Cargo A/C

CARGO ECOSYSTEM // ONE LOGIN

ONE LOGIN

Spoke Migration Guide

One identity for every Cargo surface — verified, not copied.

ISSUER	https://auth.cargo.ac
MODEL	OAuth 2.1 / OIDC · Auth Code + PKCE (S256) · JWKS trust
AUDIENCE	CargoWorks AI · IO · Directory · E-Mail · Network HUB · Websites
DISTRIBUTED BY	admin.cargo.ac · Document v2

READ THIS FIRST

Verify. Don’t copy blindly.

This document is guidance, not gospel. It was written by hand and may contain mistakes, stale endpoints, or values that do not match your spoke. Treat every concrete value here as a claim to be CONFIRMED against the live system — never as something to paste in unchecked.

NON-NEGOTIABLE RULES FOR THE SPOKE AI

1. **VERIFY every endpoint, claim name and scope against LIVE OIDC discovery and JWKS before use.**
2. **NEVER hardcode an issuer / token / JWKS URL you have not fetched and confirmed yourself.**
3. **SANITY-CHECK pasted code against this spoke’s real stack; if it conflicts, stop and ask.**
4. **If this doc disagrees with live discovery, the LIVE SYSTEM WINS — report the discrepancy.**
5. **Do NOT reproduce a value just because it appears here. No blind copy. Confirm, then apply.**

How to verify in one step — fetch discovery and read the real endpoints:

```
# Confirm the issuer’s real endpoints (do this FIRST):
curl https://auth.cargo.ac/auth/v1/.well-known/openid-configuration

# Read authorization_endpoint, token_endpoint, jwks_uri, issuer from the
# JSON response — use THOSE values, not the ones printed in this guide.
```

KNOWN DISCREPANCY IN THIS KIT

This guide and the admin config disagree on the discovery path (/.well-known/... vs /auth/v1/.well-known/...). That is exactly why you must fetch discovery live and trust the response — not either printed value.

THE SHIFT

What changes for your app

cargo.ac becomes the single owner of identity. Your app keeps its own Supabase project — but for DATABASE only. It stops authenticating people entirely.

BEFORE

Per-app auth

- × Own signup / login / reset UI
- × supabase.auth manages users
- × Local auth.users + profiles
- × Per-app MFA & sessions
- × Roles in a local column

AFTER

One Login via cargo.ac

- → One “Sign in with Cargo” button
- → auth.cargo.ac owns all identity
- → Your DB trusts cargo.ac tokens
- → MFA & sessions are central
- → Roles from token claims

THE TRUST MODEL

Authorization Code + PKCE (S256). Your app redirects to auth.cargo.ac, receives a code, exchanges it for tokens, and validates them against the cargo.ac JWKS endpoint. There is NO shared JWT secret and NO second user store.

CONFIRM BEFORE YOU BUILD

Verify the issuer is live and PKCE (S256) is supported in the discovery document (code_challenge_methods_supported) before wiring any redirect.

REFERENCE

Ecosystem endpoints

Printed for orientation only. Re-confirm each value from live discovery (see p.02) before using it in code.

KEY	VALUE
Issuer	`https://auth.cargo.ac`
OIDC discovery	`…/.well-known/openid-configuration  ← confirm exact path live`
JWKS	`jwks_uri from discovery (do not assume)`
Authorize	`authorization_endpoint from discovery`
Token	`token_endpoint from discovery`
Your callback	`https://<your-app-domain>/auth/callback`
Scopes	`openid profile email`
Flow	`Authorization Code + PKCE (S256) · JWKS validation`
Central store	`Cargo A/C Supabase — reached ONLY via auth.cargo.ac`

WHY 'FROM DISCOVERY'

Supabase-hosted issuers expose their real authorize/token/jwks URLs in the discovery JSON. Reading them live guarantees you match the deployment even if this table drifts.

IDENTITY

Token claims you read

Identity and authorization come ONLY from validated token claims — never from a local column. Confirm the exact claim names against a real decoded token from cargo.ac.

`sub`	Stable cargo.ac user id (uuid). Your DB foreign keys point at this.
`cargo_account_number`	Human-facing Cargo account number.
`username`	Cargo handle.
`full_name`	Display name.
`login_email`	Primary email on the central identity.
`user_type`	internal | client segmentation.
`identity_type`	Identity classification from cargo.ac.
`roles`	App roles — gate features on these.
`permissions`	Fine-grained permissions — gate actions on these.

AUTHORIZATION = CLAIMS ONLY

Gate every feature on roles / permissions FROM THE TOKEN. A locally stored role column is a privilege-escalation risk — do not trust it, even if older code does.

