

ECZ-ID SDK Passport Overview and Customer Guide

Persistent, resolver-verifiable identity for SDKs, software packages and release families across changing repositories, registries and versions.

SDK PASSPORT · OVERVIEW

PRIMARY READER

Buyer, technical evaluator, security architect, developer platform owner.

SECONDARY READER

Developers, enterprise architects, software supply chain, governance and procurement.

WHEN TO READ THIS

You are deciding whether an SDK/package needs durable identity and independently re-checkable publisher/distribution context.

RELEASE

2026.10 · 16 October 2026

WHAT YOU WILL KNOW AFTER READING

You can explain the package object, publisher responsibility, repository/registry/release bindings, state, evidence and safe verification boundary.

WHAT THIS DOCUMENT DOES NOT ESTABLISH

An SDK Passport does not prove code safety, vulnerability-free status, build integrity, licence compliance or permission to install/execute.

NEXT Read the AWS Getting Started Guide when you are ready to acquire and activate the free SDK Passport.

EXECUTIVE EXPLANATION

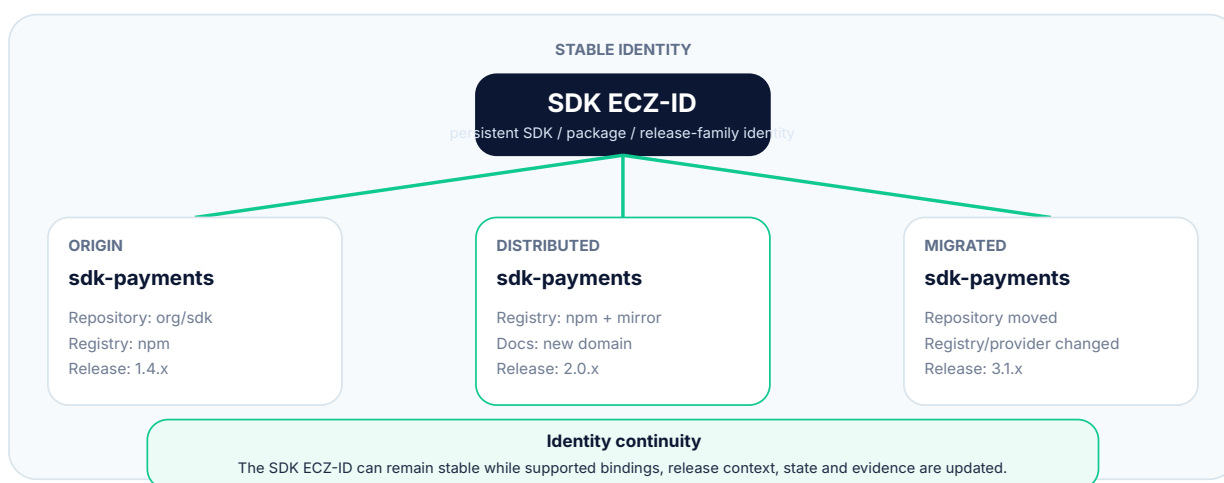
A durable identity anchor for SDKs that outlives one repository or registry

SDK Passport separates the identity of the intended SDK/package/release family from current repository, registry, documentation and release context.

PERSISTENT SDK IDENTITY

One SDK ECZ-ID; changing distribution context

Keep the canonical SDK/package identity stable while supported repository, registry, release and evidence context evolves.



WHY THIS MATTERS

A package name, repository URL or registry coordinate can be useful, but none alone necessarily provides a durable, provider-neutral identity that counterparties can independently re-check.

WHAT STAYS DURABLE

The SDK ECZ-ID for the intended package/release family where continuity is appropriate.

WHAT CAN CHANGE

Repository, registry, publisher metadata, documentation, release/version context and evidence.

