APIS v2.1 — Spec Delta (DRAFT)

Status: DRAFT, working lock-in — with a reference proof attached (§9, Herman). Supersedes nothing in v2.0; **adds** five things and **corrects** one overclaim. Cut against:

- APIS v2.0, Zenodo DOI 10.5281/zenodo.18820877 (Version 2.0, published 2026-05-04)
- PresenceOS spec §9 (Identity & Attestation Tiers)

Author: Cory M. Gibson / AetherPro Technologies LLC Last updated: 2026-06-24

---

0. Why 2.1 exists

v2.0 defined a two-pole tier model: Tier 1 (hardware TPM) and Tier 2.5 (DNS-anchored). The real fleet has a gap between those poles — VMs that **cannot reach a hardware TPM** but **can still produce a hardware-signed attestation** (AMD SEV-SNP, Intel TDX) or expose an **attestable vTPM** (some clouds, not OVH public-cloud KVM). v2.0 forces those boxes down to DNS-only, which under-claims their actual root of trust.

2.1 closes that gap, makes tier selection an algorithm instead of a label, scopes the DNS-anchor secret correctly, and adds model-access scope to the mandate.

---

1. CHANGE: insert Tier 2.0 (confidential-compute attested)

New tier between 1.5 and 2.5.

Attestation available on the node	Tier	Key property
Physical TPM 2.0, manufacturer cert chain	**1**	Signing key non-exportable; breach-containment is clean
Virtualized TPM **with** an attestable cert chain (e.g. cloud-attested vTPM)	**1.5**	Key protected by vTPM; chain rooted in cloud attestation service
Confidential-compute attestation: AMD SEV-SNP report	**2.0** *(new)*	No TPM, but the *VM measurement* is hardware-signed by the

or Intel TDX quote		CPU vendor. Key still on disk inside the enclave; attacker outside the enclave cannot forge the measurement
DNS-registered identity (TXT-published key, DKIM pattern)	**2.5**	Attribution only; security reduces to DNS-account security
Software key, no anchor	**4**	Attribution by convention only

Containment guarantee is stated per-tier (carry-over correction from v2.0 review):

- Tier 1 / 1.5: revoking all passports on a compromised machine is sufficient — keys are non-exportable.
- Tier 2.0: the *measurement* can't be forged, but the signing key lives on disk inside the guest; a compromise inside the guest can exfil it. Containment holds only until revocation propagates.
- Tier 2.5 / 4: keys on disk, no enclave. Same propagation caveat. **Do not claim single-operation breach containment at these tiers.**

---

2. CHANGE: tier resolution is an algorithm, not a hand-set label

> Rule (TPM-first, descend the stack): **a node mints at the highest tier it can actually attest, auto-detected. Never hand-pick a tier upward of what the box can prove.**

Resolution order, highest to lowest:

1. Physical TPM with cert chain → **Tier 1**
2. Attestable vTPM (cert chain present) → **Tier 1.5**
3. SEV-SNP report available → **Tier 2.0**
4. TDX quote available → **Tier 2.0**
5. DNS anchor reachable for the realm → **Tier 2.5**
6. None → **Tier 4** (and warn loudly)

The minted passport **must** carry the resolved tier and the attestation evidence reference. Verifiers and operators read the tier off the passport and **must not assume Tier 1**. (Reference implementation: `resolve_attestation_tier()` in `attestation_resolver.py`, shipped alongside this delta.)

Honest caveat baked into the resolver: **device presence is not attestability.** `/dev/tpmrm0` existing does not mean the cloud gives you a manufacturer cert chain. The resolver probes for the *evidence*, and if it can produce a report but cannot anchor the chain, it records the tier conservatively (e.g. 2.0 not 1.5, 2.5 not 1).

---

3. CHANGE: DNS anchor is issuer-written, never agent-written

This is the v2.0 review finding made normative.

- The DNS TXT anchor for an agent is **written by the issuer**, on the box where the realm signing key already lives — **not** by a `passport-init` CLI running on the agent node.
- The agent node receives a credential. It **never** holds a zone-write token.
- If DNS writes are automated, the token is scoped to **one zone, TXT records only**, and the threat model **states explicitly**: *Tier 2.5 security reduces to DNS-account security. Whoever controls the zone can rotate any agent's published key. DNSSEC protects the record in transit, not the legitimacy of its contents.*

Rationale: co-locate the DNS root-of-trust with the signing root-of-trust. Scattering a zone-write token to every agent node recreates the long-lived-secret-on-the-box failure mode (the OVH root incident) one layer up.

