Coraid SR

Reference Manual

Preliminary Edition April 14, 2016
SouthSuite, Inc.
Athens, Georgia

This book was typeset by SouthSuite using Plan 9 troff software by Joseph Ossana and Brian Kernighan. It is set in New Century Schoolbook using a set of private troff macros.

8/2/16 printing

Contents

Introduction

Coraid SR software turns commodity hardware into a high performance network storage appliance at a very affordable price. Smarter, faster, cheaper is our motto. And this software is just that.

We hope this manual will be useful, presenting information in a manner that is both easy to understand and pleasing to the eye. It lacks much of the boiler-plate fluff that is all too common in most manuals. And you will not find, for example, a section containing a plethora of clashing and visually disturbing fonts explaining how these offensive font combinations are variously used as user input, typed output, etc. We believe such information isn't helpful and the intended meaning can be derived from the context.

Principles of Operation of the SR

The SR is all about Ethernet-block storage. It's designed to be smarter, faster, and cheaper. The object of the storage is refered to as a target. SR commands deal with disk bays, RAID targets, and networking.

This manual is short because we work hard to make the product simple. Every "knob" has to earn its keep, so the number of concepts and commands you have to learn is small.

To the network, a *target* appears as a single large disk usable by any ATA-over-Ethernet initiator driver. Every target has an identifying number. The notation for an ATA-over-Ethernet target number is *major.minor*, where major is a number between 0 and 65,535 and minor is a number between 0 and 255. An example of a target number is 77.20.

Each SR chassis is known by a *shelf* number, which has the same range of values as the major numbers above. Currently, the SR will only allow you to create targets with the same major number as the shelf number. This restriction will be removed in future releases. In our example, target 77.20 will be on shelf 77.

The first step in setting up a chassis is to set its shelf number with the `shelf` command. Care should be taken to assign each shelf a unique number.

A target is constructed from one or more disk drives each occupying a disk *bay*. Disk bays are the physical slots containing storage media. On SR systems, they can be SATA, SCSI, or SSD devices. A single disk can be used in only one target. A number of disks can be combined to create a single larger target.

Targets can be a single disk. This is often quite useful in virtual machines where the target is for a single VM. This provides a private set of disk heads per virtual machine, improving performance by reducing head contention.

RAID Types

In some situations, creating a target out of several disks can improve performance. Spreading a target's data over several disks reduces the target's reliability, so the concept of Redundant Array of Inexpensive Disks, or *RAID* was invented. On the SR, one can create a RAID target with the `make` command.

The first argument to the make command is the ID of the target being created in the form of major.minor, e.g., 77.4. The next argument is the RAID type followed by a list of bay numbers that make up the disks in the RAID.

The simplist case is the "disk" RAID type. This actually isn't a RAID at all, but a single disk exported as an Ethernet block target. There is also a type "raidl" that is just a list of disks concatenated to form a longer target. Blocks are not interspersed across the drives of a raidl but form a long drive of the set of disks that have been joined together.

There are several RAID types described in David Patterson's famous paper, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," David A. Patterson, et al,

Association for Computing Machinery, 1988. RAID 0 is a simple striping of blocks across a number of disks without any redundant data included. Such a configuraion predates the acronym and was used by supercomputer operations to increase the performance of the disks. For a single system that needs high performance and recreatable data, this option can create very fast targets.

RAID 1 is a simple arrangement of two disks, each containing the same information. It's the highest level of redundancy and is the fastest to recover from a failure. The performance is the same as a single drive.

The combination of the two can be used to create a high performance, highly redundant target that consists of a RAID 0 of RAID 1s, referred to as RAID 10. A RAID 10 can survive multiple disk failures as long as two failures are not on the same stripe. Recovery time is short since only one disk worth of data needs to be copied, yet the performance is as fast as the number of stripes allows.

RAID 5 is a common raid type that consists of a number of data disks, usually around 7, combined with a parity disk. The parity disk is the exclusive-or of all the data disks. If any disk fails, its contents can be recreated using the exclusive-or operation on all the other disks. The exclusive-or, despite its exotic sounding name, is a common and cheap computer operation requiring no hardware assistance for fast performance.

With today's larger disk capacities and performance, RAID 5 appears less and less appealing. To replace a failed disk, one has to read ***all*** the data from ***all*** the other disks, which can take many days on a running system.

There is also the issue of poor performance of a RAID 5 operating in a degraded state. To read a block from the missing disk the system has to read all the remaining disks to recreate the needed data using the parity function. This means that a single read of the missing drive creates a read for every remaining drive in the RAID. Doing this while using the target also creates contention on the read/write heads, which move at a snail's pace compared with local reads and writes. Because building the missing drive causes sequential reads over all of the disks, it creates severe head contention with the normal IO to the target.

Besides RAID 5, the SR includes something called RAID 6 (the original paper from David Patterson only included RAIDs 0 through 5). A RAID 6 is a RAID where any *two* disks can fail and still function. It uses two parity disks instead of just one, and one of the disks uses Reed-Solomon error check codes.

Given the performance penalties for running RAID 6 in degraded mode, we think it would be smarter to simply use a RAID 10.

Disks: Elements of a Target

Targets then are made of a number of disks, each serving as a stripe. These disks are said to be *elements* of the target. Therefore, to specify a particular disk in a RAID target, specify a three-part element name. For example, the first disk in target 77.20 is 77.20.0, 77.20.1 is the second, and so on. Remember that targets are noted by a major.minor notation.

Some think that using RAID means you don't have to back up data since it can tolerate two disk failures. We disagree. RAID brings the reliability of a target that has several parts back to the reliability of a single drive. There is still a non-zero chance of losing all the data.

Targets are created out of disks inserted into drive bays in the chassis. The SR identifies the type of drive, whether SAS or SATA. All targets appear as an ATA drive to AoE.

