NAME

ctl - CAM Target Layer

SYNOPSIS

To compile this driver into the kernel, place the following line in your kernel configuration file:

```
device ctl
```

Alternatively, to load the driver as a module at boot time, place the following line in loader.conf(5):

```
ctl_load="YES"
```

DESCRIPTION

The **ctl** subsystem provides SCSI target devices emulation. It supports features such as:

- Disk, CD-ROM and processor device emulation
- Tagged queueing
- SCSI task attribute support (ordered, head of queue, simple tags)
- SCSI implicit command ordering support
- Full task management support (abort, query, reset, etc.)
- Support for multiple ports, initiators, targets and backing stores
- Support for VMWare VAAI and Microsoft ODX offload (COMPARE AND WRITE, XCOPY, POPULATE TOKEN/WRITE USING TOKEN, WRITE SAME and UNMAP)
- Persistent reservation support
- Extensive VPD/mode/log pages support
- Featured error reporting, error injection and basic SMART support
- High Availability clustering support with ALUA
- All I/O handled in-kernel, no userland context switch overhead

The **ctl** subsystem includes multiple frontends to provide access using different transport protocols and implementations:

camsim Provides access for local system via virtual initiator mode CAM(4) SIM.

camtgt Provides access for remote systems via target mode CAM(4) SIMs, such as Fibre Channel isp(4) and mpt(4).

cfumass Provides access for remote systems via USB Mass Storage Class Bulk Only (BBB) Transport.

ha Internal frontend used to receive requests from other node ports in High Availability cluster.

ioctl Provides access for local user-level applications via ioctl(2) based API.

iscsi Provides access for remote systems via the iSCSI protocol using cfiscsi(4).

tpc Internal frontend used to receive requests from Third Party Copy engine, implementing copy offload operations.

The **ctl** subsystem includes two backends to create logical units using different kinds of backing stores:

block Stores data in ZFS ZVOLs, files or raw block devices.

ramdisk Stores data in RAM, that makes it mostly useful for performance testing. Depending on configured capacity can work as black hole, thin or thick provisioned disk.

SYSCTL VARIABLES

The following variables are available as both sysctl(8) variables and loader(8) tunables:

kern.cam.ctl.debug

Bit mask of enabled CTL log levels:

1 log commands with errors;
2 log all commands;
4 log data for commands other then READ/WRITE.

Defaults to 0.

kern.cam.ctl.ha_id

Specifies unique position of this node within High Availability cluster. Default is 0 -- no HA, 1 and 2 -- HA enabled at specified position.

kern.cam.ctl.ha_mode

Specifies High Availability cluster operation mode:

0 Active/Standby -- primary node has backend access and processes requests, while secondary can only do basic LUN discovery and reservation;
1 Active/Active -- both nodes have backend access and process requests, while secondary node synchronizes processing with primary one;
2 Active/Active -- primary node has backend access and processes requests, while secondary node forwards all requests and data to primary one;

All above modes require established connection between HA cluster nodes. If connection is not configured, secondary node will report Unavailable state; if configured but not established -- Transitioning state. Defaults to 0.

kern.cam.ctl.ha_peer

String value, specifying method to establish connection to peer HA node. Can be "listen IP:port", "connect IP:port" or empty.

kern.cam.ctl.ha_link

Reports present state of connection between HA cluster nodes:

0 not configured;
1 configured but not established;
2 established.

kern.cam.ctl.ha_role

Specifies default role of this node:

0 primary;
1 secondary.

This role can be overridden on per-LUN basis using "ha_role" LUN option, so that for one LUN one node is primary, while for another -- another. Role change from primary to secondary for HA modes 0 and 2 closes backends, the opposite change -- opens. If there is no primary node (both nodes are secondary, or secondary node has no connection to primary one), secondary node(s) report Transitioning state. State with two primary nodes is illegal (split brain condition).

TUNABLE VARIABLES

The following variables are available as loader(8) tunables:

kern.cam.ctl.max_luns

Specifies the maximum number of LUNs we support, must be a power of 2. The default value is 1024.

kern.cam.ctl.max_ports

Specifies the maximum number of ports we support, must be a power of 2. The default value is 1024.

SEE ALSO

cfiscsi(4), cfumass(4), ctladm(8), ctld(8), ctlstat(8)

HISTORY

The **ctl** subsystem first appeared in FreeBSD 9.1.

AUTHORS

The **ctl** subsystem was originally written by Kenneth Merry <*ken@FreeBSD.org*>. Later work was done by

Alexander Motin <*mav@FreeBSD.org*>.