

## ECZ-ID API Passport Overview and Customer Guide

Persistent, resolver-verifiable identity for API and service interfaces across changing endpoints, gateways, versions and providers.

### API PASSPORT · OVERVIEW

#### PRIMARY READER

Buyer, technical evaluator, security architect, API/platform owner.

#### SECONDARY READER

Developers, enterprise architects, procurement, governance and operations.

#### WHEN TO READ THIS

You are deciding whether an API/service interface needs durable identity and independently re-checkable context.

#### RELEASE

2026.10 · 16 October 2026

#### WHAT YOU WILL KNOW AFTER READING

You can explain the API object, organisation responsibility, endpoint/description bindings, current state, evidence, authority context and verification boundary.

#### WHAT THIS DOCUMENT DOES NOT ESTABLISH

An API Passport does not authenticate callers, guarantee code safety, or grant blanket permission to invoke the API.

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

## EXECUTIVE EXPLANATION

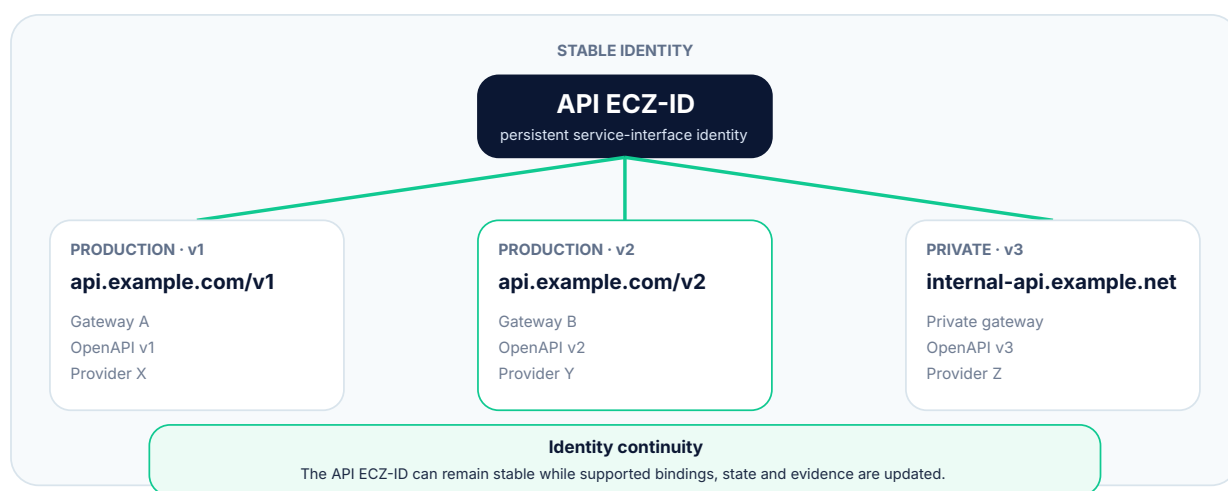
# A durable identity anchor for APIs that outlive any one endpoint

API Passport separates the identity of the intended API/service interface from its current URL, gateway, version, environment and provider context.

## PERSISTENT API IDENTITY

## One API ECZ-ID; changing delivery context

Keep the canonical API identity stable while endpoints, gateways, environments and versions evolve.



## WHY THIS MATTERS

A base URL tells you where to call now. The API ECZ-ID is designed to remain referenceable as the interface moves or changes, while supported bindings and evidence are updated.

## WHAT STAYS DURABLE

The API ECZ-ID for the intended logical API/service interface.

## WHAT CAN CHANGE

Base URL, gateway, environment, version, provider, OpenAPI description and evidence.

