

ECZ-ID SDK Passport Technical Reference

Technical semantics for package identity, publisher responsibility, bindings, release state, evidence, Resolver representation and integration.

SDK PASSPORT · TECHNICAL REFERENCE

PRIMARY READER

Developer, architect,
security engineer,
integration engineer.

SECONDARY READER

Platform engineers,
automated-system
designers, software
supply chain teams and
technical evaluators.

WHEN TO READ THIS

You understand the
product and need to
integrate or interpret
SDK Passport proof
precisely.

RELEASE

2026.10 · 16 October
2026

WHAT YOU WILL KNOW AFTER READING

You can identify the canonical object, supported
relationships/bindings, state/evidence semantics,
machine route, lifecycle and safe failure
behaviour.

WHAT THIS DOCUMENT DOES NOT ESTABLISH

This is not an internal schema dump and does not
promise fields or evidence that the current
canonical implementation does not expose.

NEXT Read the Operations Guide for lifecycle changes, stale context, duplicate prevention and troubleshooting.

CANONICAL OBJECT

What exactly does an SDK Passport represent?

The canonical object is an SDK / software package / release family - not the repository URL, registry coordinate, version string, download URL or credential used to access it.

CANONICAL IDENTITY

Persistent ECZ-ID representing the intended SDK/package object across supported context changes.

EXTERNAL REFERENCES

Repository, registry, namespace, version/release, documentation and provider IDs are bindings/context.

OBJECT BOUNDARY

A new package version does not automatically mean a new SDK identity; a genuinely different package/release-family object should not inherit the same identity merely because it replaces the old one.

STABLE

ECZ-ID where object continuity holds.

MUTABLE

Bindings, release/version context, publisher relationship, state and evidence.

IDENTITY GRAPH

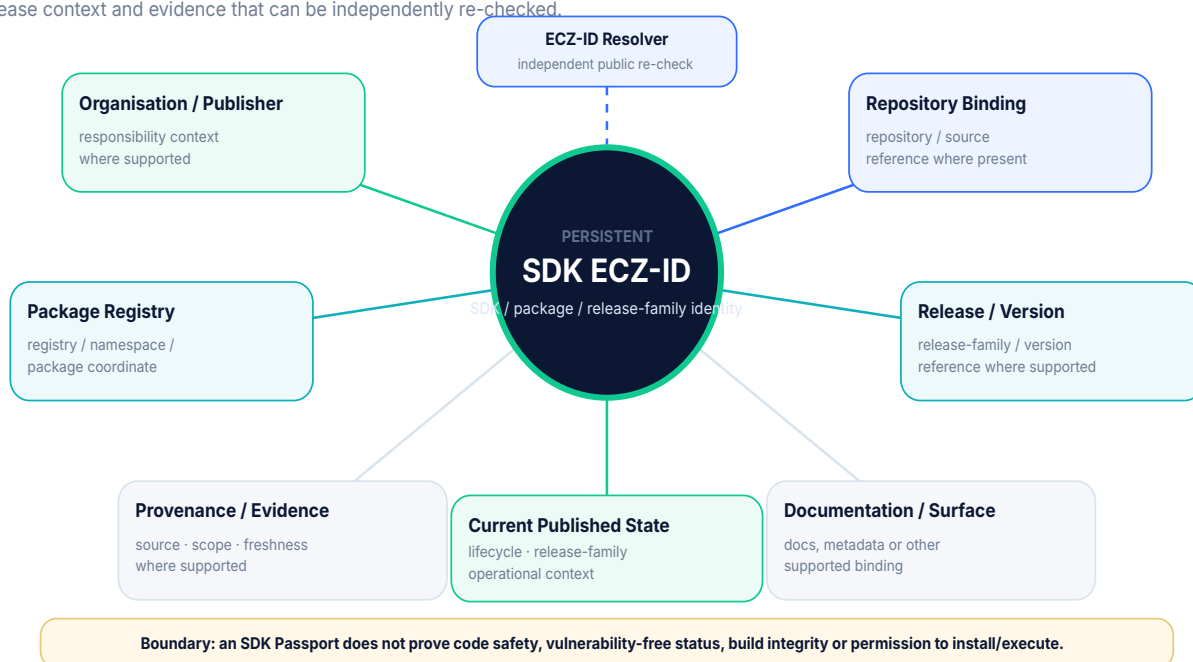
The SDK ECZ-ID at the centre of bounded context

The graph should be read as relationships and evidence around a persistent identity – not as a universal trust score.

