PanCTL User Guide

The command-line client for administering VDURA storage systems

Documentation for PanFS 12.0.0

© 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.
3. 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 12.0.0-2026-11-12-11:47:20

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
2.3 Interactive Sessions 3

3. Authentication and Sessions 6
3.1 Authentication Modes 6
3.2 Logging In (`auth login`) 6
3.3 Checking Your Identity (`auth whoami`) 9
3.4 Logging Out (`auth logout`) 9
3.5 API Tokens (`auth tokens`) 10
3.6 Using a Pre-Issued Token 10
3.7 Automatic Session Renewal 11

4. Managing Profiles (`profiles`) 12
5. Global Options 13
5.1 Persistent Flags 13
5.2 Environment Variables 14
5.3 Output Formats 15
5.4 Command Aliases 17
5.5 Exit Codes 18
5.6 Naming Items 19
5.7 Request Bodies 19
5.8 Narrowing a List (`--filter`) 20
5.9 Waiting for a Condition (`--wait-for`) 21
5.10 Destructive Commands Ask First 23
5.11 Across Tenants 23
5.12 Output Streams and Paging 24

6. Specifying Sizes and Quotas 25
7. Command Summary 26
8. Command Reference 32
8.1 `realm` 32
8.2 `nodes` 32
8.3 `storagesets` 33
8.4 `tenants` 35

8.5 `volumes` 36
8.6 `subnets` 38
8.7 `users` 40
8.8 `service-accounts` 42
8.9 `roles` 44
8.10 `s3` 45
8.11 `fs-users` 48
8.12 `events` 49
8.13 `alerts` 49
8.14 `audit` 50
8.15 `log-collections` 51
8.16 `api` 53

9. Shell Completion 55
10. Common Administrative Workflow 56
11. Troubleshooting 57
12. Scripting 59
12.1 Take the Id from the Create 59
12.2 Find Something You Did Not Just Create 59
12.3 One Line per Row 59
12.4 Empty Results and Several Matches 60
12.5 Collect and Fetch Logs Unattended 60
12.6 Run Unattended 60

1. Introduction

This chapter covers What PanCTL Is and How PanCTL Relates to the API and the Web UI.

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, StorageSets, 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 `kubectl` -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, StorageSets, 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

This chapter covers Prerequisites and Installation, Checking Your Version, and Interactive Sessions.

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.

2.3 Interactive Sessions

Running `panctl` with no arguments on a terminal opens an interactive session. Every other invocation works as it always has.

```
$ panctl
[lab] platform/admin> volumes list
[lab] platform/admin> /format json
[lab] platform/admin> volumes get vol-1
[lab] platform/admin> /exit
```

A session opens only when no command was named and both standard input and standard output are terminals. `panctl --profile lab` opens a session on the profile `lab` because the flag takes the name as its value. If terminal detection gets it wrong, as it can in a terminal multiplexer, an IDE terminal, or `ssh` without a pseudo-terminal, `panctl shell` forces a session.

The prompt and banner. The default prompt is `[<profile>] <username>>`. It carries both the profile and the authenticated user because profiles can share usernames and one username can serve several appliances. The server, profile, user, and PanCTL version are written once to standard error when the session opens, so a captured session holds command results alone. `(expired)` appears in the prompt only after a request has actually been answered with `401`, so its absence does not mean the session is valid. Set `PANCTL_PROMPT` to change the format. The fields are `{profile}`, `{username}`, `{server}`, and `{marker}`, and an unknown field is refused when the session opens.

Commands. Anything you type without a leading `/` is an ordinary PanCTL command, with the same flags, output, and exit handling as one-shot mode. Quoting is shell-like (single quotes are literal, double quotes group, and a backslash escapes), but there is no expansion, globbing, piping, or redirection, so `|` and `>` reach the command as ordinary text. A failing command reports itself and returns you to the prompt. Commands that ask for confirmation work as usual; see 5.10.

Session command	Effect
`/profile <name>`	Use another profile for the following commands. Nothing is written to disk; `profiles use`, typed without the slash, is still the persistent form.
`/format <value>`	Set the output format, using exactly the values `--format` accepts. A command's own `--format` wins for that command.
`/help`	List the session commands
`/clear`	Clear the screen
`/exit`, `/quit`	Leave the session

No session tenant. A session carries no tenant of its own because `--tenant` is not uniform across the command tree, and a sticky one would be honored by some commands and quietly ignored by others. Name the tenant on each command with `--tenant`, as in one-shot mode.

Keys and history. Tab completes commands and values from the same source the shell completion uses (see Chapter 9). The Up and Down keys recall earlier lines, which are kept in `~/.panctl/history` (created readable by you alone). Ctrl-C once cancels the running command and returns you to the prompt, and pressing it twice within a second leaves the session, as do Ctrl-D and `/exit`. `auth login` typed inside a session reads its password through the session, and interrupting it abandons that login only.

Exit status. A session exits `0` on any clean exit, whatever the commands inside it did, because the code describes the session and not the last command. Use one-shot mode when you need a command's own exit code.

3. Authentication and Sessions

This chapter covers the following topics.

3.1 Authentication Modes

PanCTL supports two distinct ways of authenticating:

Mode	How it works	Best for
Profile session	`panctl auth login` 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 `auth tokens create` or `service-accounts tokens create`) is passed directly via `--token`, `--token-stdin`, or the `PANCTL_API_TOKEN` 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
```

Usernames are tenant-qualified: `<tenant-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
```

PanWeb also reverse-proxies the API on port `443`, so `panctl auth login https://34.63.175.103:443 --insecure` reaches the same paths as port `8000` does directly. Write the `https://` form out in full because nothing is filled in for it, and `--insecure` stays a deliberate choice since it persists to the profile.

Choosing where the password comes from. There are four sources, and each exposes the password differently:

Source	What it exposes
`--password <value>`	The OS process listing and your shell history
`--password-stdin`	Nothing; the recommended path for scripts
`PANCTL_PASSWORD`	The process environment, CI logs, and shell configuration
The hidden prompt	Nothing, but it needs a terminal

The first one supplied wins, in the order shown, and it skips the prompt. `--password` and `--password-stdin` are two spellings of one secret, so passing both is refused before anything is sent.

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

A successful login prints `Login succeeded` to standard error and nothing to standard output.

As its last step, after it reports success, login downloads the API's OpenAPI specification (see 1.2) and caches it. If that download fails, the login still succeeds, a `warning:` line appears on standard error, and the exit status stays `0`. Commands then run without local request validation until the next successful login.

Other login flags:

Flag	Purpose
`--profile <name>`	Log in under a named profile instead of the default. On `login` specifically, a successful attempt also switches your *active* profile to this one.
`--insecure`	Skip TLS certificate verification for this login and all future commands run under this profile. An explicit `--insecure` or `--insecure=false` on the command line wins over the `PANCTL_API_INSECURE` environment variable, which in turn wins over the setting already saved in the profile.
`--server <addr>`	The server address, as an alternative to the positional `[server]` argument (see the address resolution rules above).
`--username <user>`	The tenant-qualified username to log in as (for example `finch/alice`).
`--password <value>`	The password on the command line. It is visible in the process listing and your shell history, so prefer `--password-stdin` or the prompt. It cannot be combined with `--password-stdin`.

Flag	Purpose
`--password-stdin`	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
```

The lines after `PROFILE` and `SERVER` are the API's own answer, so the exact fields can vary and include the roles and permissions of the identity. With a pre-issued token the first line reads `auth: <source>` (for example `env:PANCTL_API_TOKEN`) in place of the profile because no profile was consulted. Under `--as-tenant-admin` (see 5.1) it reports the delegated identity; `is_delegated` is true, `user_id` is the `platform-support` principal, and `actor_user_id` names you.

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

```
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.

`auth login` and `auth logout` refuse the token flags and ignore the environment token because they exist to change the profile session. An exported CI variable therefore does not interfere with a login.

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. By default an access token lasts 60 minutes and a refresh token lasts 7 days, and each renewal issues a new pair. A profile that goes unused for more than 7 days therefore needs a new login. 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
`profiles list`	List all saved profiles and which one is active
`profiles show [name]`	Show details of the active (or named) profile
`profiles use <name>`	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

This chapter covers the following topics.