## API PASSPORT IN ONE VIEW

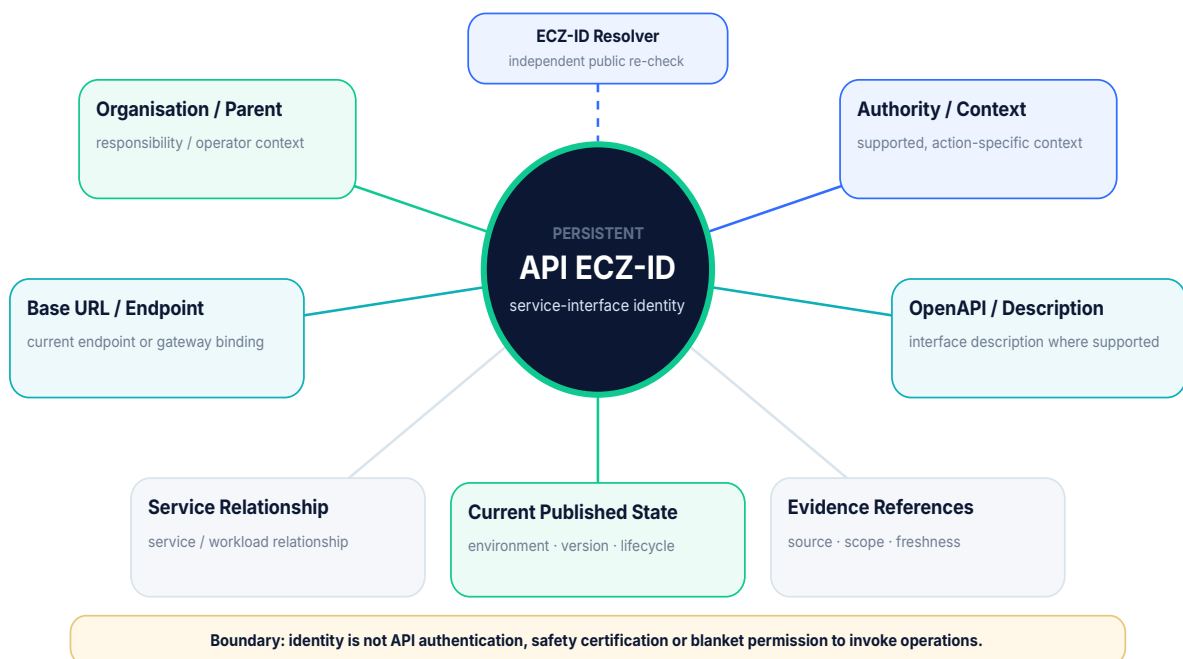
# Identity at the centre; responsibility, service context and evidence around it

The Passport can connect a persistent API ECZ-ID to organisation responsibility, service relationship, endpoint/description bindings, current state, authority context and evidence where present.

## API PASSPORT PROOF MODEL

## What an API Passport can prove and contextualise

A persistent API identity with bounded responsibility, bindings, state and evidence that can be re-checked.

**INTERPRETATION BOUNDARY**

Identity is not authentication. A published endpoint is not blanket permission to invoke it. Evidence and authority/context must be read according to source, scope and freshness.

**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

# Endpoints are discoverable; durable accountability is harder

An integration can know a URL or OpenAPI document without knowing whether it still represents the intended API, who is responsible for it, or whether the published context is current.

**WITHOUT A PERSISTENT IDENTITY**

A gateway migration, provider change, new version or environment split can make it difficult to know whether today's endpoint is the same intended API object.

Self-description is useful, but it is still only one input.

**WITH API PASSPORT**

The API can present a stable ECZ-ID, organisation context and bounded proof that counterparties can resolve again before they connect or act.

**CORE QUESTION**

Is this the API/service interface I intended to call, who operates it, what binding/evidence supports it, and is the published state current?

Persistent identity complements - rather than replaces - TLS, OAuth, API keys, mTLS, gateway policy and runtime authorization. Provider-native IDs and URLs remain bindings/context; they do not become the ECZ-ID authority root.

## RESPONSIBILITY AND OPERATOR CONTEXT

# Who claims responsibility for this API interface?

Where supported, the API record can publish a Parent/organisation relationship or operator context. That relationship is useful, but bounded.

**WHAT IT CAN ESTABLISH**

A published relationship between the API ECZ-ID and the organisation/Parent or operator context represented in the canonical record.

