

ECZ-ID SDK Passport Operations, Lifecycle and Troubleshooting Guide

Operate package identity safely over time as publisher, repository, registry,
release and evidence context changes.

SDK PASSPORT · OPERATIONS

PRIMARY READER

Developer-platform administrator, software supply chain operations, security operations, estate owner.

SECONDARY READER

Support engineers, developers, release managers and governance teams.

WHEN TO READ THIS

The SDK Passport exists and you need to operate, update, recover or troubleshoot it safely.

RELEASE

2026.10 · 16 October 2026

WHAT YOU WILL KNOW AFTER READING

You can preserve canonical identity, update distribution context, re-check Resolver proof, avoid duplicates and escalate failures without leaking secrets.

WHAT THIS DOCUMENT DOES NOT ESTABLISH

Operational changes do not automatically imply a new package identity; identity continuity must follow the actual package/release-family object boundary.

NEXT Use the Technical Reference when you need machine consumer semantics or integration detail.

FIRST OPERATIONAL CHECK

Confirm the canonical SDK/package before changing anything

Start every maintenance or troubleshooting action by identifying the intended SDK ECZ-ID and its current organisation/Parent/publisher context.

1 · IDENTIFY

Confirm SDK ECZ-ID and Passport family.

2 · INSPECT

Check Dashboard relationship, state, repository/registry/release bindings and evidence.

3 · RESOLVE

Open the same ECZ-ID in Resolver and compare public proof.

4 · SCOPE THE CHANGE

Decide whether you are changing context or the package object itself.

OPERATING LAW

Preserve the canonical identity when the intended SDK/package object remains the same; update context rather than minting a duplicate.

DASHBOARD VERSUS RESOLVER

Control privately; prove publicly

Use Dashboard for authenticated permitted actions and Resolver for public independent re-check.

DASHBOARD

Authenticated customer control:
inspect/manage permitted organisation
relationship, bindings, evidence, state and
lifecycle actions.

RESOLVER

Public, read-only proof surface for
counterparties and automation.

DO NOT CONFUSE

Do not attempt to mutate public proof through Resolver; do not expose private customer/control data merely because Resolver is public.

AFTER A CHANGE

Use Resolver to confirm the intended public
result.

BEFORE A HIGH-CONSEQUENCE DEPENDENCY DECISION

Re-resolve if freshness matters.

CHANGING REPOSITORY, REGISTRY OR DISTRIBUTION BINDINGS

Update context without fragmenting identity

Software-distribution changes are normal. Treat repository/provider/registry/release references as mutable supported context.

TYPICAL CHANGES

- repository migration or rename
- registry / namespace migration
- documentation-domain change
- distribution-provider move
- release-channel / version binding update

OPERATOR SEQUENCE

- confirm canonical SDK ECZ-ID
- update supported binding/relationship
- refresh evidence/state if needed
- re-resolve and verify public result

DO NOT MINT BY DEFAULT

A new repository, registry coordinate, provider project ID or package URL is not by itself a reason to create a new SDK Passport.

PRESERVE IDENTITY WHEN

The same intended SDK/package/release-family object persists.

RECONSIDER IDENTITY WHEN

The package object itself has genuinely changed.

RELEASE, PROVENANCE AND EVIDENCE REVIEW

Keep bounded package context current

Where supported, release/version context and evidence should be reviewed when distribution or publisher context changes or when the evidence itself becomes stale.

REVIEW TRIGGERS

- new major/minor release
- repository/registry migration
- publisher/maintainer change
- distribution or docs move
- evidence expiry or replacement

REVIEW QUESTIONS

- is this still the intended SDK/package?
- does release/binding context match current distribution?
- is evidence still in scope/fresh?
- did publisher responsibility or lifecycle state change?

DO NOT CLAIM BUILD PROVENANCE WITHOUT EVIDENCE

Repository/release context can be useful, but it is not cryptographic build provenance, artifact integrity or signature validation unless supported evidence explicitly establishes that.

REFRESH

Update supported evidence/state through the canonical control path.

VERIFY

Resolve again and inspect what counterparties can now observe.

PUBLISHER, MAINTAINER AND RESPONSIBILITY CHANGES

Change relationships carefully

Packages can move between publishers, maintainers, organisations or product teams. Preserve identity where the same SDK/package persists, while updating supported relationship/state.

BEFORE CHANGING

Confirm the intended SDK ECZ-ID and current Parent/organisation/publisher context.

AFTER CHANGING

Verify new relationship, current state and evidence in Resolver.

RESPONSIBILITY BOUNDARY

A publisher/maintainer change can coexist with one persistent SDK identity where the object remains the same. A successor or fork should not inherit identity merely because it replaces the original.

AUDIT HABIT

Record material publisher/distribution changes and re-check proof.

SECURITY HABIT

Re-evaluate package signing, registry credentials, release approvals and execution policy separately from Passport identity.

LIFECYCLE STATES

Make material package lifecycle changes visible

Operational truth is useful when counterparties can tell that an SDK/package is current, deprecated, suspended, withdrawn or archived according to supported canonical state.

CURRENT

The published record represents the intended current package/release-family context.

DEPRECATED

Use supported state/evidence to communicate deprecation where available.