5.1 Persistent Flags

These flags are available on every PanCTL command:

Flag	Purpose
`--profile <name>`	Use a specific profile for this command only
`--format`, `-o <format>`	Set the output format (see below)
`--no-color`	Disable colored table output (the `NO_COLOR` environment variable is honored too)
`--no-header`	Drop the column header row from `plain` and `table` list output, for piping into `awk` or `cut`. `get` output keeps its `KEY:` labels; `json`, `raw`, and `jsonpath` never had a header.
`--no-pager`	Never page output. By default, long output on a terminal goes through a pager.
`--request-timeout <n>`	How long one API request may take before it is abandoned: seconds, or a duration such as `30s`. The default is 30 seconds; `-1` removes the limit.
`--token <value>`	Use a pre-issued API token for this command
`--token-stdin`	Read a pre-issued API token from standard input
`--as-tenant-admin <tenant>`	Run this command inside another tenant (platform administrators only; see below)
`--yes`, `-y`	Proceed through the confirmation that destructive commands ask for, without prompting
`--version`, `-V`	Print the version and exit

Acting in another tenant. `--as-tenant-admin <tenant>` (a tenant UUID or name) is the PanCTL form of the tenant switcher. It makes one command run inside that tenant, which is how a platform administrator creates or deletes a tenant's volumes. PanCTL exchanges your token for a short-lived one scoped to the tenant (30 minutes to 12 hours), once per command, and saves nothing to your profile. Only a `platform-admin` can use it (it requires the `tenants:switch` permission), and it cannot be combined with `--tenant`. The flag is global and works on any command except `auth login`, which refuses it. It assumes that tenant's `platform-support` role, not its admin role. Commands that name the tenant in the path, such as `subnets create`, `subnets mount-info`, and `attachments`, take `--tenant` instead, and switching tenants does not help there. What the switched identity may do is set by the tenant's own policy, so check with `panctl auth whoami --as-tenant-admin <tenant>` before you rely on it for a write.

5.2 Environment Variables

The following table lists each environment variable and its effect.

Variable	Effect
`PANCTL_PROFILE`	Selects the active profile for one invocation without changing the saved default
`PANCTL_API_SERVER`	Appliance address (used as a fallback in login mode, required in token mode)
`PANCTL_USERNAME`	Default username for `auth login`
`PANCTL_PASSWORD`	Default password for `auth login` (prefer `--password-stdin`; environment variables can leak via process listings)
`PANCTL_API_TOKEN`	Pre-issued token for the whole session (requires `PANCTL_API_SERVER`)
`PANCTL_API_INSECURE`	Set to `1` / `true` / `yes` / `on` to skip TLS verification. `auth login` persists the effective value to the profile; this variable is the only way to set it in token mode.

Variable	Effect
`PANCTL_CONFIG_DIR`	The directory that holds `profiles.toml` and the OpenAPI cache. The default is `~/.panctl`. A relocated directory starts empty, which is how two shells can drive two realms with no shared state.
`PANCTL_REQUEST_TIMEOUT`	Default for `--request-timeout` (30 seconds; `-1` removes the limit)
`PANCTL_PAGER`, `PAGER`	The pager for terminal output, checked in that order, then `less -FRX`. An empty value for either disables paging.
`PANCTL_NO_PAGER`	Set to any non-empty value to never page output
`NO_COLOR`	Suppresses ANSI styling. An unset or `dumb` `TERM` does the same.
`PANCTL_WAIT_TIMEOUT`	Default for `--wait-timeout` (30 minutes); see 5.9
`PANCTL_WAIT_CHECK_INTERVAL`	Default for `--wait-check-interval` (5 seconds); see 5.9
`PANCTL_PROMPT`	The prompt format of an interactive session, with the fields `{profile}`, `{username}`, `{server}`, and `{marker}`; see 2.3

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.

Format	Description
`raw`	The API's response body, unmodified.
`json`	The response body, pretty-printed with 2-space indentation.
`jsonpath=<expression>`	Extracts specific fields using `kubectl`-style JSONPath, ideal for scripting.

Control characters are escaped. The `table` and `plain` formats, and the key and value layout of `get`, never print a control character that came from the API. A value such as a username, display name, description, or volume path can contain one, and a terminal would obey it instead of showing it. PanCTL prints a short escape instead:

Characters	Printed as
Bell, backspace, form feed, newline, carriage return, tab, vertical tab	`\a`, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`
Other characters from `U+0000` to `U+001F`, and `DEL`	`\xNN` in lowercase hex, for example `\x1b` for `ESC`
Characters from `U+0080` to `U+009F`	`\u00NN` in lowercase hex, for example `\u009b`

The escaping cannot be reversed because a backslash in the data is printed as it is. To read a value exactly as stored, use `raw`, `json`, or `jsonpath`, which are not escaped.

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'`.

5.4 Command Aliases

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

Group	Aliases
`auth`	`au`, `a`
`profiles`	`profile`, `pr`, `p`
`realm`	`rlm`
`roles`	`role`, `ro`, `r`
`volumes`	`volume`, `vol`, `v`
`nodes`	`node`, `no`, `n`
`storagesets`	`storageset`, `ss`, `s`
`tenants`	`tenant`, `tn`, `tx`, `t`
`users`	`user`, `us`, `u`
`service-accounts`	`sa`
`fs-users`	`fs-user`, `fsu`
`log-collections`	`log-collection`, `lc`
`s3`	*(none)*
`events`	`ev`, `e`
`alerts`	`al`
`subnets`	`subnet`, `sub`, `sn`
`audit`	`aud`, `ad`

5.5 Exit Codes

The following table lists each exit code and its meaning.

Code	Meaning
`0`	Success, including a JSONPath expression that matched nothing
`1`	Operational failure: the command was written correctly and the operation failed. Examples are an API refusal at any status, a missing object, a transport error, an expired session, and a declined confirmation.
`2`	Usage error, and nothing was sent. Examples are an unknown command or flag, a wrong flag type, an out-of-range value, a missing required flag or argument, an unknown `--format`, a malformed JSONPath expression, and a missing `--yes` when there is no terminal.
`3`	A pre-issued API token was rejected

Every exit `2` also means nothing was sent, so a script can tell its own bug from a condition it is meant to handle. For example, `panctl volumes get "$ID" || handle_missing_volume` takes the `handle_missing_volume` branch only when the volume is genuinely absent, and a mistyped flag gives `2`.

> **Breaking change:** seven kinds of input error used to exit `1` alongside API refusals. A script that branches on `1` for those now gets `2`. This changed under SWIFT-1793.

5.6 Naming Items

The argument of an item command takes either a name or an ID, so `panctl volumes get /vol1` and `panctl volumes get 3` both work, as do `panctl tenants get acme` and `panctl storagesets get "Demo Set"`. A value that is already in the ID form is used as it stands, and anything else costs one lookup. An unknown name fails before the request is sent, so a typo never reaches the appliance as part of a path.

Resource	The argument accepts	Matched against
`tenants`	Short name or UUID	`name`
`storagesets`	Name or integer ID	`name`
`volumes`	Path or integer ID	`pathname`
`subnets`	Name or UUID	`name`
`users`	Username or UUID	`username`
`service-accounts`	Name or UUID	`name`
`nodes`	Dataserial only	No name is accepted
`log-collections`	Collection ID only	No name is accepted
`fs-users`, `auth tokens`	Id only	No name is accepted

A human label is never matched; the `display_name` of a tenant or a service account resolves nothing. On `service-accounts`, the `--service-account` flag resolves a name too, but on `users` the `--user` flag does not, so `panctl users get alice` works while `panctl users get-roles --user alice` does not.

Wherever a command takes `--tenant`, it accepts the tenant's UUID or its short name. On the `users` commands, omitting it takes the tenant from the prefix of your profile's username.

5.7 Request Bodies

Create and update commands that accept field flags take those flags, a JSON document with `-f <file|->`, or both. When both are given, the flags override the matching fields of the document. With no field flags, the bytes of `-f` are sent unchanged; with at least one, the document is parsed, the flags are layered over it, and the result is re-serialized, so key order and formatting are not kept. A field whose flag you did not give is left out of the request, never sent as `null`.

Before sending a body, PanCTL checks it against the cached OpenAPI document where it can (see 1.2), so a missing required field or a value of the wrong type is reported with no request made. The same document feeds `--help`; each command's help ends with the operation's summary and an example request body.

5.8 Narrowing a List (`--filter`)

Every `list` command takes `--filter`, which keeps only the matching rows and renders them normally, columns and headers included. It does not change the request, so it narrows what you see and not what is fetched. `get` commands have no `--filter`.

```
panctl nodes list --filter '.role==vosd && .state!=online'
panctl audit list --filter '.action==/create|delete/'
panctl log-collections list --filter '.status==failed'
```

A filter is a path on the left, one of `==`, `!=`, `<`, `<=`, `>`, `>=` on the right, and terms joined by `&&` and `||`. Terms are folded left to right with no precedence and no grouping, so `a && b || c` means `(a && b) || c`. A leading `@` is optional, and a field that a row lacks makes its term false instead of failing.

The right-hand side can be a bare word (`ready`), a quoted string (`"with spaces"`), a number (`42`, `3.5`), `true`, `false`, `null`, or a regular expression between slashes. A regular expression accepts only `==` and `!=`. A path with no operator is an existence test, so `--filter '.reason'` keeps the rows that have a `reason` at all. Because of that, a single `=` is refused with a message listing the valid operators; otherwise `.action=delete` would be read as an existence test on a field with that odd name, match nothing, and exit `0`.