**WHAT IT DOES NOT ESTABLISH**

It does not automatically prove every organisational claim, operational control or assurance statement. Read the actual state/evidence present.

**CUSTOMER VALUE**

Responsibility context gives counterparties something stronger than an anonymous endpoint: a persistent object they can resolve together with the organisation claiming responsibility for it.

**WHEN THIS MATTERS MOST**

Third-party APIs · regulated workflows · internal service estates · multi-provider API programmes.

**RE-CHECK DISCIPLINE**

If responsibility can change, re-resolve before a consequential interaction.

## ENDPOINT AND DESCRIPTION BINDINGS

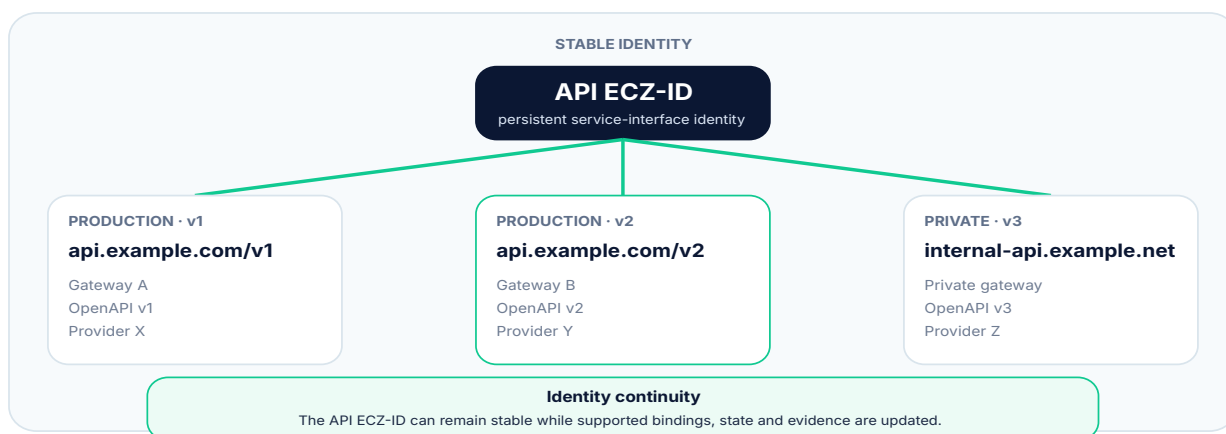
# Stable identity, changing technical surfaces

Bindings connect the canonical API ECZ-ID to supported external references such as base URLs, gateways, OpenAPI descriptions, service/workload references and provider-native identifiers.

## PERSISTENT API IDENTITY

## One API ECZ-ID; changing delivery context

Keep the canonical API identity stable while endpoints, gateways, environments and versions evolve.

**BINDING ≠ IDENTITY**

A URL, ARN, resource ID or gateway object can change. It remains an external binding/context around the canonical ECZ-ID.

**DESCRIPTION ≠ GUARANTEE**

An OpenAPI or other interface description is valuable evidence/context; it does not guarantee runtime behaviour by itself.

**GOOD OPERATING MODEL**

Preserve the API ECZ-ID when the logical interface remains the same; update the supported endpoint, gateway, environment, version and description context around it.

## STATE, ENVIRONMENT AND FRESHNESS

# Current proof should be treated as point-in-time

API state can change on a different timescale from identity: production vs non-production, version transitions, endpoint cutovers, deprecation, suspension or evidence refresh.

**IDENTITY**

The persistent API ECZ-ID answers which logical interface is being represented.

**CURRENT PUBLISHED STATE**

The point-in-time state/environment/version context exposed by the canonical system where supported.

**FRESHNESS**

Do not cache a trust conclusion indefinitely. Re-resolve when the consequence requires current proof.

**EVIDENCE**

Interpret each evidence reference according to its source, scope and freshness.

**RE-CHECK REQUIRED**

A previously valid endpoint or description can become stale. Identity continuity only helps if consumers re-check the current context when it matters.