To see what disks are in the chassis, the `disks` command will display each drive, its bay number, and other parameters about the disk, such as its size and if it's currently being used. These drives are the raw material of targets.

We should mention here, that the numbering of drive bays is problematic. Cabling, differences in disk backplanes, and the use of external expansion chassis makes it impossible to talk about bays by counting slots. The best way to deal with them is to use the `locate` command to flash the LED on the bay. Locate will show you a single drive in a bay using the number from the disks command, or it will display a single disk in a target by giving it the major.minor.element address. If you give locate only the major.minor number of the target, it will flash all of the bays of that target. Typing the command The LEDs will flash until you type "locate off."

The common operations on targets are `make`, `online`, `offline`, and `remove`.

Targets are created with the `make` command. Specify the major.minor address of the target, the RAID type, and then list the disk numbers you want to use. If the RAID type is RAID 5 or RAID 6, it will begin to initialize the parity blocks. You can use the target while this rebuilding is going on.

Once a target is created, it will not be accessible until you issue the `online` command. You can reverse this with the `offline` command.

To look at the targets you have created, use the `list` command to display them. Use the `-l` option to see details about the elements of the target.

When done with a target, its disks can be freed by using the `remove` command. The target must be offline to use the `remove` command.

You can examine the performance of a target with the `iostats` command. It will tell you throughput and latency information for a target. It's very useful to see which element is holding up performance.

When deciding how to use the disks in the system, set aside a number of disks to be used as spares. When a disk fails in a RAID, the system will look for a disk that has been marked as a spare, recruit that disk to replace the failed element, then recreate the data on that disk using the exclusive-or operation against the data from the other disks. To mark a disk as a spare, use the `spare` command.

To remove a disk from the set of spares, use the `rmspare` command.

The `fail` command can be used to take an element out of the target. If there is a spare available, the fail command will trigger its replacement, using a spare. This is a good way to remove a drive from a target without having to take the target down.

If you would rather replace the failed disk by hand, don't specify spares and use the `replace` command to do the job. This command takes the maj.min.element specifier and a bay number from the disks command.

A string can be associated with a target to help remind you how it's being used. The `label` command takes care of that. Use single quotes if the string you want to set has embedded spaces. As you might expect, the strings can be removed by issuing the `unlabel` command.

As we said earlier, when you want to create a target out of a single disk, use the `make` command, specifying the major.minor.element, a RAID type of "disk," and the

bay number from the `disks` command. This is done so often that there's a short hand for it: the `jbod` command. In this command you simply specify the major number and the bay number, which is used as the target's minor.

The Network and the SR

Without the network, networked storage would just be storage. In this section we'll talk a bit about how the SR uses networking.

The first thing to understand is that ATA-over-Ethernet doesn't use TCP or even IP. AoE runs directly over Ethernet. The protocol can only be used on the local broadcast domain, which makes it secure. The SR can have a VLAN, a virtual local area network, defined right to the target. In fact, one can have a different VLAN for each target in the SR.

By default, each Ethernet port in a system running the SR software will serve AoE requests. It is often useful to disable AoE on the first Ethernet port. Use the `aoeif` command to enable or disable a given port. Also, by default, an AoE enabled port will respond to an AoE request from any system in the broadcast domain. This can be changed by restricting a target to a list of Ethernet addresses that are allowed to access the system. If this is needed it is usually done over the protocol by some initiator that is doing fail over and wants to be able to restrict the protected system from accessing the storage. In other words, the protocol does this automatically, you never have to deal with it manually. But there is a command, `mask` that can be used to view and modify the list of allowed Ethernet addresses.

The SR is controlled via console commands. There are four ways to do this: a KVM, a serial port, Telnet, and cec. `Cec` is a program run on the initator that connects to the SR either with raw Ethernet or over TCP/IP.

By default the SR uses DHCP to look for a local IP address over its first ethernet port. This is usually the left RJ45 socket on the motherboard. While the unit doesn't need IP to serve AoE requests, IP is used as one way to log into the box and it is used to update the software on the unit from the SouthSuite servers.

While a room of servers is best managed using DHCP, you can set a static IP address using the `ipif` command.

Cec is the *Coraid Ethernet Console* protocol and there are open source versions of it that will run on Linux systems and VMware. It works directly on Ethernet, without IP, so it can be used even when the unit doesn't have an IP address. Cec can be enabled or disabled on an ethernet port with the `cecon` and `cecoff` commands, respectively. The `cecstat` command will display the current set of ports over which the cec protocol is allowed.

SR Work Flows

The simplicity of the SR is nicely illustrated by the small number of work flows and the brevity of each. We'll go over the majority of tasks that will need to be performed with your SR Ethernet block storage appliance.

Installing SR

The SR software is downloaded from the Coraid website. The downloaded file is a bit image that needs to be copied onto the Disk-on-Module (DoM) to be used in the hardware. The easiest way to do this is to **dd** the image using a Linux or similar system and then put the DoM into the appliance hardware.

```
dd if=srimage of=device
```

where *srimage* is the downloaded software and *device* is the path to the DoM.

Initial Setup

Very little setup is required. First, the shelf number needs to be set to give the appliance its major number. Then the password can be changed. The default password is *admin*.

```
Coraid unset 16:01:06 04/28/16 $ shelf 3
Coraid 3 18:36:35 04/28/16 $ passwd
old password: #####
new password: #####
again to verify: #####
Coraid 3 18:36:58 04/28/16 $
```

Notice that the prompt includes the date and time to aid in support.

A forgotten password can be reset by using the pseudo-password *ivelostit* and sending the output to SouthSuite support. SouthSuite will send you the response to the challenge.

```
Admin Password: #########
challenge: 29249
response: b653f070
Password reset to default
Coraid 3 19:03:26 04/28/16 $
```

As a result, the password will be reset to *admin*.

