Kommandozeile

Referenz und Anleitungen zur aYOUne-CLI.

139 Seiten in dieser Sammlung.

aYOUne CLI Überblick

aYOUne CLI Überblick

Die aYOUne CLI (ay) ist das offizielle Kommandozeilen-Werkzeug für die aYOUne-Plattform. Sie liefert direkten Zugriff auf alle ~602 Collections, die Deployment-Pipeline, Custom Functions und das lokale Self-Hosting — aus jedem Terminal, ohne Browser.

Wofür

- Datenzugriff: Jede Collection lesen, schreiben, durchsuchen, aggregieren.
- Automatisierung: Exports, Batch-Updates, scheduled Scripts.
- DevOps: Deployments, Logs, Pipeline-Status, Cluster-Health.
- FaaS: Custom-Functions lokal entwickeln, deployen, invoken.
- Self-Hosting: Docker-Compose-Stacks starten/stoppen, Updates einspielen.

Schnellstart

```
npm install -g @tolinax/ayoune-cli
ay setup                    # Interaktiver First-Run-Wizard
ay login                    # Authentifizierung via Browser
ay list consumers           # Erste Abfrage
```

Siehe Installation (wiki:cli/install) und Erster Login (wiki:cli/login) für Details.

Struktur dieser Dokumentation

- Einstieg (wiki:cli/getting-started) — Installation, Setup, First-Run-Tutorial
- Konzepte (wiki:cli/concepts/authentication) — Auth, Customer-Context, Output-Formate
- Befehle (wiki:cli/commands) — Vollständige Referenz aller 45 Commands, gruppiert
- Rezepte (wiki:cli/recipes) — Häufige Workflows
- Problembehebung (wiki:cli/troubleshooting) — Fehler, Debugging, bekannte Probleme

Version

Aktuelle CLI-Version: @tolinax/ayoune-cli@2026.17.0.

::: info Die CLI verwendet CalVer (Calendar Versioning) im Format YYYY.MINOR.PATCH. :::

Installation

Installation

Die aYOUne CLI wird als npm-Paket verteilt und läuft auf Node.js "e 20.

Voraussetzungen

- Node.js "e 20 (empfohlen: aktuelle LTS).
- npm "e 10 oder pnpm "e 8 oder yarn "e 4.
- Eine aYOUne-Tenant-Zugehörigkeit (Login erforderlich für fast alle Commands).

Global installieren

```
npm install -g @tolinax/ayoune-cli
```

Nach erfolgreicher Installation steht das Binary ay im Path zur Verfügung:

```
ay --version
# 2026.17.0
```

Alternativen

Ohne globale Installation

```
npx @tolinax/ayoune-cli login
```

pnpm

```
pnpm add -g @tolinax/ayoune-cli
```

yarn

```
yarn global add @tolinax/ayoune-cli
```

Shell-Completion

```
# Bash
ay completions bash > ~/.ayoune-completion.bash
echo "source ~/.ayoune-completion.bash" >> ~/.bashrc

# Zsh
ay completions zsh > ~/.ayoune-completion.zsh
echo "source ~/.ayoune-completion.zsh" >> ~/.zshrc

# Fish
ay completions fish > ~/.config/fish/completions/ay.fish
```

Update

```
npm install -g @tolinax/ayoune-cli@latest
```

::: tip Im Umfeld des Monorepos entspricht die CLI-Version der aktuellen Platform-Version. Prüfe regelmäßig auf Updates, um neue Commands und Bugfixes zu bekommen. :::

Nächster Schritt

Erst-Setup durchführen (wiki:cli/setup) — interaktiver Wizard, der Customer-Context, Default-Output-Format und Auth konfiguriert.

Erst-Setup

Erst-Setup

Der Befehl ay setup ist ein interaktiver Wizard, der alle wichtigen Einstellungen einmalig erfasst und in ~/.config/ayoune/config.json speichert.

Aufruf

```
ay setup
```

Was der Wizard abfragt

- API-Host — Default api.ayoune.app. Für Self-Hosting: eigene Domain.
- Auth-Host — Default auth.ayoune.app.
- Default-Customer-Context — falls du Zugriff auf mehrere Tenants hast, hier die primäre Auswahl.
- Default-Output-Format — json, yaml, table, csv.
- Editor für ay edit — code, vim, nano, $EDITOR (fallback).
- Locale — de oder en, beeinflusst Fehlermeldungen und Help-Texte.

Manuelle Konfiguration

Ohne Wizard direkt die Config-Datei bearbeiten:

```
ay config set apiHost api.ayoune.app
ay config set defaultFormat yaml
ay config set editor vim
ay config get                       # aktuelle Config zeigen
```

Config-File-Ort: ~/.config/ayoune/config.json (XDG-konform).

Nächster Schritt

Erster Login (wiki:cli/login) — Browser-basierte Authentifizierung gegen auth.ayoune.app.

Authentifizierung

Authentifizierung

Die meisten CLI-Befehle erfordern eine aktive Session. Die CLI nutzt auth.ayoune.app mit OAuth2-PKCE-Flow im Browser.

Anmelden

```
ay login
```

Der Befehl öffnet automatisch den Default-Browser auf https://auth.ayoune.app/?continue=.... Nach erfolgreicher Anmeldung wird das JWT lokal in ~/.aYOUne/storage/ abgelegt und automatisch alle 55 Minuten erneuert.

Aktuellen User anzeigen

```
ay whoami
```

Zeigt: User-ID, Email, aktiver Customer-Context, Rechte-Scope.

Customer-Context wechseln

Bei Multi-Tenant-Zugehörigkeit:

```
ay switch                    # interaktiver Picker
ay switch customer <slug>    # direkt
ay switch back               # zum vorherigen Context zurück
ay switch current            # aktuellen Context anzeigen
ay switch list               # alle verfügbaren Tenants
```

::: info Der Customer-Context bestimmt das _customerID-Scoping jeder nachfolgenden Query. ay list consumers zeigt NUR Consumers des aktiven Tenants. :::

Abmelden

```
ay logout              # Token löschen
```

Token-Storage inspizieren

```
ay storage set         # Einzelnen Wert setzen
ay storage clear       # Kompletten Storage leeren (forced logout)
```

Troubleshooting

- "Login Browser öffnet sich nicht" — Manuell zu https://auth.ayoune.app/?continue=http://localhost:{port} navigieren (Port wird im Terminal ausgegeben).
- "Session expired" — CLI refreshed JWT alle 55min. Wenn das Gerät länger offline war, ay login erneut ausführen.
- "No access to customer" — ay whoami prüfen, ggf. ay switch list und korrekten Context wählen.

Nächster Schritt

First-Run-Tutorial (wiki:cli/first-run-tutorial) — die ersten fünf produktiven Befehle.

First-Run-Tutorial

First-Run-Tutorial

Nach Install (wiki:cli/install), Setup (wiki:cli/setup) und Login (wiki:cli/login): die fünf ersten produktiven Befehle.

1. Module erkunden

```
ay modules
```

Interaktive Übersicht aller 25 Module (CRM, Marketing, CMS, DevOps, …) mit ihren Collections. Navigiere mit Pfeiltasten, öffne eine Collection mit Enter.

2. Erste Abfrage

```
ay list consumers --limit 5 --format table
```

Zeigt die letzten 5 Consumers als Tabelle. Flags:

- --limit N — Ergebnis-Anzahl
- --format json|yaml|table|csv — Output-Format
- --sort createdAt:-1 — Sortierung

3. Einen einzelnen Datensatz holen

```
ay get consumers 64a1b2c3d4e5f60012345678
```

Liefert den kompletten Consumer-Datensatz. Mit -f yaml für besser lesbares Format.

4. Eintrag erstellen

```
ay create projects "Neues Pilotprojekt"
```

Legt ein Projects-Dokument mit name: "Neues Pilotprojekt" an und gibt die neue ID zurück. Weitere Felder via --set field=value.

5. Aggregation ausführen

```
ay aggregate wizard
```

Interaktiver Wizard, der Collection, Pipeline-Stages und Output-Optionen Schritt-für-Schritt abfragt und am Ende das MongoDB-Pipeline-JSON + den ausgeführten Output zeigt. Gute Einstiegs-Lernkurve für Aggregations-Arbeit.

Was als Nächstes

- Konzept: Customer-Context (wiki:cli/concepts/customer-context) — Multi-Tenant-Grundlagen.
- Konzept: Output-Formate (wiki:cli/concepts/output-formats) — wann JSON vs YAML vs Table.
- Befehlsreferenz (wiki:cli/commands) — alle 45 Commands, gruppiert.

::: tip Tipp Mit ay --help oder ay <cmd> --help bekommst du jederzeit Inline-Hilfe. :::

Authentifizierung

Authentifizierung

Die CLI nutzt denselben Auth-Flow wie alle anderen aYOUne-Clients: OAuth2-PKCE gegen auth.ayoune.app, JWT-Tokens mit 60-Minuten-Lifetime und automatischem Refresh 5 Minuten vor

Ablauf.

Flow

ay login startet einen lokalen HTTP-Listener auf einem freien Port.

Browser öffnet https://auth.ayoune.app/?continue=http://localhost:{port}&code_challenge=....

Nach Login redirected der Auth-Service mit ?code=... zurück auf den Listener.

CLI tauscht Code gegen { accessToken, refreshToken } via /oauth/token.

Tokens werden in ~/.aYOUne/storage/tokens.json abgelegt (OS-Dateirechte 0600).

Token-Lifecycle

| Token | TTL | Refresh | |---|---|---| | accessToken | 60 min | Automatisch 55min nach Ausstellung | | refreshToken | 30 Tage | Nur beim Re-Login erneuert |

Wenn der Refresh fehlschlägt (z.B. Refresh-Token expired), wird der User zum erneuten ay login aufgefordert.

Service-Accounts (Headless)

Für CI/CD + scheduled Scripts: API-Keys statt Browser-Login.

```
ay login --api-key $AYOUNE_API_KEY
```

API-Keys werden pro Customer im Admin-UI angelegt (mobile-app !’ Einstellungen !’ Integrationen !’ API-Keys). Sie haben dieselben Rechte wie der erzeugende User und können widerrufen werden.

Rechte-Scope

Die CLI respektiert die serverseitigen Rechte-Gates (<module>.<entity>.<verb>, z.B. crm.consumers.read). Unzureichende Rechte resultieren in HTTP-403 mit klarer Meldung.

Prüfen, welche Module dem aktiven User zugänglich sind:

```
ay access
```

Siehe auch

- First-Run-Tutorial (wiki:cli/first-run-tutorial)
- Customer-Context (wiki:cli/concepts/customer-context)

Customer-Context

Customer-Context

aYOUne ist multi-tenant. Fast jede Collection hat ein _customerID-Feld, das den Zugriff scoped. Die CLI hält einen aktiven Customer-Context pro Session vor.

Aktiven Context anzeigen

```
  ay whoami               # zeigt auch aktiven Customer
  ay switch current       # fokussiert nur auf Customer
```

Die Terminal-Prompt-Integration (ay prompt, optional) zeigt den aktiven Customer als Präfix, z.B. [tolinax] $.

Context wechseln

```
  ay switch                             # interaktiver Picker (Tabelle mit Slug, Name,
Role)
  ay switch customer tolinax            # direkt via Customer-Slug
  ay switch customer 64a1b2c3d4e5...    # direkt via Customer-ID
  ay switch back                        # zurück auf vorherigen Context
  ay switch list                        # alle verfügbaren Tenants
```

Der neue Context persistiert bis zum nächsten ay switch oder ay logout.

Auswirkung auf Abfragen

- List/Get/Search/Aggregate filtern immer nach _customerID = activeContext.
- Create setzt _customerID = activeContext implizit.
- Update/Delete prüfen serverseitig, dass das Target-Dokument dem aktiven Customer gehört — sonst HTTP-403.

::: warning Achtung Bei ay export + ay batch ohne explizites --customer-Flag wird der aktive Context verwendet. Prüfe vor großen Operationen mit ay whoami. :::

Platform-Operator-Modus

Tolinax-Mitarbeiter mit restrictApiSuperUser-Recht können in den Platform-Scope wechseln:

```
  ay switch customer platform
```

Im Platform-Scope sehen/ändern sie Cross-Tenant-Daten (Registry, Infra-Config, Diagnostik). Siehe auch api-superuser-home (reference:api-superuser-home).

Siehe auch

- Authentifizierung (wiki:cli/concepts/authentication)
- Permissions-Befehl (wiki:cli/auth-config)

Output-Formate

Output-Formate

Fast jeder CLI-Befehl akzeptiert --format (alias -f). Der Default ist via ay setup / ay config set defaultFormat einstellbar.

Verfügbare Formate

| Format | Flag | Einsatz | |---|---|---| | json | -f json | Maschinen-lesbar, für Pipes in jq etc. | | yaml | -f yaml | Menschen-lesbar, Default für ay describe + ay edit | | table | -f table | Terminal-Übersicht, ASCII-

Grid, gekürzte Spalten | | csv | -f csv | Export für Excel / Tabellen-Tools |

Nur Payload (ohne Envelope)

```
ay list consumers -m                    # entfernt { payload, meta }-Wrapper
ay get consumers <id> -m                # nur das Consumer-Objekt
```

Pipe-Beispiele

```
# Alle Consumer-Emails extrahieren
ay list consumers -f json -m | jq -r '.[].email'

# CSV-Export direkt in Datei
ay list orders -f csv > orders.csv

# YAML-Diff zwischen zwei IDs
diff <(ay get consumers A -f yaml) <(ay get consumers B -f yaml)
```

Farbausgabe

- --no-color deaktiviert ANSI-Color-Codes.
- FORCE_COLOR=1 erzwingt Farben auch in Pipes.
- NO_COLOR=1 (Unix-Standard) respektiert die CLI.

Fortschrittsausgabe

Lang laufende Operationen (ay export, ay batch, ay db pull) zeigen Spinner + Fortschrittsbalken. In nicht-TTY-Umgebungen (CI-Logs) fallen sie auf statische Log-Lines zurück.

Siehe auch

- First-Run-Tutorial (wiki:cli/first-run-tutorial)
- Aggregate-Befehl (wiki:cli/queries)

Befehls-Referenz

Befehls-Referenz

Die CLI bündelt 45 Commands in 8 logische Gruppen. Jede Gruppe hat eine eigene Landing-Page mit Überblick + Details zu den einzelnen Befehlen.

Gruppen

| Gruppe | Anzahl | Zweck | |---|---|---| | CRUD & Core (wiki:cli/crud-core) | 12 | Daten lesen, erstellen, ändern, löschen | | Queries & Analysis (wiki:cli/queries) | 5 | Search, Aggregate, Export, Batch, Access | | Streaming & Events (wiki:cli/streaming) | 2 | Live-Updates, Platform-Events | | Auth & Config (wiki:cli/auth-config) | 8 | Login, Switch, Context, Setup, Config, Alias | | Folder-based (wiki:cli/folder-based) | 3 | local, functions, provision mit Subcommands | | DevOps & Deployment (wiki:cli/devops) | 7 | deploy, monitor, status, db, release, pm, self-host-update | | Utilities (wiki:cli/utilities) | 5 | actions, exec, ai, services, webhooks | | Miscellaneous (wiki:cli/misc) | 3+ | jobs, users, sync, permissions, templates,

credentials, rdp, doctor, completions |

Globale Flags

Jeder Command akzeptiert:

| Flag | Zweck | |---|---| | --help / -h | Inline-Hilfe zum Command | | --format / -f | Output-Format: json, yaml, table, csv | | --no-color | Deaktiviert Farbausgabe | | --verbose / -v | Mehr Log-Details | | --quiet / -q | Unterdrückt nicht-Fehler-Output | | --customer <slug> | Override aktivem Customer-Context für diesen Call |

Inline-Hilfe

```
ay --help                    # Alle Commands
ay <cmd> --help              # Command-Details
ay <cmd> <sub> --help        # Subcommand-Details
```

Aliase

Viele Commands haben Kurzformen (z.B. l für list, g für get). Siehe jede Command-Seite für die aktuellen Aliase.

CRUD & Core

CRUD & Core

Die 12 Basis-Commands für Daten-Zugriff auf alle Collections der Plattform.

Commands

| Command | Alias | Zweck | |---|---|---| | list (wiki:cli/crud-core/list) | l | Liste einer Collection mit Paginierung | | get (wiki:cli/crud-core/get) | g | Einzelner Datensatz via ID | | create (wiki:cli/crud-core/create) | c | Neuen Datensatz anlegen | | edit (wiki:cli/crud-core/edit) | e | Datensatz im Editor öffnen | | copy (wiki:cli/crud-core/copy) | cp | Datensatz duplizieren | | delete (wiki:cli/crud-core/delete) | rm | Datensatz per ID löschen | | update (wiki:cli/crud-core/update) | u | Einzelne Felder aktualisieren | | describe (wiki:cli/crud-core/describe) | d | Schema + YAML-View | | storage (wiki:cli/crud-core/storage) | s | Token- & Prefs-Speicher | | audit (wiki:cli/crud-core/audit) | history | Audit-Trail eines Docs | | modules (wiki:cli/crud-core/modules) | m | Interaktive Modul/Collection-Browser | | upload (wiki:cli/crud-core/upload) | — | File-Upload nach Documents |

Gemeinsames Pattern

Alle CRUD-Commands erwarten als erstes Argument die Collection (lowercased plural aus modelsAndRights.ts, z.B. consumers, products, orders), und — wo sinnvoll — als zweites die ObjectId bzw. ein Name/Slug.

Beispiele

```
ay list consumers                    # Liste
```

```
ay list consumers --limit 50 --sort 'createdAt:-1'
ay get products 64a1b2c3d4e5f60012345678       # Einzeldatensatz
ay create projects "Neues Projekt"             # anlegen
ay update tasks <id> --set status=done         # partielles Update
ay delete logs <id>                            # löschen
```

Siehe auch

- Konzept: Output-Formate (wiki:cli/concepts/output-formats)
- Befehl: list (wiki:cli/crud-core/list)

---

ay list

ay list (Alias: l)

Listet Dokumente einer Collection mit Paginierung, Sortierung und optionalen Filtern.

Syntax

```
ay list <collection> [flags]
```

Flags

Flag	Default	Zweck
--limit N	20	Anzahl Ergebnisse pro Page
--skip N	0	Offset
--sort <field>:<-1|1>	createdAt:-1	Sortierung
--filter '<json>'	{}	MongoDB-Filter-Objekt
--fields '<json>'	alle	Projection
--format	config-default	json / yaml / table / csv
-m / --minimal	false	Nur Payload, kein {payload, meta}-Envelope

Beispiele

```
# Einfache Liste mit 5 Einträgen
ay list consumers --limit 5

# Gefiltert, nur ausgewählte Felder
ay list orders \
  --filter '{"status":"paid","total":{"$gte":100}}' \
  --fields '{"_id":1,"total":1,"consumer":1}' \
  --sort 'createdAt:-1' \
  --limit 10

# Export zu CSV
ay list invoices -f csv -m > invoices.csv
```

Paginierung

Bei mehr als --limit Treffern zeigt die CLI:

```
Showing 20 of 1347 matching docs (page 1 of 68)
Next page: ay list consumers --skip 20
```

Performance-Tipp

```
ay list <collection> --no-sort           # schnellster Path, überspringt Sortier-Stage
ay list <collection> --skip-count        # kein Total-Count (schneller bei großen
```

`Collections)`

Fehlerfälle

- HTTP 403 — Keine Rechte auf Collection. Via ay access prüfen.
- HTTP 404 — Collection existiert nicht. ay modules für korrekten Namen.
- Filter-Syntax-Error — JSON muss valide sein; bei Shell-Escaping-Problemen via Heredoc oder File.

Siehe auch

- get (wiki:cli/crud-core/get) — einzelner Datensatz
- search (wiki:cli/queries/search) — Volltextsuche

---

ay get

ay get (Alias: g)

Liest Datensätze aus einer Collection mit Feld-Selektion und Paginierung. Im Gegensatz zu list ist get für Field-Projection und tabellarische Ausgabe optimiert.

Syntax

```
ay get <collectionOrModule> [collection] [flags]
```

Erstes Argument ist entweder die Collection (z.B. consumers) oder das Modul (z.B. crm); im zweiten Fall folgt die Collection als zweites Argument.

Flags

Flag	Default	Zweck
-p, --page <n>	1	Seite
-l, --limit <n>	20	Einträge pro Seite
-f, --from <date>	—	Ab-Datum (ISO 8601 oder 2026-04-01)
-i, --fields <fields...>	alle	Whitelist von Feldern (Projection)

Beispiele

```
# Standard-Felder einer Collection
ay get contacts

# Explizites Modul
ay get crm consumers

# Nur ausgewählte Felder, Seite 3
ay get products -i name price stock -p 3 -l 10

# Als CSV speichern
ay get orders -i _id total status createdAt -r csv --save
```

Siehe auch

- list (wiki:cli/crud-core/list) — Listen-Modus mit Filter

- search (wiki:cli/queries/search) — Volltextsuche
- describe (wiki:cli/crud-core/describe) — Schema eines Eintrags

---

ay edit

ay edit (Alias: e)

Öffnet einen einzelnen Eintrag im interaktiven CLI-Editor (Tabellen- oder Raw-JSON-Modus). Speichert beim Beenden via PUT. Gegenstück zu update (wiki:cli/crud-core/update), das non-interaktiv arbeitet.

Syntax

```
ay edit [collection] [id]
```

Beide Argumente fallen ohne Eingabe auf den zuletzt benutzten Wert (lastCollection, lastId) zurück.

Verhalten

edit ruft erst GET /<collection>/<id> auf und entscheidet dann:

- Hat die Antwort content.columns (Tabellen-Schema) !’ Tabellen-Editor mit Inline-Editing pro Feld.
- Sonst !’ Raw-JSON-Editor im Default-$EDITOR.

Beispiele

```
# Aktiven Eintrag editieren (lastCollection + lastId)
ay edit

# Konkreter Datensatz
ay edit contacts 64a1b2c3d4e5f60012345678

# Alias
ay e tasks 64a1b2c3
```

Siehe auch

- update (wiki:cli/crud-core/update) — non-interaktiv für Scripts
- get (wiki:cli/crud-core/get) — nur lesen
- audit (wiki:cli/crud-core/audit) — Änderungs-History

---

ay copy

ay copy (Alias: cp)

Dupliziert einen Datensatz innerhalb einer Collection. Server-seitig: GET !’ neue _id !’ POST. Bei Sub-Documents werden nested IDs neu generiert.

Syntax

```
ay copy [collection] [id]
```

Beide Argumente defaulten auf lastCollection / lastId.

Beispiele

```
# Konkreter Datensatz
ay copy contacts 64a1b2c3d4e5f60012345678

# Letzten Eintrag duplizieren
ay cp

# Email-Template als Vorlage kopieren
ay copy emailtemplates 6792...
```

Verhalten

- Der Server vergibt eine neue _id.
- copiedFrom wird auf die ursprüngliche _id gesetzt (sofern das Schema das Feld unterstützt).
- Audit-Log wird automatisch geschrieben.
- _customerID wird beibehalten — Cross-Tenant-Copy nur via Marketplace-Install-Worker (wiki:cli/concepts/marketplace).

Siehe auch

- create (wiki:cli/crud-core/create) — Neuanlage von Grund auf
- update (wiki:cli/crud-core/update) — Felder am Duplikat anpassen

---

ay describe

ay describe (Alias: d)

Zeigt die YAML-View eines Eintrags inkl. aller Felder. Gut für Schema-Inspektion und Sub-Resource-Extraction (Comments, Attachments, Worklogs).

Syntax

```
ay describe [collection] [id] [subResource]
```

Argumente

Argument	Default	Zweck
collection	lastCollection	Collection-Name
id	lastId	ObjectId
subResource	—	Optionales Feld, z.B. comments, attachments, worklogs

Beispiele

```
# Komplette YAML-Ausgabe
ay describe contacts 64a1b2c3d4e5

# Sub-Resource extrahieren
ay describe tasks 64a1b2c3d4e5 comments

# Sub-Resource als JSON
ay describe tasks 64a1b2c3d4e5 comments -r json
```

```
# JMESPath-Filter
ay describe tasks 64a1b2c3d4e5 --jq "subject"

# Alias auf den letzten Eintrag
ay d
```

Siehe auch

- get (wiki:cli/crud-core/get) — Tabellen-View
- audit (wiki:cli/crud-core/audit) — Änderungs-Log

---

ay create

ay create (Alias: c)

Legt einen neuen Datensatz mit einem Namen an. Für Felder über name hinaus !’ siehe update (wiki:cli/crud-core/update) direkt im Anschluss.

Syntax

```
ay create [collection] [name]
```

Wenn beide Argumente fehlen und das TTY interaktiv ist, fragt die CLI nach Collection und Name.

Beispiele

```
# Mit Namen direkt
ay create contacts "John Doe"

# Mit Alias
ay c products "Widget"

# Workflow: anlegen + sofort befüllen
ay create projects "Website-Relaunch"
ay update projects <id> --set status=planning --set budget=15000
```

Verhalten

- Server-seitig: POST /<collection> mit Body { name: "<name>" }.
- Default-Felder (_customerID, Timestamps, Access-Control) werden vom Backend gefüllt.
- Audit-Log wird geschrieben.
- Bei fehlendem Recht (<module>.<plural>.create) !’ HTTP 403, Exit-Code 5.

Siehe auch

- update (wiki:cli/crud-core/update) — Folgefelder befüllen
- copy (wiki:cli/crud-core/copy) — Duplikat statt Neuanlage
- upload (wiki:cli/crud-core/upload) — Files in Documents

---

ay update

ay update (Alias: u)

Aktualisiert einen Eintrag non-interaktiv — gemacht für Scripts und AI-Agenten. Im Gegensatz zu edit öffnet update keinen Editor; die Änderungen kommen via --set, --body, --body-file oder --body-stdin.

Syntax

```
ay update [collection] [id] [flags]
```

Flags

Flag	Default	Zweck
--set <key=value...>	—	Einzelne Felder (wiederholbar)
--body <json>	—	Komplettes Update als JSON-String
--body-file <path>	—	JSON-Datei einlesen
--body-stdin	—	JSON via stdin pipen
--merge	false	GET ’ merge ’ PUT (existing fields bleiben erhalten)

--set-Werte werden auto-gecastet: true/false ’ boolean, null ’ null, Zahlen ’ number, Rest bleibt string.

Beispiele

```
# Einzelne Felder
ay update contacts 64a1b2c3 --set firstName=Jane --set active=true

# AWS-State eines Computing Entity
ay update computingentities 64a1b2c3 --set ip=35.159.50.19 --set instanceId=i-019c...

# JSON-Body
ay update contacts 64a1b2c3 --body '{"firstName":"Jane","tags":["vip"]}'

# Aus Datei
ay update products 64a1b2c3 --body-file changes.json

# Aus stdin
echo '{"status":"active"}' | ay update contacts 64a1b2c3 --body-stdin

# Merge-Mode (GET + spread + PUT)
ay update tasks 64a1b2c3 --set done=true --merge
```

Siehe auch

- edit (wiki:cli/crud-core/edit) — interaktiver Editor
- batch (wiki:cli/utilities/batch) — Bulk-Updates
- exec (wiki:cli/utilities/exec) — Custom-Actions

---

ay delete

ay delete (Alias: rm)

Löscht einen oder mehrere Einträge per ID. Bei interaktivem TTY ist eine Bestätigung erforderlich; mit --force wird sie übersprungen.

Syntax

```
ay delete [collection] [ids] [flags]
```

ids kann eine einzelne ObjectId oder eine kommaseparierte Liste sein.

Flags

| Flag | Default | Zweck | |---|---|---| | --ids-stdin | false | IDs aus stdin lesen (zeilen- oder kommagetrennt) | | --force | false | Confirmation-Prompt überspringen |

Beispiele

```
# Einzelner Datensatz
ay delete contacts 64a1b2c3d4e5

# Force ohne Prompt
ay rm contacts 64a1b2c3d4e5 --force

# Mehrere IDs
ay delete contacts id1,id2,id3 --force

# Aus Pipeline (z.B. von ay search)
ay search tasks "status=archived" -i _id -m | ay delete tasks --ids-stdin --force
```

Verhalten

- Bei mehreren IDs wird sequenziell gelöscht; Erfolg/Fehler-Counter am Ende.
- Bei mindestens einem Fehler !’ Exit-Code 1.
- --force ist Pflicht in non-TTY-Kontexten (CI, Pipes).

