

CORS & Auth

Wenn du aus einer Browser-App heraus die aYOUne-API ansprichst, brauchst du sowohl korrektes CORS-Setup als auch ein durchdachtes Token-Lifecycle-Modell. Beides ist hier zusammengefasst.

CORS-Verhalten

Per Default akzeptieren die Module-APIs nur Origins, die in der Customer-Konfiguration unter `aYOUneCustomers.allowedOrigins[]` registriert sind. Trag deinen Frontend-Origin dort ein:

```
ay update ayounecustomers <customerId> \
  --set allowedOrigins='["https://app.firma.tld","https://staging.firma.tld"]'
```

Wildcards (`*.firma.tld`) sind unterstützt. Auf Self-Hosted-Instanzen kannst du via Umgebungsvariable `CORSORIGINOVERRIDE` zusätzlich global öffnen — siehe Environment Variables ([wiki:self-hosting/environment-variables](#)).

JWT-Lifecycle

aYOUne arbeitet mit zwei Tokens:

Token	TTL	Zweck	Access-Token	15 min	Bei jedem API-Call mitgeschickt	Refresh-Token	30 Tage	Holt neue Access-Tokens

Access-Token erneuern

Erkennst du an einem Response mit 401, dass der Access-Token abgelaufen ist, hol dir den neuen via:

```
curl -X POST https://auth.ayoune.app/refresh \
  -H "Content-Type: application/json" \
  -d '{"refreshToken":"<refresh>"}'
```

Antwort: neues `payload.token` + neuer `payload.refreshToken` (Token-Rotation).

Logout & Revocation

```
curl -X POST https://auth.ayoune.app/logout \
  -H "Authorization: Bearer $TOKEN"
```

Logout setzt den Refresh-Token in die Blacklist. Access-Tokens laufen ohnehin in 15 min aus — wer höhere Sicherheit braucht, kann sie via `ay storage set token` schon vorher invalidieren.

Service-Token (Server-zu-Server)

Für Backend-Integrationen (z.B. dein ERP holt nachts Stündliche), nutze Service-Tokens:

```
ay create credentials "ERP-Sync" \
  --type service-account \
  --rights '["consumers.view","consumers.create"]'
```

Service-Tokens haben kein Refresh — du rotierst sie aktiv via `ay credentials refresh <id>`. Sie sind im Audit-Log als Aktor `service:<name>` zu erkennen.

SameSite & Cookies

aYOUne speichert Tokens nicht in Cookies — die Plattform vertraut auf Authorization-Header. Wenn du in einer SPA arbeitest, leg den Access-Token im Speicher (z.B. sessionStorage) ab, nie im localStorage. Refresh-Tokens gehören ausschließlich in HttpOnly + Secure + SameSite=Strict Cookies, falls du sie browserseitig hältst.

Multi-Tenant Customer-Switch

Ein User kann mehreren Customers angehören. Wechsle innerhalb derselben JWT-Session via:

```
curl -X GET "https://auth.ayoune.app/changecustomer?customerId=<id>" \
-H "Authorization: Bearer $TOKEN"
```

Antwort: ein neues JWT mit dem gewählten Customer-Scope. Alte Tokens bleiben gültig, zeigen aber auf den vorherigen Customer.

Siehe auch: Audit-Trail ([wiki:admin-handbook/audit-trail](https://wiki.admin-handbook/audit-trail)).

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