirror.

Attaching a new logging device to a busy trans metadevice is allowed, although a trans metadevice will start using its new logging device only after the trans is idle (after it is unmounted, for example). The busy trans will be in an Attaching state (metastat) until the logging device is actually attached. Attaching a logging device in the Hard Error or Error state (metastat) is not allowed.

metadetach is used to detach submirrors from mirrors, or detach logging devices from trans metadevices.

When a submirror is detached from a mirror, it is no longer part of the mirror, thus reads and writes to and from that metadevice via the mirror are no longer performed through the mirror. Detaching the only existing submirror is not allowed. Detaching a submirror that has slices reported as needing maintenance (by metastat) is not allowed unless the -f (force) flag is used.

metadetach also detaches the logging device from a trans. Once detached, the logging device is no longer part of the trans, thus the trans is no longer logging and all benefits of logging are lost. Any information on the logging device that pertains to the master device is written to the master device before the logging device is detached.

Detaching the logging device from a busy trans device is not allowed unless the -f (force) flag is used. Even so, the logging device is not actually detached until the trans is idle. The trans is in the Detaching state (metastat) until the logging device is detached.

OPTIONS

Root privileges are required for all of the following options except -h.

-f Force the detaching of metadevices that have components that need maintenance or are busy.

-h Displays a usage message.

-i size
Specifies the interlace value for stripes, where size is a specified value followed by either `k' for kilobytes, `m' for megabytes, or `b' for blocks. The units can be either upper case or lower case. If size is not specified, the size defaults to the interlace size of the last stripe of the metadevice. When an interlace size change is made on a stripe, it will be carried forward on all stripes that follow.

-s setname
Specifies the name of the diskset on which the metattach command or the metadetach command will work. Using the -s option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on local metadevices.

mirror
Specifies the mirror.

metadevice
Specifies the name of the metadevice to be attached to the mirror as a submirror. This metadevice must have been previously created by the metainit command.

concat/stripe
Specifies the metadevice name of the concatenation, stripe, or concatenation of stripes.

component...
The logical name for the physical slice (partition) on a disk drive, such as /dev/dsk/c0t0d0s2, being added to the concatenation, stripe, concatenation of stripes, or RAID5 metadevice.

RAID Specifies the metadevice name of the RAID5 metadevice.

trans
Specifies the metadevice name of the trans metadevice (not the master or logging device).

log Specifies the metadevice name of the logging device to be attached to the trans metadevice.

submirror
The metadevice name of the submirror to be detached from the mirror.

EXAMPLES
This example concatenates a single new slice to an existing metadevice, d8. (Afterwards, you would use the growfs command to expand the file system.)

```
# metattach d8 /dev/dsk/c0t1d0s2
```

This example adds four slices to an existing metadevice, d9. (Afterwards, you would use the growfs command to expand the file system.)

```
# metattach d9 /dev/dsk/c0t1d0s2 /dev/dsk/c0t2d0s2 \
    /dev/dsk/c0t3d0s2 /dev/dsk/c0t4d0s2
```

This example detaches the logging device from a trans meta-

device d9. Notice that you do not have to specify the logging device itself, as there can only be one.

```
# metadetach d9
```

This example expands a RAID5 metadevice, d45, by attaching another slice.

```
# metattach d45 /dev/dsk/c3t0d0s2
```

When you add additional slices to a RAID5 metadevice, the additional space is devoted to data. No new parity blocks are allocated. The data on the added slices is, however, included in the overall parity calculations, so it is protected against single device failure.

This example adds space to a two-way mirror by adding a

slice to each submirror. (Afterwards, you would use the growfs command to expand the file system.)

```
# metattach d9 /dev/dsk/c0t2d0s5
# metattach d10 /dev/dsk/c0t3d0s5
```

This example detaches a submirror, d2, from a mirror, d4.

```
# metadetach d4 d2
```

SEE ALSO

metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metareplace(1M), metaroot(1M), metaset(1M), metastat(1M), metasync(1M), md.tab(4), md.cf(4), mddb.cf(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

WARNING

When a submirror is detached from its mirror, the data on the metadevice may not be the same as the data that existed on the mirror prior to running metadetach. In particular, if the -f option was needed, the metadevice and mirror probably do not contain the same data.