Record shape (DKIM-equivalent), normative example:

```
; published by issuer, under the realm zone
agent-name._apis.aetherpro.us.  IN  TXT  "v=APIS1; t=2.5; k=ec-p256;
p=<base64url-pubkey-or-thumbprint>"
```

`.well-known` issuance chain stays as PresenceOS §9 defines it (`alliance-root.jwk`, `apis-issuer-jwks.json`, `apis-issuer-delegation.json`).

3A. CHANGE: namespace-control proof is provider-neutral (ACME/certbot model)

§3 fixes *who* writes the anchor (the issuer). This section fixes *how* the issuer proves the right to write it — and removes any implied dependency on a specific DNS provider.

Provider-Neutral Namespace Proof

APIS does not require Cloudflare or any specific DNS provider.

APIS requires proof of namespace control. A principal, issuer, or delegated operator may satisfy this requirement through one of the supported challenge or anchor publication methods:

1. DNS TXT publication
2. HTTPS `.well-known` publication
3. DNS provider API automation
4. Manual verifier-approved publication

Provider integrations such as Cloudflare, Route53, Google Cloud DNS, Namecheap, or other DNS APIs are implementation adapters, not protocol requirements.

For DNS-based Tier 2.5 anchoring, the issuer or operator publishes a TXT record under a domain or subdomain controlled by the principal or issuer. The verifier resolves the TXT record and confirms that the published fingerprint or public key matches the passport claim.

Agent nodes must never receive DNS write credentials.

Challenge flow (ACME/certbot-equivalent)

Anchored-evidence status is granted only after the issuer verifies namespace control — exactly as ACME proves domain control before issuing a TLS certificate:

```
1. Operator requests a passport (principal, agent, mandate).
2. Issuer generates a challenge (nonce).
3. Operator publishes proof under a namespace they control.
4. Issuer verifies the proof via public resolution.
5. Issuer mints / upgrades the passport's anchored-evidence status.
```

Manual DNS-TXT MUST remain a first-class path, so an issuer whose zone is registrar-managed (e.g. Namecheap) is never blocked.

Two anchor families (both Tier 2.5; distinguished by `evidence_kind`, not by sub-tier):

```
herman._apis.syndicateai.co  IN TXT  "v=APIS1; t=2.5; k=ec-p256;
p=<thumbprint>"   # evidence_kind: dns-txt
https://syndicateai.co/.well-known/apis/agents/herman.json
# evidence_kind: https-well-known
```

`evidence_kind` ∈ { `dns-txt`, `https-well-known`, `dns-pending` }. A `dns-pending` passport is minted but not yet anchored; it MUST NOT be presented as complete Tier 2.5 until the record exists and verifies.

Identity proof vs node binding (keep separate). Namespace control proves *who the agent is*. It does **not** require the agent's host to be publicly reachable. Node reachability/attestation is a **separate, optional** field

(hostname, Tailscale node, TPM/SEV-SNP/TDX evidence per §1–2). Requiring the agent VM to expose public HTTP would break sovereign/private deployments and is therefore NOT required for identity.

Reference adapters (implementation conveniences, normatively non-binding): Cloudflare, Route53, Google Cloud DNS, Namecheap/registrar-manual, HTTPS `.well-known`.

---

4. CHANGE: add `model_scope` to the mandate

Model access becomes a **mandate-scope and revocation** concern, not a tier.

Directly addresses the Fable/Mythos export-control class: when access to a model class is suspended, you narrow or revoke the mandate; bound circuits trip; RedWatch logs it. This is APIS #9 (revocation propagation) applied to model access — and it is demoable.

