



PanCTL User Guide

The command-line client for administering VDURA storage systems

© 2002–2026 VDURA Inc. All rights reserved. VDURA, the VDURA logo, VDURA Data Platform, VeLO, VPOD, DirectFlow, PanView, PanMove, and Multi-Level Erasure Coding (MLEC) are trademarks or registered trademarks of VDURA, Inc. in the U.S. and/or other countries. All other trademarks, registered trademarks, trade names, company names, and service marks are the respective properties of their holders.

Parts of this product are derived from Berkeley Database software, licensed from Sleepycat Software. Parts of this product are derived from Berkeley BSD systems, licensed from the University of California. The BDB software is also BSD licensed, and carries the following copyrights:

- Copyright © 1990–2009 Oracle. All rights reserved.
- Copyright © 1990, 1993, 1994, 1995 The Regents of the University of California. All rights reserved.
- Copyright © 1995, 1996 The President and Fellows of Harvard University. All rights reserved.
- Copyright © 2000–2005 INRIA, France Telecom

Parts of this product are derived from gd Version 1.8.4, licensed by Boutell.Com, Inc.

Copyright © 1982, 1986, 1991, 1993 The Regents of the University of California. All rights reserved.

© UNIX System Laboratories, Inc.

All or some portions of this file are derived from material licensed to the University of California by American Telephone and Telegraph Co. or Unix System Laboratories, Inc. and are reproduced herein with the permission of UNIX System Laboratories, Inc.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Disclaimer

VDURA has carefully checked the information in this document and believes it to be accurate. However, VDURA assumes no responsibility for any inaccuracies that this document may contain. In no event will VDURA be liable for direct, indirect, special, exemplary, incidental, or consequential damages resulting from any defect or omission in this document, even if advised of the possibility of such damages.

Any alterations or inconsistent replications of this document cannot be considered accurate and are void. VDURA may not be able to support issues related to altered documents that are not approved by subject matter experts.

Document version 12.0.0-2026-10-15-18:23:37

1. Introduction	1
1.1 What PanCTL Is	1
1.2 How PanCTL Relates to the API and the Web UI	1
2. Getting Started	3
2.1 Prerequisites and Installation	3
2.2 Checking Your Version	3
3. Authentication and Sessions	4
3.1 Authentication Modes	4
3.2 Logging In (<code>auth login</code>)	4
3.3 Checking Your Identity (<code>auth whoami</code>)	6
3.4 Logging Out (<code>auth logout</code>)	6
3.5 API Tokens (<code>auth tokens</code>)	7
3.6 Using a Pre-Issued Token	7
3.7 Automatic Session Renewal	8
4. Managing Profiles (<code>profiles</code>)	9
5. Global Options	10
5.1 Persistent Flags	10
5.2 Environment Variables	10
5.3 Output Formats	11
5.4 Command Aliases	12
5.5 Exit Codes	12
6. Specifying Sizes and Quotas	14
7. Command Summary	15
8. Command Reference	20
8.1 <code>realm</code>	20
8.2 <code>nodes</code>	20
8.3 <code>storagesets</code>	21
8.4 <code>tenants</code>	22
8.5 <code>volumes</code>	23
8.6 <code>subnets</code>	24
8.7 <code>users</code>	25
8.8 <code>service-accounts</code>	26
8.9 <code>roles</code>	27
8.10 <code>s3</code>	28
8.11 <code>events</code>	29
8.12 <code>alerts</code>	30

8.13 audit	30
9. Shell Completion	32
10. Common Administrative Workflow	33
11. Troubleshooting	34

1. Introduction

1.1 What PanCTL Is

PanCTL is the official command-line client for administering a VDURA storage system, invoked as `panctl`. It is a single, statically linked executable that lets administrators manage realms, storage sets, nodes, tenants, volumes, subnets, users, service accounts, and S3 gateway configuration from a terminal or automation script, without needing the web UI.

PanCTL is organized as a tree of command **groups** (`nodes` , `volumes` , `tenants` , etc.), each with a consistent set of **verbs** (`list` , `get` , `create` , `update` , `delete` , plus resource-specific actions). Every command supports multiple output formats, including human-readable tables, JSON, and `kubect1 -style` JSONPath, so the same commands work equally well typed interactively or embedded in shell scripts.

1.2 How PanCTL Relates to the API and the Web UI

VDURA systems run a single backend REST service, commonly referred to as **PanAPI** (the underlying service is `vdura-api`). Both PanCTL and the VDURA web UI are independent clients of this same API; neither one is “more authoritative” than the other, and neither wraps the other:

- **PanCTL → PanAPI directly.** Every PanCTL command translates to one or more calls against a documented PanAPI resource (nodes, tenants, volumes, storage sets, and so on). There is no separate “CLI service” in between.
- **Same data, same rules, both ways.** Because PanCTL and the web UI talk to the same backend, changes made in one are immediately visible in the other. This is most obvious in the observability commands:
 - `panctl events list` shows the exact same event feed the web UI’s Events view shows.
 - `panctl alerts list` shows the exact same active-alert set behind the UI’s alert badge.
 - `panctl audit list` shows the exact same records behind the UI’s Audit Log.

