

ECZ-ID API Passport Technical Reference

External integration reference for representing, resolving and safely consuming API Passport proof.

API PASSPORT · TECHNICAL REFERENCE

PRIMARY READER

Developer, architect, security engineer, integration engineer and automated-system designer.

SECONDARY READER

Platform/API engineering, governance, operations and technical evaluators.

WHEN TO READ THIS

You understand the product and need the technical object model, binding semantics and machine-safe verification pattern.

RELEASE

2026.10 · 16 October 2026

WHAT YOU WILL KNOW AFTER READING

You can integrate API ECZ-ID proof without confusing identity with authentication, endpoint discovery, authority or runtime security.

WHAT THIS DOCUMENT DOES NOT ESTABLISH

This is not an internal schema dump and does not invent fields that are absent from the current canonical implementation.

NEXT Read the Operations Guide after integration so the identity, bindings, state and evidence remain current over time.

CANONICAL OBJECT

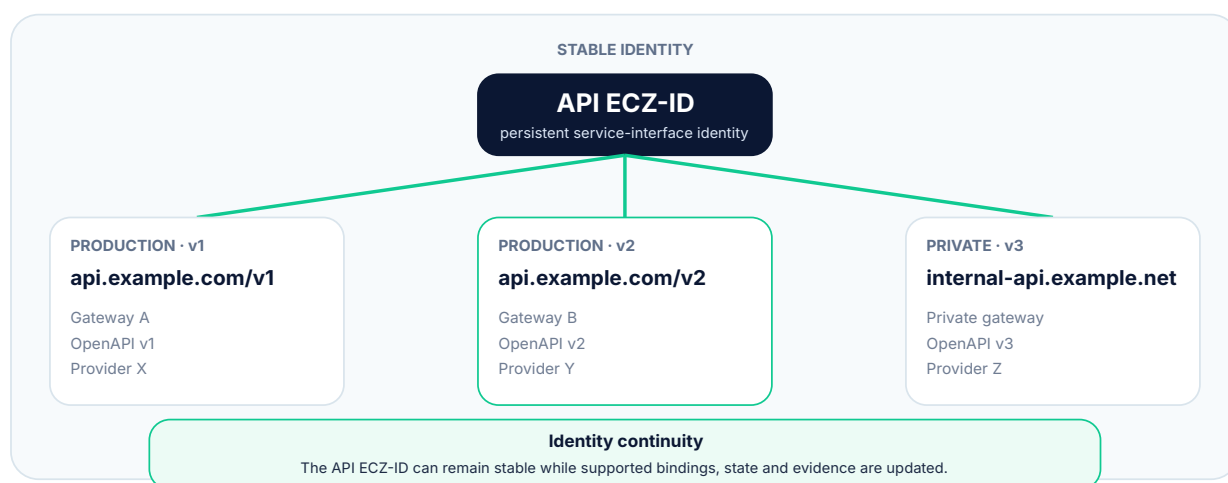
What the API ECZ-ID represents

An API Passport represents the intended logical API/service interface - not merely one transient base URL, gateway route, cloud resource ID or deployment.

PERSISTENT API IDENTITY

One API ECZ-ID; changing delivery context

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



CANONICAL IDENTITY

The persistent API ECZ-ID for the logical interface being represented.

EXTERNAL SURFACES

URLs, gateways, provider-native IDs, descriptions and environments remain bindings/context.

IMPLEMENTATION RULE

When infrastructure changes but the intended logical API remains the same, update supported context rather than minting another ECZ-ID merely because the endpoint moved.

PARENT AND OPERATOR RELATIONSHIP

Responsibility context is explicit and bounded

Where supported, the API record can relate the object to a Parent/organisation or operator context.

REPRESENTED RELATIONSHIP

The record may publish the organisation/Parent claiming responsibility for the API or relevant operator/service relationship where supported.

NOT UNIVERSAL ASSURANCE

The presence of that relationship does not prove every organisational assertion or runtime control.

CONSUMER RULE

Evaluate the published relationship and evidence actually present. Do not infer ownership, certification or authority that is not represented.

CHANGE HANDLING

If the responsible organisation changes, use the canonical control path and re-check the resulting public proof.

DUPLICATE HANDLING

Do not create a replacement API identity solely because organisation/account access changed.

SUPPORTED BINDING CLASSES

Connect stable identity to changing technical surfaces

Bindings are external references associated with the canonical API ECZ-ID according to available state/evidence.

ENDPOINT / BASE URL

Public or private address used to reach the API where supported.

