

**NAME**

cciss - HP Smart Array block driver

**SYNOPSIS**

```
modprobe cciss [ cciss_allow_hpsa=1 ]
```

**DESCRIPTION**

**cciss** is a block driver for older HP Smart Array RAID controllers.

**Options**

*cciss\_allow\_hpsa=1*: This option prevents the **cciss** driver from attempting to drive any controllers that the [hpsa\(4\)](#) driver is capable of controlling, which is to say, the **cciss** driver is restricted by this option to the following controllers:

- Smart Array 5300
- Smart Array 5i
- Smart Array 532
- Smart Array 5312
- Smart Array 641
- Smart Array 642
- Smart Array 6400
- Smart Array 6400 EM
- Smart Array 6i
- Smart Array P600
- Smart Array P400i
- Smart Array E200i
- Smart Array E200
- Smart Array E200i
- Smart Array E200i
- Smart Array E200i
- Smart Array E500

**Supported hardware**

The **cciss** driver supports the following Smart Array boards:

- Smart Array 5300
- Smart Array 5i
- Smart Array 532
- Smart Array 5312
- Smart Array 641
- Smart Array 642
- Smart Array 6400
- Smart Array 6400 U320 Expansion Module
- Smart Array 6i
- Smart Array P600
- Smart Array P800
- Smart Array E400
- Smart Array P400i
- Smart Array E200
- Smart Array E200i
- Smart Array E500
- Smart Array P700m
- Smart Array P212
- Smart Array P410
- Smart Array P410i
- Smart Array P411
- Smart Array P812
- Smart Array P712m

Smart Array P711m

### Configuration details

To configure HP Smart Array controllers, use the HP Array Configuration Utility (either **hpacuxe(8)** or **hpacucli(8)**) or the Offline ROM-based Configuration Utility (ORCA) run from the Smart Array's option ROM at boot time.

## FILES

### Device nodes

The device naming scheme is as follows:

Major numbers:

```
104 cciss0
105 cciss1
106 cciss2
105 cciss3
108 cciss4
109 cciss5
110 cciss6
111 cciss7
```

Minor numbers:

```
b7 b6 b5 b4 b3 b2 b1 b0
|---+----| |---+----|
||
| +----- Partition ID (0=wholedev, 1-15 partition)
|
+----- Logical Volume number
```

The device naming scheme is:

```
/dev/cciss/c0d0 Controller 0, disk 0, whole device
/dev/cciss/c0d0p1 Controller 0, disk 0, partition 1
/dev/cciss/c0d0p2 Controller 0, disk 0, partition 2
/dev/cciss/c0d0p3 Controller 0, disk 0, partition 3

/dev/cciss/c1d1 Controller 1, disk 1, whole device
/dev/cciss/c1d1p1 Controller 1, disk 1, partition 1
/dev/cciss/c1d1p2 Controller 1, disk 1, partition 2
/dev/cciss/c1d1p3 Controller 1, disk 1, partition 3
```

### Files in `/proc`

The files `/proc/driver/cciss/cciss[0-9]+` contain information about the configuration of each controller. For example:

```
$ cd /proc/driver/cciss
$ ls -l
total 0
-rw-r--r-- 1 root root 0 2010-09-10 10:38 cciss0
-rw-r--r-- 1 root root 0 2010-09-10 10:38 cciss1
-rw-r--r-- 1 root root 0 2010-09-10 10:38 cciss2
$ cat cciss2
cciss2: HP Smart Array P800 Controller
Board ID: 0x3223103c
Firmware Version: 7.14
IRQ: 16
Logical drives: 1
Current Q depth: 0
```

```

Current # commands on controller: 0
Max Q depth since init: 1
Max # commands on controller since init: 2
Max SG entries since init: 32
Sequential access devices: 0

cciss/c2d0: 36.38GB RAID 0

```

#### Files in /sys

```

/sys/bus/pci/devices/<dev>/ccissX/cXdY/model
    Displays the SCSI INQUIRY page 0 model for logical drive Y of controller X.

/sys/bus/pci/devices/<dev>/ccissX/cXdY/rev
    Displays the SCSI INQUIRY page 0 revision for logical drive Y of controller X.

/sys/bus/pci/devices/<dev>/ccissX/cXdY/unique_id
    Displays the SCSI INQUIRY page 83 serial number for logical drive Y of controller X.

/sys/bus/pci/devices/<dev>/ccissX/cXdY/vendor
    Displays the SCSI INQUIRY page 0 vendor for logical drive Y of controller X.

/sys/bus/pci/devices/<dev>/ccissX/cXdY/block:cciss!cXdY
    A symbolic link to /sys/block/cciss!cXdY.

/sys/bus/pci/devices/<dev>/ccissX/rescan
    When this file is written to, the driver rescans the controller to discover any new,
    removed, or modified logical drives.

/sys/bus/pci/devices/<dev>/ccissX/resettable
    A value of 1 displayed in this file indicates that the reset_devices=1 kernel parameter
    (used by kdump) is honored by this controller. A value of 0 indicates that the
    reset_devices=1 kernel parameter will not be honored. Some models of Smart Array are
    not able to honor this parameter.

/sys/bus/pci/devices/<dev>/ccissX/cXdY/lunid
    Displays the 8-byte LUN ID used to address logical drive Y of controller X.

/sys/bus/pci/devices/<dev>/ccissX/cXdY/raid_level
    Displays the RAID level of logical drive Y of controller X.

/sys/bus/pci/devices/<dev>/ccissX/cXdY/usage_count
    Displays the usage count (number of opens) of logical drive Y of controller X.