Siehe auch

- batch delete (wiki:cli/utilities/batch) — Bulk-Variante mit detailliertem Reporting
- search (wiki:cli/queries/search) — IDs zum Pipen finden

---

ay storage

ay storage (Alias: s)

Zeigt und verwaltet den lokalen CLI-Speicher: AES-256-CBC-verschlüsselte Tokens (token, refreshToken) und Klartext-Preferences (lastModule, lastCollection, activeCustomerName, …).

Sensitive Werte werden in der Anzeige automatisch redacted (z.B. eyJh••••5678). Persistierung in ~/.aYOUne/storage/.

Syntax

```
ay storage                    # Show
ay storage set <key> <value> # Set token | refreshToken
ay storage clear <key>       # Remove token | refreshToken
```

Beispiele

```
# Aktuellen Speicher inspizieren (redacted)
ay storage

# Token aus anderem Login persistieren
ay storage set token eyJhbGciOi...

# Token löschen (= sanftes Logout)
ay storage clear token

# Refresh-Token resetten
ay storage clear refreshToken
```

Verhalten

- set akzeptiert nur die Keys token und refreshToken.
- Werte müssen valide JWT-Form haben (<header>.<payload>.<signature>), sonst wird der Set abgelehnt.
- Nicht-sensitive Keys (z.B. lastCollection) werden direkt gelesen, nicht entschlüsselt.

Speicher-Order

--token <jwt> Flag (höchste Priorität)

secureStorage.token (dieser Befehl schreibt hierhin)

AYOUNE_TOKEN Environment-Variable (niedrigste Priorität)

Siehe auch

- login (wiki:cli/cmd-login) — Browser-Login
- logout (wiki:cli/auth-config/logout) — Logout-Equivalent
- whoami (wiki:cli/auth-config/whoami) — Aktiven User anzeigen

---

ay audit

ay audit (Alias: history)

Zeigt den Änderungs-Verlauf eines Eintrags. Listet alle Audit-Records, ein interaktiver Picker öffnet den Detail-View des ausgewählten Records als YAML-Diff.

Syntax

```
ay audit [collection] [id]
```

Beide Argumente defaulten auf lastCollection / lastId.

Beispiele

```
# History eines Consumers
ay audit consumers 64a1b2c3d4e5

# History des letzten benutzten Eintrags
```

```
ay history

# History einer Task
ay audit tasks 64a1b2c3
```

Verhalten

CLI lädt die Audit-Liste (GET /<collection>/<id>/audit).

Interaktiver Picker zeigt Date + User + Action.

Bei Auswahl: Detail-Call (GET /<collection>/<id>/audit/<auditId>) !' YAML-Dump in Terminal.

Audit-Records enthalten:

- _userID — wer
- createdAt — wann
- action — create / update / delete
- before / after — komplette Document-Snapshots

Hinweis

Der Befehl ist interaktiv-only — in Pipes oder CI funktioniert die Picker-Phase nicht. Für scriptbare Audit-Daten: ay list audits --filter '{"_entityID":"<id>"}'.

Siehe auch

- describe (wiki:cli/crud-core/describe) — aktuellen YAML-State
- edit (wiki:cli/crud-core/edit) — Eintrag anpassen

---

ay modules

ay modules (Alias: m)

Interaktiver Browser für Module !' Collections !' Operationen. Auch nutzbar als Direkt-Mode mit positional args (analog zu kubectl get).

Syntax

```
ay modules [module] [collection] [operation] [subject] [flags]
```

Operationen

list · get · create · delete (alle anderen !' siehe dedizierte Commands).

Flags

Flag	Default	Zweck
-p, --page <n>	1	Seite (für list)
-l, --limit <n>	20	Page-Size
-f, --from <date>	—	Ab-Datum
-a, --all	false	Alle Pages fetchen
-q, --search <term>	—	Suchbegriff

Beispiele

```
# Vollinteraktiver Browser
ay modules

# Direkt ins CRM-Modul
ay m crm

# CRM Consumers Collection
ay m crm consumers

# Direkt-Mode: Operation + Subject
ay m pm projects list
ay m pm projects create "Migration"
ay m pm projects get 64a1b2c3
ay m pm projects delete 64a1b2c3
```

Verhalten

- Im Interactive-Mode werden nur die für den User accessible Module + Collections angezeigt.
- Superuser sehen zusätzlich das su-Modul (57 System-Collections).
- Module-Namen sind toleriert: crm, CRM, Customer Relationship Management matchen alle.

Siehe auch

- access (wiki:cli/auth-config/access) — Was darf der User?
- list (wiki:cli/crud-core/list) — Direct-CRUD ohne Drilldown

---

ay upload

ay upload

Lädt eine lokale Datei in die Documents-Collection des aktiven Customers hoch. Pflicht ist eine Target-Entity, an die das Document gehängt wird.

Syntax

```
ay upload <file> --target <id> [flags]
```

Flags

Flag	Default	Zweck
-s, --state <right>	cms.downloads.edit	Upload-Kontext / Access-Right
-t, --target <id>	— (Pflicht)	ObjectId der Ziel-Entity

Beispiele

```
# Software-Installer hochladen
ay upload ./installer.exe \
  --state cms.downloads.edit \
  --target 698d574...

# PDF einem Consumer-Datensatz anhängen
ay upload ./report.pdf \
```

```
    --state crm.documents.edit \
    --target 507f1f7...

  # Receipt an Invoice
  ay upload ./receipt.pdf \
    --state accounting.invoices.edit \
    --target 64a1b2c3...
```

Verhalten

- Multipart-POST gegen config-api.ayoune.app/documents/upload.
- Server vergibt _id, schreibt File-Metadata + Storage-Pfad.
- Antwort enthält documentId, filename, size — bei --format json kompletter Output.

Fehlerfälle

- HTTP 403 — --state matched kein Right des Users. Via ay access prüfen.
- HTTP 413 — File > Customer-Quota. Customer-Limit hochsetzen.
- HTTP 422 — Target-Entity existiert nicht oder gehört anderem Customer.

Siehe auch

- list documents (wiki:cli/crud-core/list) — ay list documents
- credentials (wiki:cli/folder-based/credentials) — für Cloud-Storage-Keys

---

Queries & Analysis

Queries & Analysis

5 Commands für Volltextsuche, MongoDB-Aggregationen, Exports und Bulk-Operationen.

Commands

| Command | Alias | Zweck | |---|---|---| | search (wiki:cli/queries/search) | find | Volltextsuche + Model-Search | | aggregate (wiki:cli/queries/aggregate) | agg | MongoDB-Pipelines (7 Subcommands) | | export (wiki:cli/queries/export) | exp | Datenexport mit Format-Wahl | | batch (wiki:cli/queries/batch) | — | Bulk-Operationen | | access (wiki:cli/queries/access) | — | Zugriffsrechte überblicken |

Hero: ay aggregate

ay aggregate ist der mächtigste Query-Command. Er öffnet einen Wizard oder führt vordefinierte Pipelines aus:

```
  ay aggregate wizard                # Interaktiv
  ay aggregate run <saved-pipeline>  # Gespeicherte Pipeline
  ay aggregate exec --model Orders \
    --pipeline '[{"$match":{"status":"paid"}},{"$group":{"_id":"$country","total":
{"$sum":"$amount"}}}]'
```

Subcommands: run, exec, list, save, validate, models, wizard.

Export-Beispiele

```
ay export consumers --format csv --filter '{"active":true}' -o consumers.csv
ay export orders --format json --date-range 2026-04-01..2026-04-30
```

Siehe auch

- Konzept: Output-Formate (wiki:cli/concepts/output-formats)
- list (wiki:cli/crud-core/list)

---

ay search

ay search (Alias: find)

Volltextsuche und Field-Filter über aYOUne Search-Service. Vier Modi:

Single-Model — ay search consumers "John" (Default).

Global SSE — ay search -g "John" streamt Treffer aus allen Collections.

FindOne — --one gibt nur den ersten Treffer zurück.

Legacy — --legacy umgeht den Search-Service und nutzt das Module-API direkt (für Filter, die der Search-Index nicht unterstützt).

Syntax

```
ay search [collectionOrModule] [collectionOrQuery] [query] [flags]
```

Flags

| Flag | Default | Zweck | |---|---|---| | -g, --global <query> | — | SSE-Streaming über alle Collections | | --field <name> | — | Suche nur in einem Feld | | --one | false | Erster Treffer reicht | | --legacy | false | Module-API statt Search-Service | | --filter <key=val,…> | — | Field-Filter (=, !=, >, <, >=, <=) | | --fields <list> | — | Projection (kommagetrennt) | | --sort <field> | -createdAt | Sortierung (- für desc) | | --count | false | Nur Treffer-Anzahl ausgeben | | -l, --limit <n> | 25 | Page-Size | | -p, --page <n> | 1 | Seite |

Beispiele

```
# Volltext im Search-Index
ay search consumers "John"
ay search crm consumers "John"

# Feld-spezifisch
ay search consumers "John" --field firstName

# Filter mit Operatoren
ay search orders --filter "status=paid,total>500"
ay search invoices --filter "createdAt>2026-04-01" --sort -total

# Erster Treffer (FindOne)
ay search consumers "Doe" --one --fields _id,firstName,lastName

# Globale SSE-Suche
ay search -g "tolinax" -l 50
```

```
# Nur zählen
ay find tasks --filter "status=open" --count
```

Sanitizing

--fields wird gegen ^-?[a-zA-Z][a-zA-Z0-9.$]*$ validiert — MongoDB-Operatoren ($where, $gt, …) werden abgelehnt.

Siehe auch

- list (wiki:cli/crud-core/list) — primitive Listenanzeige
- aggregate (wiki:cli/queries/aggregate) — komplexe Pipelines
- export (wiki:cli/queries/export) — größere Datenmengen exportieren

---

ay aggregate

ay aggregate (Alias: agg)

MongoDB-Aggregation-Pipelines gegen beliebige Collections, mit 7 Subcommands für Wizard, Ausführung und Wiederverwendung.

Subcommands

Subcommand	Zweck
wizard	Interaktiver Wizard: Collection, Stages, Output
exec	Ad-hoc-Pipeline ausführen
run <name>	Gespeicherte Pipeline ausführen
list	Alle gespeicherten Pipelines
save <name>	Pipeline speichern
validate	Pipeline-Syntax prüfen
models	Collections listen, die Aggregation erlauben

Ad-hoc-Ausführung

```
ay aggregate exec \
  --model Orders \
  --pipeline '[
    {"$match":{"status":"paid"}},
    {"$group":{"_id":"$country","total":{"$sum":"$amount"}}},
    {"$sort":{"total":-1}},
    {"$limit":10}
  ]'
```

Wizard

```
ay aggregate wizard
```

Fragt Schritt-für-Schritt:

Collection (aus verfügbaren Modulen)

Stages (Match, Group, Lookup, Project, Sort, Limit …)

Output-Format

Am Ende zeigt der Wizard die fertige Pipeline-JSON + den Output und bietet an, beides als Saved-Pipeline zu speichern.

Gespeicherte Pipelines

```
ay aggregate list                          # Alle gespeicherten
ay aggregate run revenue-by-country        # Ausführen
ay aggregate save revenue-by-country < pipeline.json
```

Saved-Pipelines liegen in der Aggregations-Collection und sind customer-scoped.

Flags

| Flag | Zweck | |---|---| | --model <Collection> | Ziel-Collection (PascalCase, Plural) | | --pipeline '<json>' | Pipeline-Array als JSON | | --file pipeline.json | Pipeline aus Datei | | --explain | Pipeline-Execution-Plan statt Ergebnis | | --allowDisk | allowDiskUse: true für große Pipelines | | --format | Output-Format |

Performance-Tipp

- --explain vor großen Pipelines — zeigt, welche Stages Indizes nutzen können.
- $match möglichst früh, um Volumen zu reduzieren.
- $project vor $lookup, um Inputs schlank zu halten.

Siehe auch

- list (wiki:cli/crud-core/list) — einfache Queries
- export (wiki:cli/queries/export) — Aggregat-Output exportieren

---

ay export

ay export (Alias: exp)

Daten-Export mit Pagination, Format-Wahl und Filter-Sprache. Subcommands für Run, List, Get und Configs.

Subcommands

ay export run <collection>

Exportiert die Inhalte einer Collection in einem Rutsch (Auto-Pagination).

| Flag | Default | Zweck | |---|---|---| | --format <fmt> | csv | json / csv / yaml | | --fields <list> | alle | Projection (kommagetrennt) | | --filter <key=val,…> | — | Filter wie bei search (wiki:cli/queries/search) | | --sort <field> | -createdAt | Sortierung | | -l, --limit <n> | 0 (= alles) | Limit (0 = alle Pages) |

```
ay export run contacts --format csv --fields "firstName,lastName,email"
ay export run products --format json --filter "status=active"
ay export run invoices --format csv --filter "createdAt>2026-01-01" --save
```

ay export list

Listet bestehende Export-Jobs.

```
ay export list -l 25
```

ay export get <id>

Detail + Download-URL eines Exports.

```
ay export get 64a1b2c3
```

ay export configs

Listet konfigurierte Export-Definitionen (z.B. wiederkehrende Exports an externe Targets).

```
ay export configs
```

ay export logs

Audit-Log aller Export-Runs.

```
ay export logs -l 25
```

Hinweis

Bei --limit 0 (Default) iteriert die CLI mit limit=500 durch alle Pages und mergt die Payloads — geeignet für Collections bis ~50k Einträge. Für größere !’ db pull (wiki:cli/devops/db) oder aggregate (wiki:cli/queries/aggregate) mit $out.

Siehe auch

- search (wiki:cli/queries/search) — Filter-Sprache
- aggregate (wiki:cli/queries/aggregate) — Pipelines
- db (wiki:cli/devops/db) — Cross-DB-Sync

Streaming & Events

Streaming & Events

2 Commands für Live-Updates einer Collection und Platform-Event-Subscription.

Commands

| Command | Alias | Zweck | |---|---|---| | stream (wiki:cli/streaming/stream) | listen | Live-Änderungen einer Collection (Change-Stream) | | events (wiki:cli/streaming/events) | sub | Platform-Event-Bus-Subscription |

Beispiel: Live-Consumer-Änderungen

```
ay stream consumers
# Terminal zeigt jede Insert/Update/Delete in Echtzeit
```

Flags:

- --filter '<json>' — nur Änderungen, die einem MongoDB-Filter matchen
- --ops insert,update — nur bestimmte Operationen

Beispiel: Platform-Events

```
ay events --topic orders.paid
ay events --topic '*.failed' --since 1h
```

Topics sind in der Events-Collection definiert (events Module), Pattern-Match mit * erlaubt.

Siehe auch

- jobs (wiki:cli/misc/jobs) — Queue-Monitoring
- monitor (wiki:cli/devops/monitor) — Log-Streaming

---

ay events

ay events (Alias: sub)

Subscribed via WebSocket auf den aYOUne Customer-Event-Bus und filtert die Events auf der Client-Seite. Output formatiert als JSON, YAML oder Tabelle.

Syntax

```
ay events [flags]
```

Flags

| Flag | Default | Zweck | |---|---|---| | -f, --format <fmt> | json | json / yaml / table | | -c, --category <cat> | ` | Event-Category (z.B. sales, crm) | | -a, --action <act> | | Action (z.B. create, update, delete) | | -l, --label <label> | | Label-Filter | | -V, --value <val> | ` | Value-Filter |

* = match-all. Filter werden client-seitig ausgewertet — Server sendet alle Events des aktiven Customers.

Beispiele

```
# Alle Events
ay events

# CRM-Events als YAML
ay sub -c crm -f yaml

# Sales-Creates in Tabellenform
ay events -c sales -a create -f table

# Nur Errors
ay events -c errors -f json

# Notification-Worker im Blick behalten
ay events -c notifications -f yaml
```

Schema

Jedes Event hat:

```
{
```

```
  category: string,
  action: string,
  label: string,
  value: string,
  evt_data: { …payload }
}
```

Hinweis

Die WebSocket-Connection bleibt offen bis Ctrl+C. Auto-Reconnect ist eingebaut (mit Customer-Room-Rejoin). Für strukturierte Long-Term-Logs siehe monitor logs (wiki:cli/devops/monitor).

Siehe auch

- stream (wiki:cli/streaming/stream) — Live-Änderungen einer Collection
- monitor (wiki:cli/devops/monitor) — Persistente Log-Streams

---

ay stream

ay stream (Alias: listen)

Live-Stream aller Änderungen einer Collection. Nutzt MongoDB Change-Streams unter der Haube.

Syntax

```
ay stream <collection> [flags]
```

Flags

| Flag | Zweck | |---|---| | --filter '<json>' | Nur Änderungen, die einem Filter matchen | | --ops insert,update,delete,replace | Nur bestimmte Operation-Types | | --fields '<json>' | Projection auf geänderte Felder | | --format json\|yaml\|table | Output-Format |

Beispiele

```
# Alle Consumer-Änderungen
ay stream consumers

# Nur Inserts im CRM-Scope
ay stream consumers --ops insert

# Nur bestimmte Felder bei Updates
ay stream orders \
  --ops update \
  --filter '{"status":"paid"}' \
  --fields '{"_id":1,"total":1,"status":1}'
```

Output-Pattern

Jede Änderung kommt als JSON-Event:

```
{
  "op": "update",
  "ns": "consumers",
```

```
  "_id": "64a1b2c3...",
  "ts": "2026-04-22T14:32:11.000Z",
  "updatedFields": {"lastLoginAt": "2026-04-22T14:32:10.000Z"},
  "fullDocument": { … }
}
```

Abbruch

Ctrl-C beendet den Stream sauber.

Leistung

Change-Streams benötigen ein Replica-Set (Standard im aYOUne-Atlas-Cluster). Bei sehr hohem Traffic --filter nutzen, sonst füllt sich das Terminal schnell.

Siehe auch

- events (wiki:cli/streaming/events) — Platform-Event-Bus
- monitor (wiki:cli/devops/monitor) — Log-Streaming

Auth & Config

Auth & Config

8 Commands für Authentifizierung, Customer-Context-Wechsel, Erst-Setup und CLI-Preferences.

Commands

| Command | Alias | Zweck | |---|---|---| | login (wiki:cli/cmd-login) | auth | Browser-Login via auth.ayoune.app | | logout (wiki:cli/auth-config/logout) | signout | Abmelden, Token löschen | | whoami (wiki:cli/auth-config/whoami) | who | Aktueller User + Customer | | switch (wiki:cli/auth-config/switch) | sw | Multi-Tenant-Context-Switcher | | context (wiki:cli/auth-config/context) | ctx | Aktiven Entity-Context verwalten | | setup (wiki:cli/auth-config/setup) | — | First-Run-Wizard | | config (wiki:cli/auth-config/config) | — | CLI-Preferences (~/.config/ayoune/config.json) | | alias (wiki:cli/auth-config/alias) | — | Custom Command-Aliase |

Typische Workflow

```
ay setup                          # Einmalig
ay login                          # Bei Session-Start
ay whoami                         # Bestätigen
ay switch customer tolinax        # In anderen Tenant wechseln
ay list consumers                 # Arbeiten
ay logout                         # Session beenden (optional)
```

Siehe auch

- Konzept: Authentifizierung (wiki:cli/concepts/authentication)
- Konzept: Customer-Context (wiki:cli/concepts/customer-context)

---

ay login

ay login (Alias: auth)

Authentifiziert die CLI gegen die aYOUne-Plattform. Standard ist Browser-basiertes Login via WebSocket-Notifier. Alternativ akzeptiert die CLI ein bestehendes JWT.

Syntax

```
ay login [--token <jwt>]
```

Verhalten — Browser-Login

CLI generiert ein temporäres Token, öffnet https://auth.ayoune.app/login?cliToken=....

Im Browser anmelden (Email/Passwort, WebAuthn, OAuth, …).

Nach Erfolg pusht der Browser das Session-JWT via notifier.ayoune.app zurück.

CLI persistiert es AES-256-CBC-verschlüsselt in ~/.aYOUne/storage/.

Verhalten — JWT-Login

```
ay login --token eyJhbGciOi...
```

Validiert das JWT-Shape, decodiert es und schreibt es in secureStorage. Nutzt nicht den Browser-Flow.

Beispiele

```
# Standard
ay login

# Aliased
ay auth

# Per Token (z.B. CI-Pipeline)
ay login --token "$AYOUNE_TOKEN"
```

Token-Reihenfolge in der CLI

--token <jwt> Flag (CLI-Parameter)

secureStorage.token (durch ay login befüllt)

AYOUNE_TOKEN Environment-Variable

Siehe auch

- logout (wiki:cli/auth-config/logout) — Token löschen
- whoami (wiki:cli/auth-config/whoami) — Aktiven User prüfen
- storage (wiki:cli/crud-core/storage) — Token-Inspector

---

ay logout

ay logout (Alias: signout)

Löscht das gespeicherte Access-Token und Refresh-Token aus dem lokalen Secure-Storage. Beendet damit die CLI-Session, ohne den Server-side Token zu invalidieren.

Syntax

```
ay logout
```

Beispiele

```
# Logout
ay logout

# Alias
ay signout
```

Verhalten

- Entfernt token und refreshToken aus ~/.aYOUne/storage/.
- Andere Storage-Keys (lastModule, lastCollection, activeCustomerName) bleiben unangetastet.
- Server-seitig wird das JWT erst beim Ablauf invalid (kein Server-Logout).

Hinweis

Für vollständigen Server-Logout (z.B. nach Token-Leak): Admin-UI !’ Profil !’ Sessions !’ Revoke. Für nur den Token-Reset: storage clear token (wiki:cli/crud-core/storage).

Siehe auch

- login (wiki:cli/cmd-login) — Neu anmelden
- storage clear token (wiki:cli/crud-core/storage) — Equivalent ohne Refresh-Token-Drop

---

ay whoami

ay whoami (Alias: who)

Zeigt den aktuell authentifizierten User mit aktivem Customer-Banner. Decodiert das gespeicherte JWT und gibt User-ID, Customer, Role, Type, Sprache, Issued/Expires aus.

Syntax

```
ay whoami
```

Beispiele

```
# Standard
ay whoami

# Alias
ay who
```

Output-Beispiel

```
Good evening, Maximilian Hanrieder!
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
Active customer: Tolinax (676d…afff)   (previous: 670a… — `ay switch back` to return)
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
User ID     676d8e91…
Email       max@tolinax.com
Name        Maximilian Hanrieder
Type        superuser
Role        Admin
Issued      2026-04-26 09:14 (3 hours ago)
Expires     2026-04-26 18:14 (in 6 hours)
```

Verhalten

- Liest Token aus secureStorage.
- Decoded JWT-Payload (kein Server-Call).
- Wenn _customerID gesetzt und Company-Name nicht gecached !’ einmaliger GET auth.ayoune.app/changecustomer füllt den Cache.
- Bei type: superuser wird der Rang gelb hervorgehoben.
- Bei impersonation: true warnt ein gelbes Banner.

Fehlerfälle

- Kein Token gespeichert — CLI startet automatisch ay login und ruft sich rekursiv auf.
- Token expired — Wert wird rot dargestellt, Re-Login empfohlen.

Siehe auch

- login (wiki:cli/cmd-login) — Authentifizierung
- switch (wiki:cli/auth-config/switch) — Customer wechseln
- access (wiki:cli/auth-config/access) — Was darf der User?

---

ay alias

ay alias

Verwaltet user-eigene Command-Aliases, persistent in ~/.config/ayoune/aliases.json. Aliases werden bei jedem ay-Aufruf registriert (sofern sie nicht mit eingebauten Commands kollidieren).

Subcommands

ay alias set <name> <command>

Anlegen / Überschreiben eines Alias.

```
ay alias set ll "list leads"
ay alias set tasks-mine "list tasks --filter assignedTo=$USER_ID"
ay alias set me "get consumers $MY_CONSUMER_ID"
```

ay alias list

Listet alle aktiven Aliases.

```
ay alias list
# Ausgabe:
#   ll        -> list leads
#   tasks-mine -> list tasks --filter assignedTo=...
```

ay alias remove <name>

Löscht einen Alias.

```
ay alias remove ll
```

Verhalten

- Aliases sind File-basiert in ~/.config/ayoune/aliases.json.
- Konflikte mit eingebauten Commands !' Alias wird übersprungen, Built-in gewinnt.
- Bei Aufruf werden zusätzliche Argumente an den Alias-Body angehängt:

```
ay alias set show "get consumers"
ay show 64a1b2c3   # = ay get consumers 64a1b2c3
```

Siehe auch

- config (wiki:cli/auth-config/config) — Default-Optionen
- completions (wiki:cli/auth-config/completions) — Shell-Completion

---

ay config

ay config

Verwaltet Default-Optionen der CLI, gespeichert in ~/.config/ayoune/config.json. Defaults werden bei jedem Command-Run vor User-Flags gemerged — explizite Flags übersteuern.

Verfügbare Keys

| Key | Typ | Choices | |---|---|---| | responseFormat | string | json, csv, yaml, table | | verbosity | string | default, extended, minimal | | outPath | string | beliebig (für --save) | | hideMeta | boolean | true / false | | quiet | boolean | true / false | | force | boolean | true / false (Confirmation überspringen) | | dryRun | boolean | true / false |

Subcommands

ay config (interaktiv)

Inquirer-Wizard mit allen Keys.

```
ay config
```

ay config set <key> <value>

```
ay config set responseFormat yaml
ay config set hideMeta true
```

```
ay config set outPath ~/exports/
```

ay config get <key>

```
ay config get responseFormat
# yaml
```

ay config list

Zeigt alle Defaults in tabellarischer Form.

```
ay config list
```

ay config reset

Setzt alle Defaults zurück (Bestätigung erforderlich, mit --force skippable).

```
ay config reset
ay config reset --force
```

Beispiel-Workflow

```
# YAML als Default-Output
ay config set responseFormat yaml

# Force für CI-Scripts
ay config set force true

# Permanenten Pfad für --save
ay config set outPath /var/log/ayoune/
```

Siehe auch

- alias (wiki:cli/auth-config/alias) — Custom-Commands
- storage (wiki:cli/crud-core/storage) — Token-Storage

---

ay completions

ay completions (Alias: completion)

Generiert Shell-Completion-Scripts für Bash, Zsh, Fish und PowerShell. Standardmäßig nur Top-Level-Commands — Sub-Commands müssen via --help aufgelöst werden.

Syntax

```
ay completions <shell>
```

Shells: bash · zsh · fish · powershell (alias pwsh)

Setup

Bash

```
ay completions bash >> ~/.bashrc
source ~/.bashrc
```

Zsh

```
ay completions zsh >> ~/.zshrc
source ~/.zshrc
```

Bei Zsh muss compinit aktiv sein (in den meisten oh-my-zsh-Setups Default).

Fish

```
mkdir -p ~/.config/fish/completions
ay completions fish > ~/.config/fish/completions/ay.fish
```

PowerShell

```
ay completions powershell >> $PROFILE
. $PROFILE
```

Hinweis

Die Completion-Listen sind statisch (Hardcoded im CLI-Source) und werden bei jedem ay-Release aktualisiert. Bei neuen Commands !' CLI-Update + Re-Source des RC-Files.

Siehe auch

- alias (wiki:cli/auth-config/alias) — User-defined Aliases
- config (wiki:cli/auth-config/config) — Default-Optionen

---

ay context

ay context (Alias: ctx)

Setzt aktive Context-Slots (Project, Sprint, Customer, Consumer, …). Folgende Commands injizieren diese als implicit Filter, sodass z.B. ay list tasks automatisch nur Tasks des aktiven Sprints zeigt.

Verfügbare Slots

3 Tiers:

Core — customer, consumer, user

Domain — project, sprint, epic, release, contract, agency

AI / Automation — automation, model

Subcommands

ay context / ay context show

Zeigt aktuell gesetzten Context.

```
ay context
# Active Context
#   project   "Migration"  (projects, 64a1...)
#   sprint    "Sprint 14"  (sprints, 64b2...)
```

ay context set <slot> <name…>

Resolved Name zu ObjectId (mit Hierarchie-Validation und Multi-Match-Hinweis).

```
ay context set project "Website-Relaunch"
ay context set sprint "Sprint 14"
ay context set consumer "John Doe"
```

ay context unset <slot>

Entfernt einen Slot — kaskadiert auf abhängige Slots (z.B. unset project entfernt auch sprint und epic).

```
ay context unset project
```

ay context clear

Cleart alle Slots.

```
ay context clear
```

ay context list

Listet alle verfügbaren Slots gruppiert nach Tier mit aktiver Belegung.

```
ay context list
```

Beispiel-Workflow

```
ay context set project "Migration"
ay context set sprint "Sprint 14"

ay list tasks                 # automatisch gefiltert auf Project + Sprint
ay create tasks "Fix bug"     # automatisch _projectID + _sprintID gesetzt
ay search tasks "blocker"     # automatisch im Scope
```

Siehe auch

- switch (wiki:cli/auth-config/switch) — Customer-Switch
- whoami (wiki:cli/auth-config/whoami) — Aktiver User + Customer

---

ay access

ay access

Zeigt accessible Module + Collections für den aktuell authentifizierten User. Inklusive Bar-Chart, Modul-Drilldown und Search-Modus.