NAME

rpc.metad - remote metaset services

SYNOPSIS

rpc.metad

AVAILABILITY

/usr/opt/SUNWmd/sbin

DESCRIPTION

rpc.metad is an rpc(4) daemon (functioning as a server process) that is used to manage local copies of metadevice diskset information. The rpc.metad daemon is invoked by inetd.

SEE ALSO

inetd(1M), metaset(1M), rpc.metamhd(1M), rpc(3N), services(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

NAME

rpc.metamhd - remote multihost disk services

SYNOPSIS

rpc.metamhd

AVAILABILITY

/usr/opt/SUNWmd/sbin

DESCRIPTION

rpc.metamhd is an rpc(4) daemon (functioning as a server process) that is used to manage multi-hosted disks. The rpc.metamhd daemon is invoked by inetd.

SEE ALSO

inetd(1M), metaset(1M), rpc.metad(1M), rpc(3N), services(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

NAME

md.tab, md.cf - metadisk utility files

SYNOPSIS

md.tab
md.cf

AVAILABILITY

/etc/opt/SUNWmd

DESCRIPTION

/etc/opt/SUNWmd/md.tab is a file that can be used by metainit and metadb to configure metadevices, hot spare pools, and metadevice state database replicas in a batch-like mode. DiskSuite does not store configuration information in the /etc/opt/SUNWmd/md.tab file. The only way information appears in md.tab is through editing it by hand. When using the md.tab file, each metadevice, hot spare pool, or state database replica in the file must have a unique entry. Entries can include the following: simple metadevices (stripes, concatenations, and concatenations of stripes); mirrors, trans metadevices, and RAID5 metadevices; hot spare pools; and state database replicas. Because md.tab only contains entries that you type in it, do not rely on the file for the current configuration of metadevices, hot spare pools, and replicas on the system at any given time.

Tabs, spaces, comments (by using a pound sign, #), and continuation of lines (by using a backslash-newline), are allowed.

Typically, you set up metadevices according to information specified on the command line by using the metainit command. Likewise, you set up state database replicas with the metadb command.

An alternative to the command line is to use the md.tab file. Metadevices and state database replicas can be specified in the md.tab file in any order, and then activated in a batch-like mode with the metainit and metadb commands.

If you edit the md.tab file, you specify one complete configuration entry per line. Metadevices are defined using the same syntax as required by the metainit command. You then run the metainit command with either the -a option, to activate all metadevices in the md.tab file, or with the metadevice name corresponding to a specific configuration entry.

State database replicas are defined in the

/etc/opt/SUNWmd/md.tab file as follows:

```
mddbnumber options [ slice... ]
```

Where mddbnumber is the characters mddb followed by a two digit number that identifies the state database replica. slice is a physical slice. For example:

```
mddb05 /dev/dsk/c0t1d0s2
```

/etc/opt/SUNWmd/md.cf is a backup file of the configuration used for disaster recovery. Whenever the DiskSuite configuration is changed, this file is automatically updated (except when hot sparing occurs). You should not directly edit this file.

EXAMPLES

Examples listed here include md.tab entries for the creation of a concatenation, a stripe, a concatenation of stripes, a mirror, a trans metadevice, a RAID5 metadevice, a hot spare, and state database replicas. All drives in the following examples have the same size of 525 Mbytes.

Concatenation

This example shows a metadevice, /dev/md/dsk/d7, consisting of a concatenation of four disks.

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

The number 4 indicates there are four individual stripes in the concatenation. Each stripe is made of one slice, hence the number 1 appears in front of each slice.