SDK PASSPORT PROOF MODEL

What an SDK Passport can prove and contextualise

A persistent package identity with bounded publisher responsibility, repository/registry bindings, release context and evidence that can be independently re-checked.



GRAPH READING LAW

Each edge has its own scope. A repository binding does not prove publisher authority; publisher responsibility does not prove code safety; evidence does not become a blanket “trusted” verdict.

MACHINE CONSUMER

Inspect only fields/relationships relevant to the decision.

HUMAN REVIEWER

Read source, scope and freshness before drawing a conclusion.

PARENT, ORGANISATION AND PUBLISHER RELATION

Responsibility is represented as context, not universal assurance

Where present, a Parent/organisation relationship or publisher context links the SDK object to the organisation claiming responsibility for it.

CAN ESTABLISH

The record publishes the shown organisation/Parent or publisher responsibility context.

CANNOT ESTABLISH BY ITSELF

That every release, dependency or third-party statement has been independently verified.

AUTHORITY NOTE

Supported authority/delegation may be represented where present, but identity or publisher responsibility is not blanket authority for every package action or release decision.

USE RELATIONSHIP

As bounded accountability context.

USE EVIDENCE

To understand support, source and scope.

RE-CHECK

When publisher/organisation responsibility is material.

DO NOT INFER

Certification, code safety or release integrity beyond evidence.

BINDINGS

How the persistent identity connects to a changing software supply chain

Bindings can associate the SDK ECZ-ID with supported external surfaces and identifiers.

REPOSITORY

Source/repository reference.

REGISTRY / NAMESPACE

Package-registry coordinate or namespace.

RELEASE / VERSION

Release-family or version reference where supported.

DOCUMENTATION

Documentation or metadata surface.

PROVIDER ECOSYSTEM

Provider-native package/project identifiers as context.

OTHER SUPPORTED BINDING

Additional package/distribution references represented by the canonical system.

SEMANTICS

A binding says the ECZ-ID record is associated with the external surface according to current evidence/state. It does not make that surface the identity authority root.

RELEASE, VERSION AND CURRENT STATE

Separate persistent identity from mutable package context

Where supported, current state can carry point-in-time lifecycle and release-family context while the SDK ECZ-ID remains stable.

STATE MAY CHANGE

Release activity, deprecation/withdrawal context, publisher relationship, repository/registry bindings and evidence may change over time.

IDENTITY MAY PERSIST

The same intended SDK/package object can retain the ECZ-ID across those supported contextual changes.

FRESHNESS

Do not cache a package trust conclusion indefinitely. Re-resolve when current release/publisher/binding context matters.

CONSUMER PATTERN

Read current state and evidence before a consequential dependency or upgrade decision.

BOUNDARY

Current state is not a guarantee of code behaviour, vulnerability status or future safety.

EVIDENCE AND PROVENANCE

Evidence supports bounded claims - it is not magic provenance

Evidence can support claims about package identity, responsibility, bindings, release context and other supported relationships according to source, scope and freshness.

REPOSITORY EVIDENCE

Can support association to a source repository where represented.

REGISTRY EVIDENCE

Can support association to a package-registry coordinate.

RELEASE EVIDENCE

Can support bounded release/version relationships where represented.

PUBLISHER EVIDENCE

Can support responsibility/publisher context where represented.

PROVENANCE BOUNDARY

Do not infer signed-build provenance, reproducibility, artifact integrity, SBOM completeness or cryptographic attestation unless the specific evidence explicitly supports it.