Syntax

```
ay access [flags]
```

Flags

Flag	Default	Zweck
--module <name>	—	Detail-View für ein einzelnes Modul
--search <query>	—	Cross-Module Fuzzy-Search nach Collection-Name oder Right
--no-interactive	—	Disabled das Drilldown-Prompt

Beispiele

```
# Interaktiver Modul-Browser mit Drilldown
ay access

# Detail eines Moduls
ay access --module crm

# Suche nach Collections
ay access --search contacts

# Nur Summary-Tabelle, kein Prompt
ay access --no-interactive

# JSON für Scripts
ay access -r json
```

Output (Summary)

```
Access Summary
User: Maximilian   Type: superuser
21 modules, 566 collections accessible

MODULE        LABEL                          #    SHARE
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
config        Configuration                145    %ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ
marketing     Marketing                     67    %ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%Œ
crm           CRM                           42    %ˆ%ˆ%ˆ%ˆ%Œ
…
&  Superuser — all modules including system collections
```

Drilldown

Im interaktiven Mode !’ Modul wählen !’ Collections-Liste !’ Collection wählen !’ Quick-Commands:

```
List:     ay list crm contacts
Get:      ay get crm contacts
Search:   ay search contacts "<query>"
Describe: ay describe contacts <id>
Export:   ay export run crm contacts
```

Hinweis

access ist offline — basiert auf der lokalen modelsAndRights.ts der CLI. Effective rights des Users werden nicht zur Server-Validierung gegengecheckt; HTTP-403 beim eigentlichen Call ist die Single-Source-of-Truth.

Siehe auch

- modules (wiki:cli/crud-core/modules) — Interaktiver Modul-Browser mit Operations
- permissions (wiki:cli/misc/permissions) — Permission-Requests

ay services

ay services (Alias: svc)

Discovery für aYOUne-Plattform-Services, APIs, Gateways und ihre Endpoints.

Subcommands

ay services list

Listet alle registrierten Services (APIs + externe Services).

```
ay services list
ay services list --type api
ay services list --module crm
```

ay services endpoints <host> (Alias: ep)

Endpoints eines konkreten Hosts — gepullt aus ayouneapiactions.

```
ay services endpoints crm-api.ayoune.app
ay services endpoints ai.ayoune.app -r table
ay services endpoints config-api.ayoune.app -i operationId,method,endpoint
```

ay services health [host]

Health-Check eines Services oder aller Services parallel. Native Fetch + AbortController-Timeout (default 5000ms).

```
# Alle
ay services health

# Konkret
ay services health auth.ayoune.app

# Mit Timeout
ay services health --timeout 2000
```

Status-Werte: healthy (HTTP < 500), unhealthy ("e 500), unreachable (Network-Fail oder Timeout).

ay services describe <module> (Alias: desc)

Komprimiertes Modul-Profile mit APIs, Services, Endpoint-Anzahl je Method, Capabilities.

```
ay services describe crm
# {
#   module: "crm",
#   apis: [{ label: "CRM", host: "crm-api.ayoune.app" }],
#   endpoints: { total: 42, byMethod: { GET: 28, POST: 8, PUT: 6 } },
#   capabilities: ["consumers", "contacts", "leads", ...]
# }
```

Siehe auch

- actions (wiki:cli/utilities/actions) — Action-Discovery
- status (wiki:cli/devops/status) — Aggregierter Health-Check

---

ay errors

ay errors

Dokumentation der CLI-Exit-Codes — direkt im Terminal ablesbar. Hilfreich für Shell-Scripts, CI-Logs und AI-Agenten, die Non-Zero-Exit interpretieren müssen.

Subcommands

ay errors / ay errors list

Listet alle dokumentierten Exit-Codes.

```
ay errors
ay errors list
```

ay errors explain <code>

Detail-View mit Bedeutung und Recovery-Hinweis.

```
ay errors explain 2
ay errors explain 4
```

Exit-Code-Tabelle

| Code | Name | Bedeutung | |---|---|---| | 0 | EXITSUCCESS | Befehl erfolgreich | | 1 | EXITGENERALERROR | Allgemeiner Fehler — --debug für Details | | 2 | EXITMISUSE | Ungültige Verwendung — ay <cmd> --help | | 4 | EXITAUTHREQUIRED | Token fehlt/abgelaufen — ay login | | 5 | EXITPERMISSIONDENIED | Recht fehlt — Admin fragen, access (wiki:cli/auth-config/access) prüfen | | 78 | EXITCONFIGERROR | Config invalid — ~/.config/ayoune/config.json prüfen, doctor (wiki:cli/misc/doctor) |

Beispiele

```
# In CI-Pipeline
if ay export run consumers --format csv; then
  echo "Export ok"
elif [ $? -eq 5 ]; then
  echo "Permission denied — check rights"
elif [ $? -eq 4 ]; then
  ay login
fi

# Erklärung in der Shell
ay errors explain 5
```

Siehe auch

- doctor (wiki:cli/misc/doctor) — Environment-Health
- access (wiki:cli/auth-config/access) — Berechtigungen prüfen

---

ay switch

ay switch (Alias: sw)

Wechselt den aktiven Customer-Context der CLI-Session. Bei Multi-Tenant-Zugehörigkeit unverzichtbar.

Subcommands

| Subcommand | Zweck | |---|---| | (ohne) | Interaktiver Picker (Tabelle mit Slug, Name, Role) | | customer <slug\|id> | Direkter Wechsel | | back | Zurück auf vorherigen Context | | current | Aktiven Context zeigen | | list | Alle verfügbaren Tenants |

Beispiele

```
ay switch                          # !' Terminal-Picker
ay switch customer tolinax         # Direkt in Tolinax-Scope
ay switch customer platform        # Platform-Operator-Scope (Superuser only)
ay switch back                     # Zum Vor-Context zurück
```

Context-Persistenz

Der gewählte Context wird in ~/.aYOUne/storage/activeContext.json abgelegt und gilt für alle nachfolgenden Commands derselben Shell — und für neue Shells desselben Users.

Override für einen einzelnen Call ohne persistenten Wechsel:

```
ay list consumers --customer anderer-tenant
```

Platform-Scope

```
ay switch customer platform
```

Erfordert das restrictApiSuperUser-Recht. Im Platform-Scope siehst du cross-tenant Admin-Collections (Registry, Infra-Config, Diagnostik). Nach Abschluss unbedingt zurück:

```
ay switch back
```

Siehe auch

- Konzept: Customer-Context (wiki:cli/concepts/customer-context)
- whoami (wiki:cli/auth-config/whoami)

Folder-based Commands

Folder-based Commands

3 Command-Gruppen, die als eigene Folders implementiert sind (mehrere Subcommands):

| Gruppe | Subcommands | Zweck | |---|---|---| | local (wiki:cli/folder-based/local) | 7 | Self-Hosting via Docker-Compose | | functions (wiki:cli/folder-based/functions) (fns) | 9 | Custom Functions / FaaS | | provision (wiki:cli/folder-based/provision) (prov) | 7 | Cloud-Provisioning AWS/GCP/Azure/DigitalOcean/Hetzner |

Beispiel: ay local

```
ay local up                  # Docker-Compose up mit allen konfigurierten Profilen
ay local down                # Down
ay local restart <service>   # Einzelnen Service neustarten
ay local logs <service>      # Logs folgen
ay local ps                  # Container-Status
```

```
ay local pull                # Images pullen
ay local exec <service> <cmd># Command im Container
```

Beispiel: ay functions

```
ay functions list                       # Deployed Functions listen
ay functions create my-fn               # Neue Function anlegen
ay functions deploy my-fn               # Deployen
ay functions invoke my-fn --data '{…}'  # Aufrufen
ay functions logs my-fn                 # Logs ansehen
ay functions rollback my-fn v3          # Rollback
```

Beispiel: ay provision

```
ay provision list               # Verfügbare Templates
ay provision aws --region eu-central-1
ay provision gcp --project my-proj
ay provision hetzner --size cx11
```

Siehe auch

- deploy (wiki:cli/devops/deploy) — K8s-Deployment-Steuerung
- self-host-update (wiki:cli/devops/self-host-update) — Updates einspielen

---

ay provision

ay provision (Alias: prov)

Provisioniert eine aYOUne-Instanz auf einem Cloud-Provider. Wraps die Marketplace-Templates aus infrastructure/marketplace/{aws,gcp,azure,digitalocean,hetzner}/. Provider-Tools (aws, gcloud, az, doctl, terraform) müssen lokal installiert sein — die CLI bündelt sie nicht.

State (Parameter, Outputs) wird in ~/.ayoune/provision/<provider>.json persistiert.

Subcommands

ay provision hetzner

Hetzner Cloud via Terraform.

```
ay provision hetzner
ay provision hetzner --dry-run
```

ay provision aws

AWS via CloudFormation.

```
ay provision aws
ay provision aws --dry-run
```

ay provision gcp

Google Kubernetes Engine.

```
ay provision gcp
```

ay provision azure

Azure via ARM-Template.

```
ay provision azure
```

ay provision digitalocean

DigitalOcean App-Platform-Spec.

```
ay provision digitalocean
```

ay provision status [provider]

Zeigt aktuellen State des Deployments.

```
ay provision status hetzner
```

ay provision destroy <provider>

Räumt das provisionierte Environment ab. Bestätigung erforderlich.

```
ay provision destroy hetzner
```

Hinweis

provision schreibt nur die Cloud-Infrastruktur (VMs, K8s-Cluster, Load-Balancer). Die aYOUne-Stack selbst wird im Anschluss via Helm oder Compose deployed — siehe setup (wiki:cli/cmd-setup) und self-host-update (wiki:cli/devops/self-host-update).

Siehe auch

- setup (wiki:cli/cmd-setup) — Self-Hosted Wizard
- local (wiki:cli/folder-based/local) — Lokales Compose-Stack
- self-host-update (wiki:cli/devops/self-host-update) — Updates

---

ay functions

ay functions (Alias: fns)

Lokale Entwicklung, Deployment und Verwaltung von Custom Functions (FaaS). Sandbox via isolated-vm, 5 Trigger-Types.

Subcommands

Subcommand	Zweck
list	Deployed Functions mit Version, Status, letzter Aufruf
get <name>	Function-Details + Code + Config
create <name>	Scaffold lokal + Remote-Stub anlegen
deploy <name>	Code + Config hochladen, neue Version aktivieren
invoke <name>	Ad-hoc-Aufruf mit --data '<json>'
delete <name>	Function entfernen
rollback <name> <version>	Auf frühere Version zurück
versions <name>	Version-History
logs <name>	Letzte N Execution-Logs streamen

Beispiel: Neue Function

```
ay functions create welcome-mail
# 'œ tooling/functions/welcome-mail/{index.ts, config.yaml}

# Code editieren, dann:
ay functions deploy welcome-mail
ay functions invoke welcome-mail --data '{"consumerId":"..."}'
ay functions logs welcome-mail --follow
```

Function-Config

```
# config.yaml
name: welcome-mail
description: Sendet Willkommens-Email an neuen Consumer
trigger:
  type: event           # oder: cron, webhook, manual, scheduled
  topic: consumers.created
runtime: node20
timeout: 10s
memory: 128MB
secrets:
  - SMTP_PASSWORD       # Credentials-Vault-Key
allowedEntities:
  - Consumers
  - Emails
```

Trigger-Types

Type	Auslöser
event	Platform-Event (z.B. orders.paid)
cron	Repeatable-Schedule (Cron-Expression)
webhook	Externer HTTP-Aufruf
manual	Per ay functions invoke
scheduled	Einmaliger Zeitpunkt

Platform-SDK

Inside einer Function steht ay.* zur Verfügung:

```
export default async function({ event, ay }) {
  const consumer = await ay.db('Consumers').findById(event.consumerId);
  await ay.email.send({ to: consumer.email, template: 'welcome' });
}
```

Der SDK ist opt-in-scoped: nur Entities in allowedEntities + mit availableInSDK=true in modelsAndRights.ts sind erreichbar (Stand 2026-04: 26 kuratierte Entries).

Siehe auch

- credentials (wiki:cli/misc/credentials) — Secrets verwalten
- jobs (wiki:cli/misc/jobs) — Function-Queues beobachten

---

ay local

ay local

Cross-Platform-Wrapper um docker compose für lokale aYOUne-Self-Hosting-Stacks. Ersetzt

infrastructure/local/dev.ps1 für macOS/Linux/Windows. Nutzt automatisch das infrastructure/local/-Overlay des Monorepos (wenn vorhanden), sonst ./docker-compose.yml.

Subcommands

ay local up [profiles…]

Startet das Compose-Stack mit gewählten Profilen (core ist immer dabei).

```
ay local up                       # nur core
ay local up crm marketing         # core + crm + marketing
ay local up crm marketing pm devops
```

ay local down

Stoppt alle Services.

```
ay local down
```

ay local restart <service>

Restart eines Services.

```
ay local restart api-config
ay local restart api-crm
```

ay local logs <service>

Folgt den Logs eines Services.

```
ay local logs api-config
ay local logs api-crm -f
```

ay local ps

Zeigt laufende Container mit Status.

```
ay local ps
```

ay local pull

Pullt die neuesten Images.

```
ay local pull
```

ay local exec <service> <command…>

Befehl im Container ausführen.

```
ay local exec api-crm sh
ay local exec api-config node -e "console.log('hi')"
```

Beispiel-Workflow

```
# Erstmaliges Setup
ay setup
ay local pull

# Stack starten
```

```
ay local up crm marketing

# Service-Status prüfen
ay local ps
ay status

# Service-Logs verfolgen
ay local logs api-crm

# Heruntefahren
ay local down
```

Siehe auch

- setup (wiki:cli/cmd-setup) — .env und Compose-File generieren
- self-host-update (wiki:cli/devops/self-host-update) — Image-Updates pullen + restarten
- provision (wiki:cli/folder-based/provision) — Cloud-Deployment

---

ay credentials

ay credentials (Alias: cred)

Verwaltet den verschlüsselten Credentials-Vault des aktiven Customers — API-Keys, OAuth-Tokens, SMTP-Passwörter, Trading-Account-Logins. AES-256-CTR im Backend; Werte im List-Output sind maskiert.

23 unterstützte Provider: Slack, Telegram, Discord, SMTP, OAuth2 (Google/Microsoft/etc.), Twilio, Stripe, PayPal, Trading-Plattformen (MT4/MT5), Custom, …

Subcommands

ay credentials list

Listet alle Credentials (maskiert).

```
ay credentials list
ay credentials list --provider custom
ay credentials list --consumer 64a... --entity-type Contracts
```

ay credentials get <id>

Detail-View eines Credentials.

```
ay credentials get 66abc123def456
```

ay credentials create [flags]

Neues Credential anlegen.

```
# Slack-Bot
ay credentials create \
  --cred-name "Slack Bot" \
  --provider slack \
  --type bot_token \
  --token xoxb-...
```

```
# Trading-Account, an Contract gebunden
ay credentials create \
  --cred-name "Trading EUR" \
  --provider custom \
  --type password \
  --username trader@ex.com \
  --password s3cret \
  --consumer 66a... \
  --entity-type Contracts \
  --entity-id 66b...
```

ay credentials update <id>

Felder aktualisieren.

```
ay credentials update 66abc... --token xoxb-NEW
```

ay credentials delete <id>

Credential löschen (Bestätigung erforderlich).

```
ay credentials delete 66abc...
```

ay credentials decrypt <id> --field <name>

Entschlüsselt einen einzelnen Klartext-Wert (audit-logged).

```
ay credentials decrypt 66abc... --field password
```

ay credentials test <id>

Ruft den Provider auf und validiert die Credentials (z.B. chat.postMessage für Slack).

```
ay credentials test 66abc...
```

ay credentials refresh <id>

OAuth2-Token erneuern (via Refresh-Token).

```
ay credentials refresh 66abc...
```

ay credentials revoke <id>

Provider-seitig revoken + lokal deaktivieren.

```
ay credentials revoke 66abc...
```

ay credentials providers

Listet alle 23 Provider mit erforderlichen Feldern.

```
ay credentials providers
```

ay credentials status

Connected-Platforms-Summary.

```
ay credentials status
```

Siehe auch

- storage (wiki:cli/crud-core/storage) — CLI-Token-Storage
- users (wiki:cli/misc/users) — User + Roles

DevOps & Deployment

DevOps & Deployment

7 Commands für Deployment-Steuerung, Monitoring, Datenbank-Operationen, Release-Management und Projekt-Koordination.

Commands

Command	Alias	Zweck
deploy (wiki:cli/devops/deploy)	dp	K8s-Deployments, Pipelines, Clusters (8 Subcommands)
monitor (wiki:cli/devops/monitor)	mon	Logs, Alerts, Sessions, Uptime
status (wiki:cli/devops/status)	—	Platform-Health-Check
self-host-update (wiki:cli/devops/self-host-update)	shu	Self-hosted Instance updaten
db (wiki:cli/devops/db)	—	Copy/Pull zu externen DBs (4 Subcommands)
release (wiki:cli/devops/release)	rel	Releases, Versions, Channels
pm (wiki:cli/devops/pm)	—	Project Management (Releases, Sprints, Epics, Standup)

Hero: ay deploy

```
ay deploy clusters                  # Cluster-Liste
ay deploy pipelines                 # Aktive Pipelines
ay deploy deployments               # K8s-Deployments
ay deploy pods                      # Running Pods
ay deploy dashboard                 # Interaktives TUI-Dashboard
ay deploy alerts                    # Offene Alerts
```

Beispiel: ay db pull

```
ay db pull consumers --target staging --stream    # Streaming NDJSON
ay db pull orders --target local --since 2026-04-01
```

Unterstützt paginierten Fallback bei großen Collections und --stream für NDJSON-Output.

Beispiel: ay pm standup

```
ay pm standup                       # Formatierter Daily-Standup-Report
ay pm sprints                       # Aktive Sprints
ay pm roadmap                       # High-Level-Roadmap-Timeline
```

Siehe auch

- monitor (wiki:cli/devops/monitor) — Log-Streaming
- release (wiki:cli/devops/release) — Semantic-Release-Workflow

---

ay monitor

ay monitor (Alias: mon)

Monitoring-Befehl für Plattform-Logs, Alerts, aktive Sessions und Uptime-Checks.

Subcommands

ay monitor logs [type]

Listet recent Plattform-Logs aus der jeweiligen Logs-Collection. Default-Type: api.

| Typ | Collection | |---|---| | api | apilogs | | error | errorlogs | | mail | maillogs | | ai | ailogs | | trigger | triggerlogs | | doi | doilogs | | export | exportlogs | | sms | smslogs | | whatsapp | whatsapplogs | | production | productionlogs | | state | statelogs | | googleads | adwordslogs | | computing | computingentitieslogs |

(weitere: setup, shop, score, sensor, stock, soi, reward, merge, download, post, webreceiver, work, consumerapi, accessterminal)

```
ay monitor logs error
ay monitor logs api -l 10
ay monitor logs ai -r table
```

ay monitor alerts list

Listet Alerts.

| Flag | Default | Zweck | |---|---|---| | --severity <lvl> | — | info / warning / critical | | --type <t> | — | podcrash, oomkilled, imagepullerror, … | | --status <s> | active | active / acknowledged / resolved |

```
ay monitor alerts list --severity critical
ay monitor alerts list --type pod_crash
ay monitor alerts list --status acknowledged
```

ay monitor alerts ack <id> / resolve <id>

```
ay monitor alerts ack 64a1b2c3
ay monitor alerts resolve 64a1b2c3
```

ay monitor sessions

Aktive User-Sessions.

```
ay monitor sessions
ay monitor sessions -l 100
```

ay monitor checks

Konfigurierte Uptime/Health-Checks und ihre letzten Ergebnisse.

```
ay monitor checks
```

ay monitor activeusers (Alias: users)

Aktuell aktive User der Plattform.

```
ay monitor activeusers
```

ay monitor pagestats (Alias: pages)

Page-View-Statistiken.

```
ay monitor pagestats -l 50
```

Siehe auch

- status (wiki:cli/devops/status) — Service-Health
- events (wiki:cli/streaming/events) — Live-Events
- jobs (wiki:cli/devops/jobs) — Queue-Monitoring

---

ay jobs

ay jobs (Alias: j)

Verwaltet scheduled/queued Background-Jobs (Agenda + BullMQ), Automations, Triggers und Notifications.

Subcommands

ay jobs list

Listet Jobs.

| Flag | Default | Zweck | |---|---|---| | --status <s> | — | active, waiting, completed, failed, delayed | | -l, --limit <n> | 50 | Page-Size | | -p, --page <n> | 1 | Seite |

```
ay jobs list
ay jobs list --status failed
ay jobs list --status delayed -l 100
```

ay jobs triggers

Automation-Triggers.

```
ay jobs triggers
ay jobs triggers -l 100
```

ay jobs automations (Alias: auto)

Definierte Automations.

```
ay jobs automations
ay jobs automations --active
```

ay jobs execute <automationId> (Alias: run)

Führt eine Automation sofort aus, optional mit Payload.

```
ay jobs execute 64a1b2c3
ay jobs run 64a1b2c3 --body '{"consumerId":"66ab..."}'
```

ay jobs notifications (Alias: notify)

Recent Notifications der CRM-Notifications-Collection.

```
ay jobs notifications
ay jobs notifications -l 50
```

Hinweis

ay jobs list querit die agendajobs-Collection (legacy general-Module). Für BullMQ-spezifische Queue-Operations siehe interne Dashboards via deploy dashboard (wiki:cli/devops/deploy).

Siehe auch

- monitor (wiki:cli/devops/monitor) — Logs + Alerts
- webhooks (wiki:cli/utilities/webhooks) — Outbound-Hooks

---

ay webhooks

ay webhooks (Alias: hooks)

Verwaltet Outbound-Webhooks: Erstellen, Inspizieren, Löschen und Templates listen. Daten landen in Hooks (Module: config).

Subcommands

ay webhooks list

Listet alle registrierten Hooks.

```
ay webhooks list
ay webhooks list -l 100
```

ay webhooks get <id>

Detail-View eines Hooks.

```
ay webhooks get 64a1b2c3
```

ay webhooks create [flags]

Neuen Hook anlegen.

| Flag | Zweck | |---|---| | --body <json> | Hook-Definition als JSON-String | | --body-file <path> | Hook-Definition aus Datei |

```
ay webhooks create --body '{
  "name":"order-paid !’ fulfillment",
  "url":"https://api.fulfillment.com/orders",
  "trigger":{"event":"orders.paid"},
  "secret":"whsec_..."
}'

ay webhooks create --body-file ./hooks/order-paid.json
```

ay webhooks delete <id> (Alias: rm)

```
ay webhooks delete 64a1b2c3
```

ay webhooks templates

Listet vordefinierte Webhook-Templates.

```
ay webhooks templates
```

Hinweis

Für Update + Delivery-Logs (Pre-MVP) !’ bis dahin via ay update hooks <id> und ay list hooklogs --filter '{"_hookID":"<id>"}'.

Siehe auch

- jobs (wiki:cli/devops/jobs) — Job-Queue
- events (wiki:cli/streaming/events) — Event-Bus

---

ay rdp

ay rdp (Alias: remote)

Verbindet zu einem Computing Entity (CE) via RDP. Lädt eine signierte .rdp-Datei, injiziert die Credentials per cmdkey und startet mstsc. Funktioniert primär unter Windows; auf macOS/Linux wird die .rdp-Datei nur generiert und der Pfad ausgegeben.

Syntax

```
ay rdp [id] [flags]
```

Flags

Flag	Default	Zweck
--list	—	Listet alle CEs ohne Connect
--no-connect	—	.rdp-Datei generieren, aber nicht starten

Beispiele

```
# Interaktiver Picker
ay rdp

# Direkt zu einer CE
ay rdp 64a1b2c3d4e5

# Nur Liste
ay rdp --list

# Datei ohne Auto-Connect
ay rdp 64a1b2c3 --no-connect
```

Verhalten

CLI ruft GET config/computingentities/<id>/actions/rdp-connect auf.

Server liefert ip, username, password, instanceId, rdpFile (Templated mit IP).

CLI patcht prompt for credentials:i:1 !’ :i:0, schreibt .rdp ins os.tmpdir().

Auf Windows: cmdkey /add:<ip> /user:<u> /pass:<p> !’ mstsc <file>.

Auf anderen OS: nur Pfad-Ausgabe.

Fehlerfälle

- No public IP — CE wurde provisioniert aber Cloud-Sync läuft noch. Warten oder ay sync clusters triggern.
- HTTP 403 — Recht config.computingentities.actions.rdp-connect fehlt.
- Provider-Down — AWS/GCP-State stopped/terminated !’ CE muss erst gestartet werden.

Siehe auch

- list computingentities (wiki:cli/crud-core/list) — CE-Inventory
- sync (wiki:cli/devops/sync) — Cloud-State-Sync

---

ay sync

ay sync

Synchronisiert Plattform-Daten mit angebundenen Drittsystemen — Bitbucket/GitHub-Repos, K8s-Cluster-Inventory, Pipeline-Status.

Subcommands

ay sync repos

Pullt Repository-Listen von angebundenen Providern.

| Flag | Zweck | |---|---| | --provider <p> | Filter: bitbucket / github | | --id <repoId> | Nur ein Repo syncen |

```
ay sync repos
ay sync repos --provider bitbucket
ay sync repos --id 64a1b2c3
```

ay sync clusters

Synct K8s-Cluster-Status (Pods, Deployments, Resources).

```
ay sync clusters
ay sync clusters --id 64a1b2c3
```

ay sync pipelines

Pullt Pipeline-Runs vom CI/CD-Provider (Bitbucket/GitHub).

```
ay sync pipelines
ay sync pipelines --provider bitbucket -l 100
```

ay sync status

Zeigt aggregierten Sync-Status: Anzahl Repos, Cluster, Pipelines.

```
ay sync status
```

Verhalten

- Repos: bei --id direkt POST /repositories/<id>/sync. Ohne --id iteriert die CLI client-seitig durch alle Repos.
- Clusters: analog.
- Pipelines: Server-seitiger Bulk-Sync via POST /pipelines/sync.

Siehe auch

- deploy (wiki:cli/devops/deploy) — Deployments
- monitor alerts (wiki:cli/devops/monitor) — Alerts auf Sync-Failures
- status (wiki:cli/devops/status) — Service-Health

---

ay status

ay status

Health-Check der aYOUne-Plattform-Services. Pingt parallel alle bekannten Module-Hosts, akzeptiert HTTP < 500 mit JSON-Body als healthy (auch wenn 404, sofern Version-Metadata enthalten).

Syntax

```
ay status [flags]
```

Flags

| Flag | Zweck | |---|---| | --module <name> | Nur ein Modul checken |

Beispiele

```
# Alle Services
ay status

# Nur CRM
ay status --module crm

# JSON für Scripts
ay status -r json

# Tabelle
ay status -r table
```

Output (Pretty-Mode)

```
aYOUne Platform Status

%Ï config-api     v2026.4.1   (78ms)
%Ï auth           v2026.5.0   (102ms)
%Ï crm-api        v2026.3.2   (88ms)
%Ï marketing-api  v2026.2.1   (94ms)
%Ï devops-api     v2026.1.0   (110ms) — HTTP 503

4 healthy   1 unhealthy
```

Geprüfte Services

Core (immer): config-api, auth.

Module (alle bei ay status ohne --module): crm-api · marketing-api · hr-api · ecommerce-api · pm-api · devops-api · accounting-api · automation-api · support-api · reporting-api · monitoring-api.

Exit-Code

- 0 — alle Services healthy
- 1 — mindestens ein Service unhealthy oder unreachable

Damit ist ay status als CI-Gate verwendbar:

```
ay status --module crm && ay deploy crm-api
```

Siehe auch

- services health (wiki:cli/auth-config/services) — alternativer Health-Check
- doctor (wiki:cli/misc/doctor) — Lokale Environment-Checks

---

ay setup

ay setup

Interaktiver Wizard für Self-Hosted aYOUne-Deployments. Generiert .env (Compose) oder values.yaml (Helm) und kann das Stack direkt starten.

