Pelican Panel API Reference

> **Version:** Pelican v1 (forked from Pterodactyl Panel v1) **Status:** *Beta* — endpoints are identical to Pterodactyl v1 unless otherwise noted.\ **Base URL examples use** `https://panel.example.com`. Replace with your own panel domain.

---

1. Getting Started

Area	Client API prefix	Application API prefix
Base URL	`/api/client`	`/api/application`
Auth header	`Authorization: Bearer ptlc_xxx`	`Authorization: Bearer ptla_xxx`
Accept	`application/vnd.pterodactyl.v1+json`	same

Generate Keys:\ • *Client key*: **Account → API Credentials → New Key** (panel UI)\ • *Application key*: **Admin → API → New Key** (admin UI)

---

2. Common Conventions

- All dates are ISO-8601 (`2025-07-17T15:30:00+00:00`).
- Pagination parameters: `page` , `per_page` . Default `per_page` =50.
- `include` can expand related resources, e.g. `?include=allocations,subusers` .
- Standard response wrapper:

```
{
  "object": "list|resource",
  "data": [...],
  "meta": { "pagination": { ... } }
}
```

---

3. Client API

*Endpoints a **server owner or sub-user** can call.*

3.1 Account Management

Verb	Endpoint	Purpose
GET	`/api/client/account`	Fetch the logged-in user profile
GET	`/api/client/account/api-keys`	List API keys
POST	`/api/client/account/api-keys`	Create key (`{ description, allowed_ips[] }`)
DELETE	`/api/client/account/api-keys/{identifier}`	Revoke key
GET	`/api/client/account/two-factor`	Get 2FA QR/barcode
POST	`/api/client/account/two-factor`	Enable 2FA (`{ code }`)
POST	`/api/client/account/two-factor/disable`	Disable 2FA (`{ password }`)
POST	`/api/client/account/email`	Change e-mail (`{ email, password }`)
POST	`/api/client/account/password`	Change password (`{ current_password, password }`)

3.2 Servers

Verb	Endpoint	Purpose
GET	`/api/client`	List all accessible servers
GET	`/api/client/servers/{id}`	Server details
GET	`/api/client/servers/{id}/resources`	Live utilisation + power state
POST	`/api/client/servers/{id}/power`	Control power (`{ signal: start stop restart kill }`)
POST	`/api/client/servers/{id}/command`	Send console command (`{ command }`)
GET	`/api/client/servers/{id}/utilization`	Alias of `/resources` (legacy)
GET	`/api/client/servers/{id}/websocket`	Generate WebSocket JWT + URL

3.3 File Management

Verb	Endpoint	Purpose
GET	`/api/client/servers/{id}/files/list`	List directory (`?directory=/path`)
GET	`/api/client/servers/{id}/files/contents`	Download file contents (`?file=/path`)
GET	`/api/client/servers/{id}/files/download`	Temporary download URL (`?file=/path`)
POST	`/api/client/servers/{id}/files/write`	Save/overwrite (`{ file, contents }`)
POST	`/api/client/servers/{id}/files/upload`	Multipart upload (field `files[]`)
POST	`/api/client/servers/{id}/files/delete`	Delete array of files/dirs (`{ root, files[] }`)
POST	`/api/client/servers/{id}/files/rename`	Rename array (`{ root, files[{ from,to }] }`)
POST	`/api/client/servers/{id}/files/copy`	Duplicate (`{ location, files[] }`)
POST	`/api/client/servers/{id}/files/compress`	Compress (`{ root, files[], destination }`)
POST	`/api/client/servers/{id}/files/decompress`	De-archive (`{ root, file }`)
POST	`/api/client/servers/{id}/files/create-directory`	Make folder (`{ name }`)
POST	`/api/client/servers/{id}/files/chmod`	Change mode (`{ root, files[{ file, permissions }] }`)

3.4 Database Management

Verb	Endpoint	Purpose
GET	`/api/client/servers/{id}/databases`	List DBs
POST	`/api/client/servers/{id}/databases`	Create (`{ database, remote, host, password }`)