A value written between slashes is a regular expression, and it must match the **whole** value. This matters because states nest inside each other (`online` inside `going_online`, `delete` inside `finalize-delete`, `offline` inside `going_offline`), so an unanchored pattern would answer the opposite question. For a substring, write it out, as in `/.*offline/`. A regular expression tests strings only, so a number, boolean, or null never matches one.

A malformed expression, an invalid regular expression, or an unsupported operator is refused before any request is made, with exit `2`. `--filter` works with every output format. With a filter, `json` and `raw` output come back with their keys in alphabetical order; without one, `raw` is still the API's bytes exactly.

On `audit list` and `events list`, the appliance applies its window of recent records first, so `--limit 5 --filter '.action==delete'` finds the deletes among the five most recent records and not the five most recent deletes. PanCTL warns about this. Pair the filter with `--only`, `--skip`, `--since`, and `--until`, which narrow the records on the appliance before the window is applied.

There is no `--sort`. To sort or reshape results, pipe `--format json` into a tool such as `jq`.

5.9 Waiting for a Condition (`--wait-for`)

Provisioning is asynchronous. A StorageSet takes a while to finish joining, a tenant is not usable until its subnet has been issued an address, and a deletion completes some time after the request returns. Every `list` and `get` command takes `--wait-for`, which re-issues the read until a condition holds, so a script does not need a polling loop of its own.

The selector selects, and the condition compares. `-o jsonpath=` picks a value out of the response, and `--wait-for` compares that one value. Everything structural, such as fields, indexes, filters, `&&`, and `||`, belongs in the selector. In a condition, `it` stands for what the selector matched. It is not a path, so `it.name` and `it[0]` are errors.

```
panctl tenants get finch -o jsonpath='{.tenant.status}' --wait-for=active

panctl volumes list -o jsonpath='{.volumes[?(@.pathname == "/v1")].id}' --wait-for='it != null'
panctl volumes list -o jsonpath='{.volumes[?(@.pathname == "/v1")].id}' --wait-for=null   #
        waits for a deletion
panctl storagesets get 1 -o jsonpath='{.storageset.nodes_total}' --wait-for='it >= 6' --wait-
        timeout=60m

panctl nodes get-drain 1cc2673e772f58 -o jsonpath='{.state}' \
    --wait-for=not_draining --wait-timeout=4h --wait-check-interval=30s
```

Condition	Holds when	Short form
`it != null`	The selector matched anything	None
`it == null`	The selector matched nothing	`--wait-for=null`
`it == "active"`	The value is `active`	`--wait-for=active`
`it != "active"`	The value is anything else	None
`it >= 6`	A numeric comparison; `<`, `<=`, `>`, and `>=` all work	None

A bare word is a string, so `it == active` and `it == "active"` are the same condition. Quote a value that would otherwise read as a number or as `null`.

Flag	Default	Meaning
`--wait-for`	None	The condition; the read is re-issued until it holds
`--wait-timeout`	`30m`	`0` checks once without waiting, and `-1` waits without limit
`--wait-check-interval`	`5s`	How often the read is re-issued

A time is seconds when written bare and a duration when it carries a unit, so `3600`, `3600s`, and `1h` are the same hour. `PANCTL_WAIT_TIMEOUT` and `PANCTL_WAIT_CHECK_INTERVAL` set defaults for the two flags, and a flag on the command line wins. A wait with `--wait-timeout=0` turns the condition into an assertion about the state right now; it exits `0` if the condition holds and `1` if it does not.

What you see. Progress goes to standard error, so a captured result holds only the value. A condition that already holds on the first check prints just its result. A wait that has to sleep prints a line whenever the value changes and a heartbeat at most every 30 seconds with the time left. A timeout exits `1` and names the condition and the last value seen.

Things to know

- Quote the condition. An unquoted `>` is a shell redirection, so `--wait-for=>2` creates a file named `2` and loses the comparison. PanCTL refuses an empty condition and says so.
- `--wait-for` needs a selector (`-o jsonpath=`). Without one it is refused before any request is made.
- Any condition other than `it == null` or `it != null` compares one value, so the selector must match exactly one. A selector that matches several is refused.
- A failing request does not end the wait because an appliance that is still starting answers with errors, and the timeout is what bounds the wait. An error that will never clear, such as a rejected credential, therefore costs the whole timeout.
- Only `list` and `get` accept the wait flags. A write has nothing to wait for, and re-issuing it could create a second resource.
- `--as-tenant-admin` costs a token exchange on every check, so raise `--wait-check-interval` when you combine them.
- `s3 instances get <iid> --wait-for-observed=<state>` is the one command-specific form; it builds the selector for you.

5.10 Destructive Commands Ask First

These commands ask for confirmation before they act. The default answer is no, and the prompt names the target:

- Every `delete`, for tenants, StorageSets, volumes, subnets, subnet attachments, users, service accounts, PanFS users, and log collections, plus `s3 instances delete`, `s3 instances delete-all`, `s3 buckets delete`, `s3 buckets delete-policy`, and `s3 delete-config`
- `auth tokens revoke` and `service-accounts tokens revoke`
- `users set-roles` and `service-accounts set-roles`
- `nodes reboot` and `nodes drain`
- `tenants unassign-storagesets`

`nodes stop-drain` does not ask. When no terminal is attached there is nobody to answer, so pass `--yes` (or `-y`). Without it, the command exits `2` having sent nothing.

5.11 Across Tenants

Two flags answer different questions, and they cannot be combined.

Flag	The question it answers	Cost
`--tenant`	Whose records do I want to see? It filters a read and travels as a query parameter.	One request
`--as-tenant-admin`	Who is asking? It swaps your token for one scoped to that tenant, so a write lands there.	Two requests: a token exchange, then the command

Volume writes (`volumes create`, `update`, and `delete`) have no tenant parameter, so there the identity has to change instead. Subnets are the exception to both; every scoped subnet route names its tenant in the path, so `--tenant` carries it on reads and writes alike, and `--as-tenant-admin` loses access instead of gaining it.

`--all-tenants` is the other end of the read axis. It widens `volumes list`, `events list`, `alerts list`, and `audit list` to every tenant, and giving it together with `--tenant` is a usage error. On `volumes list`, `events list`, and `alerts list` it renders a tenant column. `--tenant` and `--all-tenants` are the platform tenant's; the API refuses any other caller that passes either, even one naming its own tenant.

```
# Read: filter, one request
panctl volumes list --tenant=acme
panctl volumes list --all-tenants

# Write: act as that tenant, two requests
panctl volumes create --as-tenant-admin=acme --name=/vol1 --storageset-id=1 --hard-quota=1GB

# Subnets: --tenant, not --as-tenant-admin
panctl subnets delete storage-net --tenant=acme
```

Combining `--as-tenant-admin` with `--tenant` fails with `error: --as-tenant-admin and --tenant are mutually exclusive: one is the tenant you act as, the other filters what you read`, exit `2`, before any request is sent. See 5.1 for what the delegated identity may do.

5.12 Output Streams and Paging

Only a command's result reaches standard output, so a redirect captures data and nothing else. Prompts, progress lines, the login banner, and the session-renewal notice all go to standard error.

On a terminal, long output goes through a pager: `$PANCTL_PAGER`, else `$PAGER`, else `less -FRX`. Short output, pipes, redirects, and errors are never paged. `--no-pager` turns paging off for one command, and any non-empty `PANCTL_NO_PAGER` turns it off for a whole shell, which matters for scripts run from an interactive terminal.

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	`--quota=500B`
`KB` / `MB` / `GB` / `TB` / `PB`	Decimal (powers of 1000)	`--quota=10GB` → 10,000,000,000 bytes
`KiB` / `MiB` / `GiB` / `TiB` / `PiB`	Binary (powers of 1024)	`--quota=10GiB` → 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, `shell` in Section 2.3, and `completion` in Chapter 9; they are not repeated in the table below.