Syntax

```
ay setup [-o <dir>]
```

Flags

Flag	Default	Zweck
-o, --output <dir>	.	Output-Verzeichnis für generierte Files

Beispiele

```
# Standard
ay setup

# In ein dediziertes Config-Verzeichnis
ay setup --output ./config

# Eigenes Verzeichnis
ay setup -o /etc/ayoune/
```

Wizard-Schritte

Deployment-Mode — Docker Compose (Single-Server) vs. Kubernetes (Helm).

Domain — z.B. ayoune.example.com.

MongoDB — Connection-String (Default: lokaler Container).

Redis — Host, Port, Passwort.

JWT-Secret — Leer für Auto-Generierung (64 Zeichen Random).

SMTP — Host, Port, User, Passwort.

License-Key — Leer für 14-Tage-Trial.

Modules — Multi-Select aus 13 Modulen (CRM, Marketing, HR, E-Commerce, PM, DevOps, Accounting, Automation, Support, Content, Communication, Reporting, Monitoring).

Generierte Files

Compose-Mode

- .env mit allen ENV-Variablen
- Optional sofortiger Launch via docker compose up -d

Kubernetes-Mode

- .env (für lokales Testing)
- values.yaml für helm install ayoune tolinax/ayoune -f values.yaml

Hinweis

ay setup darf erneut auf einem bestehenden Setup laufen — die Files werden überschrieben. Vorher Backup machen, falls Customizations vorgenommen wurden.

Siehe auch

- local (wiki:cli/folder-based/local) — Compose-Stack starten
- self-host-update (wiki:cli/devops/self-host-update) — Updates
- provision (wiki:cli/folder-based/provision) — Cloud-Provisioning

---

Utilities & Integration

Utilities & Integration

5 Commands für benutzerdefinierte Actions, KI-Services, Endpoint-Discovery und Webhook-Verwaltung.

Commands

| Command | Alias | Zweck | |---|---|---| | actions (wiki:cli/utilities/actions) | act | Custom Actions einer Collection listen | | exec (wiki:cli/utilities/exec) | x | Action per Name ausführen | | ai (wiki:cli/utilities/ai) | — | aYOUne-AI-Services (RAG, Translate, Generate) | | services (wiki:cli/utilities/services) | svc | Platform-Services + Endpoints | | webhooks (wiki:cli/utilities/webhooks) | hooks | Outbound-Webhooks |

Custom Actions

Jede Collection kann neben Standard-CRUD Custom Actions haben (POST /collection/:id/actions/

<slug>).

```
ay actions consumers                                    # Actions der Collection listen
ay exec consumers 64a1b2c3... actions send-welcome      # Action ausführen
```

AI-Services

```
ay ai translate --source de --target en --text "Hallo Welt"
ay ai rag --query "Wie exportiere ich Rechnungen?"
ay ai generate --prompt "Schreibe eine Willkommens-Email"
```

Services-Discovery

```
ay services                                             # Alle Platform-Services
ay services --domain crm                                # Nur CRM-Module
ay services endpoints crm                               # Alle Endpoints des crm-api
```

Webhooks

```
ay webhooks list                      # Aktive Webhooks
ay webhooks create --url https://...  # Neuen Webhook anlegen
ay webhooks test <id>                 # Test-Payload senden
ay webhooks logs <id>                 # Delivery-Logs
```

Siehe auch

- jobs (wiki:cli/misc/jobs) — Async-Queue-Jobs
- permissions (wiki:cli/misc/permissions) — Rechte-Verwaltung

---

ay exec

ay exec (Alias: x)

Führt eine registrierte API-Action aus, identifiziert über operationId. Ruft die apiactions-Registry des Servers ab, um die Action-Definition (Host, Endpoint, Method, Params) aufzulösen, dann der eigentliche HTTP-Call mit dem User-Token.

Syntax

```
ay exec <operationId> [args] [flags]
```

Beispiele

```
# Custom-Action für einen Consumer
ay exec sendWelcomeMail --param consumerId=64a1b2c3

# AI-Generate-Aufruf
ay exec aiGenerate --param prompt="Schreibe eine Willkommens-Email"

# Action ohne Args
ay exec recomputeKpis

# Body via JSON
ay x publishCampaign --body
'{"campaignId":"64a1...","scheduledAt":"2026-05-01T10:00:00Z"}'
```

Auflösungs-Reihenfolge

CLI fragt config-api/apiactions?q=<operationId> (für non-Admin User).

Bei 401/403 !’ Fallback auf su/apiactions (Superuser-Path).

Exact-Match auf operationId, sonst Partial-Match (erster Treffer).

Bei keinem Treffer !’ Exit 1; bei nur 403 in beiden Modulen !’ Exit 5.

Hinweis

Custom-Actions liegen unter /<collection>/:id/actions/<action-name>. Sub-Resource-GETs (z.B. /:id/logs) sind keine Actions — diese laufen via describe (wiki:cli/crud-core/describe) <collection> <id> <subResource>.

Siehe auch

- actions (wiki:cli/utilities/actions) — Action-Discovery
- curl (wiki:cli/misc/curl) — Direkter HTTP-Call

---

ay actions

ay actions (Alias: act)

Listet alle registrierten API-Actions der Plattform — Discovery-Tool für ay exec. Querit apiactions (mit config !’ su Fallback).

Syntax

```
ay actions [search] [flags]
```

Flags

| Flag | Zweck | |---|---| | --namespace <ns> | Filter: nur Actions eines Namespaces (z.B. ai, crm) | | --host <host> | Filter: nur Actions auf einem bestimmten Host | | --method <m> | Filter: GET / POST / PUT / DELETE | | --capability <cap> | Filter: nach Capability (z.B. consumers) | | --list-namespaces | Listet alle Namespaces | | --list-hosts | Listet alle Hosts | | -p, --page <n> | Seite (Default 1) | | -l, --limit <n> | Page-Size (Default 100) |

Beispiele

```
# Komplette Liste
ay actions

# Search-Mode
ay actions generate
ay actions sendMail

# Namespace-Filter
ay actions --namespace ai
ay actions --namespace crm
```

```
# POST-Actions auf einem Host
ay actions --host crm-api.ayoune.app --method POST

# Übersicht der Namespaces
ay actions --list-namespaces

# Tabellen-Ausgabe für AI-Pipelines
ay actions -r table --columns "operationId,method,endpoint,host"
```

Action-Schema

Jede Action enthält:

```
operationId: sendWelcomeMail
method: POST
endpoint: /consumers/:id/actions/send-welcome-mail
host: crm-api.ayoune.app
nameSpace: crm
capability: consumers
params: [{ name: id, in: path, required: true }]
description: Sends a welcome email to the consumer
```

Siehe auch

- exec (wiki:cli/utilities/exec) — Action ausführen
- services (wiki:cli/auth-config/services) — Service-Discovery
- curl (wiki:cli/misc/curl) — Direkter HTTP-Call

---

ay batch

ay batch

Bulk-Operationen über mehrere Datensätze — gemacht für Scripts und AI-Agenten. Kompakter als ein Loop um ay get / ay delete / ay update, mit aggregiertem Reporting.

Subcommands

ay batch get <collection> <ids>

Multi-GET via kommaseparierte IDs.

```
ay batch get contacts id1,id2,id3
echo "id1,id2,id3" | ay batch get contacts --stdin
```

ay batch delete <collection> <ids> (Alias: rm)

Multi-DELETE. --force empfohlen für non-TTY.

```
ay batch delete contacts id1,id2,id3 --force
echo "id1,id2" | ay batch delete contacts --stdin --force
```

ay batch update <collection> <ids> (analog)

Multi-UPDATE mit --set oder --body.

```
ay batch update tasks id1,id2,id3 --set status=archived --force
```

Flags (alle Subcommands)

| Flag | Zweck | |---|---| | --stdin | IDs aus stdin lesen (zeilen- oder kommagetrennt) | | --force | Confirmation überspringen |

Pipeline-Pattern

```
# Alle archivierten Tasks älter als 30 Tage löschen
ay search tasks --filter "status=archived,createdAt<2026-03-26" -i _id -m \
  | ay batch delete tasks --stdin --force

# Alle inaktiven Consumers reaktivieren
ay list consumers --filter '{"active":false}' --fields '{"_id":1}' -m \
  | ay batch update consumers --stdin --set active=true --force
```

Output

Sequenzielle Ausführung mit Status-Spinner:

```
Fetched 18/20 entries (2 errors)
```

Bei mindestens einem Fehler !’ Exit 1.

Siehe auch

- delete (wiki:cli/crud-core/delete) — Single-Delete (auch Bulk via Komma-IDs)
- update (wiki:cli/crud-core/update) — Single-Update
- search (wiki:cli/queries/search) — IDs für Pipes finden

---

ay db

ay db

Kopiert Query-Resultate in externe Datenbanken (Push) oder synct die Plattform-DB auf eine lokale MongoDB (Pull). Alle DB-Operationen laufen über den Aggregation-Service — die CLI verbindet niemals direkt zur Plattform-DB.

Subcommands

ay db copy [target] [query]

Push-Mode: führt eine gespeicherte Aggregation auf der Plattform aus und schreibt in einen pre-konfigurierten DataTarget (Slug-basiert).

| Flag | Zweck | |---|---| | --collection <name> | Target-Collection-Name overriden | | --write-mode <m> | insert, upsert, replace |

```
# Voll-interaktiv
ay db copy
```

```
# Target-only, Query interaktiv
ay db copy my-warehouse-pg

# Voll-autonom
ay db copy my-warehouse-pg active-users

# Mit Override
ay db copy my-warehouse-pg active-users --collection users_2026 --write-mode upsert
```

ay db pull [target]

CLI-side Pull via Streaming-NDJSON + mongoimport zu lokaler MongoDB.

| Flag | Zweck | |---|---| | --since <iso> | Nur Records nach Datum | | --every <duration> | Wiederholendes Pull (z.B. 5m, 1h) | | --max-iterations <n> | Limit für --every | | --write-mode <m> | Default upsert für Local-Targets |

```
ay db pull consumers --target local --stream
ay db pull orders --target local --since 2026-04-01
ay db pull tasks --target local --every 5m --max-iterations 100
```

ay db targets list (Alias: ls)

Listet konfigurierte DataTargets.

```
ay db targets list
```

ay db targets add

Neues Target anlegen.

| Flag | Zweck | |---|---| | --name <name> | Anzeigename | | --type <t> | mongodb, postgresql, rest | | --uri <uri> | Connection-URI | | --database <db> | DB-Name | | --collection <col> | Default-Collection |

```
ay db targets add \
  --name "Local MongoDB" \
  --type mongodb \
  --uri "mongodb://localhost:27017" \
  --database ayoune_dev
```

ay db targets test <slug>

Connectivity-Test eines Targets.

```
ay db targets test my-warehouse-pg
```

ay db targets remove <slug>

```
ay db targets remove my-warehouse-pg
```

Hinweis

Local-Targets (Tag local) nur via ay db pull. Bei ay db copy <local-slug> blockt die CLI mit Hinweis.

Siehe auch

- aggregate (wiki:cli/queries/aggregate) — Pipelines bauen + speichern
- export (wiki:cli/queries/export) — One-Off-Exports

ay flag

ay flag

Liest und togglet Customer-Level Feature-Flags. Backed by config-api/flags mit Right config.flags. Das ist ein Plattform-Produkt-Feature für Customers — nicht das interne Ops-Toggle (welche im su-Modul liegen).

Syntax

```
ay flag list
ay flag <name>
ay flag <name> on | off
ay flag <name> <0-100>
```

Subcommands

ay flag list

Listet alle nicht-archivierten Flags.

```
ay flag list
#   on    notifications.richcontent          [bool]
#   off   marketplace.phase2                 [bool]
#   on    aiCopilot.beta                     [percent] (50)
```

ay flag <name> — State-Show

```
ay flag aiCopilot.beta
```

ay flag <name> on / off

Toggle archived (off = archived: true).

```
ay flag marketplace.phase2 on
ay flag marketplace.phase2 off
```

ay flag <name> <0-100>

Schreibt eine Rollout-Percentage in value.

```
ay flag aiCopilot.beta 25       # 25% Rollout
ay flag aiCopilot.beta 100      # Vollroll-out
ay flag aiCopilot.beta 0        # Effektiv off
```

Mapping

IFlag hat keinen einzelnen active-Boolean. Die CLI mappt:

- on = archived: false
- off = archived: true

Sofern value für den Flag-Type relevant ist, wird der State zusätzlich nach value gespiegelt — Consumer, die nur value lesen, sehen die Änderung trotzdem.

Hinweis

Per-Environment-Targeting (environments[].strategies[]) ist nicht via CLI möglich — dafür Admin-UI nutzen. Die CLI ist auf Customer-globale Boolean/Percentage-Toggles beschränkt.

Siehe auch

- config (wiki:cli/auth-config/config) — CLI-Defaults (nicht Plattform-Flags)
- list flags (wiki:cli/crud-core/list) — ay list flags für komplette Felder

---

ay ai

ay ai

Zugriff auf aYOUne-AI-Services: RAG-Copilot, Translate, Text-Generation.

Subcommands

| Subcommand | Zweck | |---|---| | rag | RAG-Query gegen Knowledge-Base + Platform-Docs | | translate | Text zwischen Sprachen übersetzen (Multi-Provider) | | generate | Freiform-Text-Generation via LLM | | ask | Interaktives Chat-Interface |

Beispiele

```
# RAG-Query
ay ai rag --query "Wie exportiere ich Rechnungen als CSV?"

# Übersetzung
ay ai translate --source de --target en --text "Hallo Welt"

# Freiform-Generation
ay ai generate --prompt "Schreibe eine Willkommens-Email im Du-Tonus"

# Interaktives Chat
ay ai ask
> Hallo, wie kann ich alle Consumers mit fehlgeschlagenen Orders finden?
```

Flags

| Flag | Zweck | |---|---| | --provider openai\|anthropic\|google\|cohere\|ollama | LLM-Provider (Default: customer-preset) | | --model <id> | Spezifisches Modell | | --stream | Token-Streaming (Standard) | | --file <path> | Prompt aus Datei |

Output

Streaming-Text direkt auf stdout. Bei --format json: komplettes Response-Objekt mit usage-Info (Input/Output-Tokens).

Siehe auch

- Platform AI Overview (wiki:developer-portal/ai-services)

- Custom Functions — AI (wiki:cli/folder-based/functions)

Weitere Commands

Weitere Commands

Ergänzende CLI-Commands für Jobs, User, Sync, Permissions, Templates, Credentials und Diagnostik.

| Command | Alias | Zweck | |---|---|---| | jobs (wiki:cli/misc/jobs) | j | BullMQ-Queues, Triggers, Automations, Notifications | | users (wiki:cli/misc/users) | — | User-, Team-, Role-Management | | sync (wiki:cli/misc/sync) | — | Datensynchronisation | | permissions (wiki:cli/misc/permissions) | perms | Rechte + Rollen | | templates (wiki:cli/misc/templates) | tmpl | Email-/Notification-/Report-Templates | | credentials (wiki:cli/misc/credentials) | cred | Verschlüsselter Credentials-Vault | | rdp (wiki:cli/misc/rdp) | remote | Remote Desktop zu Computing Entities | | doctor (wiki:cli/misc/doctor) | — | Lokale Environment-Health-Checks | | completions (wiki:cli/misc/completions) | completion | Shell-Completion-Scripts |

Hero: ay doctor

```
ay doctor
```

Prüft in einem Rutsch:

- Node-Version "e 20
- Aktive Session + gültiges JWT
- Connectivity zu allen konfigurierten Hosts (api, auth, support-api, …)
- Config-File-Syntax
- Token-Refresh-Ring
- Verfügbarer Editor

Exit-Code 0 bei allen Checks grün, sonst nicht-null mit Fehler-Summary.

Hero: ay jobs

```
ay jobs list                          # Aktive Queues + Job-Counts
ay jobs stats                         # Aggregate über alle Queues
ay jobs get <queue> <jobId>           # Einzelner Job mit Payload + Logs
ay jobs retry <queue> <jobId>         # Failed Job erneut versuchen
```

Siehe auch

- monitor (wiki:cli/devops/monitor) — Log-Streaming
- status (wiki:cli/devops/status) — Platform-Health

ay curl

ay curl

Authentifizierter HTTP-Call gegen einen aYOUne-API-Endpoint — wie curl, aber mit dem aktuellen

JWT als Bearer-Token. Fürs Debuggen, ad-hoc Custom-Endpoints und Test-Invocations.

Syntax

```
ay curl <module> <path> [flags]
```

<module> ist der Module-Slug (z.B. crm, config, ai). Daraus wird https://<module>-api.ayoune.app gebaut. Spezial-Module (auth, aggregation, logs) gehen auf ihre dedizierten Hosts.

Flags

| Flag | Default | Zweck | |---|---|---| | -X, --method <verb> | get | get / post / put / delete / patch | | --body <json> | — | Inline-JSON-Body | | --body-file <path> | — | Body aus Datei | | --body-stdin | — | Body aus stdin | | --query <kv...> | — | Query-Param key=value (wiederholbar) | | --raw | false | Response unverändert ausgeben (für NDJSON-Streams) |

Body-Reihenfolge: --body-stdin > --body-file > --body.

Beispiele

```
# Simpler GET
ay curl crm consumers

# Mit Query-Params
ay curl pm tasks --query limit=5 --query status=open

# POST mit Body
ay curl config flags/64a1b2c3 -X put --body '{"archived":true}'

# Body aus stdin (z.B. von jq)
echo '{"prompt":"Hallo Welt"}' | ay curl ai ai/ask -X post --body-stdin

# NDJSON-Stream durchreichen
ay curl monitoring logs/stream --raw

# Custom-Action
ay curl crm "consumers/64a1b2c3/actions/send-welcome" -X post
```

Hinweis

ay curl ist absichtlich dünn — kein Auto-Pagination, keine Format-Conversion. Für strukturierte Outputs !’ die dedizierten Commands (list (wiki:cli/crud-core/list), search (wiki:cli/queries/search), exec (wiki:cli/utilities/exec)).

Siehe auch

- exec (wiki:cli/utilities/exec) — Action via operationId
- actions (wiki:cli/utilities/actions) — Action-Discovery
- services endpoints (wiki:cli/auth-config/services) — Endpoint-Discovery

---

ay open

ay open

Öffnet einen Eintrag im Admin-UI im Default-Browser. Schließt die Lücke nach create / update / search — "ich will den jetzt einfach im UI sehen".

Syntax

```
ay open <collection> <id> [--copy]
```

Flags

| Flag | Zweck | |---|---| | --copy | URL in Zwischenablage statt Browser-Launch |

Beispiele

```
# Im Browser öffnen
ay open tasks 68ab123def456
ay open consumers 507f1f77bcf86cd

# URL in Zwischenablage
ay open invoices 12345 --copy

# Workflow nach Create
NEW_ID=$(ay create tasks "Bug fix" -i _id -m -r json | jq -r ._id)
ay open tasks "$NEW_ID"
```

Environment

| Variable | Default | Zweck | |---|---|---| | ADMIN_URL | https://admin.ayoune.app | Admin-UI-Base — Override für Self-Hosted |

Hinweis

open ruft den OS-Default-Browser-Mechanismus (xdg-open/open/start). In SSH-Sessions ohne Display kommt nur die URL als Output (kein Crash).

Siehe auch

- url (wiki:cli/misc/url) — Nur URL, kein Launch (für Pipes)
- whoami (wiki:cli/auth-config/whoami) — Aktiven User prüfen

---

ay url

ay url

Druckt die Admin-UI-Edit-URL eines Eintrags nach stdout — ohne Browser-Launch und ohne Zusatz-Output. Gebaut für Pipes, xargs, AI-Agenten und $()-Substitution in Shell-Scripts.

Syntax

```
ay url <collection> <id>
```

Beispiele

```
# URL ausgeben
ay url tasks 68ab123def456
# !’ https://admin.ayoune.app/...../tasks/68ab123def456

# In Pipe
ay url tasks 68ab | xargs open
ay url tasks 68ab | wl-copy

# In Shell-Substitution
URL=$(ay url tasks 68ab)
echo "Edit: $URL"

# Bulk
ay search tasks "blocker" -i _id -m \
  | xargs -I{} ay url tasks {}
```

Environment

| Variable | Default | Zweck | |---|---|---| | ADMIN_URL | https://admin.ayoune.app | Admin-UI-Base |

Hinweis

Im Gegensatz zu open (wiki:cli/misc/open) ist url strikt minimal: kein Spinner, kein Banner, kein farbiger Output — nur die URL plus \n.

Siehe auch

- open (wiki:cli/misc/open) — Mit Browser-Launch oder --copy

---

ay users

ay users

Verwaltet User, Teams und Roles. Routes via su-Modul zum Legacy-API.

Subcommands

ay users list (Alias: ls)

| Flag | Zweck | |---|---| | --search <q> | Search nach Name oder Email | | --role <roleId> | Filter auf Role | | --active | Nur aktive User |

```
ay users list
ay users list --search max
ay users list --role 64a1b2c3 --active
```

ay users get <id>

```
ay users get 64a1b2c3
```

ay users invite

Lädt einen neuen User ein (Email mit Setup-Link).

| Flag | Pflicht | Zweck | |---|---|---| | --email <email> | ja | Email des Eingeladenen | | --role <roleId> | nein | Role zuweisen | | --firstName <name> | nein | Vorname | | --lastName <name> | nein | Nachname |

```
ay users invite --email max@example.com --role 64a1... --firstName Max
```

ay users update <id>

```
ay users update 64a1b2c3 --set role=64b2... --set active=true
```

ay users delete <id>

```
ay users delete 64a1b2c3
```

ay users teams list / ay users teams get <id>

```
ay users teams list
ay users teams get 64a1b2c3
```

ay users roles list / ay users roles get <id>

```
ay users roles list
ay users roles get 64a1b2c3
```

Hinweis

Effective rights = user.role.rights "* user.customRights.rights ") customer.packages[].package.modules "* customer.customRights[].right. Packages sind additiv — eine restriktive Package kann Rights nicht entziehen, sondern nur hinzufügen.

Siehe auch

- permissions (wiki:cli/misc/permissions) — Permission-Requests
- access (wiki:cli/auth-config/access) — Was darf der aktuelle User?

---

ay permissions

ay permissions (Alias: perms)

Verwaltet Permission-Requests, Rights und Rollen-Zuweisungen.

Subcommands

ay permissions requests list (Alias: ls)

Listet Permission-Requests.

| Flag | Zweck | |---|---| | --status <s> | pending / approved / rejected |

```
ay permissions requests list
ay perms requests list --status pending
```

ay permissions requests approve <id>

```
ay perms requests approve 64a1b2c3
```

ay permissions requests reject <id> [--reason <text>]

```
ay perms requests reject 64a1b2c3 --reason "Bitte erst Schulung absolvieren"
```

ay permissions rights list

Listet alle bekannten Rights des aktiven Customers (via ayounepackages.modules[] und ayounestates).

```
ay perms rights list
ay perms rights list --module crm
```

ay permissions rights grant <userId> <right>

Fügt einen Right zu user.customRights[] hinzu.

```
ay perms rights grant 64a1... crm.consumers.create
```

ay permissions rights revoke <userId> <right>

```
ay perms rights revoke 64a1... crm.consumers.create
```

Hinweis

ay permissions arbeitet on top of der Rights-Architektur (siehe Migration-Konzept Kapitel 1.70):

- Modul-Rights kommen aus dem ayounepackages-System (additiv).
- Custom-Rights werden direkt am User oder Customer gebunden.
- Effective Right = Union aus beidem ") Package-Restrictions.

Für Rechte, die im 3-Part-Format (module.entity.action) angegeben werden, wird die Annotation in ayounestates mit stateType: "right" gespiegelt.

Siehe auch

- users (wiki:cli/misc/users) — User + Roles
- access (wiki:cli/auth-config/access) — Was darf der aktuelle User?

---

ay templates

ay templates (Alias: tmpl)

Verwaltet Plattform-Templates: Email, Notification, Report und Store. Liegen in den jeweiligen Module-APIs (marketing für Email, config für Notification, reporting für Report, marketplace für Store).

Subcommands

ay templates email

email list

```
ay templates email list
ay templates email list --search "welcome"
```

email get <id>

```
ay templates email get 64a1b2c3
```

ay templates notification (Alias: notify)

notification list

```
ay templates notification list
ay tmpl notify list
```

notification get <id>

```
ay templates notification get 64a1b2c3
```

ay templates report

report list

```
ay templates report list
```

report get <id>

```
ay templates report get 64a1b2c3
```

ay templates store

Marketplace-Bundle-Templates (für Phase-2-Marketplace).

store list

```
ay templates store list
```

Beispiel-Workflow

```
# Welcome-Email-Template finden
ay templates email list --search welcome

# Detail anschauen
ay templates email get 64a1b2c3

# Im Editor öffnen
ay edit emailtemplates 64a1b2c3
```

Siehe auch

- edit (wiki:cli/crud-core/edit) — Interaktiv editieren
- exec (wiki:cli/utilities/exec) — sendTestEmail-Action triggern

---

ay pm

ay pm

Project-Management-Workflows — die aYOUne-Plattform dogfooded sich selbst auf ihrem PM-Modul.

Integriert mit dem context (wiki:cli/auth-config/context)-System: aktive project/sprint/release-Slots werden in Subcommands als implicit Filter genutzt.

Subcommands

ay pm release

release create <version> [--project <name>]

```
ay pm release create 2026.17.0 --project "aYOUne Platform Migration"
```

release publish <id>

```
ay pm release publish 64a1b2c3
```

release list

```
ay pm release list
```

ay pm changelog

changelog add --type <t> --title <text>

Types: feat, fix, chore, refactor, perf, infra, docs.

```
ay pm changelog add --type feat --title "Public roadmap page"
ay pm changelog add --type fix --title "Comm-sidebar Audio-Dedup"
```

changelog publish <id>

```
ay pm changelog publish 64a1b2c3
```

ay pm roadmap

```
ay pm roadmap                                  # Lane-View
ay pm roadmap move <frId> --lane next --order 3
```

ay pm board

Kanban-Board des aktiven Sprints.

```
ay pm board
ay pm board --sprint current
ay pm board --sprint 64a1b2c3
```

ay pm sprint

| Subcommand | Zweck | |---|---| | sprint list | Alle Sprints | | sprint current | Aktiver Sprint | | sprint seal | Closed-State: Velocity, Carry-Over berechnen | | sprint plan | Plan-Wizard für nächsten Sprint |

ay pm epic

```
ay pm epic list
ay pm epic tree                        # Hierarchical Tree
ay pm epic add "Marketplace Phase 2"
```

ay pm standup

Generiert einen formatierten Daily-Standup-Report (Yesterday / Today / Blockers) aus User-Activity der letzten 24h.

```
ay pm standup
```

ay pm seed

Historical-Migration-Seeder — befüllt PM-Collections aus .claude/handoffs/-Daten.

```
ay pm seed migration
ay pm seed migration --dry-run
```

Siehe auch

- context (wiki:cli/auth-config/context) — Project/Sprint-Context setzen
- release (wiki:cli/misc/release) — Plattform-Release (CLI-/Desktop-/etc.)

---

ay release

ay release (Alias: rel)

Release-Pipeline für Customer-Artefakte (Desktop-Client, Installer, Plugin-Pakete). Liest .ayoune-release.yml aus dem Projekt-Root, lädt Artefakte in den aYOUne-Update-Host, registriert Versions in Downloads und schaltet Update-Channels.

Subcommands

ay release (= release run)

Führt die volle Pipeline aus .ayoune-release.yml aus.

```
ay release
ay release --dry-run
```

ay release init

Erstellt ein .ayoune-release.yml-Skeleton für das aktuelle Projekt.

```
ay release init
```

ay release status

Zeigt aktuell veröffentlichte Versions inkl. Download-Counts.

```
ay release status
```

ay release list

Listet alle Downloads-Records mit autoUpdate: true.

```
ay release list
```

ay release publish <artifact>

Lädt eine bestehende Artefakt-Datei + registriert die Version manuell.

```
ay release publish ./installer.exe
ay release publish ./aYOUne-1.4.2.dmg
```

ay release verify

Prüft, ob das Update-Manifest auf dem Server die zuletzt veröffentlichte Version reflektiert.

```
ay release verify
```

ay release rollback <version>

Unpublished eine Version (clients fallen automatisch auf die vorherige zurück).

```
ay release rollback 2026.3.14
```

Hinweis — Desktop-Client

Der Desktop-Client released NICHT via npm run release, sondern via ay release — published auf den aYOUne-Update-Host (nicht GitHub Releases). .ayoune-release.yml regelt Channel (stable/beta/canary), Plattform-Targets und Auto-Update-Manifest.