- **Live validation, not a stale copy.** After you log in, PanCTL downloads a copy of the API's current OpenAPI specification from the appliance you just authenticated against and caches it locally. Before sending a request that creates or updates something, PanCTL checks your input against that cached specification. This means:
 - Obvious mistakes (a missing required field, a value of the wrong type) are reported instantly, with no network round-trip.
 - If the server's validation rules change (for example, a field becomes optional), PanCTL picks that up automatically the next time it refreshes its cache; you don't need to upgrade the CLI to get the new behavior.
 - PanCTL is deliberately permissive when it can't be sure: if it cannot confirm something is invalid (no cached spec, an unrecognized field, a network hiccup while refreshing), it sends the request anyway and lets the server make the final call.
- **Not a superset or replacement.** PanCTL does not expose every capability of the web UI (for example, it has no dashboards or visual topology views), and the UI does not expose every flag PanCTL supports (for example, pre-issued API tokens meant for automation). Treat them as two complementary front ends to the same system.

In short: **if you can see it in the UI, PanCTL is reading the same record; if you change it with PanCTL, the UI will show the change.**

2. Getting Started

2.1 Prerequisites and Installation

- A reachable VDURA appliance (PanAPI endpoint), typically on port `8000` .
- A tenant-qualified administrator username and password or a pre-issued API token (see [Section 3](#)).
- The `panctl` binary for your platform, placed anywhere on your `PATH` .

There is nothing to configure before first use; no config file is required to exist, and PanCTL will create one for you on first login.

2.2 Checking Your Version

```
panctl --version  
# or  
panctl -V
```

This prints the version and exits immediately; it does not contact the appliance.

3. Authentication and Sessions

3.1 Authentication Modes

PanCTL supports two distinct ways of authenticating:

Mode	How it works	Best for
Profile session	<code>panctl auth login</code> exchanges your username/password for an access token and refresh token, which are stored locally in a named profile . Later commands reuse this session automatically and renew it silently when it expires.	Interactive, day-to-day administration
Pre-issued API token	A long-lived token (created with <code>auth tokens create</code> or <code>service-accounts tokens create</code>) is passed directly via <code>--token</code> , <code>--token-stdin</code> , or the <code>PANCTL_API_TOKEN</code> environment variable. No profile is read or written.	Scripts, CI/CD, automation, service accounts

3.2 Logging In (`auth login`)

```
panctl auth login [server] [--server <addr>] [--username <user>] [--password-stdin] [--insecure] [--profile <name>]
```

Interactive example:

```
$ panctl auth login
Profile [default]:
Server [realm_dns_name]: 34.63.175.103
Username: finch/alice
Password:
Login succeeded
```

Username **are tenant-qualified**: `<tenant-short-name>/<username>` (e.g. `finch/alice`). A bare name such as `admin` logs in to the platform tenant. There is no separate `--tenant` flag on `login`; the tenant is always part of the username.

Server address resolution (first match wins): a positional argument or `--server` → the `PANCTL_API_SERVER` environment variable → the value already stored in the target profile → `http://localhost:8000`. A bare host/IP is filled in with `http://` and port `8000` if you don't specify them:

```
panctl auth login 34.63.175.103           # -> http://34.63.175.103:8000
panctl auth login --server 34.63.175.103:9000 # explicit port is kept
panctl auth login --server https://api.example.com # explicit https keeps port 443
```

Supplying a password non-interactively (avoid `--password`, which does not exist for `login`; use `--password-stdin` (below) or `PANCTL_PASSWORD`, see 5.2):

```
printf '%s' "$PASSWORD" | panctl auth login 34.63.175.103 --username finch/alice --password-stdin
```

Other login flags:

Flag	Purpose
<code>--profile <name></code>	Log in under a named profile instead of the default. On <code>login</code> specifically, a successful attempt also switches your <i>active</i> profile to this one.
<code>--insecure</code>	Skip TLS certificate verification for this login and all future commands run under this profile.

Flag	Purpose
<code>--server <addr></code>	The server address, as an alternative to the positional <code>[server]</code> argument (see the address resolution rules above).
<code>--username <user></code>	The tenant-qualified username to log in as (for example <code>finch/alice</code>).
<code>--password-stdin</code>	Read the password from standard input (see the example above).

3.3 Checking Your Identity (`auth whoami`)

```
$ panctl auth whoami
PROFILE:          lab
SERVER:           http://10.97.5.200:8000
USER ID:          u-1
TENANT ID:        t-1
IS PLATFORM ADMIN: false
```

Useful for confirming which server and account a profile is pointing at before you run something destructive.

3.4 Logging Out (`auth logout`)

```
panctl auth logout          # clears the active profile's session
panctl auth logout --profile lab # clears a specific profile
```

This removes the stored session locally (server, tokens, TLS setting); there is no server-side “logout” call, since the tokens simply expire on their own if left unused.

3.5 API Tokens (`auth tokens`)

Long-lived tokens you can hand to scripts or CI systems instead of a username/password.

Command	Purpose
<code>auth tokens create --name <label> --expires-in-days <n></code>	Create a new token (accepted lifetimes: 30, 90, 180, 365 days)
<code>auth tokens list [--active-only]</code>	List your tokens
<code>auth tokens get <token_id></code>	Show details of one token
<code>auth tokens revoke <token_id> [--reason "<text>"]</code>	Immediately invalidate a token

```
panctl auth tokens create --name ci-runner --expires-in-days 90
panctl auth tokens list --active-only
panctl auth tokens revoke jusKjSv7iJzMmaL9 --reason "rotated"

# Capture just the secret for use in a script:
TOKEN=$(panctl auth tokens create --name ci-runner --expires-in-days 90 --format 'jsonpath={.token}')
```

Important: the token secret is only ever shown once, at creation time. Store it somewhere safe immediately.