Command	Purpose
`realm get`	Show realm-wide state and configuration
`nodes list`	List all nodes
`nodes get <dataserial>`	Show one node's details
`nodes reboot <dataserial>`	Reboot a node (asks for confirmation)
`nodes drain <dataserial>`	Start draining a node (asks for confirmation)
`nodes get-drain <dataserial>`	Show a node's drain progress
`nodes stop-drain <dataserial>`	Stop a drain that is already running
`storagesets list`	List all StorageSets
`storagesets get <id>`	Show one StorageSet
`storagesets create --name <name> [--spares-requested <n>]`	Create a StorageSet
`storagesets update <id> -f <file>`	Update a StorageSet
`storagesets delete <id>`	Delete a StorageSet (asks for confirmation)
`storagesets add-storage --storageset-id <id> --dataserials <a,b,c> [options]`	Add nodes to a StorageSet
`tenants list`	List all tenants
`tenants get <tenant_id>`	Show one tenant
`tenants create --name <name> --display-name <label> --email <addr> [options]`	Create a tenant

Command	Purpose
`tenants update <tenant_id> -f <file>`	Update a tenant
`tenants delete <tenant_id>`	Delete a tenant (asks for confirmation; fails with `409` until its S3 instances, volumes, and subnets are removed)
`tenants assign-storagesets --tenant <ref> --storageset-id <id> --quota <size>`	Grant/resize a tenant's capacity in a StorageSet
`tenants unassign-storagesets --tenant <ref> --storageset-id <id>`	Remove a tenant's access to a StorageSet
`tenants list-storagesets --tenant <ref>`	List StorageSets assigned to a tenant
`tenants get-storageset --tenant <ref> --storageset-id <id>`	Show one tenant/storage-set assignment
`volumes list [--tenant <ref> | --all-tenants]`	List volumes
`volumes get <volume_id> [--tenant <ref>]`	Show one volume
`volumes create --name <path> --storageset-id <id> --hard-quota <size> [options]`	Create a volume
`volumes update <volume_id> -f <file>`	Update quotas, recovery priority, etc.
`volumes rename <volume_id> -f <file>`	Rename (change the path of) a volume
`volumes delete <volume_id>`	Delete a volume (asks for confirmation)
`subnets list [--tenant <ref>]`	List subnets (realm-wide, or one tenant's)
`subnets get <subnet> --tenant <ref>`	Show one subnet
`subnets create --tenant <ref> --name <name> --cidr <addr/prefix> [options]`	Create a subnet
`subnets update <subnet> --tenant <ref> [options]`	Update a subnet
`subnets delete <subnet> --tenant <ref>`	Delete a subnet (asks for confirmation)

Command	Purpose
`subnets mount-info <subnet> --tenant <ref>`	Show which Directors a client on the subnet mounts to
`subnets segment-type get` or `subnets segment-type set <vlan|vxlan>`	Read or set the realm-wide network segment type
`subnets attachments list <subnet> --tenant <ref>`	List client attachments on a subnet
`subnets attachments get <subnet> <attachment_id> --tenant <ref>`	Show one attachment
`subnets attachments create <subnet> --tenant <ref> --underlay-ip <ip> [options]`	Attach a client
`subnets attachments delete <subnet> <attachment_id> --tenant <ref>`	Remove an attachment (asks for confirmation)
`users list [--tenant <ref>]`	List users in a tenant
`users get <user_id> [--tenant <ref>]`	Show one user
`users create --tenant <ref> --username <name> [options]`	Create a user
`users update <user_id> --tenant <ref> [--display-name <label>] [-f <file>]`	Update a user
`users delete <user_id> --tenant <ref>`	Delete a user (asks for confirmation)
`users get-roles --user <id> --tenant <ref>`	Show a user's roles
`users set-roles --user <id> --tenant <ref> --roles <a,b>`	Replace a user's roles (asks for confirmation)
`service-accounts list`	List service accounts
`service-accounts get <id>`	Show one service account
`service-accounts create --name <name> [--display-name <name>] [--description <text>]`	Create a service account

Command	Purpose
`service-accounts update <id> [options]`	Update display name/description
`service-accounts delete <id>`	Delete a service account (asks for confirmation)
`service-accounts get-roles --service-account <id>`	Show a service account's roles
`service-accounts set-roles --service-account <id> --roles <a,b>`	Replace a service account's roles (asks for confirmation)
`service-accounts tokens create --service-account <id> --name <label> --expires-in-days <n>`	Mint an API token for the service account
`service-accounts tokens list --service-account <id> [--active-only]`	List a service account's tokens
`service-accounts tokens revoke <token_id> --service-account <id> [--reason <text>]`	Revoke a token (asks for confirmation)
`roles list`	Show all roles and their permitted actions
`s3 enabled get` / `s3 enabled set <true|false>`	Enable or disable the S3 gateway realm-wide
`s3 prefix get` / `s3 prefix set <path>`	View or set the realm-wide path prefix
`s3 create-config --tenant <ref> [options]`	Create a tenant's S3 configuration
`s3 get-config --tenant <ref>`	Show a tenant's S3 configuration
`s3 set-config --tenant <ref> [options]`	Update a tenant's S3 configuration
`s3 delete-config --tenant <ref>`	Delete a tenant's S3 configuration
`s3 aggregator <on|off> --tenant <ref>`	Turn a tenant's S3 aggregator on or off
`s3 dns <off|panfs|external> --tenant <ref>`	Set how S3 DNS is published for a tenant
`s3 tls renew --tenant <ref>`	Renew a tenant's S3 TLS certificate

Command	Purpose
`s3 instances list --tenant <ref>`	List S3 gateway instances
`s3 instances get <iid> --tenant <ref>`	Show one S3 gateway instance
`s3 instances create <iid> <dataserial> --tenant <ref>`	Create an S3 gateway instance on a node
`s3 instances start <iid> --tenant <ref>` / `start-all`	Start one or every S3 gateway instance
`s3 instances stop <iid> --tenant <ref>` / `stop-all`	Stop one or every S3 gateway instance
`s3 instances delete <iid> --tenant <ref>`	Delete an S3 gateway instance
`s3 instances delete-all --tenant <ref> --yes`	Delete every S3 gateway instance for a tenant
`s3 buckets create <name> --owner <access_key> --tenant <ref>`	Create an S3 bucket owned by a PanFS user
`s3 buckets list --tenant <ref>`	List S3 buckets
`s3 buckets delete <name> --tenant <ref>`	Delete an empty S3 bucket
`s3 buckets delete-policy <name> --tenant <ref>`	Remove a bucket's policy (asks for confirmation)
`fs-users list --tenant <ref>`	List a tenant's PanFS users and their S3 access keys
`fs-users get <fs_user_id> --tenant <ref>`	Show one PanFS user
`fs-users create --tenant <ref> --name <name> [options]`	Create a PanFS user
`fs-users update <fs_user_id> --tenant <ref> [options]`	Change a PanFS user
`fs-users delete <fs_user_id> --tenant <ref>`	Delete a PanFS user (asks for confirmation)

Command	Purpose
`fs-users s3-secret <fs_user_id> --tenant <ref>`	Show a user's S3 access key and secret
`fs-users rotate-s3-secret <fs_user_id> --tenant <ref>`	Rotate a user's S3 secret
`events list [--limit <n>] [--tenant <ref> | --all-tenants]`	List recent system events
`alerts list [--tenant <ref> | --all-tenants]`	List currently active alerts
`audit list [options]`	List administrative audit log entries
`audit get <request_id>`	Show one audit log entry's detail
`log-collections create [options]`	Start a realm log collection
`log-collections list`	List collections, newest first
`log-collections get <collection_id>`	Show one collection in full
`log-collections download <collection_id> [--output <file>]`	Fetch a collection's archive
`log-collections delete <collection_id>`	Stop a run, clean the nodes, and remove the archive (asks for confirmation)
`api <METHOD> <path> [--query k=v] [-f <file|->]`	Send one authenticated request to any PanAPI path
`api list`, `api explain <path>`, `api version`	Read the cached API specification

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
`realm get`	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
`nodes list`	List all nodes
`nodes get <dataserial>`	Show one node's details
`nodes reboot <dataserial>`	Reboot a node (asks for confirmation)
`nodes drain <dataserial>`	Start draining a node, which migrates every object off it (asks for confirmation)
`nodes get-drain <dataserial>`	Show a node's drain progress
`nodes stop-drain <dataserial>`	Stop a drain that is already running

```
$ panctl nodes list
NAME     DATASERIAL      SHELF ID           IP ADDRESS    STATUS  ROLE   STORAGESET ID  CAPACITY BYTES
ASD-1,1  485f0d3e72327a  3e7f0001402001001  10.97.26.200  online  qnode  0              0
VCH-1,2  c9f82187c13675  3e7e0004404001001  10.97.26.246  online  vosd   1              61891149824
```

```
panctl nodes get 109a58fc3162ef
panctl nodes reboot 109a58fc3162ef
panctl nodes drain 109a58fc3162ef
panctl nodes get-drain 109a58fc3162ef -o jsonpath='{.state}'
```

Notes:

- `NAME` is the first column and is the identifier that alerts carry in their `node_name` label, so it takes you from a firing alert to a node. Rows are sorted by name, with digit runs compared numerically, so `ASD-1,2` comes before `ASD-1,10`. This is the one list that sorts its rows; `--format json` keeps the API's order.
- A node is addressed by its `dataserial` only because the name is not accepted as an argument.
- A `STORAGESET ID` of `0` means the node isn't assigned to a StorageSet yet.
- `reboot` and `drain` ask for confirmation, and `--yes` skips it. `stop-drain` does not ask.
- The `drain` prompt reads `Every object on the node is migrated elsewhere, which can take hours.` followed by `Drain node <dataserial>? [y/N]:`.
- `get-drain` reports a `state` of `draining`, `not_draining`, `suspended`, or `unknown`. Wait on it with `--wait-for=not_draining` (see 5.9). Do not write `.state==/draining/` as a filter because `not_draining` also contains that word; see 5.8.
- `drain` is the most disruptive operation PanCTL exposes. It can take hours, and re-running it does not undo it; use `stop-drain` to cancel and `get-drain` to follow progress.
- `get-drain` reports the state, the bytes and objects moved, the rate, the percent complete, and an estimated time of arrival. A node that is not draining has no progress to report, and the API answers `409` instead of an empty result.