Verb	Endpoint	Purpose
POST	`/api/client/servers/{id}/databases/{db}/rotate-password`	New password
DELETE	`/api/client/servers/{id}/databases/{db}`	Delete

3.5 Backup Management

Verb	Endpoint	Purpose
GET	`/api/client/servers/{id}/backups`	List backups
POST	`/api/client/servers/{id}/backups`	Create (`{ name, ignored_files, lock, is_compressed }`)
GET	`/api/client/servers/{id}/backups/{uuid}`	Backup details
GET	`/api/client/servers/{id}/backups/{uuid}/download`	Signed download URL
POST	`/api/client/servers/{id}/backups/{uuid}/lock`	Lock (immutable)
POST	`/api/client/servers/{id}/backups/{uuid}/unlock`	Unlock
DELETE	`/api/client/servers/{id}/backups/{uuid}`	Delete backup

3.6 Scheduled Tasks

Verb	Endpoint	Purpose
GET	`/api/client/servers/{id}/schedules`	List schedules
GET	`/api/client/servers/{id}/schedules/{schedule}`	Schedule details
POST	`/api/client/servers/{id}/schedules`	Create (`{ name, minute, hour, day_of_week, day_of_month, enabled, only_when_online }`)
PATCH	`/api/client/servers/{id}/schedules/{schedule}`	Update
DELETE	`/api/client/servers/{id}/schedules/{schedule}`	Delete

Verb	Endpoint	Purpose
POST	`/api/client/servers/{id}/schedules/{schedule}/execute`	Run now

Schedule Tasks

Verb	Endpoint	Purpose
GET	`/api/client/servers/{id}/schedules/{schedule}/tasks`	List tasks
POST	`/api/client/servers/{id}/schedules/{schedule}/tasks`	Add (`{ action, payload, time_offset, sequence_id }`)
PATCH	`/api/client/servers/{id}/schedules/{schedule}/tasks/{task}`	Update
DELETE	`/api/client/servers/{id}/schedules/{schedule}/tasks/{task}`	Delete

3.7 Network (Allocations)

Verb	Endpoint	Purpose
GET	`/api/client/servers/{id}/network/allocations`	List allocations
POST	`/api/client/servers/{id}/network/allocations`	Request extra allocation
POST	`/api/client/servers/{id}/network/allocations/{allocation}/primary`	Set primary
POST	`/api/client/servers/{id}/network/allocations/{allocation}`	Update notes (`{ notes }`)
DELETE	`/api/client/servers/{id}/network/allocations/{allocation}`	Remove allocation

3.8 Sub-Users

Verb	Endpoint	Purpose
GET	`/api/client/servers/{id}/users`	List sub-users
GET	`/api/client/servers/{id}/users/{user}`	Sub-user details
POST	`/api/client/servers/{id}/users`	Invite (`{ email, permissions[] }`)
POST	`/api/client/servers/{id}/users/{user}`	Update perms (`{ permissions[] }`)

Verb	Endpoint	Purpose
DELETE	`/api/client/servers/{id}/users/{user}`	Revoke sub-user

3.9 Startup / Variables

Verb	Endpoint	Purpose
GET	`/api/client/servers/{id}/startup`	Startup variables array
GET	`/api/client/servers/{id}/startup/variable/{key}`	Variable details
PATCH	`/api/client/servers/{id}/startup/variable/{key}`	Update value (`{ value }`)

4. Application API

Administrative endpoints — require Application API key.