SDK PASSPORT IN ONE VIEW

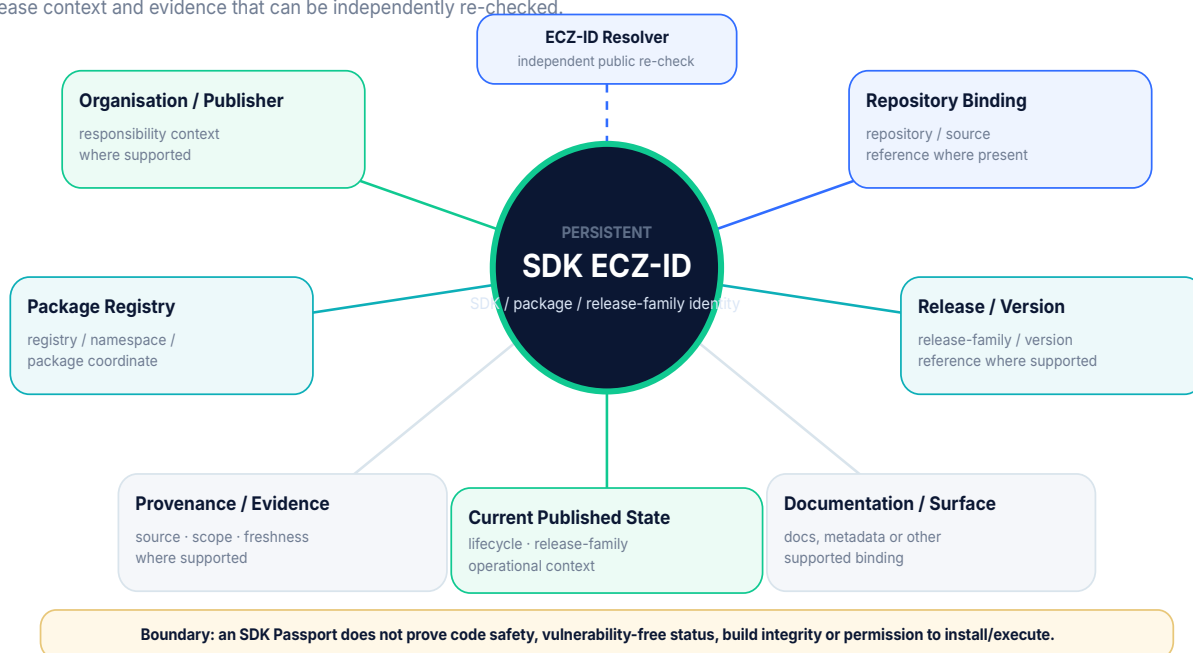
Identity at the centre; publisher, distribution and evidence around it

The Passport can connect a persistent SDK ECZ-ID to publisher/organisation context, repository/registry bindings, release context, current state and evidence where present.

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.



INTERPRETATION BOUNDARY

Do not infer code safety, vulnerability-free status, cryptographic provenance, build integrity or permission to execute unless the relevant evidence and policy explicitly establish them.

RESOLVER

Public, read-only proof a human or machine can independently re-check.

DASHBOARD

Authenticated customer control for permitted lifecycle and estate actions.

THE CUSTOMER PROBLEM

Package identity fragments across repositories, registries and releases

SDKs can move repository, registry, namespace, distribution provider and documentation surface. Versions change continuously while the underlying package/release family may remain the same object.

WITHOUT A PERSISTENT IDENTITY

Teams may rely on a package name, repository URL, registry coordinate or download location - each tied to one distribution context and potentially mutable.

WITH SDK PASSPORT

The package can present a stable ECZ-ID with bounded publisher responsibility, current state, bindings and evidence that counterparties can resolve again.

CORE QUESTION

Which SDK/package is this, who publishes or claims responsibility for it, what repository/release bindings support that identity, and can I verify it again?

SDK Passport complements - rather than replaces - package signing, software-composition analysis, vulnerability management, licence review, repository controls and runtime policy.

PUBLISHER AND ORGANISATION RESPONSIBILITY

Know who stands behind the package - without overclaiming

Where supported, the canonical record can represent organisation/Parent responsibility and publisher context for the SDK/package object.

RESPONSIBILITY CAN ANSWER

Which organisation claims responsibility for publishing or operating this SDK identity?

RESPONSIBILITY DOES NOT MEAN

Every release, dependency, security property or third-party claim has been independently verified.

SAFE INTERPRETATION

Read the published relationship and its supporting evidence. Do not promote a responsibility relationship into universal assurance.

