

ECZ-ID API Passport Operations, Lifecycle and Troubleshooting Guide

Run the API Passport safely as endpoints, gateways, versions, evidence and organisation context change over time.

API PASSPORT · OPERATIONS

PRIMARY READER

Administrator,
API/platform
operations, security
operations, support
engineer and estate
owner.

SECONDARY READER

Developers,
architecture,
governance and service
owners.

WHEN TO READ THIS

The Passport exists and
you need to keep its
public proof aligned
with the real
API/service interface
over time.

RELEASE

2026.10 · 16 October
2026

WHAT YOU WILL KNOW AFTER READING

You can operate state, bindings, evidence and
lifecycle changes without losing canonical
identity or creating duplicates.

WHAT THIS DOCUMENT DOES NOT ESTABLISH

This guide does not replace your API runtime,
gateway, authentication or incident-management
procedures.

NEXT Use the recurring one-minute runbook and re-check Resolver after material changes.

FIRST OPERATIONAL CHECK

Know the identity before you change anything

Before modifying bindings or evidence, confirm which API ECZ-ID is the canonical record for the intended logical interface.

CONFIRM IN DASHBOARD

Locate the intended API Passport and verify the object/organisation relationship before making permitted changes.

CONFIRM IN RESOLVER

Resolve the ECZ-ID publicly and note the currently published state, bindings and evidence visible to counterparties.

OPERATING PRINCIPLE

Start from the canonical identity, then change context around it. Do not start from a URL or provider resource and mint a new ECZ-ID by default.

RECORD THE INTENDED OBJECT

Which logical API/service interface is this?

RECORD THE CHANGE SCOPE

Endpoint · gateway · description · state ·
evidence · organisation · version/environment.

DASHBOARD VS RESOLVER

Control privately; prove publicly

Operational teams need to know which surface is authoritative for customer actions and which surface is used by counterparties to verify the result.

DASHBOARD · AUTHENTICATED CONTROL

Use for permitted customer/estate actions, relationships, evidence, products/entitlements and operational views available to the account.

RESOLVER · PUBLIC PROOF

Use to inspect the bounded public record and verify what a human or machine can independently re-check.

CHANGE FLOW

Make the permitted change through the canonical customer control path → confirm current state → resolve the public record → verify the intended proof is visible.

DO NOT MUTATE THROUGH RESOLVER

Resolver is intentionally read-only.

DO NOT USE DASHBOARD AS PUBLIC PROOF

A counterparty should be able to verify through Resolver without private account access.

ENDPOINT, GATEWAY AND DESCRIPTION CHANGES

Update context without losing identity continuity

API operations commonly involve URL migration, gateway replacement, provider moves, OpenAPI updates, new versions or environment cutovers.

OPERATIONS LIFECYCLE

Preserve identity; update the context around it

Most API changes should update supported bindings, state and evidence rather than create a replacement identity.

**PRESERVE ECZ-ID WHEN**

The intended logical API remains the same while supported technical bindings change.

CONSIDER OBJECT BOUNDARY WHEN

The change creates a materially distinct logical API with separate responsibility or lifecycle.

AFTER THE CHANGE

Re-resolve and confirm the new endpoint/description/state context is published as intended before counterparties depend on it.

EVIDENCE, VERSION AND ENVIRONMENT REVIEW

Keep bounded proof current

Evidence should be reviewed according to source, scope and freshness. Environment/version context should reflect the intended operational object where supported.

REVIEW TRIGGERS

- gateway/provider migration
- new API version
- OpenAPI/description change
- production/staging cutover
- evidence expiry or replacement

REVIEW QUESTIONS

- does the binding still point to the intended object?
- is the state current?
- does evidence still support the claim?
- does version/environment match the intended target?

DO NOT EQUATE DESCRIPTION WITH RUNTIME

A current OpenAPI document may support interface context, but it does not automatically prove every runtime behaviour or control.

REFRESH

Update supported evidence and state through the canonical control path.