Note: The first disk sector in all of the above devices contains a disk label. To preserve the labels on devices /dev/dsk/c0t2d0s0, /dev/dsk/c0t3d0s0, and /dev/dsk/c0t4d0s0, the metadisk driver must skip at least the first sector of those disks when mapping accesses across the concatenation boundaries. Since skipping only the first sector would create an irregular disk geometry, the entire first cylinder of these disks will be skipped. This will allow higher level file system software to optimize block allocations correctly.

Stripe

This example shows a metadevice, /dev/md/dsk/d15, consisting of two slices.

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

The number 1 indicates that one stripe is being created. Because the stripe is made of two slices, the number 2 follows next. The optional -i followed by 32k specifies the interlace size will be 32 Kbytes. If the interlace size were not specified, the stripe would use the default value of 16 Kbytes.

Concatenation of Stripes

This example shows a metadevice, /dev/md/dsk/d75, consisting of a concatenation of two stripes of three disks.

```
#
# (concatenation of two stripes, each consisting of three
disks)
#
d75 2 3 c0t1d0s2 c0t2d0s2 c0t3d0s2 -i 16k \
      3 c1t1d0s2 c1t2d0s2 c1t3d0s2 -i 32k
```

On the first line, the -i followed by 16k specifies that the stripe's interlace size is 16 Kbytes. The second set specifies the stripe interlace size will be 32 Kbytes. If the second set did not specify 32 Kbytes, the set would use default interlace value of 16 Kbytes. The blocks of each set of three disks are interlaced across three disks.

Mirroring

This example shows a three-way mirror, /dev/md/dsk/d50, consisting of three submirrors. This mirror does not contain any existing data.

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

In this example, a one-way mirror is first defined using the -m option. The one-way mirror consists of submirror d51. The other two submirrors, d52 and d53, are attached later using the metattach command. The default read and write options in this example are a round-robin read algorithm and parallel writes to all submirrors. The order in which mirrors appear in the /etc/opt/SUNWmd/md.tab file is unimportant.

Logging (trans)

This example shows a trans metadevice, /dev/md/dsk/d1, with mirrors for the master and logging devices. This trans does not contain any existing data.

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

In this example, the two mirrors, d10 and d20, are defined using the -m option. d10 is defined as the master device and d20 is defined as the logging device for the trans, d1, by using the -t option. The order in which mirrors or trans appear in the /etc/opt/SUNWmd/md.tab file is unimportant. The submirrors d12 and d22 are attached later (using the metattach command) to the d10 and d20 mirrors.

RAID5

This example shows a RAID5 metadevice, d80, consisting of three slices:

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

In this example, a RAID5 metadevice is defined using the -r option with an interlace size of 20 Kbytes. The data and

parity segments will be striped across the slices, c0t1d0s1, c1t0d0s1, and c2t0d0s1.

Hot Spare

This example shows a three-way mirror, /dev/md/dsk/d10, consisting of three submirrors and three hot spare pools.

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



```
hsp002 c3t2d0s2 c1t2d0s2 c2t2d0s2
hsp003 c1t2d0s2 c2t2d0s2 c3t2d0s2
```

In this example, a one-way mirror is first defined using the -m option. The submirrors are attached later using the metattach command. The hot spare pools to be used are tied to the submirrors with the -h option. In this example, there are three disks used as hot spares, defined in three separate hot spare pools. The hot spare pools are given the names hsp001, hsp002, and hsp003. Setting up three hot spare pools rather than assigning just one hot spare with each component helps to maximize the use of hardware. This configuration enables the user to specify selecting the most desirable hot spare first, and improves availability by having more hot spares available. At the end of the entry, the hot spares to be used are defined. Note: When using the md.tab file to associate hot spares with metadevices, the hot spare spool does not have to exist prior to the association. DiskSuite takes care of the order in which metadevices and hot spares are created when using the md.tab file.