Creating a RAID Target

A target is created with the `make` command, which combines disks from bays into a RAID array. First, use the `disks` command to choose your elements to the array.

```
Coraid 3 19:06:01 04/28/16 $ disks
```

```
 0     32.000 - free          'SSDSA2SH032G1GN INTEL'   045C8860   sata 3.0Gb/s
 1    600.127 - free      'WD       WD6001BKHG-02D22'       SR03    sas 6.0Gb/s
 2    146.816 - free           'SEAGATE  ST9146852SS'       0005    sas 6.0Gb/s
 3    600.127 - free      'WD       WD6001BKHG-02D22'       SR03    sas 6.0Gb/s
 4    500.108 - free                 ST9500530NS       SN04   sata 3.0Gb/s
 5    146.816 - free           'SEAGATE  ST9146852SS'       0005    sas 6.0Gb/s
 7    600.127 - free      'WD       WD6001BKHG-02D22'       SR03    sas 6.0Gb/s
 8    500.108 - free                 ST9500530NS       SN04   sata 3.0Gb/s
 9    600.127 - free      'WD       WD6001BKHG-02D22'       SR03    sas 6.0Gb/s
10    600.127 - free      'WD       WD6001BKHG-02D22'       SR03    sas 6.0Gb/s
11    146.816 - free           'SEAGATE  ST9146852SS'       0005    sas 6.0Gb/s
12    200.050 - free             DENRSTE251M45-0200.C       1.37   sata 3.0Gb/s
13    100.030 - free       'STEC     Z16IZF2E-100UCU'       E494    sas 6.0Gb/s
14    300.000 - free      'HITACHI  H109030SESUN300G'       A690    sas 6.0Gb/s
15    300.000 - free      'HITACHI  H109030SESUN300G'       A690    sas 6.0Gb/s
16    100.030 - free             DENRSTE251M45-0100.C       1.37   sata 3.0Gb/s
17    500.108 - free                 ST9500530NS       SN04   sata 3.0Gb/s
18    750.156 - free         'WDC WD7500BPVT-55HXZT3'   01.01A01   sata 3.0Gb/s
19    750.156 - free         'WDC WD7500BPVT-55HXZT3'   01.01A01   sata 3.0Gb/s
20     32.000 - free          'SSDSA2SH032G1GN INTEL'   045C8860   sata 3.0Gb/s
21    600.127 - free      'WD       WD6001BKHG-02D22'       SR03    sas 6.0Gb/s
22    600.127 - free      'WD       WD6001BKHG-02D22'       SR03    sas 6.0Gb/s
23    500.108 - free           'SEAGATE  ST9500620SS'       AS06    sas 6.0Gb/s
Coraid 3 19:06:04 04/28/16 $
```

The first column is the *bay* number the disk is in. Since the drive cabling and disk backplane layout can vary, the exact location of the disk bay doesn't matter. If you want to locate a particular disk drive, the `locate` command can be used as follows:

```
Coraid 3 19:09:25 04/28/16 $ locate 0
Coraid 3 19:09:29 04/28/16 $ locate
0   locate
Coraid 3 19:09:32 04/28/16 $ locate off
Coraid 3 19:09:39 04/28/16 $ locate
Coraid 3 19:09:40 04/28/16 $
```

The first line starts flashing the LED on the disk in bay 0. The second command, the one without any arguments, will show you which bays are flashing. The `locate off` command will turn off all flashing LEDs.

When choosing drives and RAID type for a target, you need to consider the purpose of the target. If the target is for a single system that will have something like a database accessing the target, you can use a wide stripe of drives. If for example we want to make a RAID 5 for a MySQL application, we would use the seven Western Digital drives.

We use the `make` command to create the target. We can then look at the targets using the `list` command, and we see that it is offline and performing the init operation. We can online the target, even though it is building its redundant parity blocks. We use the `online` command and give the target's major.minor number.

```
Coraid 3 19:13:46 04/28/16 $ make 3.8 raid5 1 3 7 9 10 21 22
Coraid 3 19:14:28 04/28/16 $ list
3.8      3600.763 GB raid5    offline initing
Coraid 3 19:14:30 04/28/16 $ online 3.8
Coraid 3 19:14:36 04/28/16 $ list
```

```
3.8      3600.763 GB raid5    online  initing
Coraid 3 19:14:44 04/28/16 $ label MySQL 3.8
Coraid 3 19:14:52 04/28/16 $ list
3.8      3600.763 GB raid5    online  initing MySQL
```

The above example also shows how to label a target so one can identify it more easily. The `label` command can add multiple words to the label if its arguments are enclosed in single quotes.

To get a closer look at the target, the `-l` option to the `list` command can be used to see the details. The first column is the element number of the constituent drives. The next column specifies the current state of the element. They are all normal in this example. The size of each drive is next, shown in gigabytes. The last column contains the bay number that each drive is in.

```
Coraid 3 19:14:54 04/28/16 $ list -l
3.8      3600.763 GB raid5    online  initing MySQL
     0: normal    600.127 GB  1
     1: normal    600.127 GB  3
     2: normal    600.127 GB  7
     3: normal    600.127 GB  9
     4: normal    600.127 GB 10
     5: normal    600.127 GB 21
     6: normal    600.127 GB 22
```

You can see that the target is busy initing the parity blocks. To get an estimate of when that initing will be finished, use the `when` command.

```
Coraid 3 19:14:58 04/28/16 $ when
LUN      COMPLETE(%) I/O RATE(KBPS) ESTIMATED TIME(h:m:s)
3.8             0.46        281,300               4:07:44
```

Rebuilding speeds will vary depending on the type of disks, the speed of the controllers, and how much IO is done to the drives while the parity is being inited. Here's what it looks like when done.

```
Coraid 3 15:07:34 04/29/16 $ when
LUN      COMPLETE(%) I/O RATE(KBPS) ESTIMATED TIME(h:m:s)
nothing rebuilding
Coraid 3 15:07:45 04/29/16 $
```

When an Element Fails

The idea of RAID is to increase the reliability of an array of drives to have the same reliability of a single drive. This means that we can still serve requests if a disk fails, albeit with some performance degradation.