## IDENTITY IS NOT API AUTHENTICATION

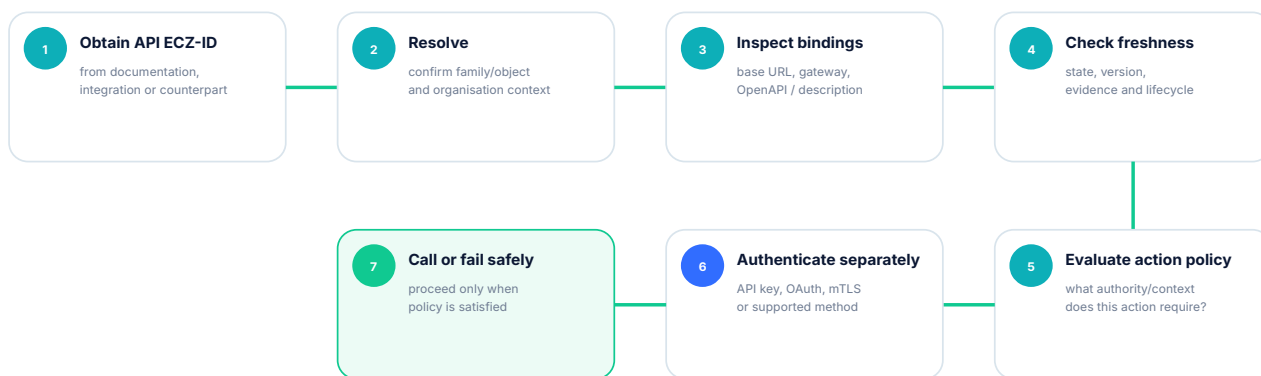
# Use the Passport before the call - not instead of credentials

API Passport can help establish which interface is intended and what context supports that identity. Authentication and runtime authorization remain separate controls.

## PRE-CALL VERIFICATION

## Verify identity and context before an API call

Use Passport proof as one input; keep API authentication and runtime authorization separate.



**Boundary: API identity and proof do not replace API authentication or authorization.**  
If required proof is absent, stale or mismatched, challenge, deny or re-resolve according to policy.

**BOUNDARY**

A valid Passport does not make an API key valid, issue an OAuth token, satisfy mTLS, or grant access to an endpoint. It is an identity/context input to the surrounding decision.

**USE PASSPORT TO ASK**

Is this the intended API, who is responsible, what endpoint/description is bound, and is proof fresh?

**USE AUTH CONTROLS TO ASK**

Is this caller authenticated and authorised for this operation right now?

## RECIPROCAL TRUST AROUND AN API CALL

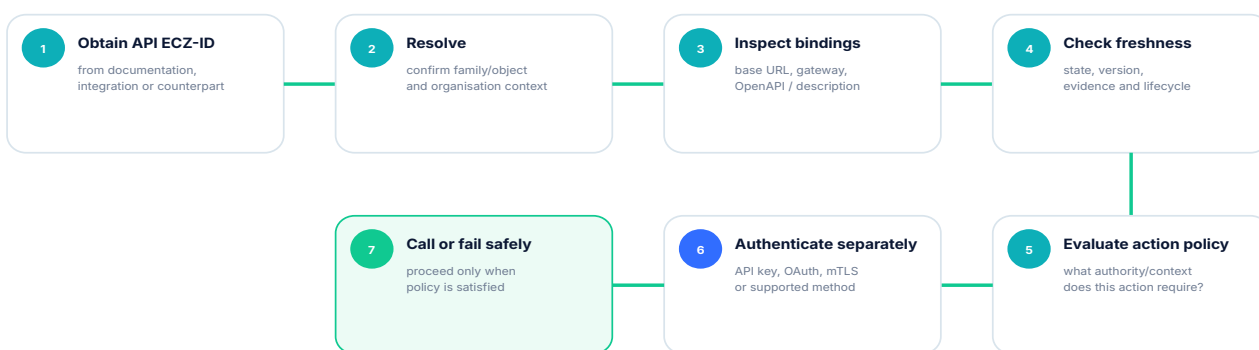
# Caller and API can each present resolvable context

A machine interaction can evaluate the API identity, the calling actor identity where available, and the relationship/evidence relevant to the intended action.