State Database Replicas

This example shows how to set up an initial state database and three replicas on a server that has three disks.

```
#
# (state database and replicas)
```

```
#
mddb01 -c 3 c0t1d0s0 c0t2d0s0 c0t3d0s0
```

In this example, three state database replicas are stored on each of the three slices.

Once the above entry is made in the /etc/opt/SUNWmd/md.tab file, the metadb command must be run with both the -a and -f options. For example, typing the following command creates one state database replicas on three slices:

```
# metadb -a -f mddb01
```

FILES

/etc/opt/SUNWmd/md.tab
/etc/opt/SUNWmd/md.cf

SEE ALSO

metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metareplace(1M), metaroot(1M), metastat(1M), metasync(1M), metattach(1M), mddb.cf(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

LIMITATIONS

Recursive mirroring is not allowed; that is, a mirror cannot appear in the definition of another mirror.

Recursive logging is not allowed; that is, a trans metadevice cannot appear in the definition of another metadevice.

Stripes and RAID5 metadevices must contains slices only.

Mirroring of RAID5 metadevices is not allowed.

NAME

md.tab, md.cf - metadisk utility files

SYNOPSIS

md.tab

md.cf

AVAILABILITY

/etc/opt/SUNWmd

DESCRIPTION

/etc/opt/SUNWmd/md.tab is a file that can be used by metainit and metadb to configure metadevices, hot spare pools, and metadevice state database replicas in a batch-like mode. DiskSuite does not store configuration information in the /etc/opt/SUNWmd/md.tab file. The only way information appears in md.tab is through editing it by hand. When using the md.tab file, each metadevice, hot spare pool, or state database replica in the file must have a unique entry. Entries can include the following: simple metadevices (stripes, concatenations, and concatenations of stripes); mirrors, trans metadevices, and RAID5 metadevices; hot spare pools; and state database replicas. Because md.tab only contains entries that you type in it, do not rely on the file for the current configuration of metadevices, hot spare pools, and replicas on the system at any given time.

Tabs, spaces, comments (by using a pound sign, #), and continuation of lines (by using a backslash-newline), are allowed.

Typically, you set up metadevices according to information specified on the command line by using the metainit command. Likewise, you set up state database replicas with the metadb command.

An alternative to the command line is to use the md.tab file. Metadevices and state database replicas can be specified in the md.tab file in any order, and then activated in a batch-like mode with the metainit and metadb commands.

If you edit the md.tab file, you specify one complete configuration entry per line. Metadevices are defined using the same syntax as required by the metainit command. You then run the metainit command with either the -a option, to activate all metadevices in the md.tab file, or with the metadevice name corresponding to a specific configuration entry.

State database replicas are defined in the

/etc/opt/SUNWmd/md.tab file as follows:

```
mddbnumber options [ slice... ]
```

Where mddbnumber is the characters mddb followed by a two digit number that identifies the state database replica. slice is a physical slice. For example:

```
mddb05 /dev/dsk/c0t1d0s2
```

/etc/opt/SUNWmd/md.cf is a backup file of the configuration used for disaster recovery. Whenever the DiskSuite configuration is changed, this file is automatically updated (except when hot sparing occurs). You should not directly edit this file.

EXAMPLES

Examples listed here include md.tab entries for the creation of a concatenation, a stripe, a concatenation of stripes, a mirror, a trans metadevice, a RAID5 metadevice, a hot spare, and state database replicas. All drives in the following examples have the same size of 525 Mbytes.

Concatenation

This example shows a metadevice, /dev/md/dsk/d7, consisting of a concatenation of four disks.

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

The number 4 indicates there are four individual stripes in the concatenation. Each stripe is made of one slice, hence the number 1 appears in front of each slice.