3.6 Using a Pre-Issued Token

```
export PANCTL_API_TOKEN=vdt1...
export PANCTL_API_SERVER=10.97.26.200
panctl volumes list
```

or per-command:

```
panctl volumes list --token-stdin < token.txt
```

In this mode, `PANCTL_API_SERVER` is required (there's no profile to fall back on), and nothing is written to disk. A rejected token produces a clear error and a distinct exit code (`3`), which is useful for detecting expired automation credentials in scripts.

3.7 Automatic Session Renewal

You generally never need to think about this: PanCTL renews a profile's session automatically, either just before it's known to have expired or transparently after a single authentication failure, and retries the original command once. If renewal itself fails, you'll see:

```
error: session expired. Run: panctl auth login --profile <name>
```

4. Managing Profiles (`profiles`)

Aliases: `profile`, `pr`, `p`

A **profile** is a named, saved connection (server + credentials) stored in `~/.panctl/profiles.toml`. Profiles let you switch between multiple VDURA systems (e.g., lab vs. production) without re-authenticating each time. All `profiles` commands are purely local; they never contact the appliance.

Command	Purpose
<code>profiles list</code>	List all saved profiles and which one is active
<code>profiles show [name]</code>	Show details of the active (or named) profile
<code>profiles use <name></code>	Switch the active profile

```
$ panctl profiles list
NAME      IN USE  SERVER                        USERNAME  AUTHENTICATED
default   false   http://localhost:8000         finch/alice true
gcp       true    https://gcp.example.com:8000 -          false

$ panctl profiles use gcp
now using profile "gcp"
```

There is no `profiles create` or `profiles delete` command: - **To create a profile**, log in with a new name: `panctl auth login --profile <name> <server>`. - **To remove a profile**, log out of it: `panctl auth logout --profile <name>`.

You can also point a single command at a different profile without switching your default or override it entirely via environment variable:

```
panctl --profile prod volumes list
PANCTL_PROFILE=prod panctl volumes list
```

5. Global Options

5.1 Persistent Flags

These flags are available on every PanCTL command:

Flag	Purpose
<code>--profile <name></code>	Use a specific profile for this command only
<code>--format , -o <format></code>	Set the output format (see below)
<code>--no-color</code>	Disable colored table output
<code>--token <value></code>	Use a pre-issued API token for this command
<code>--token-stdin</code>	Read a pre-issued API token from standard input
<code>--version , -V</code>	Print the version and exit

5.2 Environment Variables

Variable	Effect
<code>PANCTL_PROFILE</code>	Selects the active profile for one invocation without changing the saved default
<code>PANCTL_API_SERVER</code>	Appliance address (used as a fallback in login mode, required in token mode)
<code>PANCTL_USERNAME</code>	Default username for <code>auth login</code>
<code>PANCTL_PASSWORD</code>	Default password for <code>auth login</code> (prefer <code>--password-stdin</code> ; environment variables can leak via process listings)
<code>PANCTL_API_TOKEN</code>	Pre-issued token for the whole session (requires <code>PANCTL_API_SERVER</code>)

Variable	Effect
PANCTL_API_INSECURE	Set to 1 / true / yes / on to skip TLS verification in token mode

5.3 Output Formats

Use `--format` (or the shorter `-o`) to control how results are printed:

Format	Description
table	Aligned, bolded-header columns. Default when output goes to a terminal.
plain	Aligned columns without color. Default when output is piped or redirected.
raw	The API's response body, unmodified.
json	The response body, pretty-printed with 2-space indentation.
jsonpath=<expression>	Extracts specific fields using <code>kubectl -style JSONPath</code> , ideal for scripting.

JSONPath examples:

```
panctl volumes list --format 'jsonpath={.volumes[*].pathname}'

panctl volumes list --format \
'jsonpath={range .volumes[*]}{.id}{"\t"}{.pathname}{"\n"}{end}'

panctl nodes list --format \
'jsonpath={.nodes[?(@.role=="vosd" || @.role=="vhost" && @.storageset_id==0)].dataserial}'
```

Supported JSONPath features include field access, bracket-quoted keys (for hyphenated names, e.g. `{['storage-admin']}`), array indices including `[-1]`, wildcards, filter expressions (`==`, `!=`, `<`, `<=`, `>`, `>=`, joined with `&&` / `||`), and `{range ...}{end}` loops. It does **not** support recursive descent (`..`) or listing an object's keys; for that, pipe `--format json` into `jq 'keys'`.

A `yaml` format is planned but not yet available in the current release.

5.4 Command Aliases

Most command groups have one or more short aliases (spelled out exactly, no partial-name matching):

Group	Aliases
<code>auth</code>	<code>au</code> , <code>a</code>
<code>profiles</code>	<code>profile</code> , <code>pr</code> , <code>p</code>
<code>realm</code>	<code>rlm</code>
<code>roles</code>	<code>role</code> , <code>ro</code> , <code>r</code>
<code>volumes</code>	<code>volume</code> , <code>vol</code> , <code>v</code>
<code>nodes</code>	<code>node</code> , <code>no</code> , <code>n</code>
<code>storagesets</code>	<code>storage</code> , <code>ss</code> , <code>s</code>
<code>tenants</code>	<code>tenant</code> , <code>tn</code> , <code>tx</code> , <code>t</code>
<code>users</code>	<code>user</code> , <code>us</code> , <code>u</code>
<code>service-accounts</code>	<code>sa</code>
<code>s3</code>	<i>(none)</i>
<code>events</code>	<code>ev</code> , <code>e</code>
<code>alerts</code>	<code>al</code>
<code>subnets</code>	<code>subnet</code> , <code>sub</code> , <code>sn</code>
<code>audit</code>	<code>aud</code> , <code>ad</code>

