

SHARED FOUNDATION · 03/06

ECZ-ID Resolver and Passport Verification Guide

How humans and machines inspect ECZ-ID Passport proof safely, interpret freshness and fail safely when required data is absent.

PRIMARY READER

Anyone verifying an ECZ-ID, security teams, developers, auditors and counterparties

SECONDARY READER

Identity teams, risk/compliance and automated-system designers

WHEN TO READ THIS

You have an ECZ-ID and need to know what can be verified, what the record means and what conclusions are safe.

RELEASE

2026.10
16 October 2026

What you will know after reading

You can verify identity, object/family, responsibility, current state, bindings, evidence, supported authority/delegation and freshness without inventing absent proof.

What this document does not establish

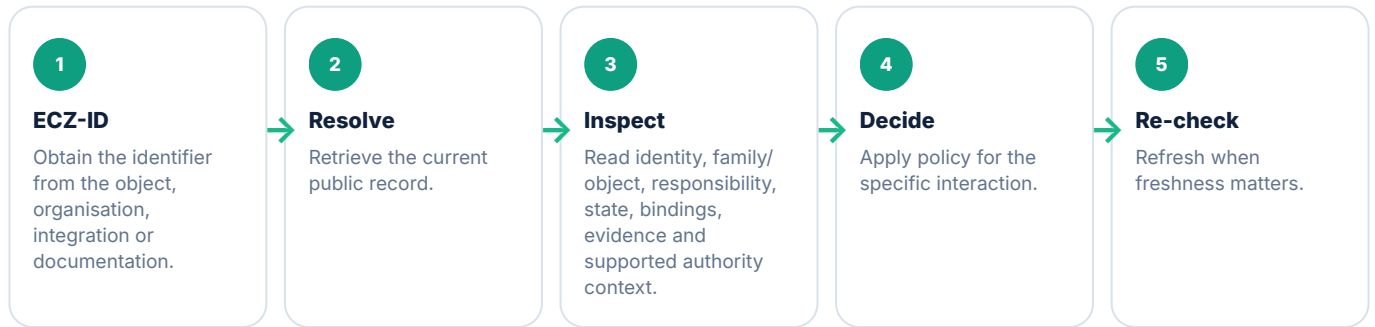
Resolver does not grant permission, mutate canonical identity, certify safety, or prove claims that are not present in the record.

Permanent route: <https://developers.ecocitizenz.com/guides/resolver-passport-verification/> · Support: support@ecocitizenz.com

RESOLVER IN ONE PAGE

Resolver is the public, read-only proof surface

Resolver exists so a person or machine can independently re-check bounded public proof without entering the authenticated customer control plane.



Resolver is proof, not control. It does not mutate the canonical identity or replace Dashboard.

VERIFY AS A PERSON

Human verification: a disciplined nine-step workflow

Use the same order every time so one attractive field does not override missing or stale context.

1

Obtain the ECZ-ID.

2

Resolve it through the official
ECZ-ID Resolver.

3

Confirm expected Passport
family/object.

4

Inspect organisation/Parent
responsibility.

5

Inspect current published state.

6

Inspect bindings relevant to
your intended interaction.

7

Inspect evidence source, scope
and freshness.

8

If authority matters, verify the
supported authority/delegation
context separately.

9

Re-check immediately before
consequence when freshness
policy requires it.

VERIFY AS SOFTWARE

Machine verification: parse, evaluate and fail safely

Automation should treat Resolver as structured evidence input, not as a single universal “trust” bit.

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

Parse stable identity

Use the ECZ-ID and the fields actually returned. Do not invent schema fields that are absent.

Check expected object

Confirm the Passport family/object and relevant relationship you expected to see.

Evaluate decision context

Inspect the bindings, state, evidence and supported authority context your policy actually requires.

Fail safely

If a required field or evidence is absent, treat it as absent. Do not infer it from marketing or prior cache.

Avoid universal verdicts

Do not convert one field into “safe”, “certified” or “approved for everything”.

Re-resolve

Refresh according to consequence and freshness policy. High-consequence paths should not rely on stale proof.

RECORD ANATOMY

Read the record by layer

Each layer answers a different question. Keep those questions separate when forming a decision.