Note: The first disk sector in all of the above devices contains a disk label. To preserve the labels on devices /dev/dsk/c0t2d0s0, /dev/dsk/c0t3d0s0, and /dev/dsk/c0t4d0s0, the metadisk driver must skip at least the first sector of those disks when mapping accesses across the concatenation boundaries. Since skipping only the first sector would create an irregular disk geometry, the entire first cylinder of these disks will be skipped. This will allow higher level file system software to optimize block allocations correctly.

Stripe

This example shows a metadevice, /dev/md/dsk/d15, consisting of two slices.

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

The number 1 indicates that one stripe is being created. Because the stripe is made of two slices, the number 2 follows next. The optional -i followed by 32k specifies the interlace size will be 32 Kbytes. If the interlace size were not specified, the stripe would use the default value of 16 Kbytes.

Concatenation of Stripes

This example shows a metadevice, /dev/md/dsk/d75, consisting of a concatenation of two stripes of three disks.

```
#
# (concatenation of two stripes, each consisting of three
disks)
#
d75 2 3 c0t1d0s2 c0t2d0s2 c0t3d0s2 -i 16k \
      3 c1t1d0s2 c1t2d0s2 c1t3d0s2 -i 32k
```

On the first line, the -i followed by 16k specifies that the stripe's interlace size is 16 Kbytes. The second set specifies the stripe interlace size will be 32 Kbytes. If the second set did not specify 32 Kbytes, the set would use default interlace value of 16 Kbytes. The blocks of each set of three disks are interlaced across three disks.

Mirroring

This example shows a three-way mirror, /dev/md/dsk/d50, consisting of three submirrors. This mirror does not contain any existing data.

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

In this example, a one-way mirror is first defined using the -m option. The one-way mirror consists of submirror d51. The other two submirrors, d52 and d53, are attached later using the metattach command. The default read and write options in this example are a round-robin read algorithm and parallel writes to all submirrors. The order in which mirrors appear in the /etc/opt/SUNWmd/md.tab file is unimportant.

Logging (trans)

This example shows a trans metadevice, /dev/md/dsk/d1, with mirrors for the master and logging devices. This trans does not contain any existing data.

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

In this example, the two mirrors, d10 and d20, are defined using the -m option. d10 is defined as the master device and d20 is defined as the logging device for the trans, d1, by using the -t option. The order in which mirrors or trans appear in the /etc/opt/SUNWmd/md.tab file is unimportant. The submirrors d12 and d22 are attached later (using the metattach command) to the d10 and d20 mirrors.

RAID5

This example shows a RAID5 metadevice, d80, consisting of three slices:

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

In this example, a RAID5 metadevice is defined using the -r option with an interlace size of 20 Kbytes. The data and parity segments will be striped across the slices, c0t1d0s1, c1t0d0s1, and c2t0d0s1.

Hot Spare

This example shows a three-way mirror, /dev/md/dsk/d10, consisting of three submirrors and three hot spare pools.

```
#
```

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



```
hsp002 c3t2d0s2 c1t2d0s2 c2t2d0s2
hsp003 c1t2d0s2 c2t2d0s2 c3t2d0s2
```

In this example, a one-way mirror is first defined using the -m option. The submirrors are attached later using the metattach command. The hot spare pools to be used are tied to the submirrors with the -h option. In this example, there are three disks used as hot spares, defined in three separate hot spare pools. The hot spare pools are given the names hsp001, hsp002, and hsp003. Setting up three hot spare pools rather than assigning just one hot spare with each component helps to maximize the use of hardware. This configuration enables the user to specify selecting the most desirable hot spare first, and improves availability by having more hot spares available. At the end of the entry, the hot spares to be used are defined. Note: When using the md.tab file to associate hot spares with metadevices, the hot spare spool does not have to exist prior to the association. DiskSuite takes care of the order in which metadevices and hot spares are created when using the md.tab file.

State Database Replicas

This example shows how to set up an initial state database and three replicas on a server that has three disks.

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

In this example, three state database replicas are stored on each of the three slices.