8.3 `storagesets`

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

Command	Purpose
`storagesets list`	List all StorageSets

Command	Purpose
`storagesets get <id>`	Show one StorageSet
`storagesets create --name <name> [--spares-requested <n>]`	Create a StorageSet
`storagesets update <id> -f <file>`	Update a StorageSet
`storagesets delete <id>`	Delete a StorageSet (asks for confirmation)
`storagesets add-storage --storageset-id <id> --dataserials <a,b,c> [--mode add_spare|migration] [--spares-requested <n>]`	Add nodes to a StorageSet (`--spares-requested` is required in the default `add_spare` mode and not allowed with `--mode migration`)

```
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:

- A StorageSet is addressed by its name or its integer ID. In `list`, the `STATUS` column is the worst active condition of the set.
- `storagesets get` also reads the set's space and shows it with the other fields, so capacity is available in JSON as `{.storageset.space.usable.free}`. If the space read fails, the set is still shown without those fields and the reason goes to standard error.
- `--mode` defaults to `add_spare`, and in that mode `--spares-requested` is required; use `0` to keep the current spare count. With `--mode migration`, `--spares-requested` is not allowed, and PanCTL refuses it before sending anything.
- `--dataserials` cannot be left empty on `add-storage`; PanCTL refuses an empty list. Always name the nodes. Add only unassigned nodes whose role is `vhost`, `vosd`, or `snode`; other roles, such as `qnode`, are refused. On a realm that runs on virtual hosts, list the `vhost` nodes, and their VPODs join with them.

- `add-storage` returns immediately, but absorbing new nodes into the set happens asynchronously in the background and takes about 15 minutes. A vhost adds its capacity later because each of its VPODs joins after it installs and boots. Wait on the node count to confirm completion (see 5.9):

```
panctl storagesets get 4 -o 'jsonpath={.storageset.nodes_total}' --wait-for='it >= 6' --
        wait-timeout=60m
```

8.4 `tenants`

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

Command	Purpose
`tenants list`	List all tenants
`tenants get <tenant_id>`	Show one tenant
`tenants create --name <name> --display-name <label> --email <addr> [--auth-mode local|pam] [--pam-service <service>] [--description <text>]`	Create a tenant
`tenants update <tenant_id> -f <file>`	Update a tenant
`tenants delete <tenant_id>`	Delete a tenant (asks for confirmation; **does not cascade**, see the note below)
`tenants assign-storagesets --tenant <ref> --storageset-id <id> --quota <size>`	Grant/resize a tenant's capacity in a StorageSet
`tenants unassign-storagesets --tenant <ref> --storageset-id <id>`	Remove a tenant's access to a StorageSet
`tenants list-storagesets --tenant <ref>`	List StorageSets assigned to a tenant
`tenants get-storageset --tenant <ref> --storageset-id <id>`	Show one tenant/storage-set assignment

```
panctl tenants create --name fintech --display-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 name; PanCTL resolves names for you. The argument of `get`, `update`, and `delete` takes the same reference, but the display name never matches.
- `--short-name` was removed when the API renamed the field. Using it exits `2` with a message saying to use `--name` for the short identifier and `--display-name` for the human label.
- `--pam-service` (default `login`) applies only with `--auth-mode pam`, and giving it with `--auth-mode local` is an error.
- `--name` is 3–10 alphanumeric characters and cannot be changed after creation. `--display-name` is the human-readable label.
- `assign-storagesets` acts as an "upsert": if the tenant already has capacity in that StorageSet, this call resizes the quota instead of failing.
- `tenants delete` does not cascade. The API refuses with `409` (for example, "Cannot delete tenant while it still owns subnets") until the tenant's S3 instances, volumes, and subnets are removed, in that order: `s3 instances delete-all`, then `volumes delete` for each volume, then `subnets delete` for each subnet. Deleting a volume needs a session for the tenant, as creating one does. The command asks for confirmation first, and `--yes` skips the prompt. The confirmation text describes the deletion as cascading, but the refusal above is what the API does.

8.5 `volumes`

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

Command	Purpose
`volumes list [--tenant <ref> | --all-tenants]`	List volumes
`volumes get <volume_id> [--tenant <ref>]`	Show one volume

Command	Purpose
`volumes create --name <path> --storageset-id <id> --hard-quota <size> [options]`	Create a volume
`volumes update <volume_id> -f <file>`	Update quotas, recovery priority, etc.
`volumes rename <volume_id> -f <file>`	Rename (change the path of) a volume
`volumes delete <volume_id>`	Delete a volume (asks for confirmation)

```
panctl volumes list --tenant fintech
panctl volumes list --all-tenants
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
`--name <path>`	Required. A leading `/` is added automatically if missing.
`--storageset-id <id>`	Required.
`--hard-quota <size>`	Required by PanCTL (even though the API treats it as optional) so a limit is always chosen deliberately. A volume in a tenant needs a value greater than 0; the API returns `400` otherwise, so `0` is not a way to create an unlimited tenant volume.
`--soft-quota <size>`	Optional. `0` means unlimited.
`--raid-type <layout>`	Optional, **cannot be changed after creation**.
`--efsa-mode <mode>`	Optional.

Flag	Notes
`--recovery-priority <0-100>`	Optional; lower number = higher priority.
`--ignore-limits`	Optional. Platform tenant only. Acts without the limits the StorageSet assignment imposes, so no assignment needs to exist, no quota ceiling applies, and free space is not checked. `--storageset-id` is still required. `volumes update` takes the same flag.

Notes:

- A volume is addressed by its path or its integer ID. `volumes get` also shows used and free space, available in JSON as `{.volume.space.used_bytes}`.
- On `list`, `--tenant` filters to one tenant and `--all-tenants` shows every tenant's volumes and adds a `TENANT ID` column. The two cannot be combined (exit `2`). Both belong to the platform tenant, and the API refuses them for any other caller. With neither, the API scopes the list by your role.
- `volumes create` always creates the volume in your own tenant; there's no `--tenant` flag on this command.
- When the API refuses a volume request because your tenant has no StorageSet assignment, PanCTL prints a hint after the refusal that names `--ignore-limits`. `ignore_limits` is a query parameter, not a request field, so putting it in a `-f` body is refused before anything is sent.
- Before creating a volume, make sure that the target StorageSet 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
`subnets list [--tenant <ref>]`	List subnets. Without `--tenant` the list is realm-wide; with it, only that tenant's subnets.
`subnets get <subnet> --tenant <ref>`	Show one subnet

Command	Purpose
`subnets create --tenant <ref> --name <name> --cidr <addr/prefix> [options]`	Create a subnet
`subnets update <subnet> --tenant <ref> [options]`	Update a subnet
`subnets delete <subnet> --tenant <ref>`	Delete a subnet (asks for confirmation)
`subnets mount-info <subnet> --tenant <ref>`	Show which Directors a client on this subnet mounts to
`subnets segment-type get`	Show the realm-wide network segment type (`vlan` , `vxlan` , or `none`)
`subnets segment-type set <vlan|vxlan>`	Set the realm-wide network segment type
`subnets attachments list <subnet> --tenant <ref>`	List client attachments on a subnet
`subnets attachments get <subnet> <attachment_id> --tenant <ref>`	Show one attachment
`subnets attachments create <subnet> --tenant <ref> --underlay-ip <ip> [options]`	Attach a client
`subnets attachments delete <subnet> <attachment_id> --tenant <ref>`	Remove an attachment (asks for confirmation)