VERIFY

Resolve again and inspect what counterparties can now observe.

ORGANISATION AND RESPONSIBILITY CHANGES

Change the relationship carefully

If the organisation/operator relationship changes, update the supported relationship through the canonical customer control path and verify the public result.

BEFORE CHANGING

Confirm the intended API ECZ-ID and the current Parent/organisation relationship.

AFTER CHANGING

Verify the new relationship, current state and relevant evidence in Resolver.

DO NOT USE RE-ISSUANCE AS ACCOUNT RECOVERY

A change of user access, sign-in or organisation administration does not automatically mean the API itself needs a new ECZ-ID.

BOUNDARY

Responsibility context is not universal assurance or blanket authority.

AUDIT HABIT

Record when the relationship changed and re-check proof at consequential boundaries.

LIFECYCLE STATES

Make material lifecycle changes visible to counterparties

Operational truth is useful when counterparties can tell that an API is current, suspended, degraded, withdrawn or otherwise changed according to the canonical state model where supported.

CURRENT

Use when the published record represents the intended active context.

DEGRADED

Use supported state/evidence to communicate material degradation where available.

SUSPENDED

A temporary state should be visible rather than hidden behind a stale “current” assumption.

WITHDRAWN / RETIRED

Retain identity continuity where appropriate while making lifecycle status explicit.

CONSUMER EXPECTATION

A counterparty should re-resolve rather than assuming an old “good” state is still current.

RESOLVER RE-CHECK POLICY

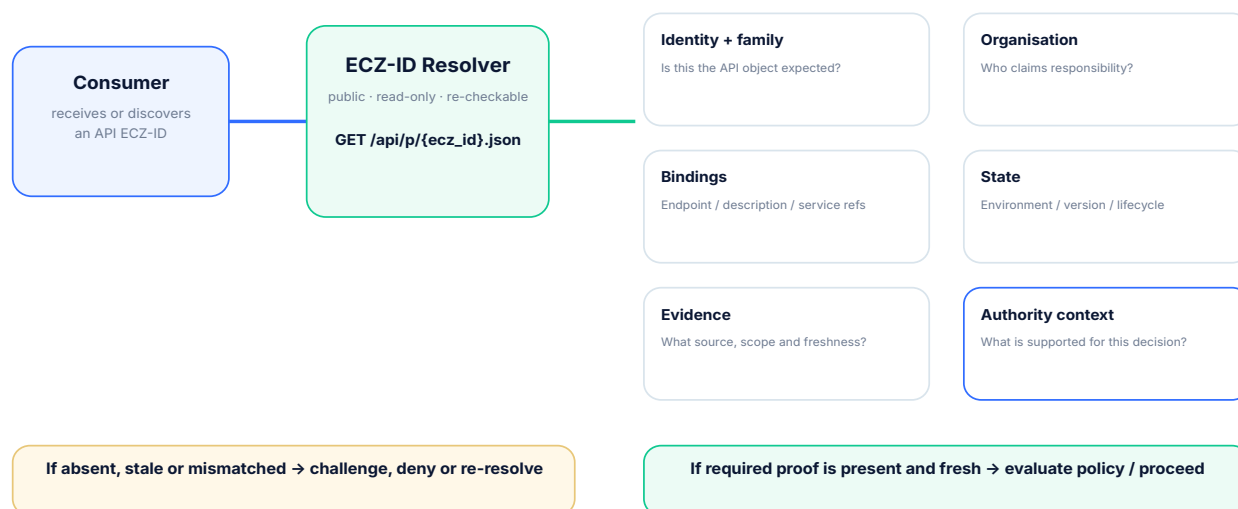
Freshness belongs in operations

Use re-resolution as an operational control whenever endpoint, version, organisation, evidence or state can change faster than your cached decision.

RESOLVER RE-CHECK

Turn a presented ECZ-ID into a bounded decision input

Resolve the current public record, inspect only what is present, and fail safely when required proof is missing.