AUTHORITY AND ADOPTION BOUNDARY

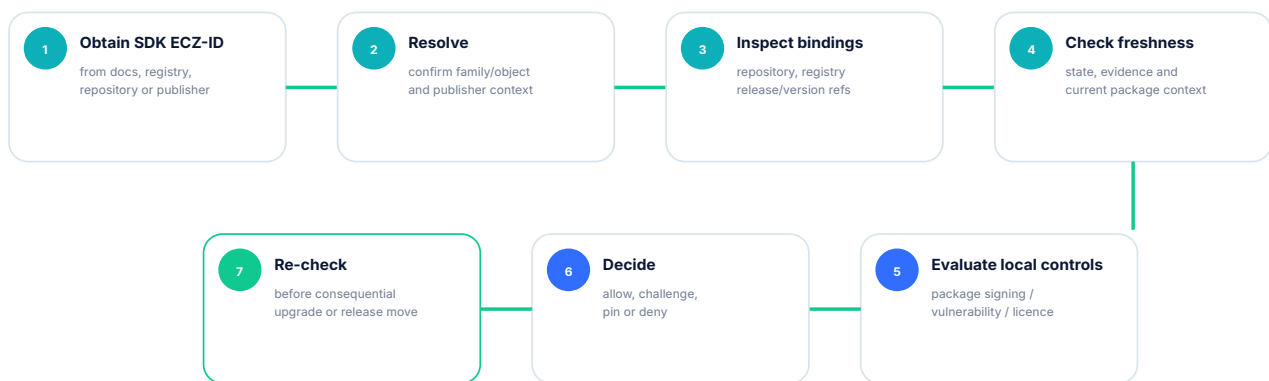
Identity is not permission to install or execute code

Some interactions require separate evidence/policy that a package may be adopted, upgraded, distributed or executed in a particular environment.

PRE-USE VERIFICATION

Verify identity and context before you adopt an SDK

Use Passport proof as one input; keep package authentication, signature verification, vulnerability review, licensing and runtime policy separate.



Boundary: SDK identity/proof does not replace artifact verification, vulnerability management, licence review or execution policy.

If required proof is absent, stale or mismatched, challenge or deny rather than infer.

DECISION BOUNDARY

Evaluate package signing/authentication, vulnerability status, licence policy, approved versions, change control and runtime restrictions separately from SDK identity.

PASSPORT CONTRIBUTION

Which package is this and what bounded publisher/distribution context is current?

LOCAL CONTROLS

Is this exact artifact/version permitted and safe enough for this environment?

RESOLVER REPRESENTATION

Public proof and machine-readable resolution

Resolver is the public, read-only proof surface. It is designed for independent re-check by humans and machines.

HUMAN USE

Inspect ECZ-ID, family/object, organisation/Parent/publisher context, current state, bindings and evidence where present.

MACHINE USE

GET
https://api.ecocitizenz.com/api/p/{ecz_id}.json
and evaluate only the returned fields relevant to policy.

READ-ONLY

Resolver does not mutate the SDK Passport. Changes flow through the authenticated/canonical control path and become visible when the public record is updated.

RE-CHECK

Use Resolver again when publisher, registry, release or evidence freshness matters.

NOT FOUND / ABSENT

Treat required missing data as not established; do not fabricate a fallback package identity.

MACHINE-CONSUMER CONTRACT

A safe integration pattern

Treat Resolver output as structured identity/proof context, not as an all-purpose “trust = true” signal.

1 · PARSE IDENTITY

Read the stable ECZ-ID and confirm SDK Passport family/object.

2 · CHECK RESPONSIBILITY

Evaluate organisation/Parent/publisher context if policy requires it.

3 · CHECK BINDINGS

Compare repository/registry/release references relevant to the dependency.

4 · CHECK STATE/EVIDENCE

Evaluate required current state, evidence scope and freshness.

5 · APPLY LOCAL CONTROLS

Signature, vulnerability, licence, version and execution policy remain separate.

6 · FAIL SAFELY

If required proof is missing, stale or mismatched, challenge/deny/pin/re-resolve.

NO UNIVERSAL SCORE

Do not reduce the record to a single “trusted package” boolean unless your own policy defines and justifies that decision from specific fields/evidence.

LIFECYCLE AND RELEASE-FAMILY CONTINUITY

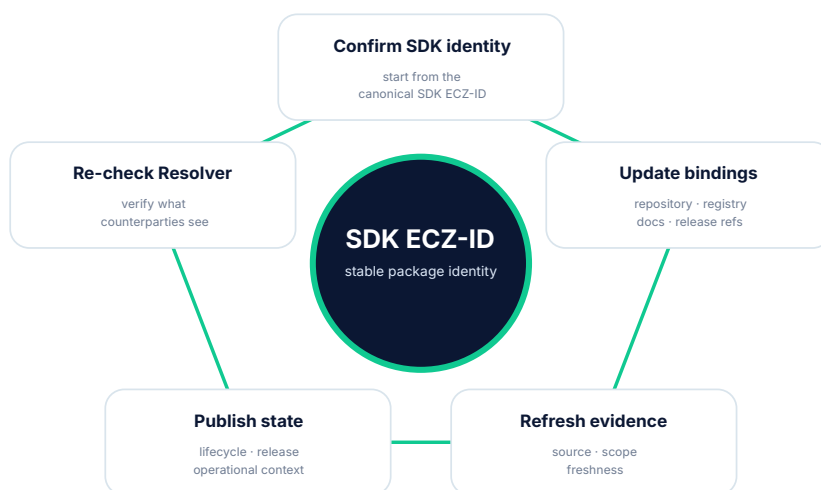
Keep the identity meaningful as distribution changes

An SDK/package can move through publication, active maintenance, major releases, deprecation, repository/registry migration, withdrawal and archival while retaining identity continuity where the object remains the same.