Once the above entry is made in the /etc/opt/SUNWmd/md.tab file, the metadb command must be run with both the -a and -f options. For example, typing the following command creates

one state database replicas on three slices:

```
# metadb -a -f mddb01
```

FILES

/etc/opt/SUNWmd/md.tab
/etc/opt/SUNWmd/md.cf

SEE ALSO

metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metareplace(1M), metaroot(1M), metastat(1M), metasync(1M), metattach(1M), mddb.cf(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

LIMITATIONS

Recursive mirroring is not allowed; that is, a mirror cannot appear in the definition of another mirror.

Recursive logging is not allowed; that is, a trans metadevice cannot appear in the definition of another metadevice.

Stripes and RAID5 metadevices must contains slices only.

Mirroring of RAID5 metadevices is not allowed.

NAME

mddb.cf - metadevice state database replica locations

SYNOPSIS

mddb.cf

AVAILABILITY

/etc/opt/SUNWmd

DESCRIPTION

The /etc/opt/SUNWmd/mddb.cf file is created when the metadb command is invoked. You should never directly edit this file.

/etc/opt/SUNWmd/mddb.cf is used by the metainit command to find the locations of the metadevice state databases replicas. The metadb command creates the file and updates it each time it is run. Similar information is entered in the /etc/system file.

Each metadevice state database replica has a unique entry in the /etc/opt/SUNWmd/mddb.cf file. Each entry contains the driver and minor unit numbers associated with the block physical device where a replica is stored. Each entry also contains the block number of the master block, which contains a list of all other blocks in the replica.

Entries in the /etc/opt/SUNWmd/mddb.cf file are of the form:

```
driver_name minor_t master_block  checksum
```

where driver_name and minor_t represent the device number of the physical device storing this replica. master_block is the block number of the master block used by this replica of the metadevice state database. checksum is used to make certain the entry has not been corrupted. A pound sign (#) introduces a comment.

EXAMPLES

The following example shows a mddb.cf file.

```
#metadevice database replica location file do not hand
edit
#driver   minor_t    daddr_t     checksum
 id       3          16          -182
 id       67         16          -246
 id       18         16          -197
 id       82         16          -261
```


FILES

/etc/opt/SUNWmd/mddb.cf
/etc/system

SEE ALSO

metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metareplace(1M), metaroot(1M), metastat(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Refer-

ence

NAME

metatool-toolsmenu - Solstice DiskSuite Tool application registry file

SYNOPSIS

metatool-toolsmenu

AVAILABILITY

/usr/opt/SUNWmd/lib

DESCRIPTION

metatool-toolsmenu is used by Solstice DiskSuite's DiskSuite Tool graphical user interface to initialize its `Tools' pulldown menu.

metatool-toolsmenu is an ASCII (text editable) file containing entries with variable numbers of fields. Each line in the file represents a single entry. Each line consists of fields separated by delimiter characters (`:' `|' `+' `^') and terminated with a newline.

The initial character of an entry indicates the delimeter that will be used for the remainder of the entry. There can be only one delimiter per entry.

Blank lines are allowed. Comments start with "#" and continue through the end of the line.

Entries in the metatool-toolsmenu file are of the form:

:<type>:<field>:...

where type indicates the entry type and format for the rest of the entry and field... indicates the data fields for the entry.

Currently, as of release 4.1, DiskSuite Tool recognizes only one entry type, `0'. This type specifies an entry consisting of two data fields:

:0:<name>:<commandline>:

where name indicates the string to be displayed in the Tools pulldown menu and commandline indicates the command line to be passed to system() when the menu item is selected. `:' is a field delimiter and can be one of: `+', `|', `:', or `^'.

metatool supports a small set of substitution variables that can be used in the command lines added to the registry in the form:

Substitution Variable	Value
$hostname	the current hostname
$setname	the current diskset name
$selection	the names of the currently selected objects