```
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 mount-info fintech-net --tenant fintech
panctl subnets attachments create fintech-net --tenant fintech --underlay-ip 10.128.9.0 --
         hostname client01
panctl subnets update fintech-net --tenant fintech --mtu 1500
panctl subnets delete fintech-net --tenant fintech
```

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.
- A subnet is addressed by its name or its UUID.
- Every scoped subnet route names its tenant in the path, so `--tenant` is **required** on every subnet command except `list`. `--all-tenants` was removed from `subnets list` and exits `2` with a message saying so.
- `--as-tenant-admin` does not help on subnets and actively hurts; no tenant role holds a grant on them, so switching into the tenant loses the access.
- `subnets attachments create` needs `--underlay-ip` unless `-f` supplies it. `--hostname` is informational only and is never interpreted. Omit `--overlay-ip` to let the backend allocate the tenant network address; it applies on first attach only, so repeating a create never moves a host. Flags override the matching fields of an `-f` document. The response is the full attachment, and on a tunnelled segment it carries the tunnel details the host needs (MAC address, UDP port, and VTEP flood list).
- `subnets attachments list` takes `--underlay-ip <ip>` to show only the attachment with that underlay address; the filter is applied by the API.
- `subnets mount-info` and the `attachments` commands need a platform role. A tenant administrator gets a permissions error, and `mount-info` also fails through `--as-tenant-admin`. The `rmlist` in the `mount-info` output is ready to use as the `mount_panfs` option.
- `--cidr` is the PanCTL spelling of two API fields, the base address and the prefix length, so the pair travels as one value.
- `--segment-id`, `--tunnel-udp-port`, and `--addr-external` can only be set at creation time.
- `subnets update` takes `--name`, `--gateway`, `--mtu`, `--range-start`, and `--range-end`, and the flags only set values. To clear a field, such as removing the gateway, pass `-f` with `null` for it, for example `{"gateway":null}`. `--mtu 0` selects the platform default. An update with no flags and no `-f` is refused before anything is sent.
- The realm, not any one subnet, owns the segment type, which decides what `--segment-id` means on `subnets create`, which is a VLAN ID under `vlan` and a VNI under `vxlan`. `segment-type get` can answer `none`, meaning the realm names no segment type. That is a state, not an error, and `subnets create` against such a realm is refused with a reason. `segment-type set` takes `vlan` or `vxlan`; `none` can be read but not set, so a realm cannot be put back into that state.

8.7 `users`

Aliases: `user`, `us`, `u`. Human user accounts.

> **Tenant users are not fully supported in this release.** The API and PanCTL accept `--tenant` on these commands, and a tenant user signs in as `<tenant-name>/<username>`. Vantage, however, manages users only in the Platform area, and a tenant has no sign-in of its own there. To administer a tenant, use a platform administrator with `--as-tenant-admin` (see 5.1) instead.

Command	Purpose
`users list [--tenant <ref>]`	List users in a tenant
`users get <user_id> [--tenant <ref>]`	Show one user
`users create --tenant <ref> --username <name> [options]`	Create a user
`users update <user_id> --tenant <ref> [--display-name <label>] [-f <file>]`	Update a user
`users delete <user_id> --tenant <ref>`	Delete a user (asks for confirmation)
`users get-roles --user <id> --tenant <ref>`	Show a user's roles
`users set-roles --user <id> --tenant <ref> --roles <a,b>`	Replace a user's roles (asks for confirmation)

```
panctl users create --tenant fintech --username jsmith --email jsmith@fintech.example.com
printf '%s' "$PW" | panctl users create --tenant fintech --username alice --password-stdin
panctl users set-roles --user <user_id> --tenant fintech --roles platform-read-only
panctl users get-roles --user <user_id> --tenant fintech
panctl users delete <user_id> --tenant fintech
```

`create` options:

Flag	Notes
`--username <name>`	Required unless `-f` supplies it. 1 to 255 characters, no whitespace, unique within the tenant.
`--display-name <label>`	Optional human label.

Flag	Notes
`--email <addr>`	Optional contact email.
`--auth-mode local|pam`	Optional; the default is `local`.
`--pam-service <service>`	Optional; the default is `login`, and it applies only with `--auth-mode pam`.
`--password <value>`	The password on the command line. It is visible in the process listing and your shell history.
`--password-stdin`	Read the password from standard input. Cannot be combined with `--password`.

Notes:

- Never pass a password as a flag. Pipe it with `--password-stdin`, or let `create` prompt for it. The password sources are tried in the order `--password`, `--password-stdin`, a `password` already in the `-f` document, and then an interactive prompt. A `-f`-only invocation never prompts.
- `--user` on `get-roles` and `set-roles` is a required flag that takes the user's UUID, not the username, even though `users get <username>` accepts the name.
- If `--tenant` is omitted, PanCTL infers it from the tenant-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.
- `platform-admin`, `platform-read-only`, and `storage-admin` are the platform roles; see 8.9 for what each allows. A tenant defines its own roles, such as the `admin` role of a tenant user, and tenant roles are not fully supported in this release.

8.8 `service-accounts`

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

Command	Purpose
`service-accounts list`	List service accounts
`service-accounts get <id>`	Show one service account
`service-accounts create --name <name> [--display-name <name>] [--description <text>]`	Create a service account

Command	Purpose
`service-accounts update <id> [options]`	Update display name/description
`service-accounts delete <id>`	Delete a service account (asks for confirmation)
`service-accounts get-roles --service-account <id>`	Show a service account's roles
`service-accounts set-roles --service-account <id> --roles <a,b>`	Replace a service account's roles (asks for confirmation)
`service-accounts tokens create --service-account <id> --name <label> --expires-in-days <n> [--description <text>]`	Mint an API token for the service account
`service-accounts tokens list --service-account <id> [--active-only]`	List a service account's tokens
`service-accounts tokens revoke <token_id> --service-account <id> [--reason <text>]`	Revoke a token (asks for confirmation)

```
panctl service-accounts create --name ci-runner --display-name "CI Runner"
panctl service-accounts set-roles --service-account ci-runner --roles storage-admin

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

Notes:

- A service account is addressed by its name or its UUID, both as the argument of `get`, `update`, and `delete` and in the `--service-account` flag. The display name never matches.
- Deleting a service account stops every token issued to it immediately.
- 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.

A user or service account holds one or more of these three platform roles. Tenant roles, defined per tenant, are not fully supported in this release (see the note in 8.7):

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

Command	Purpose
`roles list`	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
`s3 enabled get` / `s3 enabled set <true|false>`	Enable or disable the S3 gateway realm-wide
`s3 prefix get` / `s3 prefix set <path>`	View or set the realm-wide path prefix

Per-tenant configuration:

```
panctl s3 create-config --tenant fintech --prefix /vol1/fintech
panctl s3 get-config --tenant fintech
panctl s3 set-config --tenant fintech --enabled true --prefix /vol1/fintech \
  --owner uid:1000 --group gid:1000 --dir-mode 0755 --obj-mode 0644
panctl s3 aggregator on --tenant fintech
panctl s3 dns external --tenant fintech
panctl s3 tls renew --tenant fintech
panctl s3 delete-config --tenant fintech
```

`create-config` and `set-config` take the same field flags, or a JSON body with `-f <file|->` :

Flag	Sets
`--enabled <true|false>`	Whether S3 is enabled for the tenant
`--prefix <path>`	The path S3 serves. It must name an existing volume of the tenant, or a directory inside one.
`--owner <uid>` and `--group <gid>`	The default POSIX owner and group for objects: a number, optionally written `uid:<n>` or `gid:<n>`
`--dir-mode <mode>` and `--obj-mode <mode>`	The default permission modes for directories and objects: three or four octal digits, such as `0755` and `0644`

Flag	Sets
`--access-log <true|false>`	Whether the S3 access log is enabled
`--debug-log <true|false>`	Whether the S3 debug request log is enabled
`--metrics-enabled <true|false>`	Whether S3 metrics are collected

- `aggregator on` gives the tenant one stable address, served by the Directors, in front of every started instance. `aggregator off` (the default for a new configuration) leaves clients with the per-instance endpoints only. Neither is needed for reachability.
- `dns` is `off` (the default, nothing published), `panfs` (the realm publishes the records; **platform tenant only**, and the API refuses it for any other tenant), or `external` (you run your own DNS and supply the certificate).
- `tls renew` works only when the tenant's TLS mode is `realm-generated`. For a `tenant-provided` certificate it returns `409`; write a new `tls` block with `set-config -f` instead. Renewing does not restart running instances.

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

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

- A `start` does not restart an instance that is already started. To restart one, stop it, wait for `stopped`, and then start it.
- `start`, `stop`, `start-all`, `stop-all`, and `delete` record the new desired state and return without waiting. Read the `observed` state with `instances get` until it matches, or add `--wait-for-observed <state>` (for example `running` or `stopped`) to `instances get` to wait for it.
- When an instance ID runs on more than one node, `get`, `start`, `stop`, and `delete` act on every node. `--dataserial <serial>` narrows the command to one node. An empty `--dataserial` is an error.
- `delete` on an instance ID that runs on more than one node is refused (`409`, with a preview of the matches) unless you add `--confirm-fanout`. `delete-all` removes every instance of the tenant.

- Every instance listens on port 3000. `instances get` shows each instance's `endpoint`, for example `https://192.168.79.51:3000`.

