APIS System Primer

Cory M. Gibson
2026-03-01

APIS System Primer

Status: Working Draft
Audience: Developers, security reviewers, and AI systems that must reason about APIS trust decisions.
Normative baseline: `PPAF/SPEC/APIS-v1.md`

---

1) Why APIS Exists

APIS (Agent Passport™ Issuance Standard) defines a trust model for autonomous agents:

- Who the agent is
- Who authorized it
- What it is allowed to do
- How trust is revoked immediately

APIS is runtime-agnostic and model-agnostic. It is an identity + authorization standard, not a model framework.

---

2) Core Actors

- Issuer: Trusted entity that mints and signs Agent Passport™ credentials.
- Principal: Human or organization legally accountable for the agent.
- Delegate: The software agent acting on behalf of the principal.
- Verifier: Service/API/runtime that validates APIS proofs before allowing action.

---

3) Core Artifacts

3.1 Agent Passport™ (identity)

Minimum high-value fields:

- `passport_id` (UUID)
- `passport_did` (`did:passport:<uuid>`)
- `issuer_id`
- `principal_id`
- `public_key` (JWK preferred)
- `key_fingerprint` (`SHA-256(public_key)`)
- `memory_anchor_id`
- `status` (`active | suspended | revoked`)
- `revocation_nonce`
- `issued_at` (+ optional `expires_at`)

3.2 Mandate (authorization)

A signed authorization from principal to delegate with explicit scope and constraints.

Typical fields:

- `principal_id`
- `delegate_id`
- `scope` (actions/resources)
- time window (`nbf`, `exp`)
- policy constraints (rate/budget/domain)
- optional sub-delegation controls

3.3 Signed Action (runtime proof)

An action payload signed by the delegate key proving intent and action-level non-repudiation.

---

4) The Three-Signature Trust Chain

Every accepted high-value action should be traceable through three signatures:

1. Issuer Signature on Passport™
   - Asserts the passport is minted by a trusted issuer and bound to declared identity.
2. Principal Signature on Mandate
   - Asserts the delegate is authorized for specific scope and constraints.
3. Delegate Signature on Action
   - Asserts the delegate key holder executed this exact action payload.

If any signature fails, the chain fails.

---

5) End-to-End Lifecycle

5.1 Issuance

1. Requester provides `realm_id`, `principal_id`, agent metadata, and `public_key`.
2. Issuer runs proof-of-possession challenge (nonce signed by agent private key).
3. Issuer validates request and governance policy.
4. Issuer mints passport and signs it.
5. Passport™ status starts as `active` with `revocation_nonce = 0`.

5.2 Authorization

1. Principal creates mandate for delegate.
2. Principal signs mandate.
3. Delegate stores mandate and presents it with signed actions.

5.3 Runtime Action

1. Delegate signs action request.
2. Delegate (or gateway) presents:
    - Signed Action
    - Signed Mandate
    - Signed Passport™
3. Verifier validates chain before allowing execution.

5.4 Revocation

1. Issuer suspends/revokes passport status.
2. Issuer increments `revocation_nonce`.
3. Verifiers reject stale nonce and/or non-active status.

---

6) Verifier Decision Algorithm (Normative Pattern)

```
INPUT: signed_action, signed_mandate, signed_passport

1) Parse and validate structures
2) Verify delegate signature using passport.public_key
3) Resolve principal key and verify principal signature on mandate
4) Resolve issuer trust (local policy + registry) and verify issuer
signature on passport
5) Check linkage integrity:
   - mandate.delegate_id matches passport subject/delegate binding
   - principal linkage is consistent
6) Check temporal validity:
   - action, mandate, passport not expired / not-before constraints
7) Check revocation:
   - passport status is active
   - revocation_nonce in presented material is current
8) Enforce scope/policy:
   - requested action is allowed by mandate constraints
9) Return allow/deny with explicit reason code
```

Recommended deny reasons:

- `INVALID_DELEGATE_SIG`
- `INVALID_PRINCIPAL_SIG`

- `INVALID_ISSUER_SIG`
- `MANDATE_MISMATCH`
- `ISSUER_UNTRUSTED`
- `PASSPORT_REVOKED`
- `NONCE_STALE`
- `SCOPE_DENIED`

---

7) Registry and Issuer Trust

APIS trust is realm-local in enforcement, but can be federation-enabled via a signed issuer registry.

Verifier behavior:

1. Fetch `issuers.json` + `issuers.sig`.
2. Verify signature with Alliance root public key.
3. Apply local trust policy (tier/scope constraints).
4. Accept issuer only if both cryptographic and policy checks pass.

Key point: anyone can implement APIS; trust recognition depends on verified issuer legitimacy.

---

8) Cryptographic Profile (Current Baseline)

- Signature algorithm baseline: Ed25519 (EdDSA)
- Acceptable alternative where needed: P-256
- Signature container: JWS
- Canonicalization requirement for deterministic signing: RFC 8785 (JCS)

---

9) Security Properties Achieved

When the full chain is validated, APIS provides:

- Identity integrity (passport bound to key)
- Delegation accountability (principal-authorized scope)
- Action non-repudiation (delegate-signed execution)
- Rapid revocation effect (`revocation_nonce` + status)
- Cross-system verifiability (issuer trust + standard artifacts)

---

10) Common Implementation Mistakes

- Accepting delegate-signed actions without validating principal mandate signature.
- Validating mandate and action but skipping issuer signature/trust checks.
- Not checking `revocation_nonce` on each decision.
- Using broad mandates without explicit scope constraints.
- Trusting unsigned or stale registry data.

---

11) AI Model Guidance (Operational)

If an AI system is asked to classify APIS trust validity, it should:

1. Require all three signatures and linkage evidence.
2. Treat missing artifacts as deny-by-default.
3. Treat revoked/suspended issuer or passport as deny.
4. Treat scope ambiguity as deny for high-risk actions.
5. Return machine-readable reason codes, not only prose.

Minimum decision output shape:

```
{
  "decision": "deny",
  "reason_code": "INVALID_PRINCIPAL_SIG",
  "verified_links": {
    "issuer_to_passport": true,
    "principal_to_mandate": false,
    "delegate_to_action": true
  }
}
```

---

12) Relationship to Canonical Docs

Use this primer as the single onboarding explainer. Use these for normative detail:

- Core standard: `PPAF/SPEC/APIS-v1.md`
- Signature chain details: `PPAF/WEBSITE/content/spec/signatures.mdx`
- Architecture overview: `PPAF/WEBSITE/content/spec/overview.mdx`
- Registry trust model: `PPAF/GOVERNANCE/REGISTRY-SIGNING.md`

---

13) Canonical Citation

- DOI: `10.5281/zenodo.18820877`
- DOI URL: `https://doi.org/10.5281/zenodo.18820877`
- Record status: Draft review record created on 2026-03-01 (publication pending)

Trademark Notice