PUBLISHER CHANGES

A maintainer or publisher relationship can change while the canonical package identity remains stable where appropriate.

PARENT CONTEXT

Links the SDK Passport to the responsible organisation/Parent where present.

EVIDENCE

Supports bounded claims according to source, scope and freshness.

RE-CHECK

Resolve again when publisher responsibility matters to the decision.

REPOSITORY, REGISTRY AND RELEASE BINDINGS

Stable identity; changing distribution surfaces

Bindings connect the canonical SDK ECZ-ID to supported external references such as repositories, package registries, namespaces, documentation and releases.

REPOSITORY

Source repository or project reference where supported.

PACKAGE REGISTRY

Registry / namespace / package coordinate.

RELEASE / VERSION

Release-family or version reference where supported.

DOCUMENTATION

Official documentation or metadata surface where supported.

PROVIDER / ECOSYSTEM

Distribution provider or ecosystem-specific identifier as context.

EVIDENCE LINK

Reference that supports a bounded claim about the package identity.

BINDING LAW

External package identifiers and URLs are bindings/context. They do not replace ECZ-ID as the canonical identity authority root.

RELEASE STATE, EVIDENCE AND FRESHNESS

Current proof should be treated as point-in-time

SDK identity may persist while versions, release channels, registry locations, publisher metadata and evidence change.

IDENTITY PERSISTS

The ECZ-ID can remain stable across supported repository, registry and release changes when the underlying SDK/package object remains the same.

CONTEXT CHANGES

Published release state, bindings and evidence should be re-checked when freshness matters.

PROVENANCE BOUNDARY

Repository or release evidence may support provenance/context, but do not infer cryptographic build provenance, signature validity or reproducibility unless the record and evidence explicitly establish it.

GOOD CONSUMER BEHAVIOUR

Read source, scope and freshness; re-resolve before consequential adoption or upgrade decisions.

UNSAFE SHORTCUT

Assuming an old release binding or publisher context is still current because the ECZ-ID itself is unchanged.

IDENTITY IS NOT CODE SAFETY OR INSTALL AUTHORITY

Use the Passport before the consequence - not instead of software-supply-chain controls

SDK Passport can help establish which package is intended and what context supports that identity. It does not make the package safe or authorised for your environment.

IDENTITY

Which SDK/package object is this?

RESPONSIBILITY

Who claims publication/responsibility?

LOCAL POLICY

Is this package/version permitted in this environment?

SECURITY CONTROLS

Are signing, vulnerability, licence and runtime requirements satisfied?

BOUNDARY

A valid Passport is not permission to install, import or execute code. Apply package-authentication, vulnerability, licence, change-control and runtime policy separately.

HIGH-VALUE USE CASES

Where persistent, re-checkable package identity helps most

The strongest use cases are those in which distribution context changes while publisher accountability and identity continuity still matter.

REPOSITORY MIGRATION

Preserve package identity when source hosting moves.

REGISTRY MIGRATION

Keep the same ECZ-ID while supported registry/package coordinates change.

RELEASE VERIFICATION

Connect a stable package identity to current release/version context.

ENTERPRISE DEPENDENCY REVIEW

Give architecture/security teams a re-checkable identity anchor for an SDK.

MACHINE CONSUMPTION

Allow automation to resolve bounded package identity/context before dependency decisions.

CROSS-PROVIDER DISTRIBUTION

Treat provider-native package IDs as bindings rather than the authority root.

VALUE

The outcome is identity continuity plus current, independently re-checkable context - not a universal software-security score.

EXAMPLE - REGISTRY MIGRATION

One package identity through a distribution move

A publisher moves an SDK from one package registry or namespace to another while preserving the same underlying package/release family.

BEFORE

Consumers know the SDK by ECZ-ID plus repository, registry and release bindings.

AFTER

The SDK ECZ-ID remains stable; supported registry/documentation bindings and evidence are updated and Resolver reflects the current record.

WHAT CHANGED

Distribution context, not necessarily the canonical package identity.

CONSUMER ACTION

Re-resolve before adopting the new distribution location or release.

SECURITY ACTION

Validate package signatures, vulnerabilities, licence and local approval independently.

DEPLOYMENT FIT

Who should use an SDK Passport?