4.1 Users

Verb	Endpoint	Purpose
GET	`/api/application/users`	List users (`filter[username email]` support)
POST	`/api/application/users`	Create (`{ username, email, first_name, last_name, password, root_admin }`)
GET	`/api/application/users/{id}`	User details
PATCH	`/api/application/users/{id}`	Update (same fields)
DELETE	`/api/application/users/{id}`	Delete user

4.2 Servers

Verb	Endpoint	Purpose
GET	`/api/application/servers`	List all servers
POST	`/api/application/servers`	Create (complex JSON: name, user, egg, docker_image, limits, feature_limits, environment, allocation, deploy)
GET	`/api/application/servers/{id}`	Server details

Verb	Endpoint	Purpose
PATCH	`/api/application/servers/{id}/details`	Update name/owner/etc.
PATCH	`/api/application/servers/{id}/build`	Update RAM/CPU/disk
PATCH	`/api/application/servers/{id}/startup`	Update startup & env vars
POST	`/api/application/servers/{id}/suspend`	Suspend
POST	`/api/application/servers/{id}/unsuspend`	Unsuspend
POST	`/api/application/servers/{id}/reinstall`	Re-install
DELETE	`/api/application/servers/{id}`	Delete server

4.3 Nodes

Verb	Endpoint	Purpose
GET	`/api/application/nodes`	List nodes
POST	`/api/application/nodes`	Create node
GET	`/api/application/nodes/{id}`	Node details
PATCH	`/api/application/nodes/{id}`	Update node
GET	`/api/application/nodes/{id}/config`	Export wings config
DELETE	`/api/application/nodes/{id}`	Delete node
GET	`/api/application/nodes/{id}/allocations`	List allocations
POST	`/api/application/nodes/{id}/allocations`	Create allocations (`{ ip, ports[] }`)
DELETE	`/api/application/nodes/{id}/allocations/{alloc}`	Delete allocation

4.4 Locations

Verb	Endpoint	Purpose
GET	`/api/application/locations`	List locations

Verb	Endpoint	Purpose
POST	`/api/application/locations`	Create location (`{ short, long }`)
GET	`/api/application/locations/{id}`	Location details
PATCH	`/api/application/locations/{id}`	Update location
DELETE	`/api/application/locations/{id}`	Delete location

4.5 Nests & Eggs

Verb	Endpoint	Purpose
GET	`/api/application/nests`	List nests
GET	`/api/application/nests/{nest}`	Nest details
GET	`/api/application/nests/{nest}/eggs`	List eggs in a nest
GET	`/api/application/nests/{nest}/eggs/{egg}`	Egg details (environment schema, docker image list, etc.)

---

5. WebSocket Console

`wss://panel.example.com/api/client/servers/{id}/ws?token=JWT_HERE`\ Use the token returned by `GET /servers/{id}/websocket` to open a console stream (`event` + `args[]` messages).

---

6. Rate Limits

- **Client API:** 240 requests / minute / key
- **Application API:** 240 requests / minute / key\ Headers: `X-RateLimit-Limit`, `X-RateLimit-Remaining`, `X-RateLimit-Reset` (epoch).

---

7. Error Handling

Code	Meaning	Notes
400	Bad Request	Invalid JSON, missing fields
401	Unauthorized	Missing/invalid bearer token
403	Forbidden	Token exists but lacks permission
404	Not Found	Resource doesn't exist or not accessible

Code	Meaning	Notes
422	Validation Error	Field constraints failed
429	Too Many Requests	Rate-limit hit
500	Server Error	Panel or wings internal error

---

8. Pelican-Specific Notes

- **Branding**: Header `X-Pelican-Panel: true` distinguishes Pelican responses.
- **Beta Add-ons** (may change without notice):
- `POST /api/client/servers/{id}/power` now supports `"hibernate"`.
- Additional `PATCH /api/application/servers/{id}/docker-image` to change Docker image without full rebuild.

---

9. Example Agent Workflows (Minecraft)

1. **Start a server if it's offline**
2. `GET /api/client/servers/{id}/resources` → check `current_state`.
3. If `offline`, `POST /power { signal:"start" }`.
4. **Upload a Bukkit plugin**
5. `POST /files/upload` with Multipart `file` → `/plugins/Plugin.jar`.
6. `POST /power { signal:"restart" }`.
7. **Nightly backup**
8. `POST /backups { name:"nightly", lock:false }`.
9. **Rotate DB password weekly**
10. `POST /databases/{db}/rotate-password`.