GATEWAY / PROVIDER ID

API gateway, cloud resource or provider-native identifier as contextual binding.

OPENAPI / DESCRIPTION

Machine-readable or human-readable interface description where supported.

SERVICE / WORKLOAD

Relationship to the service or workload the API fronts or represents.

ENVIRONMENT / VERSION

Production/staging, version or release context where supported.

REPOSITORY / DOCUMENTATION

Supporting source or documentation references where appropriate.

BINDING LAW

Provider-native and endpoint identifiers remain bindings/context. They do not replace the ECZ-ID as the canonical identity authority.

STATE AND FRESHNESS

Point-in-time truth needs a freshness policy

The published state is useful only when consumers know whether it is fresh enough for the consequence they are about to allow.

STATE EXAMPLES

Environment, version, lifecycle status, availability/degradation context or other supported point-in-time state.

FRESHNESS POLICY

Choose re-resolution frequency according to how quickly relevant endpoint, version, evidence or operator context can change.

DO NOT CACHE A TRUST CONCLUSION FOREVER

A previously observed record can become stale. Cache only according to the use case; re-resolve before high-consequence interactions when current proof matters.

MISSING STATE

Treat required absent state as not established.

MISMATCHED STATE

If environment/version does not match the intended target, challenge or deny according to policy.

EVIDENCE SEMANTICS

Evidence is bounded by source, scope and freshness

Evidence can support claims about organisation context, bindings, state or other API properties. It does not become an unlimited “trusted” verdict merely by being present.

EVALUATE EACH REFERENCE

Source · scope · freshness · what claim it actually supports.

DO NOT INFER

Presence of evidence does not prove unrelated behaviour, security posture or operational correctness.

SAFE CONSUMPTION

Use evidence only for the decision it can legitimately support. If a required claim has no relevant evidence, treat that claim as not established.

OPENAPI EXAMPLE

An OpenAPI binding can support interface-description context; it does not prove every runtime behaviour matches the document.

PROVIDER-ID EXAMPLE

A cloud resource binding can support association to that provider object; it does not become canonical identity authority.

AUTHORITY AND API AUTHORIZATION

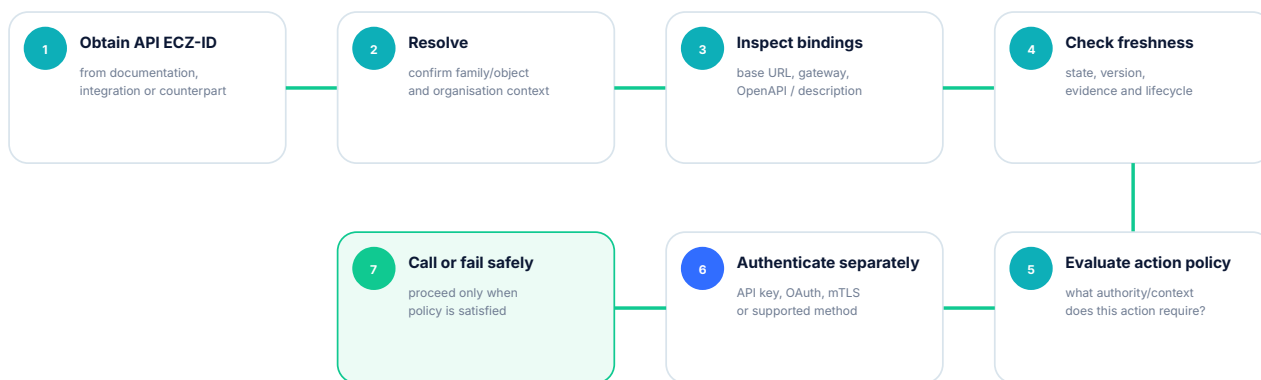
Identity is not permission to invoke an operation

API Passport can support context for an authorization decision, but it does not itself authenticate the caller or grant permission to call an endpoint.

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.

PASSPORT LAYER

Target identity, organisation, bindings, state, evidence and supported authority/context where present.

AUTHORIZATION LAYER

Caller authentication, scopes/roles, gateway policy, application policy and action-specific permission.

RULE

Evaluate authority for the specific action at the point of consequence. Never convert identity into blanket API permission.