Adopt where persistent package identity and independent re-checkability are valuable across a changing software supply chain.

STRONG FIT

SDKs/packages distributed across repositories, registries, versions or providers where durable identity matters.

SECURITY FIT

Teams that need a stable object identity before applying separate software-supply-chain controls.

DEVELOPER FIT

Tooling that needs a machine-resolvable package identity independent of one package coordinate.

LOWER-VALUE FIT

Single-use or ephemeral package artifacts where persistent cross-context identity brings little benefit.

FREE IDENTITY PRIMITIVE

The SDK Passport itself is a free identity primitive. Paid ECZ-ID capabilities are not prerequisites for basic Passport identity and Resolver verification.

HOW TO VERIFY AN SDK PASSPORT

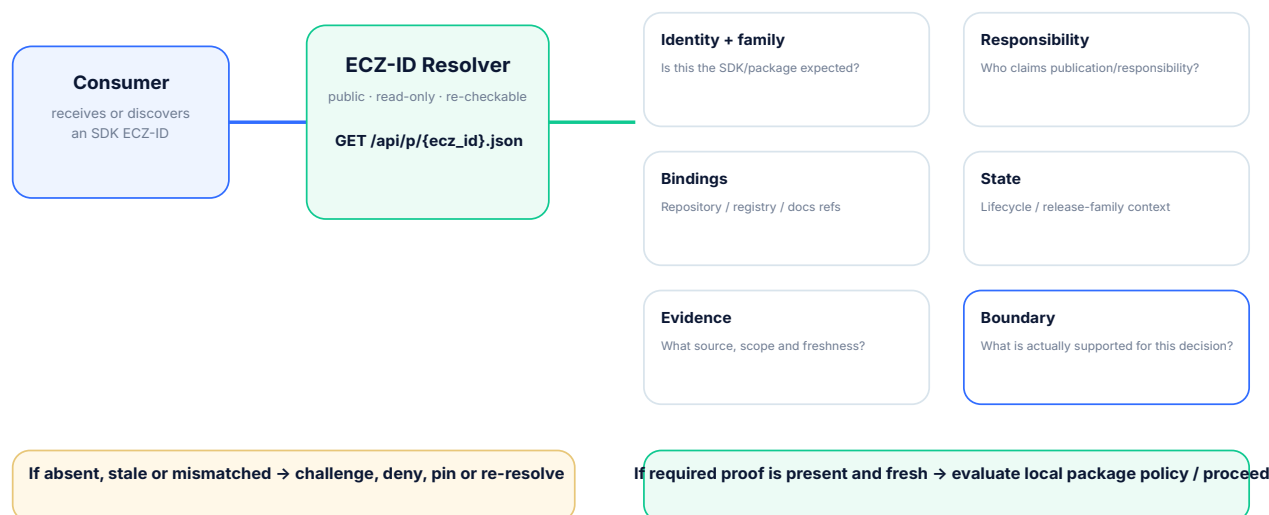
Resolve, inspect, interpret, re-check

Verification is intentionally bounded: identify the package object, inspect publisher responsibility and current distribution context, then apply your own policy for adoption.

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.



MACHINE-READABLE ROUTE

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

IF REQUIRED PROOF IS PRESENT

Evaluate the decision using the returned state, bindings and evidence plus your local package policy.

IF REQUIRED PROOF IS ABSENT

Treat it as not established; challenge, pin, deny or seek additional evidence rather than infer.

DECISION CARD

The SDK Passport in 60 seconds

Use this page as the executive summary before acquisition or architecture review.

OBJECT

SDK / software package / release family.

PERSISTENT IDENTITY

One canonical ECZ-ID can survive repository, registry and version changes where object continuity holds.

RESPONSIBILITY

Organisation/Parent and publisher context where present.

BINDINGS

Repository, registry, docs, release/version and provider references where supported.

EVIDENCE + STATE

Bounded, point-in-time context; source, scope and freshness matter.

BOUNDARY

Not code safety, vulnerability-free status, build attestation, licence approval or execution permission.

NEXT ACTION

If the model fits your software supply chain, acquire the free SDK Passport through AWS Marketplace or the supported ECZ-ID acquisition surface, then verify the resulting record in Resolver.