SUSPENDED / WITHDRAWN

Temporary or permanent withdrawal should not be hidden behind stale "current" assumptions.

ARCHIVED / RETIRED

Retain identity continuity where appropriate while making lifecycle status explicit.

CONSUMER EXPECTATION

Counterparties should re-resolve rather than assuming an old publisher, registry, release or lifecycle state remains current.

RESOLVER RE-CHECK POLICY

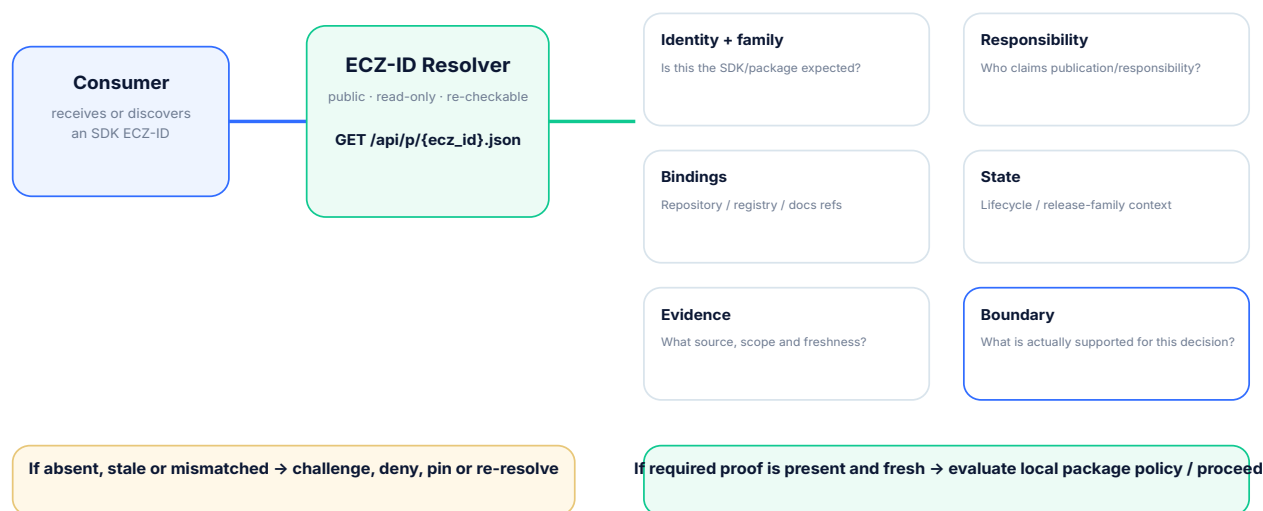
Freshness belongs in operations

Use re-resolution whenever publisher responsibility, repository/registry bindings, release context, evidence or lifecycle state can change faster than your cached decision.

RESOLVER RE-CHECK

Turn a presented SDK 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

Release change · repository/registry migration · publisher change · evidence refresh · lifecycle change.

RE-CHECK BEFORE

Consequential dependency adoption/upgrade decisions where stale identity/context would matter.

CACHE RULE

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

DUPLICATE PREVENTION

One package object, one canonical identity where supported

Duplicates create ambiguity for developers, security teams, counterparties and audit trails.

COMMON DUPLICATE TRIGGER

A new repository, registry namespace, distribution provider, AWS subscription or sign-in issue is mistaken for a need to mint another identity.

CORRECT RESPONSE

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

DUPLICATE CHECK

Before new issuance, resolve or search the known ECZ-ID and determine whether the existing record represents the same intended package/release-family object.

SAME PACKAGE

Preserve identity; update distribution context.

FORK/SUCCESSOR/DIFFERENT PACKAGE

Issue a distinct Passport when the object boundary genuinely differs.

ACCOUNT RECOVERY AND CUSTOMER ACCESS

Recover control without duplicating the package estate

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

ACCOUNT PROBLEM

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

PASSPORT PROBLEM

Investigate issuance/reuse only if the canonical package 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 SDK 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 package-security/runtime problems.

WRONG PACKAGE / ECZ-ID

Confirm object boundary and identifier before changing anything.

BINDING STALE

Update the supported repository/registry/release binding; do not mint by default.

PROVENANCE/EVIDENCE MISSING

Treat as absent unless supported evidence exists; do not infer signature/build integrity.

RESOLVER NOT FOUND

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

STATE/EVIDENCE STALE

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

PACKAGE SECURITY FAILURE

Diagnose signature, vulnerability, licence, registry-authentication or runtime-policy issues separately.

ESCALATE WITH LAYER CONTEXT

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

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
- package-registry access tokens
- AWS Marketplace registration tokens
- private signing keys / certificates
- other credentials or secrets

SUPPORT

support@ecocitizenz.com

If the issue affects a consequential production dependency decision, apply your own package-security, change-control and runtime policies while the identity/proof issue is investigated.

ONE-MINUTE OPERATOR RUNBOOK

The recurring operational rhythm

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

1 Identify

Confirm the canonical SDK ECZ-ID.

2 Inspect

Check Dashboard responsibility, bindings, state and evidence.

3 Update

Apply permitted contextual changes; preserve identity when the package 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 SDK/package identity while supported repository, registry, publisher, release, state and evidence context changes.