Here we see a target with a failed element 3. We know from the `disks` command that we have another compatible disk in bay 9. (See above.) We use the `replace` command to install a new element 3, and the target begins rebuilding its parity.

```
Coraid 3 14:15:31 05/05/16 $ list -l
3.0      3000.636 GB raid5    online  degraded
     0: normal    600.127 GB 21
     1: normal    600.127 GB 22
```

```
     2: normal    600.127 GB 10
     3: failed    600.127 GB  1
     4: normal    600.127 GB  3
     5: normal    600.127 GB  7
Coraid 3 14:15:33 05/05/16 $ replace 3.0.3 9
Coraid 3 14:16:05 05/05/16 $ list -l
3.0      3000.636 GB raid5    online  recovering,degraded
     0: normal    600.127 GB 21
     1: normal    600.127 GB 22
     2: normal    600.127 GB 10
     3: replaced  600.127 GB  9
     4: normal    600.127 GB  3
     5: normal    600.127 GB  7
Coraid 3 14:17:53 05/05/16 $ when
LUN      COMPLETE(%) I/O RATE(KBPS) ESTIMATED TIME(h:m:s)
3.0             1.32        437,084               2:15:29
```

Setting up Spares

You don't have to replace the element by hand. If you want, disks can be marked as spares. These disks will be automically claimed by targets in the event of a failed element. The `spare` command can be called with the bay containing the disk to be used as a spare.

When a disk fails and the spare automatically replaces the failed drive, the old disk is usually pulled and a new one fills its bay. The procedure then is to make the new drive the spare, leaving the old spare, now part of the target, to remain the permanent element. There is no need to move the data after the spare has been drafted into the target.

```
Coraid 3 15:43:27 05/05/16 $ spare 1
Coraid 3 15:43:31 05/05/16 $ spare
  1   600.127 GB
Coraid 3 15:43:33 05/05/16 $
```

Removing a Target

At some point, we will no longer want to use the target and will want to repurpose the disks for some other task. Removing the target is simple. Just offline it and then use the `remove` command. The disks are now free and available for another task.

```
Coraid 3 15:55:27 05/05/16 $ list
3.0      3000.636 GB raid5    online  recovering,degraded
Coraid 3 15:55:34 05/05/16 $ offline 3.0
Coraid 3 15:55:37 05/05/16 $ remove 3.0
Coraid 3 15:55:45 05/05/16 $ disks
  0     32.000 - free     'SSDSA2SH032G1GN INTEL'       045C8860  sata3.0Gb/s
  1    600.127 - spare    'WD       WD6001BKHG-02D22'       SR03   sas6.0Gb/s
  2    146.816 - free     'SEAGATE  ST9146852SS'            0005   sas6.0Gb/s
  3    600.127 - free     'WD       WD6001BKHG-02D22'       SR03   sas6.0Gb/s
  4    500.108 - free     ST9500530NS                       SN04  sata3.0Gb/s
  5    146.816 - free     'SEAGATE  ST9146852SS'            0005   sas6.0Gb/s
```

```
  7    600.127 - free     'WD       WD6001BKHG-02D22'       SR03   sas6.0Gb/s
  8    500.108 - free     ST9500530NS                       SN04  sata3.0Gb/s
  9    600.127 - free     'WD       WD6001BKHG-02D22'       SR03   sas6.0Gb/s
 10    600.127 - free     'WD       WD6001BKHG-02D22'       SR03   sas6.0Gb/s
 11    146.816 - free     'SEAGATE  ST9146852SS'            0005   sas6.0Gb/s
 12    200.050 - free     DENRSTE251M45-0200.C              1.37  sata3.0Gb/s
 13    100.030 - free     'STEC     Z16IZF2E-100UCU'        E494   sas6.0Gb/s
 14    300.000 - free     'HITACHI  H109030SESUN300G'       A690   sas6.0Gb/s
 15    300.000 - free     'HITACHI  H109030SESUN300G'       A690   sas6.0Gb/s
 16    100.030 - free     DENRSTE251M45-0100.C              1.37  sata3.0Gb/s
 17    500.108 - free     ST9500530NS                       SN04  sata3.0Gb/s
 18    750.156 - free     'WDC WD7500BPVT-55HXZT3'      01.01A01  sata3.0Gb/s
 19    750.156 - free     'WDC WD7500BPVT-55HXZT3'      01.01A01  sata3.0Gb/s
 20     32.000 - free     'SSDSA2SH032G1GN INTEL'       045C8860  sata3.0Gb/s
 21    600.127 - free     'WD       WD6001BKHG-02D22'       SR03   sas6.0Gb/s
 22    600.127 - free     'WD       WD6001BKHG-02D22'       SR03   sas6.0Gb/s
 23    500.108 - free     'SEAGATE  ST9500620SS'            AS06   sas6.0Gb/s
Coraid 3 15:55:48 05/05/16 $
```

Lexicon of Commands

This section lists all the SR commands in alphabetical order.

aoeif `[ etherN ... ]` The aoeifcs command displays the set of ethernet ports that the AoE protocol will use. If given a list of ports, it will limit any AoE transmissions to those ports. By default, all ports speak AoE.

cecoff `interface ...` The cecoff command is used to disable cec on an Ethernet interface. The interface should be specified in the format of “ether” followed by a digit, for example: cecoff ether0.

cecon `interface ...` The cecon command is used to enable cec on an Ethernet interface. The interface should be specified in the format of “ether” followed by a digit, for example: cecon ether0.