5.5 Exit Codes

Code	Meaning
<code>0</code>	Success

Code	Meaning
1	General failure (API error, validation error)
2	Unknown output format or malformed JSONPath expression
3	A pre-issued API token was rejected

6. Specifying Sizes and Quotas

Flags like `--hard-quota`, `--soft-quota`, and `--quota` accept a number with a size suffix. **Decimal (SI) and binary (IEC) suffixes are distinct**; don't mix them up:

Suffix	Meaning	Example
B (or no suffix)	Bytes	<code>--quota=500B</code>
KB / MB / GB / TB / PB	Decimal (powers of 1000)	<code>--quota=10GB</code> → 10,000,000,000 bytes
KiB / MiB / GiB / TiB / PiB	Binary (powers of 1024)	<code>--quota=10GiB</code> → 10,737,418,240 bytes

Suffixes are case-insensitive, and a space before the suffix is fine (`10 GB`). Fractional values (`1.5GB`) are rejected; use a smaller unit instead (`1500MB`).

7. Command Summary

Every PanCTL command described in the Command Reference chapter that follows, grouped the same way as its sections. Each command links to its group's own section. The `auth` commands are covered in [Chapter 3](#), the `profiles` commands in [Chapter 4](#), and `completion` in [Chapter 9](#); they are not repeated in the table below.

Command	Purpose
<code>realm get</code>	Show realm-wide state and configuration
<code>nodes list</code>	List all nodes
<code>nodes get <dataserial></code>	Show one node's details
<code>nodes reboot <dataserial></code>	Reboot a node
<code>storagesets list</code>	List all storage sets
<code>storagesets get <id></code>	Show one storage set
<code>storagesets create --name <name> [--spares-requested <n>]</code>	Create a storage set
<code>storagesets update <id> -f <file></code>	Update a storage set
<code>storagesets delete <id></code>	Delete a storage set
<code>storagesets add-storage --storage-set-id <id> --dataserials <a,b,c> [options]</code>	Add nodes to a storage set
<code>tenants list</code>	List all tenants
<code>tenants get <tenant_id></code>	Show one tenant
<code>tenants create --name <name> --short-name <name> --email <addr> [options]</code>	Create a tenant
<code>tenants update <tenant_id> -f <file></code>	Update a tenant
<code>tenants delete <tenant_id></code>	Delete a tenant (cascades to its volumes)

Command	Purpose
<code>tenants assign-storagesets --tenant <ref> --storageset-id <id> --quota <size></code>	Grant/resize a tenant's capacity in a storage set
<code>tenants unassign-storagesets --tenant <ref> --storageset-id <id></code>	Remove a tenant's access to a storage set
<code>tenants list-storagesets --tenant <ref></code>	List storage sets assigned to a tenant
<code>tenants get-storageset --tenant <ref> --storageset-id <id></code>	Show one tenant/storage-set assignment
<code>volumes list</code>	List volumes
<code>volumes get <volume_id></code>	Show one volume
<code>volumes create --name <path> --storageset-id <id> --hard-quota <size> [options]</code>	Create a volume
<code>volumes update <volume_id> -f <file></code>	Update quotas, recovery priority, etc.
<code>volumes rename <volume_id> -f <file></code>	Rename (change the path of) a volume
<code>volumes delete <volume_id></code>	Delete a volume
<code>subnets list [--tenant <ref>]</code>	List subnets
<code>subnets get <subnet_id></code>	Show one subnet
<code>subnets create --tenant <ref> --name <name> --cidr <addr/prefix> [options]</code>	Create a subnet
<code>subnets update <subnet_id> [options]</code>	Update a subnet
<code>subnets delete <subnet_id></code>	Delete a subnet
<code>subnets attachments list <subnet_id></code>	List client attachments on a subnet
<code>subnets attachments get <subnet_id> <attachment_id></code>	Show one attachment
<code>subnets attachments create <subnet_id> --underlay-ip <ip> [options]</code>	Attach a client

Command	Purpose
<code>subnets attachments delete <subnet_id> <attachment_id></code>	Remove an attachment
<code>users list [--tenant <ref>]</code>	List users in a tenant
<code>users get <user_id> [--tenant <ref>]</code>	Show one user
<code>users create --tenant <ref> --username <name> [options]</code>	Create a user
<code>users update <user_id> --tenant <ref> -f <file></code>	Update a user
<code>users delete <user_id> --tenant <ref></code>	Delete a user
<code>users get-roles --user-id <id> --tenant <ref></code>	Show a user's roles
<code>users set-roles --user-id <id> --tenant <ref> --roles <a,b></code>	Replace a user's roles
<code>service-accounts list</code>	List service accounts
<code>service-accounts get <id></code>	Show one service account
<code>service-accounts create --name <name> [--display-name <name>] [--description <text>]</code>	Create a service account
<code>service-accounts update <id> [options]</code>	Update display name/description
<code>service-accounts delete <id></code>	Delete a service account
<code>service-accounts get-roles --sa-id <id></code>	Show a service account's roles
<code>service-accounts set-roles --sa-id <id> --roles <a,b></code>	Replace a service account's roles
<code>service-accounts tokens create --sa-id <id> --name <label> --expires-in-days <n></code>	Mint an API token for the service account
<code>service-accounts tokens list --sa-id <id> [-active-only]</code>	List a service account's tokens