Buckets:

```
panctl s3 buckets create archive-2026 --owner alice --tenant fintech
panctl s3 buckets list --tenant fintech
panctl s3 buckets delete archive-2026 --tenant fintech
panctl s3 buckets delete-policy archive-2026 --tenant fintech
```

- `delete-policy` is the recovery for a bucket whose owner is locked out by its own policy. Once a bucket has a policy it is the only authority on the S3 API, so an owner whose policy omits `s3:DeleteBucketPolicy` cannot remove it there. The policy is removed, not replaced; the bucket returns to the profile baseline, under which its owner has full control again and can apply a new policy over the S3 API. It asks for confirmation because the policy document cannot be recovered. A bucket with no policy is a success, so after a `504` or a timeout you can simply repeat the request.
- A bucket belongs to a PanFS user, named by that user's S3 access key (see 8.11). Bucket commands are answered by a running gateway instance, so without one they return `503`, and the list is never shown as empty.
- A `504` on `create` or `delete` means the bucket may exist or may have been deleted, so check with `buckets list` before you retry.
- `buckets delete` is refused with `409` while the bucket still holds objects. Empty it over the S3 API first.

Notes:

- Destructive commands (`s3 instances delete`, `s3 instances delete-all`, `s3 buckets delete`, and `s3 delete-config`) ask for confirmation. Pass `--yes` (or `-y`) to proceed without prompting.
- S3 secrets are read and rotated with `fs-users s3-secret` and `fs-users rotate-s3-secret` (see 8.11). The `s3 credentials` commands are no longer part of PanCTL. Both fs-users commands print the secret in plain text, so handle the output carefully.
- The gateway's port range no longer exists, and `s3 port-range` is no longer a command.

8.11 `fs-users`

Aliases: *`fs-user`*, *`fsu`*. A tenant's PanFS (data-plane) users. A PanFS user never signs in and holds no roles. It exists so the tenant's S3 gateway has a record to resolve a signed request onto, and it carries the POSIX identity (`uid` and `gid`) that S3 requests act under. This group is also where a tenant's S3 credentials are read and rotated, and where a bucket's owner comes from.

Command	Purpose
`fs-users list`	List the tenant's PanFS users, with the ID and S3 access key of each
`fs-users get <fs_user_id>`	Show one PanFS user
`fs-users create --name <name> [--uid <n>] [--gid <n>] [--s3-profile <profile>] [--access-key <key>]`	Create a PanFS user
`fs-users update <fs_user_id> [--uid <n>] [--gid <n>] [--s3-profile <profile>] [--access-key <key>]`	Change a PanFS user
`fs-users delete <fs_user_id>`	Delete a PanFS user (asks for confirmation; `--yes` skips it)
`fs-users s3-secret <fs_user_id>`	Show the user's S3 access key and secret
`fs-users rotate-s3-secret <fs_user_id>`	Mint a new S3 secret

Every command takes `--tenant <ref>`.

```
panctl fs-users create --tenant fintech --name alice --uid 2001 --gid 2001 --s3-profile
        privileged
panctl fs-users list --tenant fintech
panctl fs-users s3-secret <fs_user_id> --tenant fintech
panctl fs-users rotate-s3-secret <fs_user_id> --tenant fintech
```

Notes:

- `--s3-profile` is `user` (the default; works with the buckets it owns), `privileged` (also creates its own buckets over S3), or `disabled` (the credential is kept, but the gateways reject it).

- The access key defaults to the user's name, and `fs-users list` shows it. `--access-key` sets a different one on `create` or `update`. It is the value `s3 buckets create --owner` takes.
- The secret is not printed when the user is created; read it with `s3-secret`. `rotate-s3-secret` keeps the access key, prints the new secret once, and stops the old secret from working (a gateway can keep accepting the old one for a short time).
- Deleting a PanFS user does not delete its buckets.

8.12 `events`

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

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

Shows up to 200 of the most recent events (`--limit` accepts 1–200); this is the exact same feed shown in the web UI. The default columns are `TIME`, `SEVERITY`, `CATEGORY`, `EVENT_CODE`, and `MESSAGE`; with `--all-tenants`, a `TENANT_ID` column comes first. The rest of each event appears in `--format json`.

`--tenant` narrows to one tenant and `--all-tenants` widens to every tenant; the two cannot be combined. Both belong to the platform tenant, and the API refuses any other caller that passes either, even one naming its own tenant. With neither flag you read your own tenant's events. For the platform tenant, that is the realm's own telemetry, so `--all-tenants` is what reaches the events that tenants own. The profile's stored tenant is deliberately not consulted.

8.13 `alerts`

Alias: `al`. Currently active alerts (read-only).

```
panctl alerts list
panctl alerts list --tenant fintech
panctl alerts list --all-tenants
```

This shows a live snapshot of active alerts (not a historical log), the same set that drives the alert badge in the web UI. There is nothing to page or limit because the set is whatever is active.

`--tenant` and `--all-tenants` scope the read the same way they do on `events` and `audit`; both belong to the platform tenant, and with neither flag you read your own tenant's alerts, which for the platform tenant is the realm's own health.

8.14 `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
`--tenant <ref>`	Filter to one tenant
`--all-tenants`	Include every tenant's records. Cannot be combined with `--tenant`.
`--user <user>`	Filter to one user, by username or UUID. A username needs a single tenant in scope, so pass a UUID together with `--all-tenants`.
`--only <a,b>` / `--skip <a,b>`	Include or exclude specific action types. They can be combined, and `--skip` wins where they overlap. An action name that does not exist matches nothing.
`--since` / `--until`	Time bounds, both inclusive. Relative means that long ago, built from the units `d`, `h`, `m` (minutes), and `s`, for example `2h`, `2d`, or `2h30m`. Absolute is RFC-3339, with a `Z` or a numeric offset, or a bare date such as `2026-09-05`, which is read as UTC.
`--limit <n>`	Maximum records to return (no upper cap, unlike `events`)

As on `events` and `alerts`, `--tenant` and `--all-tenants` belong to the platform tenant, and with neither flag you read your own tenant's records. `--only`, `--skip`, `--since`, and `--until` narrow the records on the appliance, before the window of recent records is applied. `audit list` returns a window of recent records with no pagination, and `--limit` resizes that window.

The default `audit list` columns are `TIME`, `REQUEST_ID`, `PRINCIPAL_TYPE`, `PRINCIPAL_NAME`, `CLIENT_IP`, `ACTION`, `RESOURCE`, `TARGET_ID`, and `RESULT`. `PRINCIPAL_NAME` is written as `<tenant>/<name>`. The other fields of each record, such as the status code, durations, and request body, appear in `--format json`. A `result` of `failed` (an operation that broke) or `denied` (authorization refused it) is colored on a terminal.

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

8.15 `log-collections`

Aliases: *`log-collection`*, *`lc`*. Diagnostic bundles that `collect_logs` gathers from the realm's nodes. The short alias exists because this is typed while something is already wrong.

Access follows the existing roles. `platform-admin` can start, download, and delete a collection, while `platform-read-only` and `storage-admin` can read (`list` and `get`). The API enforces this and PanCTL does not check it, so a caller without the role receives the API's refusal.

A collection carries configuration and state from every node, so handle the archives as sensitive and keep them inside support channels. Archives stay on the Director that produced them, and they do not follow the API if it moves to another Director.

Old archives are cleaned when a new run starts or when a collection is deleted, not on a timer. By default, the newest 3 runs and 24 hours are kept.

Command	Purpose
`log-collections create [-f <file|->] [options]`	Start a collection
`log-collections list`	List collections, newest first
`log-collections get <collection_id>`	Show one collection in full, including per-node detail
`log-collections download <collection_id> [--output <file>]`	Fetch the archive

Command	Purpose
`log-collections delete <collection_id>`	Stop a run if it is still going, clean the nodes, and remove the archive (asks for confirmation)

Start a collection

```
panctl lc create                                    # everything, every node (collect_logs all)
panctl lc create --scope=directors --include-cores --node-timeout-seconds=1800
panctl lc create -f collect.json                    # hosts, storage sets, age
```

With no body and no flags, the API collects everything from every node; PanCTL does not invent a default. One collection runs at a time, and a second `create` is refused by the API with `409`. `create` answers with the collection it started, so `--format 'jsonpath={.id}'` captures the ID.

`create` flag	Meaning
`--scope`	The class of nodes to collect from: `all`, `repset`, `directors`, or `storage`
`--include-cores`	Include corefiles
`--include-databases`	Include system databases
`--include-sed`	Include self-encrypting-drive data
`--run-diagnostics`	Run diagnostics as part of the collection
`--wait-for-savecore`	Wait for savecore to finish first
`--node-timeout-seconds`	The per-node timeout, 0 to 86400
`-f <file|->`	The JSON body. It carries `hosts`, `storagesets`, and `age`, which have no flags. Flags override its fields.

`--include-cores=false` is sent as an instruction, while a flag you did not give is left out of the request.

Follow it to the end

```
panctl lc list
panctl lc get <id> -o jsonpath='{.status}' --wait-for='it != running' --wait-timeout=2h
```

`list` shows ID, STATUS, STARTED AT, FINISHED AT, HOST COUNT, and ARCHIVE SIZE. The size is in binary units, so 304138240 bytes shows as 290.0 MiB, and the status is one of `running`, `succeeded`, or `failed`. On a terminal a `failed` row is red and a `running` row is amber. `get` adds the options the collection was started with, the error behind a failure, who requested it, the archive's SHA-256, the duration, and the result for each node.