Siehe auch

- pm release (wiki:cli/misc/pm) — Plattform-internes PM-Release
- self-host-update (wiki:cli/devops/self-host-update) — Self-Hosted Updates

---

ay doctor

ay doctor

Diagnostik-Tool: prüft die lokale CLI-Installation + Environment + Connectivity in einem Rutsch.

Aufruf

```
ay doctor
```

Checks

| Kategorie | Check | |---|---| | Runtime | Node "e 20, npm "e 10 | | Installation | CLI-Version aktuell (latest verfügbar?) | | Config | ~/.config/ayoune/config.json vorhanden + valides JSON | | Session | JWT vorhanden + gültig (expires_at future) | | Token-Refresh | Refresh-Token-Ring OK | | Connectivity | HTTP/HTTPS zu allen konfigurierten Hosts (api, auth, support-api, cm, bc, …) | | Customer-Context | Aktiver Context noch gültig (Customer nicht gelöscht) | | Editor | $EDITOR existiert im PATH | | Completion | Shell-Completion installiert? | | File-Permissions | ~/.aYOUne/storage/* 0600 |

Output-Format

```
'  Node v20.11.1
'  CLI version 2026.17.0 (latest)
'  Config valid
'  Session active (expires in 42m)
'  Token refresh ring healthy
'  api.ayoune.app unreachable (DNS timeout)
```

```
'  auth.ayoune.app OK (103ms)
'  support-api.ayoune.app OK (87ms)
&  Editor $EDITOR not set, falling back to 'nano'
'  Completion installed for bash
'  Storage file permissions OK

2 issues found:
  - api.ayoune.app unreachable
  - $EDITOR not set (non-blocking)

Exit: 1
```

Exit-Codes

| Code | Bedeutung | |---|---| | 0 | Alle Checks grün | | 1 | Mindestens ein Error (blockierend) | | 2 | Nur Warnings |

CI-Integration

```
ay doctor && npm run build       # nur bauen wenn Doctor OK
ay doctor --json > health.json   # CI-Reporting
```

Siehe auch

- status (wiki:cli/devops/status) — Platform-Health-Check (serverseitig)
- Troubleshooting (wiki:cli/troubleshooting)

Authentifizierung

Authentifizierung

Die CLI nutzt denselben Auth-Flow wie alle anderen aYOUne-Clients: OAuth2-PKCE gegen auth.ayoune.app, JWT-Tokens mit 60-Minuten-Lifetime und automatischem Refresh 5 Minuten vor Ablauf.

Flow

1. ay login startet einen lokalen HTTP-Listener auf einem freien Port.
2. Browser öffnet https://auth.ayoune.app/?continue=http://localhost:{port}&code_challenge=....
3. Nach Login redirected der Auth-Service mit ?code=... zurück auf den Listener.
4. CLI tauscht Code gegen { accessToken, refreshToken } via /oauth/token.
5. Tokens werden in ~/.aYOUne/storage/tokens.json abgelegt (OS-Dateirechte 0600).

Token-Lifecycle

| Token | TTL | Refresh | |---|---|---| | accessToken | 60 min | Automatisch 55min nach Ausstellung | | refreshToken | 30 Tage | Nur beim Re-Login erneuert |

Wenn der Refresh fehlschlägt (z.B. Refresh-Token expired), wird der User zum erneuten ay login aufgefordert.

Service-Accounts (Headless)

Für CI/CD + scheduled Scripts: API-Keys statt Browser-Login.

```
ay login --api-key $AYOUNE_API_KEY
```

API-Keys werden pro Customer im Admin-UI angelegt (mobile-app !’ Einstellungen !’ Integrationen !’ API-Keys). Sie haben dieselben Rechte wie der erzeugende User und können widerrufen werden.

Rechte-Scope

Die CLI respektiert die serverseitigen Rechte-Gates (<module>.<entity>.<verb>, z.B. crm.consumers.read). Unzureichende Rechte resultieren in HTTP-403 mit klarer Meldung.

Prüfen, welche Module dem aktiven User zugänglich sind:

```
ay access
```

Siehe auch

- First-Run-Tutorial (wiki:cli/first-run-tutorial)
- Customer-Context (wiki:cli/concepts/customer-context)

---

aYOUne CLI Überblick

aYOUne CLI Überblick

Die aYOUne CLI (ay) ist das offizielle Kommandozeilen-Werkzeug für die aYOUne-Plattform. Sie liefert direkten Zugriff auf alle ~602 Collections, die Deployment-Pipeline, Custom Functions und das lokale Self-Hosting — aus jedem Terminal, ohne Browser.

Wofür

- Datenzugriff: Jede Collection lesen, schreiben, durchsuchen, aggregieren.
- Automatisierung: Exports, Batch-Updates, scheduled Scripts.
- DevOps: Deployments, Logs, Pipeline-Status, Cluster-Health.
- FaaS: Custom-Functions lokal entwickeln, deployen, invoken.
- Self-Hosting: Docker-Compose-Stacks starten/stoppen, Updates einspielen.

Schnellstart

```
npm install -g @tolinax/ayoune-cli
ay setup                    # Interaktiver First-Run-Wizard
ay login                    # Authentifizierung via Browser
ay list consumers           # Erste Abfrage
```

Siehe Installation (wiki:cli/install) und Erster Login (wiki:cli/login) für Details.

Struktur dieser Dokumentation

- Einstieg (wiki:cli/getting-started) — Installation, Setup, First-Run-Tutorial

- Konzepte (wiki:cli/concepts/authentication) — Auth, Customer-Context, Output-Formate
- Befehle (wiki:cli/commands) — Vollständige Referenz aller 45 Commands, gruppiert
- Rezepte (wiki:cli/recipes) — Häufige Workflows
- Problembehebung (wiki:cli/troubleshooting) — Fehler, Debugging, bekannte Probleme

Version

Aktuelle CLI-Version: @tolinax/ayoune-cli@2026.17.0.

::: info Die CLI verwendet CalVer (Calendar Versioning) im Format YYYY.MINOR.PATCH. :::

Befehls-Referenz

Befehls-Referenz

Die CLI bündelt 45 Commands in 8 logische Gruppen. Jede Gruppe hat eine eigene Landing-Page mit Überblick + Details zu den einzelnen Befehlen.

Gruppen

| Gruppe | Anzahl | Zweck | |---|---|---| | CRUD & Core (wiki:cli/crud-core) | 12 | Daten lesen, erstellen, ändern, löschen | | Queries & Analysis (wiki:cli/queries) | 5 | Search, Aggregate, Export, Batch, Access | | Streaming & Events (wiki:cli/streaming) | 2 | Live-Updates, Platform-Events | | Auth & Config (wiki:cli/auth-config) | 8 | Login, Switch, Context, Setup, Config, Alias | | Folder-based (wiki:cli/folder-based) | 3 | local, functions, provision mit Subcommands | | DevOps & Deployment (wiki:cli/devops) | 7 | deploy, monitor, status, db, release, pm, self-host-update | | Utilities (wiki:cli/utilities) | 5 | actions, exec, ai, services, webhooks | | Miscellaneous (wiki:cli/misc) | 3+ | jobs, users, sync, permissions, templates, credentials, rdp, doctor, completions |

Globale Flags

Jeder Command akzeptiert:

| Flag | Zweck | |---|---| | --help / -h | Inline-Hilfe zum Command | | --format / -f | Output-Format: json, yaml, table, csv | | --no-color | Deaktiviert Farbausgabe | | --verbose / -v | Mehr Log-Details | | --quiet / -q | Unterdrückt nicht-Fehler-Output | | --customer <slug> | Override aktivem Customer-Context für diesen Call |

Inline-Hilfe

```
ay --help                    # Alle Commands
ay <cmd> --help              # Command-Details
ay <cmd> <sub> --help        # Subcommand-Details
```

Aliase

Viele Commands haben Kurzformen (z.B. l für list, g für get). Siehe jede Command-Seite für die aktuellen Aliase.

Auth & Config

Auth & Config

8 Commands für Authentifizierung, Customer-Context-Wechsel, Erst-Setup und CLI-Preferences.

Commands

| Command | Alias | Zweck | |---|---|---| | login (wiki:cli/cmd-login) | auth | Browser-Login via auth.ayoune.app | | logout (wiki:cli/auth-config/logout) | signout | Abmelden, Token löschen | | whoami (wiki:cli/auth-config/whoami) | who | Aktueller User + Customer | | switch (wiki:cli/auth-config/switch) | sw | Multi-Tenant-Context-Switcher | | context (wiki:cli/auth-config/context) | ctx | Aktiven Entity-Context verwalten | | setup (wiki:cli/auth-config/setup) | — | First-Run-Wizard | | config (wiki:cli/auth-config/config) | — | CLI-Preferences (~/.config/ayoune/config.json) | | alias (wiki:cli/auth-config/alias) | — | Custom Command-Aliase |

Typische Workflow

```
ay setup                          # Einmalig
ay login                          # Bei Session-Start
ay whoami                         # Bestätigen
ay switch customer tolinax        # In anderen Tenant wechseln
ay list consumers                 # Arbeiten
ay logout                         # Session beenden (optional)
```

Siehe auch

- Konzept: Authentifizierung (wiki:cli/concepts/authentication)
- Konzept: Customer-Context (wiki:cli/concepts/customer-context)

---

ay access

ay access

Zeigt accessible Module + Collections für den aktuell authentifizierten User. Inklusive Bar-Chart, Modul-Drilldown und Search-Modus.

Syntax

```
ay access [flags]
```

Flags

| Flag | Default | Zweck | |---|---|---| | --module <name> | — | Detail-View für ein einzelnes Modul | | --search <query> | — | Cross-Module Fuzzy-Search nach Collection-Name oder Right | | --no-interactive | — | Disabled das Drilldown-Prompt |

Beispiele

```
# Interaktiver Modul-Browser mit Drilldown
```

```
ay access

# Detail eines Moduls
ay access --module crm

# Suche nach Collections
ay access --search contacts

# Nur Summary-Tabelle, kein Prompt
ay access --no-interactive

# JSON für Scripts
ay access -r json
```

Output (Summary)

```
Access Summary
User: Maximilian   Type: superuser
21 modules, 566 collections accessible

MODULE        LABEL                          #    SHARE
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
config        Configuration                145    %ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ
marketing     Marketing                     67    %ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%ˆ%Œ
crm           CRM                           42    %ˆ%ˆ%ˆ%ˆ%Œ
…
&  Superuser — all modules including system collections
```

Drilldown

Im interaktiven Mode !’ Modul wählen !’ Collections-Liste !’ Collection wählen !’ Quick-Commands:

```
List:     ay list crm contacts
Get:      ay get crm contacts
Search:   ay search contacts "<query>"
Describe: ay describe contacts <id>
Export:   ay export run crm contacts
```

Hinweis

access ist offline — basiert auf der lokalen modelsAndRights.ts der CLI. Effective rights des Users werden nicht zur Server-Validierung gegengecheckt; HTTP-403 beim eigentlichen Call ist die Single-Source-of-Truth.

Siehe auch

- modules (wiki:cli/crud-core/modules) — Interaktiver Modul-Browser mit Operations
- permissions (wiki:cli/misc/permissions) — Permission-Requests

---

ay alias

ay alias

Verwaltet user-eigene Command-Aliases, persistent in ~/.config/ayoune/aliases.json. Aliases werden bei jedem ay-Aufruf registriert (sofern sie nicht mit eingebauten Commands kollidieren).

Subcommands

ay alias set <name> <command>

Anlegen / Überschreiben eines Alias.

```
ay alias set ll "list leads"
ay alias set tasks-mine "list tasks --filter assignedTo=$USER_ID"
ay alias set me "get consumers $MY_CONSUMER_ID"
```

ay alias list

Listet alle aktiven Aliases.

```
ay alias list
# Ausgabe:
#   ll         -> list leads
#   tasks-mine -> list tasks --filter assignedTo=...
```

ay alias remove <name>

Löscht einen Alias.

```
ay alias remove ll
```

Verhalten

- Aliases sind File-basiert in ~/.config/ayoune/aliases.json.
- Konflikte mit eingebauten Commands !' Alias wird übersprungen, Built-in gewinnt.
- Bei Aufruf werden zusätzliche Argumente an den Alias-Body angehängt:

```
ay alias set show "get consumers"
ay show 64a1b2c3   # = ay get consumers 64a1b2c3
```

Siehe auch

- config (wiki:cli/auth-config/config) — Default-Optionen
- completions (wiki:cli/auth-config/completions) — Shell-Completion

---

ay completions

ay completions (Alias: completion)

Generiert Shell-Completion-Scripts für Bash, Zsh, Fish und PowerShell. Standardmäßig nur Top-Level-Commands — Sub-Commands müssen via --help aufgelöst werden.

Syntax

```
ay completions <shell>
```

Shells: bash · zsh · fish · powershell (alias pwsh)

Setup

Bash

```
ay completions bash >> ~/.bashrc
source ~/.bashrc
```

Zsh

```
ay completions zsh >> ~/.zshrc
source ~/.zshrc
```

Bei Zsh muss compinit aktiv sein (in den meisten oh-my-zsh-Setups Default).

Fish

```
mkdir -p ~/.config/fish/completions
ay completions fish > ~/.config/fish/completions/ay.fish
```

PowerShell

```
ay completions powershell >> $PROFILE
. $PROFILE
```

Hinweis

Die Completion-Listen sind statisch (Hardcoded im CLI-Source) und werden bei jedem ay-Release aktualisiert. Bei neuen Commands !' CLI-Update + Re-Source des RC-Files.

Siehe auch

- alias (wiki:cli/auth-config/alias) — User-defined Aliases
- config (wiki:cli/auth-config/config) — Default-Optionen

---

ay config

ay config

Verwaltet Default-Optionen der CLI, gespeichert in ~/.config/ayoune/config.json. Defaults werden bei jedem Command-Run vor User-Flags gemerged — explizite Flags übersteuern.

Verfügbare Keys

| Key | Typ | Choices | |---|---|---| | responseFormat | string | json, csv, yaml, table | | verbosity | string | default, extended, minimal | | outPath | string | beliebig (für --save) | | hideMeta | boolean | true / false | | quiet | boolean | true / false | | force | boolean | true / false (Confirmation überspringen) | | dryRun | boolean | true / false |

Subcommands

ay config (interaktiv)

Inquirer-Wizard mit allen Keys.

```
ay config
```

ay config set <key> <value>

```
ay config set responseFormat yaml
ay config set hideMeta true
ay config set outPath ~/exports/
```

ay config get <key>

```
ay config get responseFormat
# yaml
```

ay config list

Zeigt alle Defaults in tabellarischer Form.

```
ay config list
```

ay config reset

Setzt alle Defaults zurück (Bestätigung erforderlich, mit --force skippable).

```
ay config reset
ay config reset --force
```

Beispiel-Workflow

```
# YAML als Default-Output
ay config set responseFormat yaml

# Force für CI-Scripts
ay config set force true

# Permanenten Pfad für --save
ay config set outPath /var/log/ayoune/
```

Siehe auch

- alias (wiki:cli/auth-config/alias) — Custom-Commands
- storage (wiki:cli/crud-core/storage) — Token-Storage

---

ay context

ay context (Alias: ctx)

Setzt aktive Context-Slots (Project, Sprint, Customer, Consumer, …). Folgende Commands injizieren diese als implicit Filter, sodass z.B. ay list tasks automatisch nur Tasks des aktiven Sprints zeigt.

Verfügbare Slots

3 Tiers:

Core — customer, consumer, user

Domain — project, sprint, epic, release, contract, agency

AI / Automation — automation, model

Subcommands

ay context / ay context show

Zeigt aktuell gesetzten Context.

```
ay context
# Active Context
#   project   "Migration"  (projects, 64a1...)
#   sprint    "Sprint 14"  (sprints, 64b2...)
```

ay context set <slot> <name…>

Resolved Name zu ObjectId (mit Hierarchie-Validation und Multi-Match-Hinweis).

```
ay context set project "Website-Relaunch"
ay context set sprint "Sprint 14"
ay context set consumer "John Doe"
```

ay context unset <slot>

Entfernt einen Slot — kaskadiert auf abhängige Slots (z.B. unset project entfernt auch sprint und epic).

```
ay context unset project
```

ay context clear

Cleart alle Slots.

```
ay context clear
```

ay context list

Listet alle verfügbaren Slots gruppiert nach Tier mit aktiver Belegung.

```
ay context list
```

Beispiel-Workflow

```
ay context set project "Migration"
ay context set sprint "Sprint 14"

ay list tasks                 # automatisch gefiltert auf Project + Sprint
ay create tasks "Fix bug"     # automatisch _projectID + _sprintID gesetzt
ay search tasks "blocker"     # automatisch im Scope
```

Siehe auch

- switch (wiki:cli/auth-config/switch) — Customer-Switch
- whoami (wiki:cli/auth-config/whoami) — Aktiver User + Customer

---

ay errors

ay errors

Dokumentation der CLI-Exit-Codes — direkt im Terminal ablesbar. Hilfreich für Shell-Scripts, CI-Logs

und AI-Agenten, die Non-Zero-Exit interpretieren müssen.

Subcommands

ay errors / ay errors list

Listet alle dokumentierten Exit-Codes.

```
ay errors
ay errors list
```

ay errors explain <code>

Detail-View mit Bedeutung und Recovery-Hinweis.

```
ay errors explain 2
ay errors explain 4
```

Exit-Code-Tabelle

| Code | Name | Bedeutung | |---|---|---| | 0 | EXITSUCCESS | Befehl erfolgreich | | 1 | EXITGENERALERROR | Allgemeiner Fehler — --debug für Details | | 2 | EXITMISUSE | Ungültige Verwendung — ay <cmd> --help | | 4 | EXITAUTHREQUIRED | Token fehlt/abgelaufen — ay login | | 5 | EXITPERMISSIONDENIED | Recht fehlt — Admin fragen, access (wiki:cli/auth-config/access) prüfen | | 78 | EXITCONFIGERROR | Config invalid — ~/.config/ayoune/config.json prüfen, doctor (wiki:cli/misc/doctor) |

Beispiele

```
# In CI-Pipeline
if ay export run consumers --format csv; then
  echo "Export ok"
elif [ $? -eq 5 ]; then
  echo "Permission denied — check rights"
elif [ $? -eq 4 ]; then
  ay login
fi

# Erklärung in der Shell
ay errors explain 5
```

Siehe auch

- doctor (wiki:cli/misc/doctor) — Environment-Health
- access (wiki:cli/auth-config/access) — Berechtigungen prüfen

---

ay login

ay login (Alias: auth)

Authentifiziert die CLI gegen die aYOUne-Plattform. Standard ist Browser-basiertes Login via WebSocket-Notifier. Alternativ akzeptiert die CLI ein bestehendes JWT.

Syntax

```
ay login [--token <jwt>]
```

Verhalten — Browser-Login

CLI generiert ein temporäres Token, öffnet https://auth.ayoune.app/login?cliToken=....

Im Browser anmelden (Email/Passwort, WebAuthn, OAuth, …).

Nach Erfolg pusht der Browser das Session-JWT via notifier.ayoune.app zurück.

CLI persistiert es AES-256-CBC-verschlüsselt in ~/.aYOUne/storage/.

Verhalten — JWT-Login

```
ay login --token eyJhbGciOi...
```

Validiert das JWT-Shape, decodiert es und schreibt es in secureStorage. Nutzt nicht den Browser-Flow.

Beispiele

```
# Standard
ay login

# Aliased
ay auth

# Per Token (z.B. CI-Pipeline)
ay login --token "$AYOUNE_TOKEN"
```

Token-Reihenfolge in der CLI

--token <jwt> Flag (CLI-Parameter)

secureStorage.token (durch ay login befüllt)

AYOUNE_TOKEN Environment-Variable

Siehe auch

- logout (wiki:cli/auth-config/logout) — Token löschen
- whoami (wiki:cli/auth-config/whoami) — Aktiven User prüfen
- storage (wiki:cli/crud-core/storage) — Token-Inspector

---

ay logout

ay logout (Alias: signout)

Löscht das gespeicherte Access-Token und Refresh-Token aus dem lokalen Secure-Storage. Beendet damit die CLI-Session, ohne den Server-side Token zu invalidieren.

Syntax

```
ay logout
```

Beispiele

```
# Logout
ay logout

# Alias
ay signout
```

Verhalten

- Entfernt token und refreshToken aus ~/.aYOUne/storage/.
- Andere Storage-Keys (lastModule, lastCollection, activeCustomerName) bleiben unangetastet.
- Server-seitig wird das JWT erst beim Ablauf invalid (kein Server-Logout).

Hinweis

Für vollständigen Server-Logout (z.B. nach Token-Leak): Admin-UI !’ Profil !’ Sessions !’ Revoke. Für nur den Token-Reset: storage clear token (wiki:cli/crud-core/storage).

Siehe auch

- login (wiki:cli/cmd-login) — Neu anmelden
- storage clear token (wiki:cli/crud-core/storage) — Equivalent ohne Refresh-Token-Drop

---

ay services

ay services (Alias: svc)

Discovery für aYOUne-Plattform-Services, APIs, Gateways und ihre Endpoints.

Subcommands

ay services list

Listet alle registrierten Services (APIs + externe Services).

```
ay services list
ay services list --type api
ay services list --module crm
```

ay services endpoints <host> (Alias: ep)

Endpoints eines konkreten Hosts — gepullt aus ayouneapiactions.

```
ay services endpoints crm-api.ayoune.app
ay services endpoints ai.ayoune.app -r table
ay services endpoints config-api.ayoune.app -i operationId,method,endpoint
```

ay services health [host]

Health-Check eines Services oder aller Services parallel. Native Fetch + AbortController-Timeout (default 5000ms).

```
# Alle
ay services health

# Konkret
ay services health auth.ayoune.app

# Mit Timeout
ay services health --timeout 2000
```

Status-Werte: healthy (HTTP < 500), unhealthy ("e 500), unreachable (Network-Fail oder Timeout).

ay services describe <module> (Alias: desc)

Komprimiertes Modul-Profile mit APIs, Services, Endpoint-Anzahl je Method, Capabilities.

```
ay services describe crm
# {
#   module: "crm",
#   apis: [{ label: "CRM", host: "crm-api.ayoune.app" }],
#   endpoints: { total: 42, byMethod: { GET: 28, POST: 8, PUT: 6 } },
#   capabilities: ["consumers", "contacts", "leads", ...]
# }
```

Siehe auch

- actions (wiki:cli/utilities/actions) — Action-Discovery
- status (wiki:cli/devops/status) — Aggregierter Health-Check

---

ay switch

ay switch (Alias: sw)

Wechselt den aktiven Customer-Context der CLI-Session. Bei Multi-Tenant-Zugehörigkeit unverzichtbar.

Subcommands

Subcommand	Zweck
(ohne)	Interaktiver Picker (Tabelle mit Slug, Name, Role)
customer <slug|id>	Direkter Wechsel
back	Zurück auf vorherigen Context
current	Aktiven Context zeigen
list	Alle verfügbaren Tenants

Beispiele

```
ay switch                          # !’ Terminal-Picker
ay switch customer tolinax         # Direkt in Tolinax-Scope
ay switch customer platform        # Platform-Operator-Scope (Superuser only)
ay switch back                     # Zum Vor-Context zurück
```

Context-Persistenz

Der gewählte Context wird in ~/.aYOUne/storage/activeContext.json abgelegt und gilt für alle nachfolgenden Commands derselben Shell — und für neue Shells desselben Users.

Override für einen einzelnen Call ohne persistenten Wechsel:

```
ay list consumers --customer anderer-tenant
```

Platform-Scope

```
ay switch customer platform
```

Erfordert das restrictApiSuperUser-Recht. Im Platform-Scope siehst du cross-tenant Admin-Collections (Registry, Infra-Config, Diagnostik). Nach Abschluss unbedingt zurück:

```
ay switch back
```

Siehe auch

- Konzept: Customer-Context (wiki:cli/concepts/customer-context)
- whoami (wiki:cli/auth-config/whoami)

---

ay whoami

ay whoami (Alias: who)

Zeigt den aktuell authentifizierten User mit aktivem Customer-Banner. Decodiert das gespeicherte JWT und gibt User-ID, Customer, Role, Type, Sprache, Issued/Expires aus.

Syntax

```
ay whoami
```

Beispiele

```
# Standard
ay whoami

# Alias
ay who
```

Output-Beispiel

```
  Good evening, Maximilian Hanrieder!
  % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
  Active customer: Tolinax (676d…afff)   (previous: 670a… — `ay switch back` to return)
  % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
  User ID     676d8e91…
  Email       max@tolinax.com
  Name        Maximilian Hanrieder
  Type        superuser
  Role        Admin
  Issued      2026-04-26 09:14 (3 hours ago)
  Expires     2026-04-26 18:14 (in 6 hours)
```

Verhalten

- Liest Token aus secureStorage.
- Decoded JWT-Payload (kein Server-Call).
- Wenn _customerID gesetzt und Company-Name nicht gecached !' einmaliger GET auth.ayoune.app/changecustomer füllt den Cache.
- Bei type: superuser wird der Rang gelb hervorgehoben.

- Bei impersonation: true warnt ein gelbes Banner.

Fehlerfälle

- Kein Token gespeichert — CLI startet automatisch ay login und ruft sich rekursiv auf.
- Token expired — Wert wird rot dargestellt, Re-Login empfohlen.

Siehe auch

- login (wiki:cli/cmd-login) — Authentifizierung
- switch (wiki:cli/auth-config/switch) — Customer wechseln
- access (wiki:cli/auth-config/access) — Was darf der User?

CRUD & Core

CRUD & Core

Die 12 Basis-Commands für Daten-Zugriff auf alle Collections der Plattform.

Commands

| Command | Alias | Zweck | |---|---|---| | list (wiki:cli/crud-core/list) | l | Liste einer Collection mit Paginierung | | get (wiki:cli/crud-core/get) | g | Einzelner Datensatz via ID | | create (wiki:cli/crud-core/create) | c | Neuen Datensatz anlegen | | edit (wiki:cli/crud-core/edit) | e | Datensatz im Editor öffnen | | copy (wiki:cli/crud-core/copy) | cp | Datensatz duplizieren | | delete (wiki:cli/crud-core/delete) | rm | Datensatz per ID löschen | | update (wiki:cli/crud-core/update) | u | Einzelne Felder aktualisieren | | describe (wiki:cli/crud-core/describe) | d | Schema + YAML-View | | storage (wiki:cli/crud-core/storage) | s | Token- & Prefs-Speicher | | audit (wiki:cli/crud-core/audit) | history | Audit-Trail eines Docs | | modules (wiki:cli/crud-core/modules) | m | Interaktive Modul/Collection-Browser | | upload (wiki:cli/crud-core/upload) | — | File-Upload nach Documents |

Gemeinsames Pattern

Alle CRUD-Commands erwarten als erstes Argument die Collection (lowercased plural aus modelsAndRights.ts, z.B. consumers, products, orders), und — wo sinnvoll — als zweites die ObjectId bzw. ein Name/Slug.

Beispiele

```
ay list consumers                              # Liste
ay list consumers --limit 50 --sort 'createdAt:-1'
ay get products 64a1b2c3d4e5f60012345678       # Einzeldatensatz
ay create projects "Neues Projekt"             # anlegen
ay update tasks <id> --set status=done         # partielles Update
ay delete logs <id>                            # löschen
```

Siehe auch

- Konzept: Output-Formate (wiki:cli/concepts/output-formats)

- Befehl: list (wiki:cli/crud-core/list)

---

ay audit

ay audit (Alias: history)

Zeigt den Änderungs-Verlauf eines Eintrags. Listet alle Audit-Records, ein interaktiver Picker öffnet den Detail-View des ausgewählten Records als YAML-Diff.

Syntax

```
ay audit [collection] [id]
```

Beide Argumente defaulten auf lastCollection / lastId.

Beispiele

```
# History eines Consumers
ay audit consumers 64a1b2c3d4e5

# History des letzten benutzten Eintrags
ay history

# History einer Task
ay audit tasks 64a1b2c3
```

Verhalten

CLI lädt die Audit-Liste (GET /<collection>/<id>/audit).

Interaktiver Picker zeigt Date + User + Action.