RESOLVER REPRESENTATION

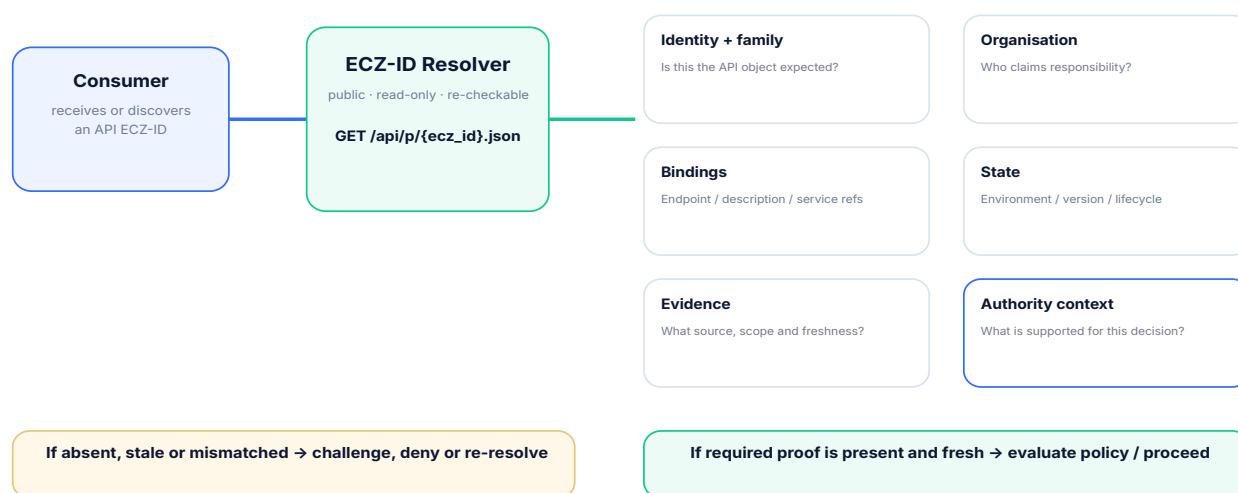
Public proof and machine resolution

Resolver is the public, read-only proof surface. A customer or machine can independently re-check the API record without using the Dashboard control plane.

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.



MACHINE-READABLE ROUTE

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

CONSUMER BEHAVIOUR

Parse only fields actually returned; check family/object and context relevant to your decision.

FAILURE BEHAVIOUR

Distinguish not-found, missing evidence, mismatch and network/client failure. Do not fabricate fallback identity.

INTEGRATION PATTERN

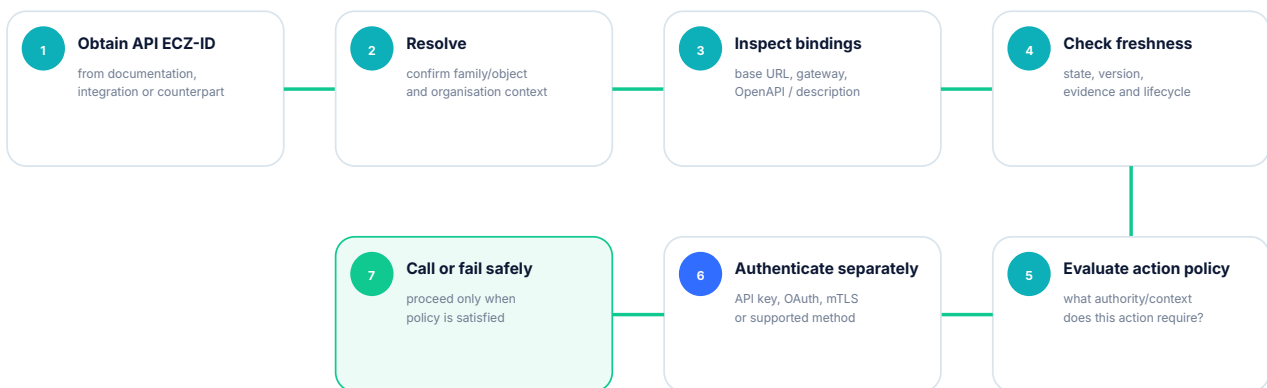
A machine-safe pre-call sequence

Use the Passport before connection or invocation when identity continuity, target context or freshness matters.

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.

REFERENCE ORDER

Resolve identity → confirm object/organisation → inspect endpoint/description bindings → check state/evidence freshness → evaluate action policy → authenticate separately → call or fail safely.

DO NOT INFER

Do not infer authentication, permission or safety from the Passport alone.

DO RE-RESOLVE

Re-resolve when endpoint, state, evidence or consequence requires current proof.