Download the archive

```
panctl lc download <id>                       # a name from the server, with progress
panctl lc download <id> --output case-4711.tar
panctl lc download <id> --output - | tar -tv   # to standard output
```

The archive is a tar file holding one `.tgz` per node, and it is routinely hundreds of megabytes. The download is streamed to disk and not held in memory, and its progress shows on standard error when that is a terminal. The file is named from the server's `Content-Disposition` header, for example `realm-logs_<realm>_<time>.tar`, so it says which realm and when. If the server gives no name, one is derived from the collection ID.

- **Interrupted downloads resume.** The file is written to a temporary file beside its destination and renamed once the whole archive has arrived, so an interrupted download never leaves a partial file that looks complete. Run the same command again and it continues from the kept partial.
- **The result is verified.** The archive is checked against its published length and SHA-256, and the bytes already on disk are included in that check. A partial that belongs to something else is caught and fetched again from the start. If the collection publishes no digest, the check is by length alone, and PanCTL says so.
- `--request-timeout` **bounds only getting a response.** A slow transfer is not cut off, but a transfer that stops delivering bytes fails.

Use `log-collections download` for archives, not `panctl api` (see 8.16), which reads a whole response into memory and would truncate one.

8.16 `api`

A raw request to any PanAPI path, for an operation that has no dedicated command. It carries the same credentials, session renewal, `--as-tenant-admin` delegation, and request deadline as every other command, and it validates the body whenever the cached specification describes the operation.

Otherwise the request is simply sent.

```
panctl api GET /api/v1/subnets/segment-type
panctl api GET /api/v1/volumes --query tenant_id=<uuid>
panctl api PUT /api/v1/subnets/segment-type -f - <<< '{"segment_type":"vxlan"}'
```

Command	Purpose
`api <METHOD> <path> [--query k=v] [-f <file|->]`	One authenticated request. The method is not case sensitive, and `--query` can be repeated.
`api list [--filter <expr>]`	The operations the cached specification declares: method, path, and summary. No network call.
`api explain <path> [--recurse]`	How to call a path: its methods, parameters and whether each is required, the body fields, and the response codes. It takes the template that `api list` prints or a concrete path copied from a log, and `--recurse` covers everything beneath it.
`api version`	Which cached specification validation and `--help` are reading, from which server, and when it was retrieved. For the live answer, use `panctl api GET /api/v1/about`.

`api list`, `api explain`, and `api version` read the document cached at login. With nothing cached they say so and point to `auth login`.

The response is written as the API returned it because PanCTL cannot know the shape of something it was not told about. Add `--format json` to pretty-print it, or `-o jsonpath=` to extract a value. Prefer a real command wherever one exists; `panctl api` resolves no names, supplies no tenant, renders no table, and completes nothing, so you give it a concrete path with IDs already in it.

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)
```

The bash script needs bash 4 or later. macOS ships bash 3.2, so install a newer bash and `bash-completion` with Homebrew there. Use `source <(panctl completion bash)` or `eval "$(panctl completion bash)"`. An unquoted `eval $(panctl completion bash)` exits `0` but registers nothing because word-splitting mangles the script.

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 platform/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 --storageset-id <id> --dataserials <ds1>,<ds2>,<ds3>
# add-storage is asynchronous: wait until the nodes are absorbed (see 8.3 for a polling
        example), then until the realm reports
# panfs_mounted (panctl realm get); tenant and volume create return 503 until both are true

# 4. Create a tenant and give it capacity
panctl tenants create --name fintech --display-name FinTech --email admin@fintech.example.com
panctl tenants assign-storagesets --tenant fintech --storageset-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
# The tenant becomes active once a node address has been issued on the subnet (usually
        seconds). Volume create returns 409 until then.
panctl tenants get <tenant_id> -o 'jsonpath={.tenant.status}' --wait-for=active

# 6. Create a volume inside the tenant, acting as a platform administrator
panctl volumes create --as-tenant-admin fintech --name /data --storageset-id <id> --hard-quota
        5TB

# 7. Optional: create a tenant administrator and use it instead.
#    Tenant users are not fully supported in this release (see 8.7).
panctl users create --tenant fintech --username jsmith --email jsmith@fintech.example.com
panctl users set-roles --user <user_id> --tenant fintech --roles admin
panctl auth login <server> --username fintech/jsmith --profile fintech
panctl --profile fintech volumes create --name /data2 --storageset-id <id> --hard-quota 5TB
```

11. Troubleshooting

The following table lists each symptom and its likely cause and fix.

Symptom	Likely Cause / Fix
`error: profile "<name>" not authenticated. Run: panctl auth login ...`	The session for that profile has never logged in or has been logged out. Run `auth login` again.
`error: storage set "Demo Sett" not found: no storage set has that name (pass the id, or check 'panctl storagesets list')`	A name that resolved to nothing. A path that contains an unresolved name is never sent, so this is a typo and not a missing object.
`error: tenant reference "acme" is ambiguous: 2 tenants share that short name`	More than one item has that name, so PanCTL refuses to pick one. Pass the UUID or ID instead.
`error: unknown command "create" for "panctl nodes"`	Exit `2`. A command exists only where the API defines a matching operation, so `nodes create` does not exist. Nothing was sent. For an operation with no command, use `panctl api` (see 8.16).
`error: profile "<name>" not found in ~/.panctl/profiles.toml`	Check for a typo in `--profile` or `PANCTL_PROFILE`, or the profile was never created.
`error: -f <file|-> is required`	This command needs a JSON body; pass a file path or `-` to read from standard input.
`error: <field> is required`	A required positional argument or flag is missing.
`error: session expired. Run: panctl auth login --profile <name>`	Automatic renewal failed (e.g., the refresh token itself expired). Log in again.
`dial tcp ...: connection refused`	The appliance is unreachable; check the server address, network path, and firewall rules.

Symptom	Likely Cause / Fix
A raw JSON body, for example `{"detail":"Volume not found"}`	This is the API's own refusal, relayed exactly and with no `error:` prefix, so it stays valid JSON. Pipe it through `jq` for easier reading, e.g. `panctl volumes create ... 2>&1 | jq .`
`error: this operation requires confirmation; pass --yes to proceed`	Exit `2`. A destructive command ran with no terminal to ask. Add `--yes` (or `-y`) when that is what you mean.

Messages that start with `error:` come from PanCTL itself, and API refusals (not found, already exists, permission denied, quota exceeded) do not. A search for `^error:` therefore does not match every failure, so use the exit code to detect failure.

12. Scripting

The table layout is for reading. Machines should read JSON, and scripts should test the exit code (see 5.5).

12.1 Take the Id from the Create

```
tenant_id=$(panctl tenants create --name=acme --display-name="ACME Corp" \
  --email=admin@acme.example.com --format jsonpath='{.id}')
lc_id=$(panctl lc create --scope=storage --format jsonpath='{.id}')
```

A create returns the new object, so no second call is needed, and `{.id}` is the identifier on every resource.

12.2 Find Something You Did Not Just Create

```
tenant_id=$(panctl tenants list --format 'jsonpath={.tenants[?(@.name=="acme")].id}')

# Every addable node: unassigned, and of a role that carries storage
panctl nodes list --format \
  'jsonpath={.nodes[?(@.role=="vosd" || @.role=="vhost" && @.storageset_id==0)].dataserial}'
```

Single-quote the expression. Terms fold from left to right with no precedence and no grouping, so `a && b || c` means `(a && b) || c`. To read a narrowed list instead of capturing a field, use `--filter` (see 5.8). The two compose.

12.3 One Line per Row

```
panctl volumes list --format \
  'jsonpath={range .volumes[*]}{.id}{"\t"}{.pathname}{"\n"}{end}'
panctl volumes list --no-header | awk '{print $2}'
```

12.4 Empty Results and Several Matches

- An empty capture is silent. A selector that matches nothing prints nothing and exits `0`, and a failed create leaves standard output empty, so check for an empty variable yourself.
- Several matches are joined with a space. Use `read -r -a` to split them, or reject values that contain one.

12.5 Collect and Fetch Logs Unattended

```
id=$(panctl lc create --format jsonpath='{.id}') || exit
panctl lc get "$id" -o jsonpath='{.status}' --wait-for='it != running' --wait-timeout=2h
panctl lc download "$id" --output "logs-$(date +%F).tar"
```

12.6 Run Unattended

```
export PANCTL_API_TOKEN=vdt1_...
export PANCTL_API_SERVER=10.97.26.200
export PANCTL_NO_PAGER=1
panctl volumes list          # no login step, and no profile is read
panctl volumes delete 7 --yes
```

Token mode reads no `profiles.toml`, renews nothing, and exits `3` if the token is rejected. A destructive command run with no terminal needs `--yes` (see 5.10).

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