Bei Auswahl: Detail-Call (GET /<collection>/<id>/audit/<auditId>) !' YAML-Dump in Terminal.

Audit-Records enthalten:

- _userID — wer
- createdAt — wann
- action — create / update / delete
- before / after — komplette Document-Snapshots

Hinweis

Der Befehl ist interaktiv-only — in Pipes oder CI funktioniert die Picker-Phase nicht. Für scriptbare Audit-Daten: ay list audits --filter '{"_entityID":"<id>"}'.

Siehe auch

- describe (wiki:cli/crud-core/describe) — aktuellen YAML-State
- edit (wiki:cli/crud-core/edit) — Eintrag anpassen

---

ay copy

ay copy (Alias: cp)

Dupliziert einen Datensatz innerhalb einer Collection. Server-seitig: GET !’ neue _id !’ POST. Bei Sub-Documents werden nested IDs neu generiert.

Syntax

```
ay copy [collection] [id]
```

Beide Argumente defaulten auf lastCollection / lastId.

Beispiele

```
# Konkreter Datensatz
ay copy contacts 64a1b2c3d4e5f60012345678

# Letzten Eintrag duplizieren
ay cp

# Email-Template als Vorlage kopieren
ay copy emailtemplates 6792...
```

Verhalten

- Der Server vergibt eine neue _id.
- copiedFrom wird auf die ursprüngliche _id gesetzt (sofern das Schema das Feld unterstützt).
- Audit-Log wird automatisch geschrieben.
- _customerID wird beibehalten — Cross-Tenant-Copy nur via Marketplace-Install-Worker (wiki:cli/concepts/marketplace).

Siehe auch

- create (wiki:cli/crud-core/create) — Neuanlage von Grund auf
- update (wiki:cli/crud-core/update) — Felder am Duplikat anpassen

---

ay create

ay create (Alias: c)

Legt einen neuen Datensatz mit einem Namen an. Für Felder über name hinaus !’ siehe update (wiki:cli/crud-core/update) direkt im Anschluss.

Syntax

```
ay create [collection] [name]
```

Wenn beide Argumente fehlen und das TTY interaktiv ist, fragt die CLI nach Collection und Name.

Beispiele

```
# Mit Namen direkt
```

```
ay create contacts "John Doe"

# Mit Alias
ay c products "Widget"

# Workflow: anlegen + sofort befüllen
ay create projects "Website-Relaunch"
ay update projects <id> --set status=planning --set budget=15000
```

Verhalten

- Server-seitig: POST /<collection> mit Body { name: "<name>" }.
- Default-Felder (_customerID, Timestamps, Access-Control) werden vom Backend gefüllt.
- Audit-Log wird geschrieben.
- Bei fehlendem Recht (<module>.<plural>.create) !' HTTP 403, Exit-Code 5.

Siehe auch

- update (wiki:cli/crud-core/update) — Folgefelder befüllen
- copy (wiki:cli/crud-core/copy) — Duplikat statt Neuanlage
- upload (wiki:cli/crud-core/upload) — Files in Documents

---

ay delete

ay delete (Alias: rm)

Löscht einen oder mehrere Einträge per ID. Bei interaktivem TTY ist eine Bestätigung erforderlich; mit --force wird sie übersprungen.

Syntax

```
ay delete [collection] [ids] [flags]
```

ids kann eine einzelne ObjectId oder eine kommaseparierte Liste sein.

Flags

| Flag | Default | Zweck | |---|---|---| | --ids-stdin | false | IDs aus stdin lesen (zeilen- oder kommagetrennt) | | --force | false | Confirmation-Prompt überspringen |

Beispiele

```
# Einzelner Datensatz
ay delete contacts 64a1b2c3d4e5

# Force ohne Prompt
ay rm contacts 64a1b2c3d4e5 --force

# Mehrere IDs
ay delete contacts id1,id2,id3 --force

# Aus Pipeline (z.B. von ay search)
```

```
ay search tasks "status=archived" -i _id -m | ay delete tasks --ids-stdin --force
```

Verhalten

- Bei mehreren IDs wird sequenziell gelöscht; Erfolg/Fehler-Counter am Ende.
- Bei mindestens einem Fehler !’ Exit-Code 1.
- --force ist Pflicht in non-TTY-Kontexten (CI, Pipes).

Siehe auch

- batch delete (wiki:cli/utilities/batch) — Bulk-Variante mit detailliertem Reporting
- search (wiki:cli/queries/search) — IDs zum Pipen finden

---

ay describe

ay describe (Alias: d)

Zeigt die YAML-View eines Eintrags inkl. aller Felder. Gut für Schema-Inspektion und Sub-Resource-Extraction (Comments, Attachments, Worklogs).

Syntax

```
ay describe [collection] [id] [subResource]
```

Argumente

| Argument | Default | Zweck | |---|---|---| | collection | lastCollection | Collection-Name | | id | lastId | ObjectId | | subResource | — | Optionales Feld, z.B. comments, attachments, worklogs |

Beispiele

```
# Komplette YAML-Ausgabe
ay describe contacts 64a1b2c3d4e5

# Sub-Resource extrahieren
ay describe tasks 64a1b2c3d4e5 comments

# Sub-Resource als JSON
ay describe tasks 64a1b2c3d4e5 comments -r json

# JMESPath-Filter
ay describe tasks 64a1b2c3d4e5 --jq "subject"

# Alias auf den letzten Eintrag
ay d
```

Siehe auch

- get (wiki:cli/crud-core/get) — Tabellen-View
- audit (wiki:cli/crud-core/audit) — Änderungs-Log

---

ay edit

ay edit (Alias: e)

Öffnet einen einzelnen Eintrag im interaktiven CLI-Editor (Tabellen- oder Raw-JSON-Modus). Speichert beim Beenden via PUT. Gegenstück zu update (wiki:cli/crud-core/update), das non-interaktiv arbeitet.

Syntax

```
ay edit [collection] [id]
```

Beide Argumente fallen ohne Eingabe auf den zuletzt benutzten Wert (lastCollection, lastId) zurück.

Verhalten

edit ruft erst GET /<collection>/<id> auf und entscheidet dann:

- Hat die Antwort content.columns (Tabellen-Schema) !' Tabellen-Editor mit Inline-Editing pro Feld.
- Sonst !' Raw-JSON-Editor im Default-$EDITOR.

Beispiele

```
# Aktiven Eintrag editieren (lastCollection + lastId)
ay edit

# Konkreter Datensatz
ay edit contacts 64a1b2c3d4e5f60012345678

# Alias
ay e tasks 64a1b2c3
```

Siehe auch

- update (wiki:cli/crud-core/update) — non-interaktiv für Scripts
- get (wiki:cli/crud-core/get) — nur lesen
- audit (wiki:cli/crud-core/audit) — Änderungs-History

---

ay get

ay get (Alias: g)

Liest Datensätze aus einer Collection mit Feld-Selektion und Paginierung. Im Gegensatz zu list ist get für Field-Projection und tabellarische Ausgabe optimiert.

Syntax

```
ay get <collectionOrModule> [collection] [flags]
```

Erstes Argument ist entweder die Collection (z.B. consumers) oder das Modul (z.B. crm); im zweiten Fall folgt die Collection als zweites Argument.

Flags

Flag	Default	Zweck
-p, --page <n>	1	Seite
-l, --limit <n>	20	Einträge pro Seite
-f, --from <date>	—	Ab-Datum (ISO 8601 oder 2026-04-01)
-i, --fields <fields...>	alle	Whitelist von Feldern (Projection)

Beispiele

```
# Standard-Felder einer Collection
ay get contacts

# Explizites Modul
ay get crm consumers

# Nur ausgewählte Felder, Seite 3
ay get products -i name price stock -p 3 -l 10

# Als CSV speichern
ay get orders -i _id total status createdAt -r csv --save
```

Siehe auch

- list (wiki:cli/crud-core/list) — Listen-Modus mit Filter
- search (wiki:cli/queries/search) — Volltextsuche
- describe (wiki:cli/crud-core/describe) — Schema eines Eintrags

---

ay list

ay list (Alias: l)

Listet Dokumente einer Collection mit Paginierung, Sortierung und optionalen Filtern.

Syntax

```
ay list <collection> [flags]
```

Flags

Flag	Default	Zweck
--limit N	20	Anzahl Ergebnisse pro Page
--skip N	0	Offset
--sort <field>:<-1|1>	createdAt:-1	Sortierung
--filter '<json>'	{}	MongoDB-Filter-Objekt
--fields '<json>'	alle	Projection
--format	config-default	json / yaml / table / csv
-m / --minimal	false	Nur Payload, kein {payload, meta}-Envelope

Beispiele

```
# Einfache Liste mit 5 Einträgen
ay list consumers --limit 5

# Gefiltert, nur ausgewählte Felder
ay list orders \
  --filter '{"status":"paid","total":{"$gte":100}}' \
  --fields '{"_id":1,"total":1,"consumer":1}' \
```

```
    --sort 'createdAt:-1' \
    --limit 10

  # Export zu CSV
  ay list invoices -f csv -m > invoices.csv
```

Paginierung

Bei mehr als --limit Treffern zeigt die CLI:

```
  Showing 20 of 1347 matching docs (page 1 of 68)
  Next page: ay list consumers --skip 20
```

Performance-Tipp

```
  ay list <collection> --no-sort           # schnellster Path, überspringt Sortier-Stage
  ay list <collection> --skip-count        # kein Total-Count (schneller bei großen
Collections)
```

Fehlerfälle

- HTTP 403 — Keine Rechte auf Collection. Via ay access prüfen.
- HTTP 404 — Collection existiert nicht. ay modules für korrekten Namen.
- Filter-Syntax-Error — JSON muss valide sein; bei Shell-Escaping-Problemen via Heredoc oder File.

Siehe auch

- get (wiki:cli/crud-core/get) — einzelner Datensatz
- search (wiki:cli/queries/search) — Volltextsuche

---

ay modules

ay modules (Alias: m)

Interaktiver Browser für Module !’ Collections !’ Operationen. Auch nutzbar als Direkt-Mode mit positional args (analog zu kubectl get).

Syntax

```
  ay modules [module] [collection] [operation] [subject] [flags]
```

Operationen

list · get · create · delete (alle anderen !’ siehe dedizierte Commands).

Flags

| Flag | Default | Zweck | |---|---|---| | -p, --page <n> | 1 | Seite (für list) | | -l, --limit <n> | 20 | Page-Size | | -f, --from <date> | — | Ab-Datum | | -a, --all | false | Alle Pages fetchen | | -q, --search <term> | — | Suchbegriff |

Beispiele

```
# Vollinteraktiver Browser
ay modules

# Direkt ins CRM-Modul
ay m crm

# CRM Consumers Collection
ay m crm consumers

# Direkt-Mode: Operation + Subject
ay m pm projects list
ay m pm projects create "Migration"
ay m pm projects get 64a1b2c3
ay m pm projects delete 64a1b2c3
```

Verhalten

- Im Interactive-Mode werden nur die für den User accessible Module + Collections angezeigt.
- Superuser sehen zusätzlich das su-Modul (57 System-Collections).
- Module-Namen sind toleriert: crm, CRM, Customer Relationship Management matchen alle.

Siehe auch

- access (wiki:cli/auth-config/access) — Was darf der User?
- list (wiki:cli/crud-core/list) — Direct-CRUD ohne Drilldown

---

ay storage

ay storage (Alias: s)

Zeigt und verwaltet den lokalen CLI-Speicher: AES-256-CBC-verschlüsselte Tokens (token, refreshToken) und Klartext-Preferences (lastModule, lastCollection, activeCustomerName, …).

Sensitive Werte werden in der Anzeige automatisch redacted (z.B. eyJh••••5678). Persistierung in ~/.aYOUne/storage/.

Syntax

```
ay storage                    # Show
ay storage set <key> <value> # Set token | refreshToken
ay storage clear <key>       # Remove token | refreshToken
```

Beispiele

```
# Aktuellen Speicher inspizieren (redacted)
ay storage

# Token aus anderem Login persistieren
ay storage set token eyJhbGciOi...

# Token löschen (= sanftes Logout)
ay storage clear token
```

```
# Refresh-Token resetten
ay storage clear refreshToken
```

Verhalten

- set akzeptiert nur die Keys token und refreshToken.
- Werte müssen valide JWT-Form haben (<header>.<payload>.<signature>), sonst wird der Set abgelehnt.
- Nicht-sensitive Keys (z.B. lastCollection) werden direkt gelesen, nicht entschlüsselt.

Speicher-Order

--token <jwt> Flag (höchste Priorität)

secureStorage.token (dieser Befehl schreibt hierhin)

AYOUNE_TOKEN Environment-Variable (niedrigste Priorität)

Siehe auch

- login (wiki:cli/cmd-login) — Browser-Login
- logout (wiki:cli/auth-config/logout) — Logout-Equivalent
- whoami (wiki:cli/auth-config/whoami) — Aktiven User anzeigen

---

ay update

ay update (Alias: u)

Aktualisiert einen Eintrag non-interaktiv — gemacht für Scripts und AI-Agenten. Im Gegensatz zu edit öffnet update keinen Editor; die Änderungen kommen via --set, --body, --body-file oder --body-stdin.

Syntax

```
ay update [collection] [id] [flags]
```

Flags

| Flag | Default | Zweck | |---|---|---| | --set <key=value...> | — | Einzelne Felder (wiederholbar) | | --body <json> | — | Komplettes Update als JSON-String | | --body-file <path> | — | JSON-Datei einlesen | | --body-stdin | — | JSON via stdin pipen | | --merge | false | GET !’ merge !’ PUT (existing fields bleiben erhalten) |

--set-Werte werden auto-gecastet: true/false !’ boolean, null !’ null, Zahlen !’ number, Rest bleibt string.

Beispiele

```
# Einzelne Felder
ay update contacts 64a1b2c3 --set firstName=Jane --set active=true

# AWS-State eines Computing Entity
ay update computingentities 64a1b2c3 --set ip=35.159.50.19 --set instanceId=i-019c...
```

```
# JSON-Body
ay update contacts 64a1b2c3 --body '{"firstName":"Jane","tags":["vip"]}'

# Aus Datei
ay update products 64a1b2c3 --body-file changes.json

# Aus stdin
echo '{"status":"active"}' | ay update contacts 64a1b2c3 --body-stdin

# Merge-Mode (GET + spread + PUT)
ay update tasks 64a1b2c3 --set done=true --merge
```

Siehe auch

- edit (wiki:cli/crud-core/edit) — interaktiver Editor
- batch (wiki:cli/utilities/batch) — Bulk-Updates
- exec (wiki:cli/utilities/exec) — Custom-Actions

---

ay upload

ay upload

Lädt eine lokale Datei in die Documents-Collection des aktiven Customers hoch. Pflicht ist eine Target-Entity, an die das Document gehängt wird.

Syntax

```
ay upload <file> --target <id> [flags]
```

Flags

Flag	Default	Zweck
-s, --state <right>	cms.downloads.edit	Upload-Kontext / Access-Right
-t, --target <id>	— (Pflicht)	ObjectId der Ziel-Entity

Beispiele

```
# Software-Installer hochladen
ay upload ./installer.exe \
  --state cms.downloads.edit \
  --target 698d574...

# PDF einem Consumer-Datensatz anhängen
ay upload ./report.pdf \
  --state crm.documents.edit \
  --target 507f1f7...

# Receipt an Invoice
ay upload ./receipt.pdf \
  --state accounting.invoices.edit \
  --target 64a1b2c3...
```

Verhalten

- Multipart-POST gegen config-api.ayoune.app/documents/upload.
- Server vergibt _id, schreibt File-Metadata + Storage-Pfad.
- Antwort enthält documentId, filename, size — bei --format json kompletter Output.

Fehlerfälle

- HTTP 403 — --state matched kein Right des Users. Via ay access prüfen.
- HTTP 413 — File > Customer-Quota. Customer-Limit hochsetzen.
- HTTP 422 — Target-Entity existiert nicht oder gehört anderem Customer.

Siehe auch

- list documents (wiki:cli/crud-core/list) — ay list documents
- credentials (wiki:cli/folder-based/credentials) — für Cloud-Storage-Keys

---

DevOps & Deployment

DevOps & Deployment

7 Commands für Deployment-Steuerung, Monitoring, Datenbank-Operationen, Release-Management und Projekt-Koordination.

Commands

| Command | Alias | Zweck | |---|---|---| | deploy (wiki:cli/devops/deploy) | dp | K8s-Deployments, Pipelines, Clusters (8 Subcommands) | | monitor (wiki:cli/devops/monitor) | mon | Logs, Alerts, Sessions, Uptime | | status (wiki:cli/devops/status) | — | Platform-Health-Check | | self-host-update (wiki:cli/devops/self-host-update) | shu | Self-hosted Instance updaten | | db (wiki:cli/devops/db) | — | Copy/Pull zu externen DBs (4 Subcommands) | | release (wiki:cli/devops/release) | rel | Releases, Versions, Channels | | pm (wiki:cli/devops/pm) | — | Project Management (Releases, Sprints, Epics, Standup) |

Hero: ay deploy

```
ay deploy clusters                  # Cluster-Liste
ay deploy pipelines                 # Aktive Pipelines
ay deploy deployments               # K8s-Deployments
ay deploy pods                      # Running Pods
ay deploy dashboard                 # Interaktives TUI-Dashboard
ay deploy alerts                    # Offene Alerts
```

Beispiel: ay db pull

```
ay db pull consumers --target staging --stream    # Streaming NDJSON
ay db pull orders --target local --since 2026-04-01
```

Unterstützt paginierten Fallback bei großen Collections und --stream für NDJSON-Output.

Beispiel: ay pm standup

```
ay pm standup                    # Formatierter Daily-Standup-Report
ay pm sprints                    # Aktive Sprints
ay pm roadmap                    # High-Level-Roadmap-Timeline
```

Siehe auch

- monitor (wiki:cli/devops/monitor) — Log-Streaming
- release (wiki:cli/devops/release) — Semantic-Release-Workflow

---

ay jobs

ay jobs (Alias: j)

Verwaltet scheduled/queued Background-Jobs (Agenda + BullMQ), Automations, Triggers und Notifications.

Subcommands

ay jobs list

Listet Jobs.

| Flag | Default | Zweck | |---|---|---| | --status <s> | — | active, waiting, completed, failed, delayed | | -l, --limit <n> | 50 | Page-Size | | -p, --page <n> | 1 | Seite |

```
ay jobs list
ay jobs list --status failed
ay jobs list --status delayed -l 100
```

ay jobs triggers

Automation-Triggers.

```
ay jobs triggers
ay jobs triggers -l 100
```

ay jobs automations (Alias: auto)

Definierte Automations.

```
ay jobs automations
ay jobs automations --active
```

ay jobs execute <automationId> (Alias: run)

Führt eine Automation sofort aus, optional mit Payload.

```
ay jobs execute 64a1b2c3
ay jobs run 64a1b2c3 --body '{"consumerId":"66ab..."}'
```

ay jobs notifications (Alias: notify)

Recent Notifications der CRM-Notifications-Collection.

```
ay jobs notifications
ay jobs notifications -l 50
```

Hinweis

ay jobs list querit die agendajobs-Collection (legacy general-Module). Für BullMQ-spezifische Queue-Operations siehe interne Dashboards via deploy dashboard (wiki:cli/devops/deploy).

Siehe auch

- monitor (wiki:cli/devops/monitor) — Logs + Alerts
- webhooks (wiki:cli/utilities/webhooks) — Outbound-Hooks

---

ay monitor

ay monitor (Alias: mon)

Monitoring-Befehl für Plattform-Logs, Alerts, aktive Sessions und Uptime-Checks.

Subcommands

ay monitor logs [type]

Listet recent Plattform-Logs aus der jeweiligen Logs-Collection. Default-Type: api.

| Typ | Collection | |---|---| | api | apilogs | | error | errorlogs | | mail | maillogs | | ai | ailogs | | trigger | triggerlogs | | doi | doilogs | | export | exportlogs | | sms | smslogs | | whatsapp | whatsapplogs | | production | productionlogs | | state | statelogs | | googleads | adwordslogs | | computing | computingentitieslogs |

(weitere: setup, shop, score, sensor, stock, soi, reward, merge, download, post, webreceiver, work, consumerapi, accessterminal)

```
ay monitor logs error
ay monitor logs api -l 10
ay monitor logs ai -r table
```

ay monitor alerts list

Listet Alerts.

| Flag | Default | Zweck | |---|---|---| | --severity <lvl> | — | info / warning / critical | | --type <t> | — | podcrash, oomkilled, imagepullerror, … | | --status <s> | active | active / acknowledged / resolved |

```
ay monitor alerts list --severity critical
ay monitor alerts list --type pod_crash
ay monitor alerts list --status acknowledged
```

ay monitor alerts ack <id> / resolve <id>

```
ay monitor alerts ack 64a1b2c3
ay monitor alerts resolve 64a1b2c3
```

ay monitor sessions

Aktive User-Sessions.

```
ay monitor sessions
ay monitor sessions -l 100
```

ay monitor checks

Konfigurierte Uptime/Health-Checks und ihre letzten Ergebnisse.

```
ay monitor checks
```

ay monitor activeusers (Alias: users)

Aktuell aktive User der Plattform.

```
ay monitor activeusers
```

ay monitor pagestats (Alias: pages)

Page-View-Statistiken.

```
ay monitor pagestats -l 50
```

Siehe auch

- status (wiki:cli/devops/status) — Service-Health
- events (wiki:cli/streaming/events) — Live-Events
- jobs (wiki:cli/devops/jobs) — Queue-Monitoring

---

ay rdp

ay rdp (Alias: remote)

Verbindet zu einem Computing Entity (CE) via RDP. Lädt eine signierte .rdp-Datei, injiziert die Credentials per cmdkey und startet mstsc. Funktioniert primär unter Windows; auf macOS/Linux wird die .rdp-Datei nur generiert und der Pfad ausgegeben.

Syntax

```
ay rdp [id] [flags]
```

Flags

Flag	Default	Zweck
--list	—	Listet alle CEs ohne Connect
--no-connect	—	.rdp-Datei generieren, aber nicht starten

Beispiele

```
# Interaktiver Picker
ay rdp

# Direkt zu einer CE
ay rdp 64a1b2c3d4e5

# Nur Liste
ay rdp --list
```

```
# Datei ohne Auto-Connect
ay rdp 64a1b2c3 --no-connect
```

Verhalten

CLI ruft GET config/computingentities/<id>/actions/rdp-connect auf.

Server liefert ip, username, password, instanceId, rdpFile (Templated mit IP).

CLI patcht prompt for credentials:i:1 !’ :i:0, schreibt .rdp ins os.tmpdir().

Auf Windows: cmdkey /add:<ip> /user:<u> /pass:<p> !’ mstsc <file>.

Auf anderen OS: nur Pfad-Ausgabe.

Fehlerfälle

- No public IP — CE wurde provisioniert aber Cloud-Sync läuft noch. Warten oder ay sync clusters triggern.
- HTTP 403 — Recht config.computingentities.actions.rdp-connect fehlt.
- Provider-Down — AWS/GCP-State stopped/terminated !’ CE muss erst gestartet werden.

Siehe auch

- list computingentities (wiki:cli/crud-core/list) — CE-Inventory
- sync (wiki:cli/devops/sync) — Cloud-State-Sync

---

ay setup

ay setup

Interaktiver Wizard für Self-Hosted aYOUne-Deployments. Generiert .env (Compose) oder values.yaml (Helm) und kann das Stack direkt starten.

Syntax

```
ay setup [-o <dir>]
```

Flags

Flag	Default	Zweck
-o, --output <dir>	.	Output-Verzeichnis für generierte Files

Beispiele

```
# Standard
ay setup

# In ein dediziertes Config-Verzeichnis
ay setup --output ./config

# Eigenes Verzeichnis
ay setup -o /etc/ayoune/
```

Wizard-Schritte

Deployment-Mode — Docker Compose (Single-Server) vs. Kubernetes (Helm).

Domain — z.B. ayoune.example.com.

MongoDB — Connection-String (Default: lokaler Container).

Redis — Host, Port, Passwort.

JWT-Secret — Leer für Auto-Generierung (64 Zeichen Random).

SMTP — Host, Port, User, Passwort.

License-Key — Leer für 14-Tage-Trial.

Modules — Multi-Select aus 13 Modulen (CRM, Marketing, HR, E-Commerce, PM, DevOps, Accounting, Automation, Support, Content, Communication, Reporting, Monitoring).

Generierte Files

Compose-Mode

- .env mit allen ENV-Variablen
- Optional sofortiger Launch via docker compose up -d

Kubernetes-Mode

- .env (für lokales Testing)
- values.yaml für helm install ayoune tolinax/ayoune -f values.yaml

Hinweis

ay setup darf erneut auf einem bestehenden Setup laufen — die Files werden überschrieben. Vorher Backup machen, falls Customizations vorgenommen wurden.

Siehe auch

- local (wiki:cli/folder-based/local) — Compose-Stack starten
- self-host-update (wiki:cli/devops/self-host-update) — Updates
- provision (wiki:cli/folder-based/provision) — Cloud-Provisioning

---

ay status

ay status

Health-Check der aYOUne-Plattform-Services. Pingt parallel alle bekannten Module-Hosts, akzeptiert HTTP < 500 mit JSON-Body als healthy (auch wenn 404, sofern Version-Metadata enthalten).

Syntax

```
ay status [flags]
```

Flags

| Flag | Zweck | |---|---| | --module <name> | Nur ein Modul checken |

Beispiele

```
# Alle Services
ay status

# Nur CRM
ay status --module crm

# JSON für Scripts
ay status -r json

# Tabelle
ay status -r table
```

Output (Pretty-Mode)

```
aYOUne Platform Status

%Ï config-api     v2026.4.1   (78ms)
%Ï auth           v2026.5.0   (102ms)
%Ï crm-api        v2026.3.2   (88ms)
%Ï marketing-api  v2026.2.1   (94ms)
%Ï devops-api     v2026.1.0   (110ms) — HTTP 503

4 healthy   1 unhealthy
```

Geprüfte Services

Core (immer): config-api, auth.

Module (alle bei ay status ohne --module): crm-api · marketing-api · hr-api · ecommerce-api · pm-api · devops-api · accounting-api · automation-api · support-api · reporting-api · monitoring-api.

Exit-Code

- 0 — alle Services healthy
- 1 — mindestens ein Service unhealthy oder unreachable

Damit ist ay status als CI-Gate verwendbar:

```
ay status --module crm && ay deploy crm-api
```

Siehe auch

- services health (wiki:cli/auth-config/services) — alternativer Health-Check
- doctor (wiki:cli/misc/doctor) — Lokale Environment-Checks

---

ay sync

ay sync

Synchronisiert Plattform-Daten mit angebundenen Drittsystemen — Bitbucket/GitHub-Repos, K8s-Cluster-Inventory, Pipeline-Status.

Subcommands

ay sync repos

Pullt Repository-Listen von angebundenen Providern.

| Flag | Zweck | |---|---| | --provider <p> | Filter: bitbucket / github | | --id <repoId> | Nur ein Repo syncen |

```
ay sync repos
ay sync repos --provider bitbucket
ay sync repos --id 64a1b2c3
```

ay sync clusters

Synct K8s-Cluster-Status (Pods, Deployments, Resources).

```
ay sync clusters
ay sync clusters --id 64a1b2c3
```

ay sync pipelines

Pullt Pipeline-Runs vom CI/CD-Provider (Bitbucket/GitHub).

```
ay sync pipelines
ay sync pipelines --provider bitbucket -l 100
```

ay sync status

Zeigt aggregierten Sync-Status: Anzahl Repos, Cluster, Pipelines.

```
ay sync status
```

Verhalten

- Repos: bei --id direkt POST /repositories/<id>/sync. Ohne --id iteriert die CLI client-seitig durch alle Repos.
- Clusters: analog.
- Pipelines: Server-seitiger Bulk-Sync via POST /pipelines/sync.

Siehe auch

- deploy (wiki:cli/devops/deploy) — Deployments
- monitor alerts (wiki:cli/devops/monitor) — Alerts auf Sync-Failures
- status (wiki:cli/devops/status) — Service-Health

---

ay webhooks

ay webhooks (Alias: hooks)

Verwaltet Outbound-Webhooks: Erstellen, Inspizieren, Löschen und Templates listen. Daten landen in Hooks (Module: config).