Mandate schema addition (additive, backward-compatible):

```
{
  "mandate": {
    "scope": ["..."],
    "model_scope": {
      "allowed_classes": ["general", "voice", "stem-agentic"],
      "allowed_models": ["grm-2.6-plus", "qwen3.6-27b", "qwen3.6-35b-
a3b"],
      "denied_models": ["claude-fable-5", "claude-mythos-5"],
      "policy": "deny-overrides"
    }
  }
}
```

Semantics:

- `policy: deny-overrides` — a model in `denied_models` is blocked even if its class is in `allowed_classes`.
- A passport with a `model_scope` that no longer permits a model class is, for that class, equivalent to revoked. The PEP enforces it at the gateway (Aether Gateway) before the call reaches the model server.
- Revoking/narrowing `model_scope` propagates through the same bus + RedWatch ledger path as any other revocation.

---

5. POSITIONING (no schema change): APIS-APP is the certbot/ACME of agents

Keep this framing explicit in 2.1 prose. APIS-APP : agent passports :: ACME + certbot : TLS certs. Prove control of the realm/domain, get the passport auto-issued and renewed. Tier 2.5 anchor publication is the DKIM analog (public key in DNS). This is the language for adoption and it's accurate.

---

6. CORRECTION: soften the CMMC claim (defensibility)

v2.0 abstract says the model "enabl[es] CMMC Level 2 compliance." That is an overclaim a spec cannot make — compliance is determined by a formal assessment, not a document. 2.1 language:

> "APIS is **designed to support** NIST SP 800-171 / CMMC Level 2 **control objectives** for AI-agent identity and audit. Compliance is established by formal assessment, not by adopting this standard."

Same correction class as the v2.0 "legally binding → legally attributable" fix.

---

7. Before publishing 2.1 — open items to verify (not assumptions)

- ☒ **Verified (Herman mint, 2026-06-24):** the live mint endpoint accepts `model_scope` nested inside the mandate object, persists it, and the minted passport carries it verbatim and validates against the published JWKS. Remaining: the endpoint hardcodes a 90-day `exp` and has no `evidence_kind` field — add both so a passport can carry `dns-pending` vs `dns-txt` and a custom expiry (see §9).
- ☐ Confirm which fleet boxes can actually produce SEV-SNP / TDX evidence. OVH public-cloud KVM: **assume no** until probed. The resolver tells you.
- ☐ Confirm the home node CPU (when built) supports SEV-SNP or TDX, or you stay at 2.5 there too.
- ☐ Decide publish strategy (see note below).

---

8. Publish-vs-prove (decision note)

You said you didn't want to publish 2.1 until it actually works — so it's proof, not a paper. Both can be true and they're not coupled:

- The **tier/model_scope spec text** can publish as 2.1 now. It's defensible on its own.

- The **“it works” proof** is a separate artifact: a recorded run where the resolver mints Herman at his real resolved tier, a `model_scope` narrowing trips a bound circuit, and RedWatch logs it. That’s your demo, and it’s stronger as evidence *attached to* the published 2.1 than gating it.

Recommendation: publish 2.1 spec text + ship the resolver, then record the proof run against it. Don’t let the demo gate the spec.

---

9. Reference Proof: Herman (first conforming mint)

The proof run §8 calls for is done. The first APIS v2.1 agent passport was minted under the `syndicate` realm and validates end-to-end. Receipt, not diagram.

- DID: `did:passport:syndicate:herman`
- Principal: `AetherPro Technologies LLC` (ORGANIZATION, US)
- Tier: 2.5 · mandate carries `model_scope` with `policy: deny-overrides` (allow `grm-2.6-plus` / `qwen3.6-27b` / `qwen3.6-35b-a3b`; deny `claude-fable-5` / `claude-mythos-5`)
- Issuer: realm signing key `apis-realm-issuer-2026-05-09`, authorized by a root-signed delegation
- Passport signature: ES256, **verified against the published issuer JWKS** (`passportalliance.org/.well-known/apis-issuer-jwks.json`)
- Anchor: `herman._apis.syndicateai.co`
- Anchor method: DNS TXT
- DNS provider: **Namecheap** (registrar-managed zone; no Cloudflare, no DNS API)
- Result: the TXT record resolved publicly and its `p=` value matched the passport’s `public_key_fingerprint` claim exactly (`sha256/hBN9-eo9H_igHiwalPt1Ft7kPUA4YRNGvXLIIcnY7iU`)

Conclusion: the full chain — root → issuer delegation → realm issuer → agent passport → DNS anchor — validated from published artifacts alone, and **Tier 2.5 DNS anchoring is provider-neutral and does not require Cloudflare.** Proven by hand on a registrar-managed zone.