Command	Purpose
<code>service-accounts tokens revoke <token_id> --sa-id <id> [--reason <text>]</code>	Revoke a token
<code>roles list</code>	Show all roles and their permitted actions
<code>s3 enabled get / s3 enabled set <true\ false></code>	Enable/disable the S3 gateway realm-wide
<code>s3 port-range get / s3 port-range set <start> <end></code>	View/set the gateway's port range
<code>s3 prefix get / s3 prefix set <path></code>	View/set the realm-wide path prefix
<code>s3 get-config --tenant <ref></code>	Show a tenant's S3 configuration
<code>s3 set-config --tenant <ref> [options]</code>	Update a tenant's S3 configuration
<code>s3 dns <mode> --tenant <ref></code>	Set how S3 DNS is published for a tenant
<code>s3 tls renew --tenant <ref></code>	Renew a tenant's S3 TLS certificate
<code>s3 instances list --tenant <ref></code>	List S3 gateway instances
<code>s3 instances create <iid> <dataserial> --tenant <ref></code>	Create an S3 gateway instance on a node
<code>s3 instances start <iid> --tenant <ref></code>	Start an S3 gateway instance
<code>s3 instances stop-all --tenant <ref></code>	Stop every S3 gateway instance for a tenant
<code>s3 instances delete-all --tenant <ref> --yes</code>	Delete every S3 gateway instance for a tenant
<code>s3 buckets create <name> --tenant <ref></code>	Create an S3 bucket
<code>s3 buckets list --tenant <ref></code>	List S3 buckets
<code>s3 credentials reveal <access_key> --tenant <ref></code>	Show the current secret for an access key
<code>s3 credentials rotate <access_key> --tenant <ref></code>	Rotate the secret for an access key

Command	Purpose
<code>events list [--limit <n>] [--tenant <ref>]</code>	List recent system events
<code>alerts list</code>	List currently active alerts
<code>audit list [options]</code>	List administrative audit log entries
<code>audit get <request_id></code>	Show one audit log entry's detail

8. Command Reference

Unless noted otherwise, all `list` / `get` commands are read-only and safe to run at any time.

8.1 realm

Alias: `rlm`. The realm is the VDURA cluster as a whole; there is only one, so this group has a single, argument-free verb.

Command	Purpose
<code>realm get</code>	Show realm-wide state and configuration

```
panctl realm get
panctl realm get -o 'jsonpath={.realm.state}'
```

Requires the `platform-admin` role. Fields include `state`, `locked`, `clientmode_current`, `heartbeat_period_seconds`, `volume_max_vols_realm`, and more.

8.2 nodes

Aliases: `node`, `no`, `n`. Physical/virtual storage nodes in the realm, identified by their `dataserial` (not a name).

Command	Purpose
<code>nodes list</code>	List all nodes
<code>nodes get <dataserial></code>	Show one node's details
<code>nodes reboot <dataserial></code>	Reboot a node

```
$ panctl nodes list
DATASERIAL      IP ADDRESS    STATUS  ROLE   STORAGESET ID  CAPACITY BYTES
109a58fc3162ef  10.97.5.237  online  vosd   0               55650025472
```

```
panctl nodes get 109a58fc3162ef
panctl nodes reboot 109a58fc3162ef
```

Notes: - A `STORAGESSET ID` of `0` means the node isn't assigned to a storage set yet. - `reboot` sends the request and returns immediately; there is no confirmation prompt.

8.3 storagesets

Aliases: `storageset`, `ss`, `s`. Pools of nodes that back tenant capacity and volumes.

Command	Purpose
<code>storagesets list</code>	List all storage sets
<code>storagesets get <id></code>	Show one storage set
<code>storagesets create --name <name> [--spares-requested <n>]</code>	Create a storage set
<code>storagesets update <id> -f <file></code>	Update a storage set
<code>storagesets delete <id></code>	Delete a storage set
<code>storagesets add-storage --storageset-id <id> --dataserials <a,b,c> [--mode add_spare\ migration] [--spares-requested <n>]</code>	Add nodes to a storage set

```
panctl storagesets create --name "Demo Set" --spares-requested 4
panctl storagesets get 4
panctl storagesets add-storage --storageset-id 4 \
  --dataserials 109a58fc3162ef,3f8889d66162f5 --spares-requested 2
panctl storagesets delete 4
```

Notes: - `--dataserials` cannot be left empty on `add-storage`; an empty list would mean "add every unassigned node in the realm," so PanCTL blocks it to prevent an accidental mass operation. - `add-storage` returns immediately, but absorbing new nodes into the set happens asynchronously in the background. Poll to confirm completion: `bash until [ "$(panctl storagesets get 4 -o 'jsonpath={.storageset.nodes_online}')" = "6" ]; do sleep 5; done`

8.4 tenants

Aliases: `tenant`, `tn`, `tx`, `t`. Tenants are the isolation boundary for users, capacity, and volumes.