cecstat The cecstat command displays whether cec is enabled or disabled for each Ethernet interface.

date `[ YYYYMMDDHHMM ]` The date command displays the current date and time. The time is taken from the real-time clock on the motherboard.

disks `[ -cm ] [ major.minor ... ]` The disks command displays information about the disks in the appliance. The information displayed for each disk includes bay, size, role, model, firmware, and mode. The option -c displays configuration strings on the disks. The -m option groups disk models together for ease of use in making new targets or adding spares.

eject `major.minor ...` The eject command is used to eject one or more targets. It is similar to the remove command, except eject does not clear the RAID configuration on the component drives of a target. The eject command is useful when you want to physically move the drives in a target from one appliance to another without shutting down the appliance.

exit The exit command terminates an active login to the CLI, and re-prompts for the user password.

fail `major.minor.elem` The fail command marks an element of a target as failed. This command may be used to test how the appliance behaves in a failure condition.

halt The halt command stops all services and halts the appliance. Once halted, physical user intervention is required to restart the system by pressing the power or reset buttons on the front panel.

help `[ -s ] [ command ... ]` The help command displays a list of available commands. Given one or more command arguments, it will print the usage and a description of each command. If given the optional -s, it will print an abbreviated summary.

ifstat `[ -a ] [ interface ... ]` The ifstat command displays the status of all local interface ports. The display list can be restricted by providing one or more port arguments. The -a flag displays additional port statistics often useful for SouthSuite Technical Support.

iomode `[ major.minor ... ]` The iomode command displays the read access pattern optimization setting for configured targets. When set to sequential, a target is optimized for sequential reads using a prefetch mechanism. When set to random, a target is optimized for random read workloads by disabling prefetch. The default I/O mode for supported RAID types differs whether the target is comprised of all SSD drives or spinning drives. For RAID types “disk” and “raid0” the default for spinning disks is “sequential.” If those RAID types contain nothing but solid state disks, the iomode is “random.” For “raid1” and “raid10” the default is always “random.” For “raid5” and “raid6sr” the default is always “sequential.”

Targets of RAID types “raid5” and “raid6rs” do not support random iomode. If no targets are specified, the iomode state of all targets on the appliance is displayed. The iomode may be changed with the setiomode command. (see “help setiomode”)

iostats `[ -dl ] [ -s secs ] [ major.minor ... ]` The iostats command displays throughput and latency information for configured targets and the drives backing them. The MB/s, average latency, and maximum latency for reads and writes are displayed. The -s flag followed by a number between 1 and 32 specifies the number of seconds in the past to pull statistics. The default value is four seconds. The -d flag displays throughput and latency information for the drives. The -l flag displays throughput and latency information for the targets.

jbod `major.bay ...` The jbod command will create a new target of type “disk.” The LUN number defaults to the disk number, so it must not clash with an existing LUN number.

label `label maj.min ...` The label command sets a descriptive text label for one or more specified targets. The length of the label cannot exceed 16 characters. A label that contains spaces must be enclosed in single quotes (’).

list `[ -lc ] [ major.minor ... ]` The list command displays the targets configured on the appliance. With a -l option, the targets and component drive states are displayed. The -c option displays the following target configuration meta-data: target number, version, QueryConfig length, serial number, label, VLAN association, and flags. If no targets are specified, all targets on the appliance are displayed.

locate `[ off | major.minor[.element] | bay | spares ]` The locate command will allow you to locate a specified drive, target element, or all elements in a target on the appliance by flashing the LEDs on the bays for the argument specified. To turn off all flashing LEDs, type “locate off.”

make `major.minor raidtype bay ...` The make command creates a target.

Each target is an aggregate of one or more disks that form a RAID of “raidtype.” Raidtype can be raidL, raid0, raid1, raid10, raid5, raid6sr or “disk.” The raidtype “disk” is a single drive target. The “bay” arguments are the bay numbers from the “disk” command. Newly-created targets are offline by default to allow you to configure an Ethernet fence to manage its visibility to initiators (see “help mask”).

mask `[ major.minor ... ] [ +|-Ethernet ... ]` The mask command displays Ethernet address masks for all targets. The display list can be restricted by providing one or more target arguments. The displayed status includes target and address list. The mask command also modifies address lists by providing +ethernet and -ethernet arguments. To add an Ethernet address, prefix the address with a plus (+). To remove an address, prefix the address with a minus (-).

move `[ -l ] old-major.minor new-major.minor` The move command changes the major.minor numbers associated with a target. It can be used to change major.minor numbers on an appliance and to move targets between appliances.

offline `[ major.minor ... ]` The offline command disables SAN access for one or more specified targets.

online `[ major.minor ... ]` The online command enables SAN access for one or more specified targets.

passwd The passwd command sets the administrator (user admin) login password. Password length is restricted to 27 characters.

reboot The reboot command shuts down all services and reboots the appliance.

release The release command displays the current release of the software running the appliance.

remove `major.minor ...` The remove command removes one or more LUNs. Issuing the remove command clears the RAID configuration on the target’s component drives and releases for reuse all drives used in the component RAID. A target must be placed offline before it can be removed (see “help offline”). It is important to note that all data on a target will be lost after it is removed.

replace `major.minor.element bay` The replace command replaces a failed component in RAID 1, RAID 5, RAID 6, and RAID 10 LUNs with a drive not in use. After the failed drive has been replaced, the RAID is reconstructed. The replacement drive you specify must be listed when you issue the drives command, not in a failed state, and not in use as a spare or RAID component, with one exception: it is permissible to replace a RAID component with itself to force recovery of the existing component drive.

restore `[ -l ]` The restore command reads the configuration from all drives in the appliance and assembles targets and spares. When issued without arguments, the restore command ignores any disks that do not belong to the configured shelf address of the appliance (such as drives removed from another SR appliance). The -l flag reads the configuration from the drives in the appliance and displays the actions that it would perform, without actually carrying them out.