## PRE-CALL VERIFICATION

## Verify identity and context before an API call

Use Passport proof as one input; keep API authentication and runtime authorization separate.



**Boundary: API identity and proof do not replace API authentication or authorization.**

If required proof is absent, stale or mismatched, challenge, deny or re-resolve according to policy.

## API SIDE

Persistent API ECZ-ID, organisation context, endpoint/description bindings, state and evidence.

## CALLER SIDE

Agent, service, workload or organisation identity and action-specific authority/context where separately supported.

## RECIPROCAL TRUST

The decision can evaluate both relevant actors and the relationship between them instead of relying on one opaque self-assertion.

## HIGH-VALUE USE CASES

# Where API Passport adds the most practical value

The strongest use cases are those where an interface must remain identifiable across technical change or where counterparties need independent re-check.

**API GATEWAY MIGRATION**

Preserve one identity while gateway/provider bindings change.

**THIRD-PARTY INTEGRATION**

Let customers independently re-check the API identity and organisation context.

**MULTI-ENVIRONMENT ESTATES**

Separate stable identity from production/staging endpoints and versions.

**AGENT-TO-API INTERACTION**

Give an agent a resolvable target identity before a consequential API call.

**SERVICE CATALOGUES**

Reference one durable API identity across documentation and automation.

**PROVIDER PORTABILITY**

Treat cloud resource IDs as bindings rather than the canonical identity.

## WHAT API PASSPORT DOES NOT PROVE

# Boundaries are part of the product, not footnotes

A strong identity system is useful partly because it is explicit about conclusions that require separate controls or evidence.

**NOT AUTOMATICALLY ESTABLISHED**

- API safety or vulnerability-free status
- caller authentication
- permission to invoke any operation
- correct future behaviour
- standards compliance or certification

**ALSO NOT IMPLIED**

- every external claim independently verified
- every OpenAPI statement enforced at runtime
- unlimited organisation assurance
- provider endorsement
- permanent freshness of cached proof

**SAFE INTERPRETATION**

Use the Passport to identify and re-check bounded context. Use authentication, authorization, policy, runtime security and relevant evidence for the decision actually being made.

If a required field or evidence is absent, treat it as absent. Do not infer it from marketing copy, endpoint ownership or historical state.

## SHOULD YOU DEPLOY API PASSPORT?

# A quick decision card

API Passport is most relevant where an API/service interface should remain recognisable longer than any one URL, gateway, version or provider context.

**STRONG FIT**

- APIs used by multiple teams or organisations
- interfaces likely to move gateway/provider
- agent or machine consumers
- need for independent identity re-check
- governed service/API catalogues

**LESS VALUE IF**

- the interface is disposable and never referenced again
- no counterparty needs durable identity
- all decisions are entirely local and short-lived

**DEPLOYMENT FRICTION**

The Passport identity primitive is free. AWS Marketplace acquisition is designed to hand the subscriber into the canonical ECZ-ID account/Parent/Passport flow.

**FIRST OUTCOME**

A canonical API ECZ-ID you can control through Dashboard.

**VERIFICATION OUTCOME**

A public Resolver record counterparties can independently re-check.

## NEXT STEPS

# From evaluation to working proof

The free Passport is designed to move from discovery to verification without requiring a paid capability.

## 1 Acquire

Use AWS Marketplace or another supported acquisition surface.

## 2 Activate

Reuse or create the supported account and Parent, then issue or reuse API Passport.

## 3 Operate

Maintain current endpoint/description bindings, state and evidence.

## 4 Verify

Share the ECZ-ID and re-check through Resolver.

### READ NEXT

AWS subscriber → API Passport AWS Marketplace Getting Started and Technical Guide.  
Developer/integrator → API Passport Technical Reference. Operator → API Passport Operations, Lifecycle and Troubleshooting Guide.

**Support:** [support@ecocitizenz.com](mailto:support@ecocitizenz.com)