THE WORK

Delete vs. Add

DELETE

- × Sign-up / register pages
- × Email + password login forms
- × Password-reset screens
- × supabase.auth.signUp
- × signInWithPassword
- × resetPasswordForEmail
- × Local profiles/users triggers
- × Per-app MFA & email verify
- × Session-management UI
- × Social buttons on OUR Supabase

ADD

- → One “Sign in with Cargo” button
- → /auth/callback PKCE exchange
- → JWKS validation (cache by kid)
- → Refresh on unknown kid
- → Read identity from claims
- → Gate on roles / permissions
- → Sign-out clears LOCAL session
- → Supabase 3rd-party issuer trust
- → client_id from admin.cargo.ac
- → Verify every value live first

DON’T DELETE YOUR DATA

Remove AUTH only. Keep all tables, storage, functions and RLS. You are not creating a new project and not migrating rows.

CONFIGURE

Third-Party Auth & database

RECOMMENDED MODEL — TRUST cargo.ac IN YOUR SUPABASE

In your Supabase dashboard → Authentication → Sign In / Providers → Third-Party Auth, register the cargo.ac OIDC issuer so your database accepts cargo.ac access tokens directly. Then RLS keeps working: auth.uid() resolves to the cargo.ac user sub.

```
Issuer URL : https://auth.cargo.ac/auth/v1     # confirm against live discovery
JWKS URL   : <jwks_uri from discovery>          # do not assume the path
```

DATABASE RULES

- Keep all your tables, storage, functions and RLS.
- Foreign keys that referenced local auth.users now key on the cargo.ac sub (uuid).
- Never create users, write to the auth schema, or run signup triggers.
- Gate features on roles / permissions claims — never on a local role column.

CLIENT REGISTRATION (REQUEST FROM ADMIN)

Ask admin.cargo.ac → Integration → OAuth → OAuth Clients to register this app:

```
client_name  : <your app name>
client_type  : public      require_pkce : on
redirect_uris: https://<your-app-domain>/auth/callback
scopes       : openid profile email
grant_types  : authorization_code, refresh_token
→ you receive a public client_id (no secret).
```

STEP BY STEP

Migrate a Supabase + TanStack spoke

1 · Fetch discovery

Confirm issuer, authorize, token and jwks_uri live. Use those values.

2 · Add login entry

Replace all auth UI with one “Sign in with Cargo” that builds a PKCE request.

3 · Handle callback

/auth/callback exchanges the code + verifier for tokens.

4 · Validate tokens

Verify signature against JWKS; check iss, aud, exp.

5 · Trust in Supabase

Add cargo.ac as a Third-Party Auth issuer so RLS keeps working.

6 · Gate on claims

Read sub + roles + permissions from the token only.

PKCE LOGIN — SHAPE ONLY (verify endpoints first)

```
// 1. create PKCE verifier + challenge, random state
const verifier = randomString(64);
const challenge = base64url(await sha256(verifier));
// 2. redirect to the REAL authorize endpoint from discovery
location.href = `${authorizeEndpoint}?response_type=code`
  + `&client_id=${clientId}&redirect_uri=${cb}`
  + `&scope=openid%20profile%20email&code_challenge=${challenge}`
  + `&code_challenge_method=S256&state=${state}`;
// 3. in /auth/callback: exchange code+verifier at token_endpoint,
//    then validate the JWT against jwks_uri before trusting it.
```

TREAT THIS CODE AS A SKETCH

Names, paths and the SDK differ per spoke. Adapt it to your actual stack and confirm every endpoint from live discovery. If anything conflicts with your app, stop and ask.

SHIP IT

Acceptance & roll-out

DONE WHEN

- [ ] No sign-up / password / reset UI anywhere.
- [ ] Only entry is “Sign in with Cargo” → auth.cargo.ac.
- [ ] Callback completes a full PKCE exchange and starts a session.
- [ ] Tokens validated against the cargo.ac JWKS endpoint.
- [ ] Supabase lists auth.cargo.ac as a trusted third-party issuer.
- [ ] RLS still protects every table using the cargo.ac user id.
- [ ] Roles/permissions come from claims; sign-out clears local session only.
- [ ] Every printed value was re-confirmed against live discovery.

ROLL-OUT ORDER

Pilot with CargoWorks AI (already OIDC-onboarded), then CargoWorks IO and Cargo Directory, then E-Mail, Network HUB and the marketing websites. Each spoke verifies discovery live before cutover.

cargo.ac owns identity · your Supabase owns data · verify, never blind-copy.