rmspare `bay ...` The rmspare command removes drives from the spare pool.

setiomode `random | sequential major.minor ...` The setiomode command sets the access pattern optimization of the specified targets to “sequential” or “random.” When set to sequential, a target is optimized for sequential reads using a prefetch mechanism. When set to random a target is optimized for random read workloads by disabling prefetch. See the entry for iomode for a list of default settings.

shelf `[ shelfno | unset ]` The shelf command assigns the appliance the shelf address that you specify. Choose a number between 0 and 65534 inclusive that is unique among all AoE storage devices attached to the network. The appliance shelf address must not conflict with other shelf addresses on the SAN. Before you set the appliance shelf address, unset appears in the appliance prompt. After you set the shelf address, unset is replaced by the shelf address that you specify.

sos `[ -f file | -r ]` The sos command captures and displays SouthSuite Technical Support information concerning the status of the appliance. By default, the output is printed to the screen, but it can be stored in a file if provided the -f flag and a file name. The file name will be created under the /tmp directory, and will not accept ’/’ in the path. Alternatively, if the -r flag is provided, the sos command will dial into SouthSuite, and dump the contents of the sos into a file there for ease of access by our technical staff.

The sos command only reports information regarding configuration and status of the appliance. It does NOT include any information about the data stored through the appliance.

spare `bay ...` The spare command lists all drives currently in the spare pool. If given a list of drives, it adds them to the spare pool. If a drive fails in a RAID 1, RAID 5, RAID 6, or RAID 10 target, the appliance recruits a drive from the spare pool to replace it.

syslog `[ unset | ipaddress ]` The syslog command configures the destination server IP for syslog UDP messages. Without arguments, syslog displays the destination server address. All syslog messages are sent to the configured IP address at UDP port 514. To clear the syslog destination server address, type ’syslog unset’.

telnet `[ on | off ]` The telnet command displays whether Telnet is enabled or disabled. Given on or off as a command argument will enable or disable Telnet on the appliance.

unlabel `maj.min ...` The unlabel command removes a label from one or more specified targets.

update
The update command changes the appliance OS to a different release. You no longer need to create an update LUN into which you would copy the update software, just run the update command. Update first dials into etherdrive.com, and hence requires that IP be set up on the appliance as well as DNS (see "help ipif"). You are then prompted for your user name and your access code (into etherdrive.com), which are both given you by SouthSuite. Once authorized, the

update command retrieves a list of available updates and prints them out. You then select the desired update (usually the latest) from the list by typing the index number from the first column, and press enter to start the update. The update command will download the software to your appliance, then reboot into that release.

uptime The uptime command displays the amount of time the appliance has been running since the last reboot.

vlan `[ maj.min vlan-id | clear ]` The vlan command displays targets and their IEEE 802.1Q VLAN associations. If given a lun and vlan-id argument, it will associate the given target with the specified VLAN. The association limits the target to communication with a single vlan-id. Multiple targets may be associated with a single vlan-id. A valid vlan-id is a number inclusively between 1 and 4094. The clear argument removes all VLAN associations from a target.

when The when command lists targets in either an initializing or recovering state. The information displayed for each target includes percentage complete, I/O rate, and estimated time to completion.

zap `-f bay` The zap command will erase the first six sectors (3072 bytes) of a drive, thereby removing any SouthSuite configuration from the drive. It does not erase your data.

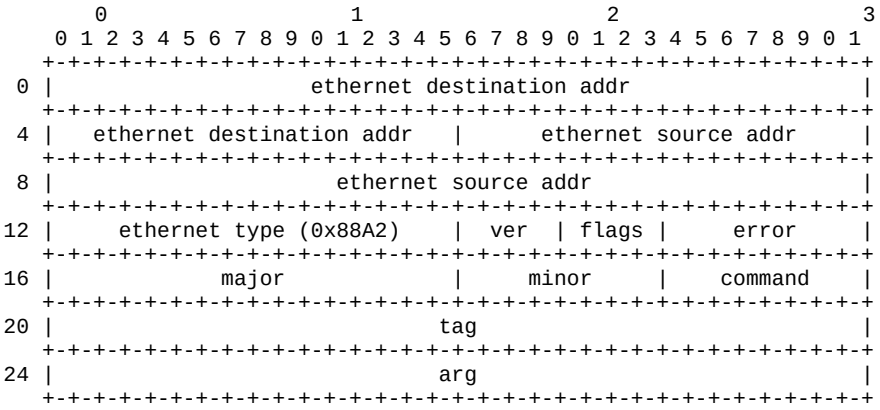
ATA over Ethernet (AoE)

Introduction. AoE is used to achieve a basic level RPC mechanism between an initiator and a target. The target accepts commands and generates responses based on a command code in the AoE header.

This section describes the format of the AoE header and the command set. All values for which a byte order is applicable are in network byte order. Reserved fields must be set to zero in all messages sent.

Common Header Format. An AoE transaction consists of a request message made by an *initiator* sent to a *target*, where it is processed and a response message is returned to the initiator using the source address of the request message. Each message contains a header followed by an argument field. The format of the argument field is defined based on the header command code.

AoE is not a connection based protocol. Each message sent to a target should be considered unique and unreliable. It's up to the initiator to resend messages that have gone astray.



AoE Header Format

The messages are carried in Ethernet frames, one message per frame. We include the Ethernet header as part of our definition. AoE has a registered Ethernet type of 0x88A2. The version, *ver* field defines the AoE header format as well as a set of command codes. All future versions must contain this field in the header, in this precise location. This document describes version 1. The Query Config Information response includes the version number a target supports.

The *flags* field is four bits. Bit 3 is set when the message is a response. Bit 2 is set in a response message if the associated command message generated an AoE protocol error. The remaing bits are zero, and reserved.