Identity

Which persistent ECZ-ID is being presented?

Family / object

What class of Passport or object does this record represent?

Responsibility

Which organisation / Parent relationship is published?

Current state

What point-in-time state is published now?

Bindings

Which external surfaces or provider-native references are associated?

Evidence

What supporting evidence references are present, with what source/scope?

Authority / delegation

What supported action-specific context is present, if any?

Receipts where present

Where receipt evidence is exposed, interpret it as bounded evidence with source/time/scope rather than universal assurance.

CURRENT PROOF

Freshness is part of verification, not an afterthought

The ECZ-ID may persist while the context around it changes.

What can remain stable

The persistent ECZ-ID for the same legitimate object.

What can change

State, bindings, provider references, evidence, responsibility context or supported authority/delegation.

1

Observe

Record what the Resolver says now.



2

Use

Apply it only within the policy window appropriate to the action.



3

Expire

Do not keep a trust conclusion indefinitely.



4

Re-check

Resolve again when the action or time window demands current proof.

RE-CHECK REQUIRED: if the consequence depends on current state, authority or evidence, a stale cached record is not enough.

BINDINGS & EVIDENCE

Bindings and evidence support context - they do not erase boundaries

A binding shows an association; evidence supports a bounded claim. Both require contextual interpretation.

Safe conclusion

- The record publishes the shown endpoint / provider / repository / manifest / package / runtime relationship.
- The evidence reference exists and can be evaluated by its source, scope and freshness.
- The ECZ-ID remains the canonical identity being resolved.

Not established

- The external system is automatically trusted because it is bound.
- The provider-native identifier has become the ECZ-ID authority root.
- Every external declaration is independently verified because an evidence item exists.

AUTHORITY

Authority/delegation must be evaluated for the action

Identity answers who/what. Authority answers whether the represented actor may perform a particular consequence.

Ask separately

Does the record contain supported authority or delegation context relevant to this exact action? What local policy applies? What runtime/IAM control will enforce the decision?

Do not infer

Do not infer permission to spend, call arbitrary APIs, invoke tools, control devices, deploy software or change systems merely because the identity is valid.

Identity is not blanket authority.

INTERPRETATION

Safe conclusions versus unsafe leaps

This is the shortest way to audit whether a verification workflow is over-claiming what the record proves.

OBSERVED	SAFE CONCLUSION	UNSAFE LEAP
Agent ECZ-ID + organisation relationship	The record presents that Agent identity and publishes the relationship.	The organisation has independently verified all agent behaviour.
Current state shown	This is the current published state at the time resolved.	The state will remain unchanged indefinitely.
Authority context shown	Evaluate it for the specific action and policy.	The actor can perform every action.
Evidence missing	The required evidence is not established by this record.	Assume it exists elsewhere.

FAIL-SAFE BEHAVIOUR

Failure handling: distinguish absent proof from technical failure

A mature verifier distinguishes “not established” from “could not retrieve” and handles both explicitly.

ECZ-ID does not resolve

- Confirm the identifier was copied correctly.
- Confirm the official Resolver/machine-readable route.
- Distinguish “not found” from network/client failure.
- Do not fabricate a fallback identity.

Expected evidence is absent

- Treat it as absent.
- Do not infer it from marketing or previous screenshots.
- Request the appropriate evidence through the relevant security/business process if needed.

Resolver unavailable

- Apply the local fail-safe policy for the consequence.
- Do not silently downgrade a verification requirement.
- Retry or route to a controlled exception process where policy permits.

Support escalation

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

QUICK REFERENCE

60-second verification checklist

Use this as the minimum manual or automated control sequence.

Verify

- Correct ECZ-ID
- Expected family/object
- Expected responsibility/Parent relationship
- Current published state
- Relevant binding(s)
- Required evidence
- Required authority/delegation context
- Freshness acceptable for the consequence

Then decide

- Proceed only on what the record actually supports.
- Challenge or deny when required context is absent.
- Re-resolve when freshness matters.
- Keep authentication/authorisation/runtime policy separate.

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

Next: After acquisition, use the Customer Dashboard and Operations Guide. For policy engineering, use Security, Trust Boundaries and Safe Interpretation.