Command	Purpose
<code>tenants list</code>	List all tenants
<code>tenants get <tenant_id></code>	Show one tenant
<code>tenants create --name <name> --short-name <name> --email <addr> [--auth-mode local pam] [--network ...]</code>	Create a tenant
<code>tenants update <tenant_id> -f <file></code>	Update a tenant
<code>tenants delete <tenant_id></code>	Delete a tenant (cascades to its volumes)
<code>tenants assign-storagesets --tenant <ref> --storageset-id <id> --quota <size></code>	Grant/resize a tenant's capacity in a storage set
<code>tenants unassign-storagesets --tenant <ref> --storageset-id <id></code>	Remove a tenant's access to a storage set
<code>tenants list-storagesets --tenant <ref></code>	List storage sets assigned to a tenant
<code>tenants get-storageset --tenant <ref> --storageset-id <id></code>	Show one tenant/storage-set assignment

```
panctl tenants create --name FinTech --short-name fintech --email admin@fintech.example.com
panctl tenants assign-storagesets --tenant fintech --storageset-id 4 --quota 10GB
panctl tenants list-storagesets --tenant fintech
panctl tenants delete <tenant_id>
```

Notes: - Multi-tenancy is not supported on InfiniBand. `tenants create` is refused with a `409 Conflict` on a realm whose clients connect over InfiniBand, because tenant isolation needs VLAN or VXLAN encapsulation. Ethernet realms are not affected, and a realm that already runs without tenants is unchanged. - `--tenant` accepts either the tenant's UUID or its short name; PanCTL resolves short names for you. - `--short-name` is 3-10 alphanumeric characters and cannot be changed after creation. - `assign-storagesets` acts as an "upsert": if the tenant already has capacity in that storage set, this call

resizes the quota instead of failing. – Creating a volume successfully is not, by itself, proof that `assign-storagesets` succeeded; confirm assignment separately with `list-storagesets` if you need to be certain. – `tenants delete` has **no confirmation prompt** and deletes the tenant’s volumes along with it.

8.5 volumes

Aliases: `volume`, `vol`, `v`. File-system volumes within your own tenant.

Command	Purpose
<code>volumes list</code>	List volumes
<code>volumes get <volume_id></code>	Show one volume
<code>volumes create --name <path> --storageset-id <id> --hard-quota <size> [options]</code>	Create a volume
<code>volumes update <volume_id> -f <file></code>	Update quotas, recovery priority, etc.
<code>volumes rename <volume_id> -f <file></code>	Rename (change the path of) a volume
<code>volumes delete <volume_id></code>	Delete a volume

```
panctl volumes create --name /data --storageset-id 1 --hard-quota 10GiB \
  --soft-quota 8GiB --raid-type raid5+ --recovery-priority 1

echo '{"new_pathname":"/archive/volume1"}' | panctl volumes rename 3 -f -
echo '{"hard_quota_bytes":21474836480}' | panctl volumes update 3 -f -
panctl volumes delete 3
```

create options:

Flag	Notes
<code>--name <path></code>	Required. A leading <code>/</code> is added automatically if missing.
<code>--storageset-id <id></code>	Required.

Flag	Notes
<code>--hard-quota <size></code>	Required by PanCTL (even though the API treats it as optional) so a limit is always chosen deliberately. <code>0</code> means unlimited.
<code>--soft-quota <size></code>	Optional. <code>0</code> means unlimited.
<code>--raid-type <layout></code>	Optional, cannot be changed after creation .
<code>--efsa-mode <mode></code>	Optional.
<code>--recovery-priority <0-100></code>	Optional; lower number = higher priority.

Notes: - `volumes create` always creates the volume in your own tenant; there's no `--tenant` flag on this command. - Before creating a volume, make sure: the target storage set has enough online capacity for the chosen RAID layout, your tenant has capacity assigned there (see `tenants assign-storagesets`), and your tenant has a subnet with an available address. - `rename` only changes the volume's path; `update` changes everything else (quotas, priority) but not the path or RAID type.

8.6 subnets

Aliases: `subnet`, `sub`, `sn`. Per-tenant IP address ranges used by clients to reach volumes, plus their client **attachments**.

Command	Purpose
<code>subnets list [--tenant <ref>]</code>	List subnets
<code>subnets get <subnet_id></code>	Show one subnet
<code>subnets create --tenant <ref> --name <name> --cidr <addr/prefix> [options]</code>	Create a subnet
<code>subnets update <subnet_id> [options]</code>	Update a subnet
<code>subnets delete <subnet_id></code>	Delete a subnet
<code>subnets attachments list <subnet_id></code>	List client attachments on a subnet
<code>subnets attachments get <subnet_id> <attachment_id></code>	Show one attachment

Command	Purpose
<code>subnets attachments create <subnet_id> --underlay-ip <ip> [options]</code>	Attach a client
<code>subnets attachments delete <subnet_id> <attachment_id></code>	Remove an attachment

```
panctl subnets create --tenant fintech --name fintech-net --cidr 10.71.0.0/16 \
  --gateway 10.71.255.254 --mtu 9000 --range-start 10.71.0.1 --range-end 10.71.1.255

panctl subnets attachments create <subnet_id> --underlay-ip 10.128.9.0 --hostname client01
panctl subnets update <subnet_id> --mtu 1500
panctl subnets delete <subnet_id>
```

Notes: - **Size your CIDR generously**; use at least a /16 for a tenant subnet, not a /24 . Internal addressing places storage-node addresses 256 addresses into the range, and a /24 will silently strand them with no error message. - Subnets are identified by UUID, not a numeric ID. - Attachment `create` / `delete` are tenant-scoped operations; even a platform administrator must be logged in as a user of that tenant to manage attachments (a platform-admin session gets a permissions error, even though it can create/delete the subnet itself). - `--segment-id` , `--tunnel-udp-port` , and `--addr-external` can only be set at creation time.