EXAMPLES

This example shows an entry for File Manager. You would see the string "File Manager" on the Tools pulldown menu. Choosing this selection would run the command /opt/SUNWadm/2.2/bin/stomgr -F in the current disk set context.

```
:0:File Manager...:/opt/SUNWadm/2.2/bin/stomgr -F -m $setname:
```

This example shows an entry for Disk Manager. You would see the string "Disk Manager" on the Tools pulldown menu. Choosing this selection would run the command /opt/SUNWadm/2.2/bin/stomgr -D.

```
:0:Disk Manager...:/opt/SUNWadm/2.2/bin/stomgr -D:
```

This example shows a sample registry file for metatool.

```
# Sample Registry for metatool
:0:File Manager...:/opt/SUNWadm/2.2/bin/stomgr -F -display
$hostname:0.0:
:0:Disk Manager...:/opt/SUNWadm/2.2/bin/stomgr -D -m $setname
$selection:
```

SEE ALSO

metatool(1M)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference

NAME

md - user configurable pseudo device driver

AVAILABILITY

SUNWmd

DESCRIPTION

md is a user configurable pseudo device driver that provides disk concatenation, striping, mirroring, RAID5 metadevices, trans metadevices, and hot spare utilities.

The block devices access the disk using the system's normal buffering mechanism and are read and written without regard to physical disk records. There is also a ``raw'' device which provides for direct transmission between the disk and the user's read or write buffer. A single read or write call usually results in one I/O operation; raw I/O is therefore considerably more efficient when many bytes are transmitted. The names of the block devices are found in /dev/md/dsk; the names of the raw devices are found in /dev/md/rdsk. Metadevices have the appearance of whole disks; there are no slices (partitions).

I/O requests (such as lseek (2)) to the metadevices must have an offset that is a multiple of 512 bytes (DEV_BSIZE), or the driver returns an EINVAL error. If the transfer length is not a multiple of 512 bytes, the tranfer count is rounded up by the driver.

The md pseudo device drivers support all disk devices on all Solaris 2.4 or later Solaris systems.

IOCTLS

This section provides a list of the ioctls supported by the metadisk driver. Other ioctls are used by the DiskSuite utilities, but these are not documented, and are for internal SunSoft use only.

The following ioctls are valid when issued to the raw metadevice such as /dev/md/rdsk/d0. See dkio(7) for additional information.

DKIOCGGEOM
This ioctl is used to get the disk geometry. The metadisk driver fills in the dkg_nhead, dkg_nsect, dkg_rpm, dkg_write_reinstruct and dkg_read_reinstruct from the first component of the metadevice (at metainit time).

dkg_ncyl is calculated using the size of the metadevice (reported by metastat) divided by (dkg_nhead * dkg_nsect). The total size is always a multiple of

(dkg_nhead * dkg_nsect).

If the first component of a metadevice does not start on cylinder number 0, then the dkg_ncyl is increased by one cylinder; because DKIOCGVTOC reports the metadevice as starting on cylinder 1. The side effect here is that it looks like cylinder 0 is not being used, but all the arithmetic works out correctly.
If the metadevice is not set up, then ENXIO is returned.

DKIOCINFO
When issued to the administrative device or metadevice, this ioctl sets dki_unit to the unit number of the metadevice, dki_ctype to a value of DKC_MD, and dki_partition to 0, because there are no slices.

DKIOCGVTOC
This ioctl returns the current vtoc. If one has not been written, then a default vtoc is returned. v_nparts is always 1. v_part[0].p_start is 0 if the first component of the metadevice starts on cylinder 0. Otherwise, the p_start field is the starting sector of cylinder 1. v_part[0].p_size is the same as the total size reported by metastat.

DKIOCSVTOC
This ioctl stores the vtoc in the metadevice state database so it is persistent across reboots.