OPERATIONS LIFECYCLE

Operate one durable SDK identity through change

Update supported distribution context and evidence without fragmenting the canonical package identity.



Repository, registry, docs and release changes update context; they do not automatically require a new SDK ECZ-ID.

OBJECT CONTINUITY

Update supported bindings/state/evidence when distribution context changes. Mint a distinct identity when the package object/release-family boundary genuinely differs.

PRESERVE

Same intended SDK/package object; context changes.

SEPARATE

Different package/object masquerading as replacement or successor.

PROVIDER-SPECIFIC BINDINGS

Package platforms are context, not canonical authority

Provider identifiers can be useful for navigation and proof, but ECZ-ID remains the canonical identity layer.

EXAMPLES OF PROVIDER CONTEXT

Package-registry coordinates, repository/provider project IDs, documentation URLs, artifact-host references and release metadata where supported.

ARCHITECTURE RULE

Provider-native identifiers become bindings/evidence/relationships - never the ECZ-ID authority root.

PORTABILITY

The SDK ECZ-ID can remain referenceable even if distribution moves between repositories, registries or providers.

USEFUL FOR

Cross-provider dependency inventories, verification and migration.

STILL REQUIRED

Provider/package authentication and local software-supply-chain controls.

FAILURE HANDLING

Treat absence and ambiguity explicitly

Safe integrations distinguish not-found, missing-evidence, stale-context, wrong-object and transport/client errors.

NOT FOUND

Confirm ECZ-ID and distinguish missing record from network/client failure.

WRONG FAMILY/OBJECT

Do not treat another Passport family or package as equivalent.

BINDING MISMATCH

Challenge if repository/registry/release reference does not match the intended package.

EVIDENCE ABSENT

Treat required provenance/context as not established.

STATE STALE

Re-resolve or require fresh evidence according to policy.

PACKAGE SECURITY FAILURE

Signature/vulnerability/licence/runtime failure remains a separate control outcome.

FAIL CLOSED WHEN REQUIRED

If your decision requires a field/evidence and it is absent, stale or mismatched, do not infer it away.

WORKED INTEGRATION EXAMPLES

Apply the model to real package-distribution patterns

Examples show how the same persistent identity can support different operational contexts without claiming more than the record proves.

REGISTRY MIGRATION

Same SDK ECZ-ID; supported registry binding updates.

REPOSITORY MOVE

Same package identity; repository/source binding changes.

MAJOR RELEASE

Same release-family identity where object continuity is intended; version context updates.

ENTERPRISE DEPENDENCY GATE

Resolve identity, publisher/bindings/state/evidence, then apply local signature/vulnerability/licence policy.

MIRROR / SECONDARY REGISTRY

Registry mirror is a binding/context; canonical identity remains ECZ-ID.

PACKAGE SUCCESSOR

A genuinely different package should receive its own identity even if it replaces the original in a product stack.

EXAMPLE LAW

Examples illustrate the identity model; they do not promise every binding or evidence type is always present.

DEVELOPER QUICK REFERENCE

The external integration contract in one page

Use this page when implementing a verifier or integration around SDK Passport proof.

OBJECT

SDK / software package / release family.

STABLE IDENTIFIER

ECZ-ID.

MACHINE ROUTE

GET
https://api.ecocitizenz.com/api/p/{ecz_id}.json

KEY CONTEXT

Parent/publisher · state ·
repository/registry/release bindings · evidence.

MUST NOT INFER

Code safety · vulnerability-free status · build
provenance · licence approval · install/execute
authority.

FAILURE RULE

Required absent/stale/mismatched proof →
challenge, deny, pin or re-resolve.

NEXT

Read the Operations, Lifecycle and Troubleshooting Guide before implementing package-identity changes
or production support workflows.