If flags bit 2 is set, the *error* field contains an error code defined as follows. A value of 1 means the command is unrecognized. A value of 2 means an improper value exists somewhere in the *args* field. A value of 3 means the target can no longer accept ATA commands. A value of 4 means the target cannot set the config

string because it is non-empty. (See Query/Config.) A value of 5 means the target does not understand the version number in *ver*. And a value of 6 means the command can not be completed because the target is reserved.

Major and minor numbers. Each AoE target possesses a major and minor address. Before processing the header Command the target must validate its major and minor address with the *major* and *minor* fields in the header. A target will accept a command message for processing if the following two tests are true:

The major field in the header is the target's major address or all ones. (0xffff).

The minor field in the header is the target's minor address or all ones. (0xff).

Any command messages failing either of these two tests must be ignored by the target. The target must supply its major and minor address in every response, even if the corresponding request has a value of all ones.

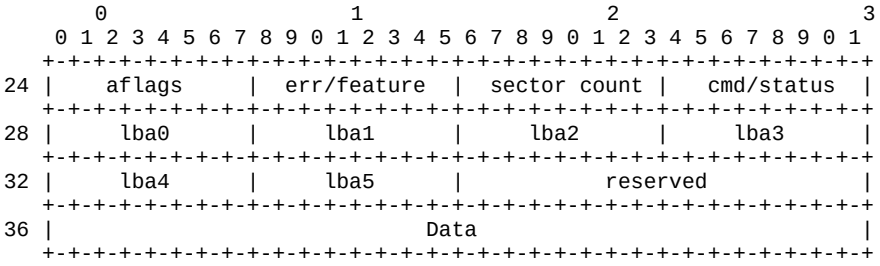
Command. This field contains the command code for the message. The following command codes are defined for this protocol version: 0: issue ATA command, 1: query/config command, 2: MAC mask list command, 3: reserve/release command. Command codes 240-255 are reserved for vendor specific use.

Tag. The Tag field gives an initiator the means to correlate responses with their appropriate commands. It is copied into the response message by the target and is otherwise ignored.

Arg. The Arg field contents serve as input for the specified command code.

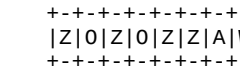
ATA Command (code 0) Command 0 is used to issue an ATA command to an attached ATA device. Any data associated with a command must fit into a single message. Using standard 1520 byte Ethernet frames will limit a device read/write to a maximum of two sectors. Using jumbo 9000 byte frames allows you to fit eight sectors into the frame.

ATA reads/writes to a target must be gated by the reserve list. If a target cannot be accessed due to reservation restriction, AoE error 6 must be returned.



ATA Command Format

AFlags is defined as follows:

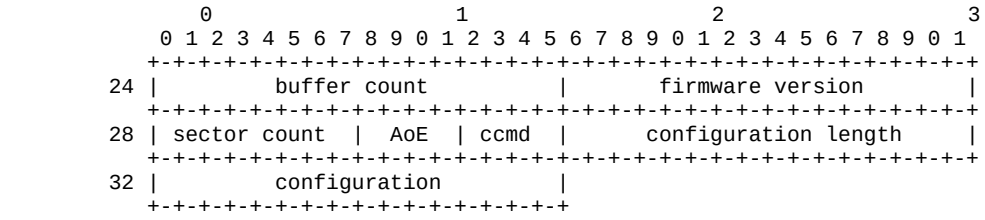


The W bit must be set when the ATA command requires data to be written to the device. The A bit must be set when a write request is to be done asynchronously. The O bits are obsolete and are ignored. The Z bits are reserved and must be set to zero.

The W bit and the Sector Count field together determine whether data is to be transferred to or from the device. If data is to be written to the device, the W bit must be set and Data must contain Sector Count * 512 bytes of data. If data is to be read from the device, the W bit must be cleared and Sector Count must specify the number of sectors to be read from the device. If the command succeeds, the Data field in the response message will contain the data read. If no data is to be transferred, Sector Count must be zero and the W bit is ignored. If both the A and W bits are set, the target may cache the write request in memory and respond immediately, returning the Arg field unchanged. The target may issue the hardware request whenever convenient, provided a subsequent read of the same sector returns the cached data. If the target experiences a power failure, the data in the target's write cache may be lost. No response is sent when the write request completes, even if an error occurred.

The target will behave as if it has transfered the contents of each field corresponding to the ATA registers as implied by the register names. The response should contain values as if the ATA operation has taken place. For example, *cmd* is the ATA command on the transaction and the same field in the response is the *status*. Note that the *sector count* register is only one byte, not two as in the ATA spec.

Query/Config (command 1) Query/Config provides a mechanism for locating a target. It serves as the Address Resolution Protocol for AoE, with the additional benefit that it can query targets who have been configured to be used by specific initiators. A 1,024 byte string can be set, queried and cleared to provide this function.



Query/Config Command Format

Buffer count is the maximum number of outstanding messages the target can queue for processing. Messages in excess of this value may be dropped. *Firmware version* is the version of the firmware in the target. *Sector count*, if non-zero, is the maximum number of sectors the target can handle in a single ATA request. A value of zero is equivalent to a value of two. *AoE* is 1. *Ccmd* is the particular query/config command to execute (see below). *Configuration length* is the number of bytes in the configuration buffer.

Configuration is a sequence of 8-bit bytes. While it is common to have them be UTF-8 strings, they are not limited to displaying text.

On a request message, the fields buffer count, firmware version and AoE must be set to zero by the initiator and are ignored by the target. When an initiator sends a query/config request message to a target, the ccmd code determines its behaviour.