CALLER ↔ API RELATIONSHIPS

Reciprocal trust across machine actors

A calling Agent, service or workload and the API can each carry resolvable identity. Policy can evaluate both sides and the relevant relationship/evidence rather than one opaque assertion.

CALLING SIDE

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

API SIDE

API ECZ-ID, responsible organisation, endpoint/description bindings, current state and evidence.

DECISION BOUNDARY

The interaction can compare the intended caller, intended API and the relationship/policy relevant to this operation. Authentication and runtime controls still apply.

EXAMPLE

An agent resolves the API Passport before invoking a consequential endpoint, then authenticates and presents the permissions required for that specific operation.

BENEFIT

The target identity remains independently re-checkable even when the endpoint or gateway changes.

ENVIRONMENT AND VERSION SEMANTICS

Treat environment/version as context around one logical API

An API may expose multiple environments or versions. Whether they share one logical identity or require distinct identities depends on the intended object boundary; do not decide by URL alone.

SAME LOGICAL API

A production endpoint moves gateway/provider or version while remaining the same intended service interface → preserve ECZ-ID and update bindings/state where appropriate.

DIFFERENT LOGICAL OBJECT

A materially distinct API/service interface with different responsibility or object boundary may legitimately need its own Passport.

OBJECT-BOUNDARY RULE

Identity follows the intended logical object, not a mechanical “one URL = one Passport” rule.

CONSUMER CHECK

Confirm the returned environment/version is appropriate for the intended interaction.

OPERATOR CHECK

Keep supported environment/version bindings and evidence current through the canonical control plane.

LIFECYCLE AND MUTATION MODEL

Change context, preserve identity

When the logical API remains the same, update supported context instead of minting another ECZ-ID merely because infrastructure moved.

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.

**TYPICAL MUTATIONS**

Base URL · gateway · OpenAPI/description · provider ID · environment/version · evidence · organisation relationship · published state.

TYPICAL NON-MUTATION

The canonical API ECZ-ID remains stable when those contextual elements change for the same logical API.

AFTER EVERY MATERIAL CHANGE

Re-resolve the public record and confirm counterparties can observe the intended current context.

FAILURE HANDLING

Do not fabricate identity when proof is missing

Integrations should distinguish identity/proof problems from transport/authentication problems and apply fail-safe behaviour appropriate to the consequence.

NOT FOUND

Confirm identifier and official Resolver route; do not create a fallback identity in the caller.

FAMILY/OBJECT MISMATCH

Reject or challenge if the record is not the API object expected.

MISSING BINDING/EVIDENCE

Treat it as absent; do not infer from historical configuration or marketing copy.

STALE STATE

Re-resolve or require fresh evidence before proceeding when consequence demands it.

NETWORK/CLIENT FAILURE

Distinguish inability to fetch proof from proof that the API identity is invalid.

AUTHENTICATION FAILURE

Handle in the API auth layer; it is not proof the Passport identity is wrong.

FAIL-SAFE PRINCIPLE

When required identity/context is absent or too stale for the action, challenge, deny, delay or seek fresh proof according to policy - never silently infer.

DEVELOPER QUICK REFERENCE

The external contract in one page

Keep this model in mind when integrating API Passport into a client, gateway, agent or policy path.

OBJECT Logical API/service interface	CANONICAL ID Persistent API ECZ-ID
PUBLIC PROOF ECZ-ID Resolver	MACHINE ROUTE GET /api/p/{ecz_id}.json
RESPONSIBILITY Parent/organisation context where supported	BINDINGS Endpoint, gateway/provider IDs, OpenAPI/description, service refs
STATE Point-in-time published context where supported	AUTHORITY Action-specific context where present; never blanket permission
AUTH Separate API authentication/authorization layer	FAILURE Do not infer missing proof; fail safely according to consequence

NEXT DOCUMENT

Operate the Passport over time

The integration is only useful if endpoint bindings, state, evidence and responsibility context remain current.

READ NEXT

ECZ-ID API Passport Operations, Lifecycle and Troubleshooting Guide

OPERATOR OUTCOME

Maintain one canonical identity while API infrastructure and evidence evolve.

CONSUMER OUTCOME

Counterparties can continue to re-check current public proof rather than depending on stale integration assumptions.

SUPPORT

support@ecocitizenz.com · Include the ECZ-ID, failed step, visible error and occurrence time. Never send passwords, API keys, tokens, certificates or other secrets.

Related shared reading: Resolver & Passport Verification Guide · Security, Trust Boundaries & Safe Interpretation · Customer Dashboard & Operations Guide