Subcommands

ay webhooks list

Listet alle registrierten Hooks.

```
ay webhooks list
ay webhooks list -l 100
```

ay webhooks get <id>

Detail-View eines Hooks.

```
ay webhooks get 64a1b2c3
```

ay webhooks create [flags]

Neuen Hook anlegen.

| Flag | Zweck | |---|---| | --body <json> | Hook-Definition als JSON-String | | --body-file <path> | Hook-Definition aus Datei |

```
ay webhooks create --body '{
  "name":"order-paid !’ fulfillment",
  "url":"https://api.fulfillment.com/orders",
  "trigger":{"event":"orders.paid"},
  "secret":"whsec_..."
}'

ay webhooks create --body-file ./hooks/order-paid.json
```

ay webhooks delete <id> (Alias: rm)

```
ay webhooks delete 64a1b2c3
```

ay webhooks templates

Listet vordefinierte Webhook-Templates.

```
ay webhooks templates
```

Hinweis

Für Update + Delivery-Logs (Pre-MVP) !’ bis dahin via ay update hooks <id> und ay list hooklogs --filter '{"_hookID":"<id>"}'.

Siehe auch

- jobs (wiki:cli/devops/jobs) — Job-Queue
- events (wiki:cli/streaming/events) — Event-Bus

Folder-based Commands

Folder-based Commands

3 Command-Gruppen, die als eigene Folders implementiert sind (mehrere Subcommands):

| Gruppe | Subcommands | Zweck | |---|---|---| | local (wiki:cli/folder-based/local) | 7 | Self-Hosting via

Docker-Compose | | functions (wiki:cli/folder-based/functions) (fns) | 9 | Custom Functions / FaaS | | provision (wiki:cli/folder-based/provision) (prov) | 7 | Cloud-Provisioning AWS/GCP/Azure/DigitalOcean/Hetzner |

Beispiel: ay local

```
ay local up                  # Docker-Compose up mit allen konfigurierten Profilen
ay local down                # Down
ay local restart <service>   # Einzelnen Service neustarten
ay local logs <service>      # Logs folgen
ay local ps                  # Container-Status
ay local pull                # Images pullen
ay local exec <service> <cmd># Command im Container
```

Beispiel: ay functions

```
ay functions list                       # Deployed Functions listen
ay functions create my-fn               # Neue Function anlegen
ay functions deploy my-fn               # Deployen
ay functions invoke my-fn --data '{…}'  # Aufrufen
ay functions logs my-fn                 # Logs ansehen
ay functions rollback my-fn v3          # Rollback
```

Beispiel: ay provision

```
ay provision list               # Verfügbare Templates
ay provision aws --region eu-central-1
ay provision gcp --project my-proj
ay provision hetzner --size cx11
```

Siehe auch

- deploy (wiki:cli/devops/deploy) — K8s-Deployment-Steuerung
- self-host-update (wiki:cli/devops/self-host-update) — Updates einspielen

---

ay credentials

ay credentials (Alias: cred)

Verwaltet den verschlüsselten Credentials-Vault des aktiven Customers — API-Keys, OAuth-Tokens, SMTP-Passwörter, Trading-Account-Logins. AES-256-CTR im Backend; Werte im List-Output sind maskiert.

23 unterstützte Provider: Slack, Telegram, Discord, SMTP, OAuth2 (Google/Microsoft/etc.), Twilio, Stripe, PayPal, Trading-Plattformen (MT4/MT5), Custom, …

Subcommands

ay credentials list

Listet alle Credentials (maskiert).

```
ay credentials list
```

```
ay credentials list --provider custom
ay credentials list --consumer 64a... --entity-type Contracts
```

ay credentials get <id>

Detail-View eines Credentials.

```
ay credentials get 66abc123def456
```

ay credentials create [flags]

Neues Credential anlegen.

```
# Slack-Bot
ay credentials create \
  --cred-name "Slack Bot" \
  --provider slack \
  --type bot_token \
  --token xoxb-...

# Trading-Account, an Contract gebunden
ay credentials create \
  --cred-name "Trading EUR" \
  --provider custom \
  --type password \
  --username trader@ex.com \
  --password s3cret \
  --consumer 66a... \
  --entity-type Contracts \
  --entity-id 66b...
```

ay credentials update <id>

Felder aktualisieren.

```
ay credentials update 66abc... --token xoxb-NEW
```

ay credentials delete <id>

Credential löschen (Bestätigung erforderlich).

```
ay credentials delete 66abc...
```

ay credentials decrypt <id> --field <name>

Entschlüsselt einen einzelnen Klartext-Wert (audit-logged).

```
ay credentials decrypt 66abc... --field password
```

ay credentials test <id>

Ruft den Provider auf und validiert die Credentials (z.B. chat.postMessage für Slack).

```
ay credentials test 66abc...
```

ay credentials refresh <id>

OAuth2-Token erneuern (via Refresh-Token).

```
ay credentials refresh 66abc...
```

ay credentials revoke <id>

Provider-seitig revoken + lokal deaktivieren.

```
ay credentials revoke 66abc...
```

ay credentials providers

Listet alle 23 Provider mit erforderlichen Feldern.

```
ay credentials providers
```

ay credentials status

Connected-Platforms-Summary.

```
ay credentials status
```

Siehe auch

- storage (wiki:cli/crud-core/storage) — CLI-Token-Storage
- users (wiki:cli/misc/users) — User + Roles

---

ay functions

ay functions (Alias: fns)

Lokale Entwicklung, Deployment und Verwaltung von Custom Functions (FaaS). Sandbox via isolated-vm, 5 Trigger-Types.

Subcommands

Subcommand	Zweck
list	Deployed Functions mit Version, Status, letzter Aufruf
get <name>	Function-Details + Code + Config
create <name>	Scaffold lokal + Remote-Stub anlegen
deploy <name>	Code + Config hochladen, neue Version aktivieren
invoke <name>	Ad-hoc-Aufruf mit --data '<json>'
delete <name>	Function entfernen
rollback <name> <version>	Auf frühere Version zurück
versions <name>	Version-History
logs <name>	Letzte N Execution-Logs streamen

Beispiel: Neue Function

```
ay functions create welcome-mail
# 'œ tooling/functions/welcome-mail/{index.ts, config.yaml}

# Code editieren, dann:
ay functions deploy welcome-mail
ay functions invoke welcome-mail --data '{"consumerId":"..."}'
ay functions logs welcome-mail --follow
```

Function-Config

```
# config.yaml
name: welcome-mail
description: Sendet Willkommens-Email an neuen Consumer
trigger:
  type: event           # oder: cron, webhook, manual, scheduled
```

```
  topic: consumers.created
runtime: node20
timeout: 10s
memory: 128MB
secrets:
  - SMTP_PASSWORD       # Credentials-Vault-Key
allowedEntities:
  - Consumers
  - Emails
```

Trigger-Types

Type	Auslöser
event	Platform-Event (z.B. orders.paid)
cron	Repeatable-Schedule (Cron-Expression)
webhook	Externer HTTP-Aufruf
manual	Per ay functions invoke
scheduled	Einmaliger Zeitpunkt

Platform-SDK

Inside einer Function steht ay.* zur Verfügung:

```
export default async function({ event, ay }) {
  const consumer = await ay.db('Consumers').findById(event.consumerId);
  await ay.email.send({ to: consumer.email, template: 'welcome' });
}
```

Der SDK ist opt-in-scoped: nur Entities in allowedEntities + mit availableInSDK=true in modelsAndRights.ts sind erreichbar (Stand 2026-04: 26 kuratierte Entries).

Siehe auch

- credentials (wiki:cli/misc/credentials) — Secrets verwalten
- jobs (wiki:cli/misc/jobs) — Function-Queues beobachten

---

ay local

ay local

Cross-Platform-Wrapper um docker compose für lokale aYOUne-Self-Hosting-Stacks. Ersetzt infrastructure/local/dev.ps1 für macOS/Linux/Windows. Nutzt automatisch das infrastructure/local/-Overlay des Monorepos (wenn vorhanden), sonst ./docker-compose.yml.

Subcommands

ay local up [profiles…]

Startet das Compose-Stack mit gewählten Profilen (core ist immer dabei).

```
ay local up                      # nur core
ay local up crm marketing        # core + crm + marketing
ay local up crm marketing pm devops
```

ay local down

Stoppt alle Services.

```
ay local down
```

ay local restart <service>

Restart eines Services.

```
ay local restart api-config
ay local restart api-crm
```

ay local logs <service>

Folgt den Logs eines Services.

```
ay local logs api-config
ay local logs api-crm -f
```

ay local ps

Zeigt laufende Container mit Status.

```
ay local ps
```

ay local pull

Pullt die neuesten Images.

```
ay local pull
```

ay local exec <service> <command…>

Befehl im Container ausführen.

```
ay local exec api-crm sh
ay local exec api-config node -e "console.log('hi')"
```

Beispiel-Workflow

```
# Erstmaliges Setup
ay setup
ay local pull

# Stack starten
ay local up crm marketing

# Service-Status prüfen
ay local ps
ay status

# Service-Logs verfolgen
ay local logs api-crm

# Heruntefahren
ay local down
```

Siehe auch

- setup (wiki:cli/cmd-setup) — .env und Compose-File generieren
- self-host-update (wiki:cli/devops/self-host-update) — Image-Updates pullen + restarten

- provision (wiki:cli/folder-based/provision) — Cloud-Deployment

ay provision

ay provision (Alias: prov)

Provisioniert eine aYOUne-Instanz auf einem Cloud-Provider. Wraps die Marketplace-Templates aus infrastructure/marketplace/{aws,gcp,azure,digitalocean,hetzner}/. Provider-Tools (aws, gcloud, az, doctl, terraform) müssen lokal installiert sein — die CLI bündelt sie nicht.

State (Parameter, Outputs) wird in ~/.ayoune/provision/<provider>.json persistiert.

Subcommands

ay provision hetzner

Hetzner Cloud via Terraform.

```
ay provision hetzner
ay provision hetzner --dry-run
```

ay provision aws

AWS via CloudFormation.

```
ay provision aws
ay provision aws --dry-run
```

ay provision gcp

Google Kubernetes Engine.

```
ay provision gcp
```

ay provision azure

Azure via ARM-Template.

```
ay provision azure
```

ay provision digitalocean

DigitalOcean App-Platform-Spec.

```
ay provision digitalocean
```

ay provision status [provider]

Zeigt aktuellen State des Deployments.

```
ay provision status hetzner
```

ay provision destroy <provider>

Räumt das provisionierte Environment ab. Bestätigung erforderlich.

```
ay provision destroy hetzner
```

Hinweis

provision schreibt nur die Cloud-Infrastruktur (VMs, K8s-Cluster, Load-Balancer). Die aYOUne-Stack selbst wird im Anschluss via Helm oder Compose deployed — siehe setup (wiki:cli/cmd-setup) und self-host-update (wiki:cli/devops/self-host-update).

Siehe auch

- setup (wiki:cli/cmd-setup) — Self-Hosted Wizard
- local (wiki:cli/folder-based/local) — Lokales Compose-Stack
- self-host-update (wiki:cli/devops/self-host-update) — Updates

---

Queries & Analysis

Queries & Analysis

5 Commands für Volltextsuche, MongoDB-Aggregationen, Exports und Bulk-Operationen.

Commands

| Command | Alias | Zweck | |---|---|---| | search (wiki:cli/queries/search) | find | Volltextsuche + Model-Search | | aggregate (wiki:cli/queries/aggregate) | agg | MongoDB-Pipelines (7 Subcommands) | | export (wiki:cli/queries/export) | exp | Datenexport mit Format-Wahl | | batch (wiki:cli/queries/batch) | — | Bulk-Operationen | | access (wiki:cli/queries/access) | — | Zugriffsrechte überblicken |

Hero: ay aggregate

ay aggregate ist der mächtigste Query-Command. Er öffnet einen Wizard oder führt vordefinierte Pipelines aus:

```
  ay aggregate wizard                # Interaktiv
  ay aggregate run <saved-pipeline>  # Gespeicherte Pipeline
  ay aggregate exec --model Orders \
    --pipeline '[{"$match":{"status":"paid"}},{"$group":{"_id":"$country","total":
{"$sum":"$amount"}}}]'
```

Subcommands: run, exec, list, save, validate, models, wizard.

Export-Beispiele

```
  ay export consumers --format csv --filter '{"active":true}' -o consumers.csv
  ay export orders --format json --date-range 2026-04-01..2026-04-30
```

Siehe auch

- Konzept: Output-Formate (wiki:cli/concepts/output-formats)
- list (wiki:cli/crud-core/list)

---

ay aggregate

ay aggregate (Alias: agg)

MongoDB-Aggregation-Pipelines gegen beliebige Collections, mit 7 Subcommands für Wizard, Ausführung und Wiederverwendung.

Subcommands

| Subcommand | Zweck | |---|---| | wizard | Interaktiver Wizard: Collection, Stages, Output | | exec | Ad-hoc-Pipeline ausführen | | run <name> | Gespeicherte Pipeline ausführen | | list | Alle gespeicherten Pipelines | | save <name> | Pipeline speichern | | validate | Pipeline-Syntax prüfen | | models | Collections listen, die Aggregation erlauben |

Ad-hoc-Ausführung

```
ay aggregate exec \
  --model Orders \
  --pipeline '[
    {"$match":{"status":"paid"}},
    {"$group":{"_id":"$country","total":{"$sum":"$amount"}}},
    {"$sort":{"total":-1}},
    {"$limit":10}
  ]'
```

Wizard

```
ay aggregate wizard
```

Fragt Schritt-für-Schritt:

- Collection (aus verfügbaren Modulen)
- Stages (Match, Group, Lookup, Project, Sort, Limit …)
- Output-Format

Am Ende zeigt der Wizard die fertige Pipeline-JSON + den Output und bietet an, beides als Saved-Pipeline zu speichern.

Gespeicherte Pipelines

```
ay aggregate list                          # Alle gespeicherten
ay aggregate run revenue-by-country        # Ausführen
ay aggregate save revenue-by-country < pipeline.json
```

Saved-Pipelines liegen in der Aggregations-Collection und sind customer-scoped.

Flags

| Flag | Zweck | |---|---| | --model <Collection> | Ziel-Collection (PascalCase, Plural) | | --pipeline '<json>' | Pipeline-Array als JSON | | --file pipeline.json | Pipeline aus Datei | | --explain | Pipeline-Execution-Plan statt Ergebnis | | --allowDisk | allowDiskUse: true für große Pipelines | | --format | Output-Format |

Performance-Tipp

- --explain vor großen Pipelines — zeigt, welche Stages Indizes nutzen können.

- $match möglichst früh, um Volumen zu reduzieren.
- $project vor $lookup, um Inputs schlank zu halten.

Siehe auch

- list (wiki:cli/crud-core/list) — einfache Queries
- export (wiki:cli/queries/export) — Aggregat-Output exportieren

---

ay export

ay export (Alias: exp)

Daten-Export mit Pagination, Format-Wahl und Filter-Sprache. Subcommands für Run, List, Get und Configs.

Subcommands

ay export run <collection>

Exportiert die Inhalte einer Collection in einem Rutsch (Auto-Pagination).

| Flag | Default | Zweck | |---|---|---| | --format <fmt> | csv | json / csv / yaml | | --fields <list> | alle | Projection (kommagetrennt) | | --filter <key=val,…> | — | Filter wie bei search (wiki:cli/queries/search) | | --sort <field> | -createdAt | Sortierung | | -l, --limit <n> | 0 (= alles) | Limit (0 = alle Pages) |

```
ay export run contacts --format csv --fields "firstName,lastName,email"
ay export run products --format json --filter "status=active"
ay export run invoices --format csv --filter "createdAt>2026-01-01" --save
```

ay export list

Listet bestehende Export-Jobs.

```
ay export list -l 25
```

ay export get <id>

Detail + Download-URL eines Exports.

```
ay export get 64a1b2c3
```

ay export configs

Listet konfigurierte Export-Definitionen (z.B. wiederkehrende Exports an externe Targets).

```
ay export configs
```

ay export logs

Audit-Log aller Export-Runs.

```
ay export logs -l 25
```

Hinweis

Bei --limit 0 (Default) iteriert die CLI mit limit=500 durch alle Pages und mergt die Payloads — geeignet für Collections bis ~50k Einträge. Für größere !’ db pull (wiki:cli/devops/db) oder aggregate (wiki:cli/queries/aggregate) mit $out.

Siehe auch

- search (wiki:cli/queries/search) — Filter-Sprache
- aggregate (wiki:cli/queries/aggregate) — Pipelines
- db (wiki:cli/devops/db) — Cross-DB-Sync

---

ay search

ay search (Alias: find)

Volltextsuche und Field-Filter über aYOUne Search-Service. Vier Modi:

Single-Model — ay search consumers "John" (Default).

Global SSE — ay search -g "John" streamt Treffer aus allen Collections.

FindOne — --one gibt nur den ersten Treffer zurück.

Legacy — --legacy umgeht den Search-Service und nutzt das Module-API direkt (für Filter, die der Search-Index nicht unterstützt).

Syntax

```
ay search [collectionOrModule] [collectionOrQuery] [query] [flags]
```

Flags

| Flag | Default | Zweck | |---|---|---| | -g, --global <query> | — | SSE-Streaming über alle Collections | | --field <name> | — | Suche nur in einem Feld | | --one | false | Erster Treffer reicht | | --legacy | false | Module-API statt Search-Service | | --filter <key=val,…> | — | Field-Filter (=, !=, >, <, >=, <=) | | --fields <list> | — | Projection (kommagetrennt) | | --sort <field> | -createdAt | Sortierung (- für desc) | | --count | false | Nur Treffer-Anzahl ausgeben | | -l, --limit <n> | 25 | Page-Size | | -p, --page <n> | 1 | Seite |

Beispiele

```
# Volltext im Search-Index
ay search consumers "John"
ay search crm consumers "John"

# Feld-spezifisch
ay search consumers "John" --field firstName

# Filter mit Operatoren
ay search orders --filter "status=paid,total>500"
ay search invoices --filter "createdAt>2026-04-01" --sort -total

# Erster Treffer (FindOne)
ay search consumers "Doe" --one --fields _id,firstName,lastName

# Globale SSE-Suche
```

```
ay search -g "tolinax" -l 50

# Nur zählen
ay find tasks --filter "status=open" --count
```

Sanitizing

--fields wird gegen ^-?[a-zA-Z][a-zA-Z0-9.$]*$ validiert — MongoDB-Operatoren ($where, $gt, …) werden abgelehnt.

Siehe auch

- list (wiki:cli/crud-core/list) — primitive Listenanzeige
- aggregate (wiki:cli/queries/aggregate) — komplexe Pipelines
- export (wiki:cli/queries/export) — größere Datenmengen exportieren

---

Streaming & Events

Streaming & Events

2 Commands für Live-Updates einer Collection und Platform-Event-Subscription.

Commands

| Command | Alias | Zweck | |---|---|---| | stream (wiki:cli/streaming/stream) | listen | Live-Änderungen einer Collection (Change-Stream) | | events (wiki:cli/streaming/events) | sub | Platform-Event-Bus-Subscription |

Beispiel: Live-Consumer-Änderungen

```
ay stream consumers
# Terminal zeigt jede Insert/Update/Delete in Echtzeit
```

Flags:

- --filter '<json>' — nur Änderungen, die einem MongoDB-Filter matchen
- --ops insert,update — nur bestimmte Operationen

Beispiel: Platform-Events

```
ay events --topic orders.paid
ay events --topic '*.failed' --since 1h
```

Topics sind in der Events-Collection definiert (events Module), Pattern-Match mit * erlaubt.

Siehe auch

- jobs (wiki:cli/misc/jobs) — Queue-Monitoring
- monitor (wiki:cli/devops/monitor) — Log-Streaming

---

ay events

ay events (Alias: sub)

Subscribed via WebSocket auf den aYOUne Customer-Event-Bus und filtert die Events auf der Client-Seite. Output formatiert als JSON, YAML oder Tabelle.

Syntax

```
ay events [flags]
```

Flags

| Flag | Default | Zweck | |---|---|---| | -f, --format <fmt> | json | json / yaml / table | | -c, --category <cat> | ` | Event-Category (z.B. sales, crm) | | -a, --action <act> | | Action (z.B. create, update, delete) | | -l, --label <label> | | Label-Filter | | -V, --value <val> | ` | Value-Filter |

* = match-all. Filter werden client-seitig ausgewertet — Server sendet alle Events des aktiven Customers.

Beispiele

```
# Alle Events
ay events

# CRM-Events als YAML
ay sub -c crm -f yaml

# Sales-Creates in Tabellenform
ay events -c sales -a create -f table

# Nur Errors
ay events -c errors -f json

# Notification-Worker im Blick behalten
ay events -c notifications -f yaml
```

Schema

Jedes Event hat:

```
{
  category: string,
  action: string,
  label: string,
  value: string,
  evt_data: { …payload }
}
```

Hinweis

Die WebSocket-Connection bleibt offen bis Ctrl+C. Auto-Reconnect ist eingebaut (mit Customer-Room-Rejoin). Für strukturierte Long-Term-Logs siehe monitor logs (wiki:cli/devops/monitor).

Siehe auch

- stream (wiki:cli/streaming/stream) — Live-Änderungen einer Collection
- monitor (wiki:cli/devops/monitor) — Persistente Log-Streams

---

ay stream

ay stream (Alias: listen)

Live-Stream aller Änderungen einer Collection. Nutzt MongoDB Change-Streams unter der Haube.

Syntax

```
ay stream <collection> [flags]
```

Flags

| Flag | Zweck | |---|---| | --filter '<json>' | Nur Änderungen, die einem Filter matchen | | --ops insert,update,delete,replace | Nur bestimmte Operation-Types | | --fields '<json>' | Projection auf geänderte Felder | | --format json\|yaml\|table | Output-Format |

Beispiele

```
# Alle Consumer-Änderungen
ay stream consumers

# Nur Inserts im CRM-Scope
ay stream consumers --ops insert

# Nur bestimmte Felder bei Updates
ay stream orders \
  --ops update \
  --filter '{"status":"paid"}' \
  --fields '{"_id":1,"total":1,"status":1}'
```

Output-Pattern

Jede Änderung kommt als JSON-Event:

```
{
  "op": "update",
  "ns": "consumers",
  "_id": "64a1b2c3...",
  "ts": "2026-04-22T14:32:11.000Z",
  "updatedFields": {"lastLoginAt": "2026-04-22T14:32:10.000Z"},
  "fullDocument": { … }
}
```

Abbruch

Ctrl-C beendet den Stream sauber.

Leistung

Change-Streams benötigen ein Replica-Set (Standard im aYOUne-Atlas-Cluster). Bei sehr hohem Traffic --filter nutzen, sonst füllt sich das Terminal schnell.

Siehe auch

- events (wiki:cli/streaming/events) — Platform-Event-Bus
- monitor (wiki:cli/devops/monitor) — Log-Streaming

Utilities & Integration

Utilities & Integration

5 Commands für benutzerdefinierte Actions, KI-Services, Endpoint-Discovery und Webhook-Verwaltung.

Commands

| Command | Alias | Zweck | |---|---|---| | actions (wiki:cli/utilities/actions) | act | Custom Actions einer Collection listen | | exec (wiki:cli/utilities/exec) | x | Action per Name ausführen | | ai (wiki:cli/utilities/ai) | — | aYOUne-AI-Services (RAG, Translate, Generate) | | services (wiki:cli/utilities/services) | svc | Platform-Services + Endpoints | | webhooks (wiki:cli/utilities/webhooks) | hooks | Outbound-Webhooks |

Custom Actions

Jede Collection kann neben Standard-CRUD Custom Actions haben (POST /collection/:id/actions/<slug>).

```
ay actions consumers                                    # Actions der Collection listen
ay exec consumers 64a1b2c3... actions send-welcome      # Action ausführen
```

AI-Services

```
ay ai translate --source de --target en --text "Hallo Welt"
ay ai rag --query "Wie exportiere ich Rechnungen?"
ay ai generate --prompt "Schreibe eine Willkommens-Email"
```

Services-Discovery

```
ay services                                             # Alle Platform-Services
ay services --domain crm                                # Nur CRM-Module
ay services endpoints crm                               # Alle Endpoints des crm-api
```

Webhooks

```
ay webhooks list                      # Aktive Webhooks
ay webhooks create --url https://...  # Neuen Webhook anlegen
ay webhooks test <id>                 # Test-Payload senden
ay webhooks logs <id>                 # Delivery-Logs
```

Siehe auch

- jobs (wiki:cli/misc/jobs) — Async-Queue-Jobs

- permissions (wiki:cli/misc/permissions) — Rechte-Verwaltung

ay actions

ay actions (Alias: act)

Listet alle registrierten API-Actions der Plattform — Discovery-Tool für ay exec. Querit apiactions (mit config !' su Fallback).

Syntax

```
ay actions [search] [flags]
```

Flags

| Flag | Zweck | |---|---| | --namespace <ns> | Filter: nur Actions eines Namespaces (z.B. ai, crm) | | --host <host> | Filter: nur Actions auf einem bestimmten Host | | --method <m> | Filter: GET / POST / PUT / DELETE | | --capability <cap> | Filter: nach Capability (z.B. consumers) | | --list-namespaces | Listet alle Namespaces | | --list-hosts | Listet alle Hosts | | -p, --page <n> | Seite (Default 1) | | -l, --limit <n> | Page-Size (Default 100) |

Beispiele

```
# Komplette Liste
ay actions

# Search-Mode
ay actions generate
ay actions sendMail

# Namespace-Filter
ay actions --namespace ai
ay actions --namespace crm

# POST-Actions auf einem Host
ay actions --host crm-api.ayoune.app --method POST

# Übersicht der Namespaces
ay actions --list-namespaces

# Tabellen-Ausgabe für AI-Pipelines
ay actions -r table --columns "operationId,method,endpoint,host"
```

Action-Schema

Jede Action enthält:

```
operationId: sendWelcomeMail
method: POST
endpoint: /consumers/:id/actions/send-welcome-mail
host: crm-api.ayoune.app
nameSpace: crm
capability: consumers
params: [{ name: id, in: path, required: true }]
description: Sends a welcome email to the consumer
```

Siehe auch

- exec (wiki:cli/utilities/exec) — Action ausführen
- services (wiki:cli/auth-config/services) — Service-Discovery
- curl (wiki:cli/misc/curl) — Direkter HTTP-Call

---

ay ai

ay ai

Zugriff auf aYOUne-AI-Services: RAG-Copilot, Translate, Text-Generation.

Subcommands

| Subcommand | Zweck | |---|---| | rag | RAG-Query gegen Knowledge-Base + Platform-Docs | | translate | Text zwischen Sprachen übersetzen (Multi-Provider) | | generate | Freiform-Text-Generation via LLM | | ask | Interaktives Chat-Interface |

Beispiele

```
# RAG-Query
ay ai rag --query "Wie exportiere ich Rechnungen als CSV?"

# Übersetzung
ay ai translate --source de --target en --text "Hallo Welt"

# Freiform-Generation
ay ai generate --prompt "Schreibe eine Willkommens-Email im Du-Tonus"

# Interaktives Chat
ay ai ask
> Hallo, wie kann ich alle Consumers mit fehlgeschlagenen Orders finden?
```

Flags

| Flag | Zweck | |---|---| | --provider openai\|anthropic\|google\|cohere\|ollama | LLM-Provider (Default: customer-preset) | | --model <id> | Spezifisches Modell | | --stream | Token-Streaming (Standard) | | --file <path> | Prompt aus Datei |

Output

Streaming-Text direkt auf stdout. Bei --format json: komplettes Response-Objekt mit usage-Info (Input/Output-Tokens).

Siehe auch

- Platform AI Overview (wiki:developer-portal/ai-services)
- Custom Functions — AI (wiki:cli/folder-based/functions)

---

ay batch

ay batch

Bulk-Operationen über mehrere Datensätze — gemacht für Scripts und AI-Agenten. Kompakter als ein Loop um ay get / ay delete / ay update, mit aggregiertem Reporting.

Subcommands

ay batch get <collection> <ids>

Multi-GET via kommaseparierte IDs.

```
ay batch get contacts id1,id2,id3
echo "id1,id2,id3" | ay batch get contacts --stdin
```

ay batch delete <collection> <ids> (Alias: rm)

Multi-DELETE. --force empfohlen für non-TTY.

```
ay batch delete contacts id1,id2,id3 --force
echo "id1,id2" | ay batch delete contacts --stdin --force
```

ay batch update <collection> <ids> (analog)

Multi-UPDATE mit --set oder --body.

```
ay batch update tasks id1,id2,id3 --set status=archived --force
```

Flags (alle Subcommands)

| Flag | Zweck | |---|---| | --stdin | IDs aus stdin lesen (zeilen- oder kommagetrennt) | | --force | Confirmation überspringen |

Pipeline-Pattern

```
# Alle archivierten Tasks älter als 30 Tage löschen
ay search tasks --filter "status=archived,createdAt<2026-03-26" -i _id -m \
  | ay batch delete tasks --stdin --force

# Alle inaktiven Consumers reaktivieren
ay list consumers --filter '{"active":false}' --fields '{"_id":1}' -m \
  | ay batch update consumers --stdin --set active=true --force
```

Output

Sequenzielle Ausführung mit Status-Spinner:

```
Fetched 18/20 entries (2 errors)
```

Bei mindestens einem Fehler !’ Exit 1.