RE-CHECK AFTER

Material configuration changes ·
provider/gateway cutovers · version changes ·
organisation changes · evidence refresh ·
lifecycle changes.

RE-CHECK BEFORE

High-consequence calls or policy decisions
where stale target identity/context would
matter.

CACHE RULE

Cache only according to use case. Re-resolve when freshness policy requires current proof.

DUPLICATE PREVENTION

One logical API, one canonical identity where supported

Duplicates create ambiguity for operators, developers, consumers and counterparties.

COMMON DUPLICATE TRIGGER

A gateway migration, new AWS subscription, sign-in issue, provider move or version change is mistaken for a need to mint another identity.

CORRECT RESPONSE

Identify the intended logical API, recover or reuse the canonical account/Parent/Passport where supported, then update bindings/state instead.

DUPLICATE CHECK

Before any new issuance, search/resolve the known ECZ-ID and determine whether the existing record represents the same intended logical API.

IF SAME OBJECT

Preserve identity; update context.

IF GENUINELY DISTINCT OBJECT

Issue a separate Passport only when the object boundary legitimately differs.

ACCOUNT RECOVERY AND ACCESS

Recover control without duplicating the estate

A sign-in or recovery problem is not evidence that the API Passport should be reissued.

ACCOUNT PROBLEM

Recover supported account access and confirm the organisation/Parent context.

PASSPORT PROBLEM

Only investigate issuance/reuse if the canonical API identity itself is missing or incorrect.

RECOVERY LAW

Restore access to the canonical estate first. Do not create replacement accounts, Parents or Passports merely to work around a sign-in failure.

AFTER RECOVERY

Confirm Dashboard shows the intended API Passport.

THEN VERIFY

Resolve the same ECZ-ID publicly and confirm the record remains consistent.

COMMON FAILURES

Diagnose by layer

Separate acquisition/account/control problems from proof/binding/freshness and API runtime problems.

WRONG API / WRONG ECZ-ID

Confirm object boundary and identifier before changing anything.

ENDPOINT BINDING STALE

Update the supported binding; do not mint a replacement identity by default.

OPENAPI/DESCRIPTION MISSING

Treat it as absent unless supported evidence exists; do not infer.

RESOLVER NOT FOUND

Check identifier and acquisition/issuance state; distinguish from network failure.

STATE/EVIDENCE STALE

Refresh through the canonical control path, then re-resolve.

API AUTH FAILURE

Diagnose OAuth/API key/mTLS/gateway policy separately from Passport identity.

ESCALATE WITH LAYER CONTEXT

Tell support whether the failure is acquisition, account access, Passport issuance/reuse, Dashboard control, Resolver proof, binding/state or API runtime/authentication.

SUPPORT ESCALATION

Send enough context - never secrets

Good support context shortens diagnosis while preserving security.

INCLUDE

- ECZ-ID / Passport ID if available
- AWS account context if relevant
- failed stage
- visible error
- occurrence time
- expected vs observed behaviour

NEVER INCLUDE

- passwords
- API keys or OAuth tokens
- AWS Marketplace registration tokens
- private keys / certificates
- other credentials or secrets

SUPPORT

support@ecocitizenz.com

If the issue affects a consequential production interaction, apply your own operational and security controls while the identity/proof issue is investigated.

ONE-MINUTE OPERATOR RUNBOOK

The recurring operational rhythm

Use this as the minimum repeatable check for an API Passport in service.

1 Identify

Confirm the canonical API ECZ-ID.

2 Inspect

Check Dashboard relationship, bindings, state and evidence.

3 Update

Apply permitted contextual changes; preserve identity when the logical API is the same.

4 Resolve

Re-check public proof after material change.

5 Challenge

If required proof is missing/stale/mismatched, do not silently infer.

6 Escalate

Send bounded context to support - never secrets.

OPERATING OUTCOME

Counterparties can continue to resolve one durable API identity while the supported endpoint, description, environment, state and evidence around it change.