8.7 users

Aliases: `user` , `us` , `u` . Human user accounts. All users are platform-level; there are no separate tenant users.

Command	Purpose
<code>users list [--tenant <ref>]</code>	List users in a tenant
<code>users get <user_id> [--tenant <ref>]</code>	Show one user
<code>users create --tenant <ref> --username <name> [options]</code>	Create a user
<code>users update <user_id> --tenant <ref> -f <file></code>	Update a user

Command	Purpose
<code>users delete <user_id> --tenant <ref></code>	Delete a user
<code>users get-roles --user-id <id> --tenant <ref></code>	Show a user's roles
<code>users set-roles --user-id <id> --tenant <ref> --roles <a,b></code>	Replace a user's roles

```
panctl users create --tenant fintech --username jsmith --email jsmith@fintech.example.com
panctl users set-roles --user-id <user_id> --tenant fintech --roles platform-read-only
panctl users get-roles --user-id <user_id> --tenant fintech
panctl users delete <user_id> --tenant fintech
```

Notes: - If `--tenant` is omitted, PanCTL infers it from the short-name prefix of your own logged-in username. It's safer to always pass `--tenant` explicitly in scripts. - `set-roles` **replaces the entire role set**, it does not add to it. Pass `--roles ''` to revoke all roles from a user. - There are no tenant roles. The only roles are `platform-admin`, `platform-read-only`, and `storage-admin`, and every user is a platform-level user. See [8.9](#) for what each role allows.

8.8 service-accounts

Alias: `sa`. Non-human accounts intended for automation, scoped to the platform rather than a tenant.

Command	Purpose
<code>service-accounts list</code>	List service accounts
<code>service-accounts get <id></code>	Show one service account
<code>service-accounts create --name <name> [--display-name <name>] [--description <text>]</code>	Create a service account
<code>service-accounts update <id> [options]</code>	Update display name/description
<code>service-accounts delete <id></code>	Delete a service account
<code>service-accounts get-roles --sa-id <id></code>	Show a service account's roles

Command	Purpose
<code>service-accounts set-roles --sa-id <id> --roles <a,b></code>	Replace a service account's roles
<code>service-accounts tokens create --sa-id <id> --name <label> --expires-in-days <n></code>	Mint an API token for the service account
<code>service-accounts tokens list --sa-id <id> [-active-only]</code>	List a service account's tokens
<code>service-accounts tokens revoke <token_id> --sa-id <id> [--reason <text>]</code>	Revoke a token

```

panctl service-accounts create --name ci-runner --display-name "CI Runner"
panctl service-accounts set-roles --sa-id <sa_id> --roles storage-admin

TOKEN=$(panctl service-accounts tokens create --sa-id <sa_id> --name ci --expires-in-days 90 -o
    jsonpath='{.token}')
PANCTL_API_SERVER=<server> panctl auth whoami --token "$TOKEN"

```

Notes: – A service account's `--name` is immutable after creation; only display name and description can be updated. – Available roles: `platform-admin`, `platform-read-only`, and `storage-admin`. See [8.9](#) for what each one allows. – This is the recommended pattern for giving an automation pipeline its own credentials, separate from any individual administrator's login.

8.9 roles

Aliases: `role`, `ro`, `r`. A read-only catalogue of what each role is permitted to do.

There are no tenant roles. A user or service account holds one or more of these three platform roles:

Role	What it allows
<code>platform-admin</code>	Full control. Create, update, and delete every resource (tenants, users, service accounts, StorageSets, volumes, S3 configuration, and more), change role assignments, and switch into a tenant.

Role	What it allows
<code>platform-read-only</code>	List and view every resource, and create or delete API tokens.
<code>storage-admin</code>	Everything <code>platform-read-only</code> allows, plus create, update, delete, and run actions on StorageSets and volumes.

Command	Purpose
<code>roles list</code>	Show all roles and their permitted actions

```
$ panctl roles list
ROLE          RESOURCE      ACTIONS
platform-admin volumes        list, view, create, update, delete, action
...
platform-read-only volumes        list, view
...
storage-admin  volumes        list, view, create, update, delete, action
...
```

The sample output is abbreviated; the command lists every resource for every role, one row per role and resource pair.

Useful for confirming exactly what a role grants before assigning it to a user or service account.

8.10 `s3`

The S3-compatible object gateway. Some commands are **realm-wide** (platform administrators only) and some are **per-tenant** (`--tenant <ref>`).