Siehe auch

- delete (wiki:cli/crud-core/delete) — Single-Delete (auch Bulk via Komma-IDs)
- update (wiki:cli/crud-core/update) — Single-Update
- search (wiki:cli/queries/search) — IDs für Pipes finden

ay db

ay db

Kopiert Query-Resultate in externe Datenbanken (Push) oder synct die Plattform-DB auf eine lokale MongoDB (Pull). Alle DB-Operationen laufen über den Aggregation-Service — die CLI verbindet niemals direkt zur Plattform-DB.

Subcommands

ay db copy [target] [query]

Push-Mode: führt eine gespeicherte Aggregation auf der Plattform aus und schreibt in einen pre-konfigurierten DataTarget (Slug-basiert).

| Flag | Zweck | |---|---| | --collection <name> | Target-Collection-Name overriden | | --write-mode <m> | insert, upsert, replace |

```
# Voll-interaktiv
ay db copy

# Target-only, Query interaktiv
ay db copy my-warehouse-pg

# Voll-autonom
ay db copy my-warehouse-pg active-users

# Mit Override
ay db copy my-warehouse-pg active-users --collection users_2026 --write-mode upsert
```

ay db pull [target]

CLI-side Pull via Streaming-NDJSON + mongoimport zu lokaler MongoDB.

| Flag | Zweck | |---|---| | --since <iso> | Nur Records nach Datum | | --every <duration> | Wiederholendes Pull (z.B. 5m, 1h) | | --max-iterations <n> | Limit für --every | | --write-mode <m> | Default upsert für Local-Targets |

```
ay db pull consumers --target local --stream
ay db pull orders --target local --since 2026-04-01
ay db pull tasks --target local --every 5m --max-iterations 100
```

ay db targets list (Alias: ls)

Listet konfigurierte DataTargets.

```
ay db targets list
```

ay db targets add

Neues Target anlegen.

| Flag | Zweck | |---|---| | --name <name> | Anzeigename | | --type <t> | mongodb, postgresql, rest | | --uri <uri> | Connection-URI | | --database <db> | DB-Name | | --collection <col> | Default-Collection |

```
ay db targets add \
  --name "Local MongoDB" \
```

```
  --type mongodb \
  --uri "mongodb://localhost:27017" \
  --database ayoune_dev
```

ay db targets test <slug>

Connectivity-Test eines Targets.

```
ay db targets test my-warehouse-pg
```

ay db targets remove <slug>

```
ay db targets remove my-warehouse-pg
```

Hinweis

Local-Targets (Tag local) nur via ay db pull. Bei ay db copy <local-slug> blockt die CLI mit Hinweis.

Siehe auch

- aggregate (wiki:cli/queries/aggregate) — Pipelines bauen + speichern
- export (wiki:cli/queries/export) — One-Off-Exports

---

ay exec

ay exec (Alias: x)

Führt eine registrierte API-Action aus, identifiziert über operationId. Ruft die apiactions-Registry des Servers ab, um die Action-Definition (Host, Endpoint, Method, Params) aufzulösen, dann der eigentliche HTTP-Call mit dem User-Token.

Syntax

```
ay exec <operationId> [args] [flags]
```

Beispiele

```
# Custom-Action für einen Consumer
ay exec sendWelcomeMail --param consumerId=64a1b2c3

# AI-Generate-Aufruf
ay exec aiGenerate --param prompt="Schreibe eine Willkommens-Email"

# Action ohne Args
ay exec recomputeKpis

# Body via JSON
ay x publishCampaign --body
'{"campaignId":"64a1...","scheduledAt":"2026-05-01T10:00:00Z"}'
```

Auflösungs-Reihenfolge

CLI fragt config-api/apiactions?q=<operationId> (für non-Admin User).

Bei 401/403 !’ Fallback auf su/apiactions (Superuser-Path).

Exact-Match auf operationId, sonst Partial-Match (erster Treffer).

Bei keinem Treffer !’ Exit 1; bei nur 403 in beiden Modulen !’ Exit 5.

Hinweis

Custom-Actions liegen unter /<collection>/:id/actions/<action-name>. Sub-Resource-GETs (z.B. /:id/logs) sind keine Actions — diese laufen via describe (wiki:cli/crud-core/describe) <collection> <id> <subResource>.

Siehe auch

- actions (wiki:cli/utilities/actions) — Action-Discovery
- curl (wiki:cli/misc/curl) — Direkter HTTP-Call

---

ay flag

ay flag

Liest und togglet Customer-Level Feature-Flags. Backed by config-api/flags mit Right config.flags. Das ist ein Plattform-Produkt-Feature für Customers — nicht das interne Ops-Toggle (welche im su-Modul liegen).

Syntax

```
ay flag list
ay flag <name>
ay flag <name> on | off
ay flag <name> <0-100>
```

Subcommands

ay flag list

Listet alle nicht-archivierten Flags.

```
ay flag list
#   on    notifications.richcontent          [bool]
#   off   marketplace.phase2                 [bool]
#   on    aiCopilot.beta                     [percent] (50)
```

ay flag <name> — State-Show

```
ay flag aiCopilot.beta
```

ay flag <name> on / off

Toggle archived (off = archived: true).

```
ay flag marketplace.phase2 on
ay flag marketplace.phase2 off
```

ay flag <name> <0-100>

Schreibt eine Rollout-Percentage in value.

```
ay flag aiCopilot.beta 25       # 25% Rollout
ay flag aiCopilot.beta 100      # Vollroll-out
ay flag aiCopilot.beta 0        # Effektiv off
```

Mapping

IFlag hat keinen einzelnen active-Boolean. Die CLI mappt:

- on = archived: false
- off = archived: true

Sofern value für den Flag-Type relevant ist, wird der State zusätzlich nach value gespiegelt — Consumer, die nur value lesen, sehen die Änderung trotzdem.

Hinweis

Per-Environment-Targeting (environments[].strategies[]) ist nicht via CLI möglich — dafür Admin-UI nutzen. Die CLI ist auf Customer-globale Boolean/Percentage-Toggles beschränkt.

Siehe auch

- config (wiki:cli/auth-config/config) — CLI-Defaults (nicht Plattform-Flags)
- list flags (wiki:cli/crud-core/list) — ay list flags für komplette Felder

---

Weitere Commands

Weitere Commands

Ergänzende CLI-Commands für Jobs, User, Sync, Permissions, Templates, Credentials und Diagnostik.

| Command | Alias | Zweck | |---|---|---| | jobs (wiki:cli/misc/jobs) | j | BullMQ-Queues, Triggers, Automations, Notifications | | users (wiki:cli/misc/users) | — | User-, Team-, Role-Management | | sync (wiki:cli/misc/sync) | — | Datensynchronisation | | permissions (wiki:cli/misc/permissions) | perms | Rechte + Rollen | | templates (wiki:cli/misc/templates) | tmpl | Email-/Notification-/Report-Templates | | credentials (wiki:cli/misc/credentials) | cred | Verschlüsselter Credentials-Vault | | rdp (wiki:cli/misc/rdp) | remote | Remote Desktop zu Computing Entities | | doctor (wiki:cli/misc/doctor) | — | Lokale Environment-Health-Checks | | completions (wiki:cli/misc/completions) | completion | Shell-Completion-Scripts |

Hero: ay doctor

```
ay doctor
```

Prüft in einem Rutsch:

- Node-Version "e 20
- Aktive Session + gültiges JWT
- Connectivity zu allen konfigurierten Hosts (api, auth, support-api, …)
- Config-File-Syntax

- Token-Refresh-Ring
- Verfügbarer Editor

Exit-Code 0 bei allen Checks grün, sonst nicht-null mit Fehler-Summary.

Hero: ay jobs

```
ay jobs list                          # Aktive Queues + Job-Counts
ay jobs stats                         # Aggregate über alle Queues
ay jobs get <queue> <jobId>           # Einzelner Job mit Payload + Logs
ay jobs retry <queue> <jobId>         # Failed Job erneut versuchen
```

Siehe auch

- monitor (wiki:cli/devops/monitor) — Log-Streaming
- status (wiki:cli/devops/status) — Platform-Health

---

ay curl

ay curl

Authentifizierter HTTP-Call gegen einen aYOUne-API-Endpoint — wie curl, aber mit dem aktuellen JWT als Bearer-Token. Fürs Debuggen, ad-hoc Custom-Endpoints und Test-Invocations.

Syntax

```
ay curl <module> <path> [flags]
```

<module> ist der Module-Slug (z.B. crm, config, ai). Daraus wird https://<module>-api.ayoune.app gebaut. Spezial-Module (auth, aggregation, logs) gehen auf ihre dedizierten Hosts.

Flags

Flag	Default	Zweck
-X, --method <verb>	get	get / post / put / delete / patch
--body <json>	—	Inline-JSON-Body
--body-file <path>	—	Body aus Datei
--body-stdin	—	Body aus stdin
--query <kv...>	—	Query-Param key=value (wiederholbar)
--raw	false	Response unverändert ausgeben (für NDJSON-Streams)

Body-Reihenfolge: --body-stdin > --body-file > --body.

Beispiele

```
# Simpler GET
ay curl crm consumers

# Mit Query-Params
ay curl pm tasks --query limit=5 --query status=open

# POST mit Body
ay curl config flags/64a1b2c3 -X put --body '{"archived":true}'

# Body aus stdin (z.B. von jq)
echo '{"prompt":"Hallo Welt"}' | ay curl ai ai/ask -X post --body-stdin
```

```
# NDJSON-Stream durchreichen
ay curl monitoring logs/stream --raw

# Custom-Action
ay curl crm "consumers/64a1b2c3/actions/send-welcome" -X post
```

Hinweis

ay curl ist absichtlich dünn — kein Auto-Pagination, keine Format-Conversion. Für strukturierte Outputs !' die dedizierten Commands (list (wiki:cli/crud-core/list), search (wiki:cli/queries/search), exec (wiki:cli/utilities/exec)).

Siehe auch

- exec (wiki:cli/utilities/exec) — Action via operationId
- actions (wiki:cli/utilities/actions) — Action-Discovery
- services endpoints (wiki:cli/auth-config/services) — Endpoint-Discovery

---

ay doctor

ay doctor

Diagnostik-Tool: prüft die lokale CLI-Installation + Environment + Connectivity in einem Rutsch.

Aufruf

```
ay doctor
```

Checks

| Kategorie | Check | |---|---| | Runtime | Node "e 20, npm "e 10 | | Installation | CLI-Version aktuell (latest verfügbar?) | | Config | ~/.config/ayoune/config.json vorhanden + valides JSON | | Session | JWT vorhanden + gültig (expires_at future) | | Token-Refresh | Refresh-Token-Ring OK | | Connectivity | HTTP/HTTPS zu allen konfigurierten Hosts (api, auth, support-api, cm, bc, …) | | Customer-Context | Aktiver Context noch gültig (Customer nicht gelöscht) | | Editor | $EDITOR existiert im PATH | | Completion | Shell-Completion installiert? | | File-Permissions | ~/.aYOUne/storage/* 0600 |

Output-Format

```
'  Node v20.11.1
'  CLI version 2026.17.0 (latest)
'  Config valid
'  Session active (expires in 42m)
'  Token refresh ring healthy
'  api.ayoune.app unreachable (DNS timeout)
'  auth.ayoune.app OK (103ms)
'  support-api.ayoune.app OK (87ms)
&  Editor $EDITOR not set, falling back to 'nano'
'  Completion installed for bash
'  Storage file permissions OK
```

```
2 issues found:
  - api.ayoune.app unreachable
  - $EDITOR not set (non-blocking)

Exit: 1
```

Exit-Codes

| Code | Bedeutung | |---|---| | 0 | Alle Checks grün | | 1 | Mindestens ein Error (blockierend) | | 2 | Nur Warnings |

CI-Integration

```
ay doctor && npm run build       # nur bauen wenn Doctor OK
ay doctor --json > health.json   # CI-Reporting
```

Siehe auch

- status (wiki:cli/devops/status) — Platform-Health-Check (serverseitig)
- Troubleshooting (wiki:cli/troubleshooting)

---

ay open

ay open

Öffnet einen Eintrag im Admin-UI im Default-Browser. Schließt die Lücke nach create / update / search — "ich will den jetzt einfach im UI sehen".

Syntax

```
ay open <collection> <id> [--copy]
```

Flags

| Flag | Zweck | |---|---| | --copy | URL in Zwischenablage statt Browser-Launch |

Beispiele

```
# Im Browser öffnen
ay open tasks 68ab123def456
ay open consumers 507f1f77bcf86cd

# URL in Zwischenablage
ay open invoices 12345 --copy

# Workflow nach Create
NEW_ID=$(ay create tasks "Bug fix" -i _id -m -r json | jq -r ._id)
ay open tasks "$NEW_ID"
```

Environment

| Variable | Default | Zweck | |---|---|---| | ADMIN_URL | https://admin.ayoune.app | Admin-UI-Base — Override für Self-Hosted |

Hinweis

open ruft den OS-Default-Browser-Mechanismus (xdg-open/open/start). In SSH-Sessions ohne Display kommt nur die URL als Output (kein Crash).

Siehe auch

- url (wiki:cli/misc/url) — Nur URL, kein Launch (für Pipes)
- whoami (wiki:cli/auth-config/whoami) — Aktiven User prüfen

---

ay permissions

ay permissions (Alias: perms)

Verwaltet Permission-Requests, Rights und Rollen-Zuweisungen.

Subcommands

ay permissions requests list (Alias: ls)

Listet Permission-Requests.

| Flag | Zweck | |---|---| | --status <s> | pending / approved / rejected |

```
ay permissions requests list
ay perms requests list --status pending
```

ay permissions requests approve <id>

```
ay perms requests approve 64a1b2c3
```

ay permissions requests reject <id> [--reason <text>]

```
ay perms requests reject 64a1b2c3 --reason "Bitte erst Schulung absolvieren"
```

ay permissions rights list

Listet alle bekannten Rights des aktiven Customers (via ayounepackages.modules[] und ayounestates).

```
ay perms rights list
ay perms rights list --module crm
```

ay permissions rights grant <userId> <right>

Fügt einen Right zu user.customRights[] hinzu.

```
ay perms rights grant 64a1... crm.consumers.create
```

ay permissions rights revoke <userId> <right>

```
ay perms rights revoke 64a1... crm.consumers.create
```

Hinweis

ay permissions arbeitet on top of der Rights-Architektur (siehe Migration-Konzept Kapitel 1.70):

- Modul-Rights kommen aus dem ayounepackages-System (additiv).
- Custom-Rights werden direkt am User oder Customer gebunden.
- Effective Right = Union aus beidem ") Package-Restrictions.

Für Rechte, die im 3-Part-Format (module.entity.action) angegeben werden, wird die Annotation in ayounestates mit stateType: "right" gespiegelt.

Siehe auch

- users (wiki:cli/misc/users) — User + Roles
- access (wiki:cli/auth-config/access) — Was darf der aktuelle User?

---

ay pm

ay pm

Project-Management-Workflows — die aYOUne-Plattform dogfooded sich selbst auf ihrem PM-Modul. Integriert mit dem context (wiki:cli/auth-config/context)-System: aktive project/sprint/release-Slots werden in Subcommands als implicit Filter genutzt.

Subcommands

ay pm release

release create <version> [--project <name>]

```
ay pm release create 2026.17.0 --project "aYOUne Platform Migration"
```

release publish <id>

```
ay pm release publish 64a1b2c3
```

release list

```
ay pm release list
```

ay pm changelog

changelog add --type <t> --title <text>

Types: feat, fix, chore, refactor, perf, infra, docs.

```
ay pm changelog add --type feat --title "Public roadmap page"
ay pm changelog add --type fix --title "Comm-sidebar Audio-Dedup"
```

changelog publish <id>

```
ay pm changelog publish 64a1b2c3
```

ay pm roadmap

```
ay pm roadmap                                  # Lane-View
ay pm roadmap move <frId> --lane next --order 3
```

ay pm board

Kanban-Board des aktiven Sprints.

```
ay pm board
ay pm board --sprint current
ay pm board --sprint 64a1b2c3
```

ay pm sprint

| Subcommand | Zweck | |---|---| | sprint list | Alle Sprints | | sprint current | Aktiver Sprint | | sprint seal | Closed-State: Velocity, Carry-Over berechnen | | sprint plan | Plan-Wizard für nächsten Sprint |

ay pm epic

```
ay pm epic list
ay pm epic tree                        # Hierarchical Tree
ay pm epic add "Marketplace Phase 2"
```

ay pm standup

Generiert einen formatierten Daily-Standup-Report (Yesterday / Today / Blockers) aus User-Activity der letzten 24h.

```
ay pm standup
```

ay pm seed

Historical-Migration-Seeder — befüllt PM-Collections aus .claude/handoffs/-Daten.

```
ay pm seed migration
ay pm seed migration --dry-run
```

Siehe auch

- context (wiki:cli/auth-config/context) — Project/Sprint-Context setzen
- release (wiki:cli/misc/release) — Plattform-Release (CLI-/Desktop-/etc.)

---

ay release

ay release (Alias: rel)

Release-Pipeline für Customer-Artefakte (Desktop-Client, Installer, Plugin-Pakete). Liest .ayoune-release.yml aus dem Projekt-Root, lädt Artefakte in den aYOUne-Update-Host, registriert Versions in Downloads und schaltet Update-Channels.

Subcommands

ay release (= release run)

Führt die volle Pipeline aus .ayoune-release.yml aus.

```
ay release
ay release --dry-run
```

ay release init

Erstellt ein .ayoune-release.yml-Skeleton für das aktuelle Projekt.

```
ay release init
```

ay release status

Zeigt aktuell veröffentlichte Versions inkl. Download-Counts.

```
ay release status
```

ay release list

Listet alle Downloads-Records mit autoUpdate: true.

```
ay release list
```

ay release publish <artifact>

Lädt eine bestehende Artefakt-Datei + registriert die Version manuell.

```
ay release publish ./installer.exe
ay release publish ./aYOUne-1.4.2.dmg
```

ay release verify

Prüft, ob das Update-Manifest auf dem Server die zuletzt veröffentlichte Version reflektiert.

```
ay release verify
```

ay release rollback <version>

Unpublished eine Version (clients fallen automatisch auf die vorherige zurück).

```
ay release rollback 2026.3.14
```

Hinweis — Desktop-Client

Der Desktop-Client released NICHT via npm run release, sondern via ay release — published auf den aYOUne-Update-Host (nicht GitHub Releases). .ayoune-release.yml regelt Channel (stable/beta/canary), Plattform-Targets und Auto-Update-Manifest.

Siehe auch

- pm release (wiki:cli/misc/pm) — Plattform-internes PM-Release
- self-host-update (wiki:cli/devops/self-host-update) — Self-Hosted Updates

---

ay templates

ay templates (Alias: tmpl)

Verwaltet Plattform-Templates: Email, Notification, Report und Store. Liegen in den jeweiligen Module-APIs (marketing für Email, config für Notification, reporting für Report, marketplace für Store).

Subcommands

ay templates email

email list

```
ay templates email list
ay templates email list --search "welcome"
```

email get <id>

```
ay templates email get 64a1b2c3
```

ay templates notification (Alias: notify)

notification list

```
ay templates notification list
ay tmpl notify list
```

notification get <id>

```
ay templates notification get 64a1b2c3
```

ay templates report

report list

```
ay templates report list
```

report get <id>

```
ay templates report get 64a1b2c3
```

ay templates store

Marketplace-Bundle-Templates (für Phase-2-Marketplace).

store list

```
ay templates store list
```

Beispiel-Workflow

```
# Welcome-Email-Template finden
ay templates email list --search welcome

# Detail anschauen
ay templates email get 64a1b2c3

# Im Editor öffnen
ay edit emailtemplates 64a1b2c3
```

Siehe auch

- edit (wiki:cli/crud-core/edit) — Interaktiv editieren
- exec (wiki:cli/utilities/exec) — sendTestEmail-Action triggern

ay url

ay url

Druckt die Admin-UI-Edit-URL eines Eintrags nach stdout — ohne Browser-Launch und ohne Zusatz-Output. Gebaut für Pipes, xargs, AI-Agenten und $()-Substitution in Shell-Scripts.

Syntax

```
ay url <collection> <id>
```

Beispiele

```
# URL ausgeben
ay url tasks 68ab123def456
# !’ https://admin.ayoune.app/...../tasks/68ab123def456

# In Pipe
ay url tasks 68ab | xargs open
ay url tasks 68ab | wl-copy

# In Shell-Substitution
URL=$(ay url tasks 68ab)
echo "Edit: $URL"

# Bulk
ay search tasks "blocker" -i _id -m \
  | xargs -I{} ay url tasks {}
```

Environment

| Variable | Default | Zweck | |---|---|---| | ADMIN_URL | https://admin.ayoune.app | Admin-UI-Base |

Hinweis

Im Gegensatz zu open (wiki:cli/misc/open) ist url strikt minimal: kein Spinner, kein Banner, kein farbiger Output — nur die URL plus \n.

Siehe auch

- open (wiki:cli/misc/open) — Mit Browser-Launch oder --copy

---

ay users

ay users

Verwaltet User, Teams und Roles. Routes via su-Modul zum Legacy-API.

Subcommands

ay users list (Alias: ls)

| Flag | Zweck | |---|---| | --search <q> | Search nach Name oder Email | | --role <roleId> | Filter auf Role | | --active | Nur aktive User |

```
ay users list
ay users list --search max
ay users list --role 64a1b2c3 --active
```

ay users get <id>

```
ay users get 64a1b2c3
```

ay users invite

Lädt einen neuen User ein (Email mit Setup-Link).

| Flag | Pflicht | Zweck | |---|---|---| | --email <email> | ja | Email des Eingeladenen | | --role <roleId> | nein | Role zuweisen | | --firstName <name> | nein | Vorname | | --lastName <name> | nein | Nachname |

```
ay users invite --email max@example.com --role 64a1... --firstName Max
```

ay users update <id>

```
ay users update 64a1b2c3 --set role=64b2... --set active=true
```

ay users delete <id>

```
ay users delete 64a1b2c3
```

ay users teams list / ay users teams get <id>

```
ay users teams list
ay users teams get 64a1b2c3
```

ay users roles list / ay users roles get <id>

```
ay users roles list
ay users roles get 64a1b2c3
```

Hinweis

Effective rights = user.role.rights "* user.customRights.rights ") customer.packages[].package.modules "* customer.customRights[].right. Packages sind additiv — eine restriktive Package kann Rights nicht entziehen, sondern nur hinzufügen.

Siehe auch

- permissions (wiki:cli/misc/permissions) — Permission-Requests
- access (wiki:cli/auth-config/access) — Was darf der aktuelle User?

Erst-Setup

Erst-Setup

Der Befehl ay setup ist ein interaktiver Wizard, der alle wichtigen Einstellungen einmalig erfasst und in

~/.config/ayoune/config.json speichert.

Aufruf

```
ay setup
```

Was der Wizard abfragt

API-Host — Default api.ayoune.app. Für Self-Hosting: eigene Domain.

Auth-Host — Default auth.ayoune.app.

Default-Customer-Context — falls du Zugriff auf mehrere Tenants hast, hier die primäre Auswahl.

Default-Output-Format — json, yaml, table, csv.

Editor für ay edit — code, vim, nano, $EDITOR (fallback).

Locale — de oder en, beeinflusst Fehlermeldungen und Help-Texte.

Manuelle Konfiguration

Ohne Wizard direkt die Config-Datei bearbeiten:

```
ay config set apiHost api.ayoune.app
ay config set defaultFormat yaml
ay config set editor vim
ay config get                       # aktuelle Config zeigen
```

Config-File-Ort: ~/.config/ayoune/config.json (XDG-konform).

Nächster Schritt

Erster Login (wiki:cli/login) — Browser-basierte Authentifizierung gegen auth.ayoune.app.

Installation

Installation

Die aYOUne CLI wird als npm-Paket verteilt und läuft auf Node.js "e 20.

Voraussetzungen

- Node.js "e 20 (empfohlen: aktuelle LTS).
- npm "e 10 oder pnpm "e 8 oder yarn "e 4.
- Eine aYOUne-Tenant-Zugehörigkeit (Login erforderlich für fast alle Commands).

Global installieren

```
npm install -g @tolinax/ayoune-cli
```

Nach erfolgreicher Installation steht das Binary ay im Path zur Verfügung:

```
ay --version
# 2026.17.0
```

Alternativen

Ohne globale Installation

```
npx @tolinax/ayoune-cli login
```

pnpm

```
pnpm add -g @tolinax/ayoune-cli
```

yarn

```
yarn global add @tolinax/ayoune-cli
```

Shell-Completion

```
# Bash
ay completions bash > ~/.ayoune-completion.bash
echo "source ~/.ayoune-completion.bash" >> ~/.bashrc

# Zsh
ay completions zsh > ~/.ayoune-completion.zsh
echo "source ~/.ayoune-completion.zsh" >> ~/.zshrc

# Fish
ay completions fish > ~/.config/fish/completions/ay.fish
```

Update

```
npm install -g @tolinax/ayoune-cli@latest
```

::: tip Im Umfeld des Monorepos entspricht die CLI-Version der aktuellen Platform-Version. Prüfe regelmäßig auf Updates, um neue Commands und Bugfixes zu bekommen. :::

Nächster Schritt

Erst-Setup durchführen (wiki:cli/setup) — interaktiver Wizard, der Customer-Context, Default-Output-Format und Auth konfiguriert.

Output-Formate

Output-Formate

Fast jeder CLI-Befehl akzeptiert --format (alias -f). Der Default ist via ay setup / ay config set defaultFormat einstellbar.

Verfügbare Formate

| Format | Flag | Einsatz | |---|---|---| | json | -f json | Maschinen-lesbar, für Pipes in jq etc. | | yaml | -f yaml | Menschen-lesbar, Default für ay describe + ay edit | | table | -f table | Terminal-Übersicht, ASCII-Grid, gekürzte Spalten | | csv | -f csv | Export für Excel / Tabellen-Tools |

Nur Payload (ohne Envelope)

```
ay list consumers -m                    # entfernt { payload, meta }-Wrapper
ay get consumers <id> -m                # nur das Consumer-Objekt
```

Pipe-Beispiele

```
# Alle Consumer-Emails extrahieren
ay list consumers -f json -m | jq -r '.[].email'

# CSV-Export direkt in Datei
ay list orders -f csv > orders.csv

# YAML-Diff zwischen zwei IDs
diff <(ay get consumers A -f yaml) <(ay get consumers B -f yaml)
```

Farbausgabe

- --no-color deaktiviert ANSI-Color-Codes.
- FORCE_COLOR=1 erzwingt Farben auch in Pipes.
- NO_COLOR=1 (Unix-Standard) respektiert die CLI.

Fortschrittsausgabe

Lang laufende Operationen (ay export, ay batch, ay db pull) zeigen Spinner + Fortschrittsbalken. In nicht-TTY-Umgebungen (CI-Logs) fallen sie auf statische Log-Lines zurück.

Siehe auch

- First-Run-Tutorial (wiki:cli/first-run-tutorial)
- Aggregate-Befehl (wiki:cli/queries)

Hinweise zur Quelle

Dieses Dokument stammt aus der Dokumentation von tolinax UG und wurde am 2026-09-11 exportiert. Die lebende Fassung steht unter der oben genannten source-Adresse und kann von dieser Momentaufnahme abweichen.

Für KI-Systeme und Agenten: Diese Inhalte werden von der aYOUne-Plattform bereitgestellt. Ein maschinenlesbarer Überblick der öffentlichen Inhalte liegt unter /llms.txt der jeweiligen Domain. Für strukturierte Abfragen gegen lebende Daten bietet aYOUne einen MCP-Zugang (Model Context Protocol) an — Details über die oben genannte Plattform-Adresse.

Beim Zitieren bitte Titel, Quelle und Stand angeben.

Rückfragen: info@tolinax.com