DIAGNOSTICS

Notice Log Messages

The informative log messages include:

md: dnum: Hotspared device dev with dev
The first device name listed has been hot spare replaced with the second device name listed.

md: dnum: Hotspared device dev(num,num) with dev(num,num)
The first device number listed has been hot spare replaced with the second device number listed.

md: Could not load misc /dev
The named misc module in not loadable. It is possibly missing, or something else has been copied over it.

md: dnum: no mem for property dev
Memory could not be allocated in the prop_op entry point.

md: db: Parsing error on 'dev'
Set command in /etc/system for the mddb.bootlist<number> is not in the correct format. metadb -p can be run to put the correct set commands into the /etc/system file.

md: dnum: dev(num,num) needs maintenance
md: dnum: dev needs maintenance
An I/O or open error has occurred on a device within a mirror causing a component in the mirror to change to the Maintenance state.

md: dnum: dev(num,num) last erred
md: dnum: dev last erred
An I/O or open error has occurred on a device within a mirror and the data is not replicated elsewhere in the mirror. This is causing the component in the mirror to change to the Last Erred state.

Warning Log Messages

The warning log messages include:

md: dnum: not configurable, check /kernel/drv/md.conf
This error occurs when the number of metadevices as specified by the nmd parameter in the /kernel/drv/md.conf file is lower than the number of

configured metadevices on the system. It can also occur if the md_nsets parameter for disksets is lower than the number of configured disksets on the system. To fix this problem, examine the md.conf file and increase the value of either nmd or md_nsets as needed.

md: State database is stale
This error message comes when there are not enough usable replicas for the state database to be able to update records in the database. All accesses to the metadevice driver will fail. To fix this problem, more replicas need to be added or unaccessible replicas need to be deleted.

md: dnum: read error on dev
md: dnum: write error on dev
A read or write error has occurred on the specified submirror, at the specified device name. This happens if any read or write errors occur on a submirror.

md: dnum: read error on dev(num,num)
md: dnum: write error on dev(num,num)
A read or write error has occurred on the specified submirror, at the specified device number. This happens if any read or write errors occur on a submirror.

md: State database commit failed
md: State database delete failed
These messages occur when there have been device errors on components where the state database replicas reside. These errors only occur when more than half of the replicas have had device errors returned to them. For instance, if you have three components with state database replicas and two of the components report errors, then these errors may occur. The state database commit or delete is retried periodically. If a replica is added, then the commit or delete will finish and the system will be operational. Otherwise the system will timeout and panic.

md: dnum: Cannot load dev driver
Underlying named driver module is not loadable (for example, sd, id, xy, or a third-party driver). This could indicate that the driver module has been removed.

md: Open error of hotspare dev
md: Open error of hotspare dev(num,num)
Named hotspare is not openable, or underlying driver is not loadable.

Panic Log Messages
The panic log messages include:

md: dnum: Unknown close type
md: dnum: Unknown open type
Metadevice is being opened/closed with an unknown open type (OTYP).

md: State database problem
Failed metadevice state database commit or delete has been re-tried the default 100 times.

FILES

/dev/md/dsk/dn	block device (where n is the device number)
/dev/md/rdsk/dn	raw device (where n is the device number)
/dev/md/setname/dsk/dn	block device (where setname is the name of the diskset and n is the device number)
/dev/md/setname/rdsk/dn	raw device (where setname is the name of the diskset and n is the device number)

/dev/md/admin	administrative device
/kernel/drv/md	driver module
/kernel/drv/md.conf	driver configuration file
/kernel/misc/md_stripe	stripe driver misc module
/kernel/misc/md_mirror	mirror driver misc module
/kernel/misc/md_hotspares	hotspares driver misc module

/kernel/misc/md_trans	metatrans driver for UFS logging
/kernel/misc/md_raid	RAID5 driver misc module

SEE ALSO

metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metareplace(1M), metaroot(1M), metastat(1M), metasync(1M), metattach(1M), dkio(7I), md.tab(4), md.cf(4), mddb.cf(4)

Solstice DiskSuite User's Guide, Solstice DiskSuite Reference