Realm-wide configuration:

Command	Purpose
<code>s3 enabled get</code> / <code>s3 enabled set <true false></code>	Enable/disable the S3 gateway realm-wide

Command	Purpose
<code>s3 port-range get / s3 port-range set <start> <end></code>	View/set the gateway's port range
<code>s3 prefix get / s3 prefix set <path></code>	View/set the realm-wide path prefix

Per-tenant configuration:

```
panctl s3 get-config --tenant fintech
panctl s3 set-config --tenant fintech --enabled true --prefix /vol1/fintech \
  --owner alice --group staff --dir-mode 0755 --obj-mode 0644
panctl s3 dns panfs --tenant fintech
panctl s3 tls renew --tenant fintech
```

Instances (gateway processes running on specific nodes, identified by an instance ID `iid`):

```
panctl s3 instances list --tenant fintech
panctl s3 instances create abc12 <dataserial> --tenant fintech
panctl s3 instances start abc12 --tenant fintech
panctl s3 instances stop-all --tenant fintech
panctl s3 instances delete-all --tenant fintech --yes
```

Buckets and credentials:

```
panctl s3 buckets create archive-2026 --tenant fintech
panctl s3 buckets list --tenant fintech
panctl s3 credentials reveal <access_key> --tenant fintech
panctl s3 credentials rotate <access_key> --tenant fintech
```

Notes: - `s3 instances delete-all` requires `--yes` (this is the only destructive command in PanCTL that enforces an explicit confirmation flag). - `credentials reveal` returns the current secret without changing it; `credentials rotate` generates a brand-new secret (the old one stops working immediately). Both print the secret in plain text, so handle the output carefully.

8.11 events

Aliases: `ev`, `e`. Recent system events (read-only).

```
panctl events list
panctl events list --limit 25
panctl events list --tenant fintech
```

Shows up to 200 of the most recent events (`--limit` accepts 1–200); this is the exact same feed shown in the web UI.

8.12 alerts

Alias: `al` . Currently active alerts (read-only, no flags).

```
panctl alerts list
```

This shows a live snapshot of active alerts (not a historical log), the same set that drives the alert badge in the web UI.

8.13 audit

Aliases: `aud` , `ad` . The administrative audit log (read-only).

```
panctl audit list
panctl audit list --tenant fintech --since 2h
panctl audit list --user <user_id> --only create,delete
panctl audit list --since 2026-09-04T22:00:00Z --until 2026-09-04T23:59:59Z
panctl audit get <request_id>
```

Flag	Purpose
<code>--tenant <ref></code>	Filter to one tenant
<code>--user <user_id></code>	Filter to one user
<code>--only <a,b> / --skip <a,b></code>	Include or exclude specific action types
<code>--since / --until</code>	Relative (<code>2h</code> , <code>2d</code>) or absolute (RFC-3339) time bounds

Flag	Purpose
<code>--limit <n></code>	Maximum records to return (no upper cap, unlike <code>events</code>)

`audit get <request_id>` drills into a single request; for a failed request, it also prints diagnostic details and relevant backend log lines.

9. Shell Completion

PanCTL can generate tab-completion scripts for common shells:

```
panctl completion bash|zsh|fish|powershell
```

Bash example:

```
source <(panctl completion bash)
```

Zsh needs `compinit` initialized first:

```
autoload -U compinit; compinit  
source <(panctl completion zsh)
```

Completion is context-aware: it will suggest live values (tenant IDs, storage-set IDs, node dataserials, etc.) by making a quick, silent lookup against your active session, in addition to suggesting fixed values like `--format` options and role names.

10. Common Administrative Workflow

A typical first-time setup, from an empty system to a usable tenant volume:

```
# 1. Log in as a platform administrator
panctl auth login <server> --username admin

# 2. Survey the realm
panctl nodes list
panctl realm get

# 3. Create a storage set and add nodes to it
panctl storagesets create --name "Primary"
panctl storagesets add-storage --storage-set-id <id> --data-serials <ds1>,<ds2>,<ds3>
# add-storage is asynchronous: wait until the nodes are absorbed (see 8.3 for a polling
  example) before continuing

# 4. Create a tenant and give it capacity
panctl tenants create --name FinTech --short-name fintech --email admin@fintech.example.com
panctl tenants assign-storagesets --tenant fintech --storage-set-id <id> --quota 10TB

# 5. Give the tenant network access
panctl subnets create --tenant fintech --name fintech-net --cidr 10.71.0.0/16 --gateway
  10.71.255.254

# 6. Create a tenant administrator
panctl users create --tenant fintech --username jsmith --email jsmith@fintech.example.com
panctl users set-roles --user-id <user_id> --tenant fintech --roles admin

# 7. Log in as the tenant admin and create a volume
panctl auth login <server> --username fintech/jsmith --profile fintech
panctl --profile fintech volumes create --name /data --storage-set-id <id> --hard-quota 5TB
```

11. Troubleshooting

Symptom	Likely Cause / Fix
<code>error: profile "<name>" not authenticated.</code> Run: <code>panctl auth login ...</code>	The session for that profile has never logged in or has been logged out. Run <code>auth login</code> again.
<code>error: profile "<name>" not found in</code> <code>~/.panctl/profiles.toml</code>	Check for a typo in <code>--profile</code> or <code>PANCTL_PROFILE</code> , or the profile was never created.
<code>error: -f <file\ -> is required</code>	This command needs a JSON body; pass a file path or <code>-</code> to read from standard input.
<code>error: <field> is required</code>	A required positional argument or flag is missing.
<code>error: session expired. Run: panctl auth</code> <code>login --profile <name></code>	Automatic renewal failed (e.g., the refresh token itself expired). Log in again.
<code>dial tcp ...: connection refused</code>	The appliance is unreachable; check the server address, network path, and firewall rules.
An <code>error:</code> message with a raw JSON body	This is the API's own error response passed through unmodified. Pipe it through <code>jq</code> for easier reading, e.g. <code>panctl volumes create ... 2>&1 jq .</code>

This guide reflects the command set and behavior of PanCTL as of the version documented in the `panctl` repository. Run `panctl <command> --help` for the most current flag list on your installed version.