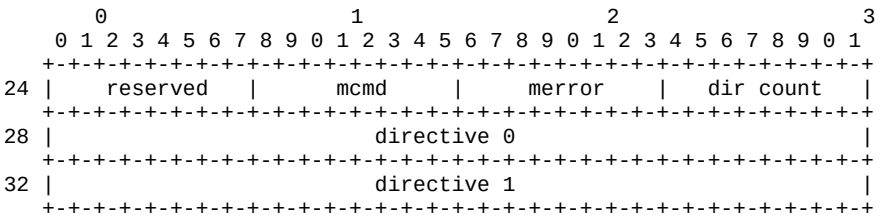
A **ccmd of 0** causes the target to return the current configration. A **ccmd of 1** will cause the target to response to the request only if the configuration in the request message completely matches the target's configuration. The target will

respond to a **ccmd of 2** if the request's configuration is a valid prefix of the target's configuration. In other words, all the bytes in the request message configuration must match the bytes in the target's configuration, but doesn't have to be as long as the target's configuration. This can be used to find subsets of configurations in a network of targets.

Two commands set the configuration. **Ccmd of 3** sets the configuration of a target **only if** the target configuration length is currently zero. This is an automatic command. If two initiators try to set the configuration of a single target, only one will succeed. **Ccmd of 4** will force the configuration on the target to be set unconditionally.

Ethernet Address Fencing (command 2). Normally a target will respond to any initiator's request. However, a target can limit the intiators to which it will respond, causing all other initiators to be fenced off. This is done by setting an Ethernet address mask list. When this list is non-empty the target will only respond to the request messages from an Ethernet address in the mask list. All others are masked out.

The Ethernet Address Fencing command has a 32 bit header followed by a number of address directives.

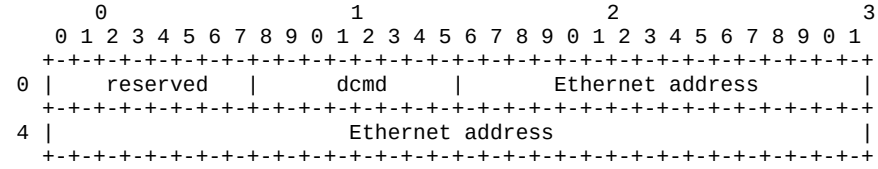


Ethernet Address Fenching Command

Mcmd informs the target if the request is to read the list, in which case the value is 0, or edit the mask list, in which case the value will be a 1. The *merror* byte will be non-zero in the case of an error. A value of 1 indicates that something unspecified was wrong. The value 2 indicates that the *dcmd* value was unknown, and an merror of 3 indicates the list is full and a directive tried to install a new address.

Dir count contains the number of directive entries in a request and the number of addresses in a response.

Each Ethernet address directive is eight bytes.



Ethernet Address Directive

Dcmd is the directive command; 0 means empty directive entry, 1 indicates an Ethernet mask should be added to the list, and a 2 indicates the Ethernet address should be deleted.

On a request message with mcmd set to zero, the initiator will return a list of directive entries with the Ethernet addresses filled in.

The target will process the list of directives for a request message if the *mcmd* field is 1. Targets will evaluate the directives in the order they appear in the request. If an error is encountered the target should cease processing the request and return a response with the *dir count* field set to the directive that caused the error, and the *merror* field set to the cause of the error.

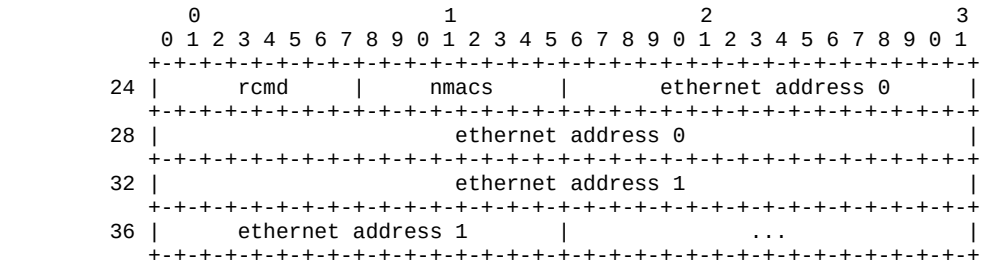
If no errors occurred in processing the directive list, the target must return the Ethernet address list in the response, including setting *dir count* field to the number of addresses. It is not an error to add an Ethernet address to a list more than once. Similarly, it is not an error to remove an Ethernet address from a list that does not contain the address. Targets must ignore duplicate additions and deletions.

There is no provision for reading a list containing more than 255 entries. Targets supporting this command are recommended to limit their mac mask lists to 255 entries for clarity. A target is not required, however, to support 255 mask list entries.

It should be noted that if an initiator adds addresses to a list and the initiator is not included, the initiator will no longer be able to communicate with the target. It is recommended that any modification of the list include the initiator source Ethernet address as the first directive.

Target Reservations (command 3) Some initiators would like to share a target. To do this, a means to coordinate access between initiators accessing the same target must be provided. Reservations are a way of doing that.

When reserved, an AoE target will only process ATA read and write commands if the source mac address in the command frame is in the reserve Ethernet address list; other commands (query config, ata ident, etc) are processed regardless of the source mac address. When a target cannot be accessed due to target reservation, an AoE error is returned with an error code of 6: Target is reserved.



Reservation Command Format

All operations return the current Ethernet address list of initiators allowed to do reads and writes on the target. As expected, *rcmd* contains the operation required of the target. A *rcmd* value of **0** is a no-op and is used to retrieve the current address list. *Rcmd* **1** sets the current list to the set of addresses in the request message and removes the previous list. A reservation can be removed by setting the target to an empty list. To make the reservation automatic, only initiators in the reservation list can set the list. If two initiators try to set the list at the same time, only one will succeed. The other will get AoE error 6.

A host can force set the reservation list with a *rcmd* command of **2**. Its primary use is to modify a stale reserve list.

Nmacs has the count of Ethernet addresses in a message.