```

#### SCSI tape drive and medium changer support

SCSI sequential access devices and medium changer devices are supported and appropriate device nodes are automatically created (e.g., `/dev/st0`, `/dev/st1`, etc.; see [st\(4\)](#) for more details.) You must enable SCSI tape drive support for Smart Array 5xxx and SCSI support in your kernel configuration to be able to use SCSI tape drives with your Smart Array 5xxx controller.

Additionally, note that the driver will not engage the SCSI core at init time. The driver must be directed to dynamically engage the SCSI core via the `/proc` filesystem entry, which the block side of the driver creates as `/proc/driver/cciss/cciss*` at run time. This is because at driver init time, the SCSI core may not yet be initialized (because the driver is a block driver) and attempting to register it with the SCSI core in such a case would cause a hang. This is best done via an initialization script (typically in `/etc/init.d`, but could vary depending on distribution). For example:

```

for x in /proc/driver/cciss/cciss[0-9]*
do
echo engage scsi > $x
done

```

Once the SCSI core is engaged by the driver, it cannot be disengaged (except by unloading the driver, if it happens to be linked as a module.)

Note also that if no sequential access devices or medium changers are detected, the SCSI core will not be engaged by the action of the above script.

### Hot plug support for SCSI tape drives

Hot plugging of SCSI tape drives is supported, with some caveats. The `cciss` driver must be informed that changes to the SCSI bus have been made. This may be done via the `/proc` filesystem. For example:

```
echo rescan > /proc/scsi/cciss0/1
```

This causes the driver to:

1. query the adapter about changes to the physical SCSI buses and/or fibre channel arbitrated loop, and
2. make note of any new or removed sequential access devices or medium changers.

The driver will output messages indicating which devices have been added or removed and the controller, bus, target, and lun used to address each device. The driver then notifies the SCSI midlayer of these changes.

Note that the naming convention of the `/proc` filesystem entries contains a number in addition to the driver name (e.g., `cciss0` instead of just `cciss`, which you might expect).

Note: *Only* sequential access devices and medium changers are presented as SCSI devices to the SCSI midlayer by the `cciss` driver. Specifically, physical SCSI disk drives are *not* presented to the SCSI midlayer. The only disk devices that are presented to the kernel are logical drives that the array controller constructs from regions on the physical drives. The logical drives are presented to the block layer (not to the SCSI midlayer). It is important for the driver to prevent the kernel from accessing the physical drives directly, since these drives are used by the array controller to construct the logical drives.

### SCSI error handling for tape drives and medium changers

The Linux SCSI midlayer provides an error-handling protocol that is initiated whenever a SCSI command fails to complete within a certain amount of time (which can vary depending on the command). The `cciss` driver participates in this protocol to some extent. The normal protocol is a four-step process:

- \* First, the device is told to abort the command.
- \* If that doesn't work, the device is reset.
- \* If that doesn't work, the SCSI bus is reset.
- \* If that doesn't work, the host bus adapter is reset.

The `cciss` driver is a block driver as well as a SCSI driver and only the tape drives and medium changers are presented to the SCSI midlayer. Furthermore, unlike more straightforward SCSI drivers, disk I/O continues through the block side during the SCSI error-recovery process. Therefore, the `cciss` driver implements only the first two of these actions, aborting the command, and resetting the device. Note also that most tape drives will not oblige in aborting commands, and sometimes it appears they will not even obey a reset command, though in most circumstances they will. If the command cannot be aborted and the device cannot be reset, the device will be set offline.

In the event that the error-handling code is triggered and a tape drive is successfully reset or the tardy command is successfully aborted, the tape drive may still not allow I/O to continue until some command is issued that positions the tape to a known position. Typically you must rewind the tape (by issuing `mt -f /dev/st0 rewind` for example) before I/O can proceed again to a tape drive that was reset.

### SEE ALSO

`cciss_vol_status(8)`, [hpsa\(4\)](#), [hpacucli\(8\)](#), [hpacuxe\(8\)](#),

[Unknown](#), and *Documentation/blockdev/cciss.txt* and *Documentation/ABI/testing/sysfs-bus-pci-devices-cciss* in the Linux kernel source tree

## COLOPHON

This page is part of release 3.74 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at <http://www.kernel.org/doc/man-pages/>.